



**Università
di Genova**

MSc Computer Engineering
Software and Computing Platforms Curriculum

**Design and development of an application
for tracking and evaluating processes and
procedures**

Candidate:

Fabio Mora

Student ID: 4373007

Advisor:

Prof. Enrico Russo

Co-advisors:

Dr. Giacomo Longo

Dr. Samuele Bertelli

October, 2024

Abstract

The growing number of cyber threats has prompted international agencies to strengthen national cybersecurity infrastructures, improving their ability to prevent, monitor and respond to incidents. These organisations require specialised personnel and established processes to achieve their operational objectives. Security exercises set in virtual systems such as cyber ranges are used to test security procedures and train technical personnel. These exercises typically involve a Blue team defending an infrastructure and a Red team launching attacks against it. The correction process of these simulations requires the intervention of trained personnel who make up the White team. This thesis aims to extend the scenarios that can be used during the exercises and to provide a system that can automatically track and evaluate the actions of the participants in real time. This system was tested through the reproduction of an exercise to verify the achievement of objectives.

Table of Contents

List of Figures	5
List of Tables	6
Chapter 1 Introduction	7
1.1 Motivations and objectives	7
1.2 Contributions	9
1.3 Overview of the Thesis	10
Chapter 2 Background	11
2.1 Process Mining	11
2.2 Business Process Modeling Notation	13
Chapter 3 Related Work	15
3.1 Scalable and automated Evaluation of Blue Team cyber posture in Cyber Ranges	15
3.2 A Framework for the Evaluation of Trainee Performance in Cyber Range Exercises	16
Chapter 4 Methodology	18
4.0.1 BPMN to structured data converter	20
4.0.2 Activities to structured data converter	23
4.0.3 Procedural Awareness	24
4.0.4 Dashboard	26

Chapter 5 Architecture implementation	27
5.0.1 BPMN to structured data converter	27
5.0.2 Activities to structured data converter	34
5.0.3 Procedural Awareness	35
5.0.4 Dashboard	40
Chapter 6 Experimental Evaluation	42
6.1 Case Study Overview	42
6.2 Case study implementation	45
6.2.1 BPMN to structured data converter	45
6.2.2 Activities to structured data converter	47
6.2.3 Procedural Awareness	47
6.2.4 Dashboard	48
6.3 Discussion	51
Chapter 7 Conclusion and Future Works	53
7.1 Conclusion	53
7.2 Future Works	54
Listing of acronyms	56

List of Figures

4.1	Logical Architecture	19
4.2	Application architecture design	20
4.3	List of components that the user can use to construct the BPMN graph . .	22
4.4	Example of a BPMN graph which meets the definition of well-structured. .	23
5.1	Example of BPMN graph.	33
5.2	Example of a Dashboard.	40
6.1	Cybersecurity exercise scenario.	42
6.2	Cybersecurity exercise scenario with representation of planned actions. . .	44
6.3	Model of the response procedure to a ransomware attack.	45
6.4	Progress status after 30 minutes by the Blue One team.	48
6.5	Progress status after 60 minutes by the Blue One team.	49
6.6	Progress status after 90 minutes by the Blue One team.	49
6.7	Progress status after 120 minutes by the Blue One team.	49
6.8	Progress status after 30 minutes by the Blue One team.	50
6.9	Progress status after 60 minutes by the Blue One team.	50
6.10	Progress status after 90 minutes by the Blue One team.	50
6.11	Progress status after 120 minutes by the Blue One team.	51
6.12	Original BPMN graph used during the exercise.	51

List of Tables

6.1	Blue One after 30 minutes	48
6.2	Blue Two after 30 minutes	48
6.3	Blue One after 60 minutes	48
6.4	Blue Two after 60 minutes	48
6.5	Blue One after 90 minutes	48
6.6	Blue Two after 90 minutes	48
6.7	Blue One after 120 minutes	48
6.8	Blue Two after 120 minutes	48

Chapter 1

Introduction

Over the past few years, the rise of cybersecurity threats has been one of the most significant challenges for global digital security. In 2023, as reported by the Clusit association [Clu24], a total of 2,779 severe cyber attacks were analyzed globally, representing a 12% increase compared to 2022. On a monthly basis, an average of 232 attacks were detected, with a maximum peak of 270 in April, which also represents the maximum value measured over the years. These attacks, ranging from targeted intrusions to large-scale data breaches, have caused significant damage in terms of loss of sensitive data, financial resources, and compromise of corporate reputation. From this perspective, examining the dynamics of these attacks and the countermeasures to be adopted becomes a global priority. To achieve these, teams of experts have been created, i.e. CSIRT, with the aim of managing cyber incidents. Virtualised environments such as cyber ranges are used to allow these teams to train and test processes and procedures in response to incidents. The objective of this thesis is to create a system capable of tracking and evaluating the activities of teams responsible for handling computer incidents during the execution of a procedure.

1.1 Motivations and objectives

In recent years, directives have been defined at European and global level to increase the level of national cybersecurity infrastructures. In 2016, the Network and Information Security (NIS) released the "Directive (EU) 2016/1148 of the European Parliament and of the Council" which aims to achieve a high common level of network and information systems

security throughout the EU [Cor24]. In this document, provisions are provided for the establishment of a national authority responsible for managing cybersecurity and responding to cyber incidents. This authority is called Computer Security Incident Response Team (CSIRT).

CSIRT : is an organised team of cybersecurity experts whose main objective is the management of cyber incidents. A CSIRT therefore offers all those services aimed at preventing, mitigating and resolving the impacts of cyber incidents.

Following Decree of the President of the Council of Ministers No. 65 of 8 August 2019 (NIS Directive), Italy has set up a national CSIRT at the National Cybersecurity Agency (ACN) called CSIRT Italia. One of its tasks is to promote and coordinate the implementation of the national network of CSIRTs and thus of the coordinated response to cyber incidents. In this regard, in August 2023, the ACN released the guidelines for the implementation of a CSIRT [ACN23]. This document defines the reference model for the implementation or enhancement of a CSIRT, which consists of the following 4 dimensions:

1. **Service model**: describes the catalogue of services offered by a CSIRT to fulfil its mandate to the Constituency;
2. **Processes**: identifies sequences of actions assigned to specific professional figures to carry out activities with the support of appropriate technological tools;
3. **Organisational model and professional figures**: describes the organisational structure of a CSIRT, in terms of an organisational chart, detailing the professional figures needed to provide the services offered and their roles and responsibilities, as well as the way in which the different figures iterate;
4. **Tools**: describes the tools used by the CSIRT staff to achieve the objectives within the mandate.

The task of the Processes part is to define, execute and monitor the processes that support the activities of the CSIRT. Hence the need for a system to test these processes and to train technical personnel to execute them. Modern security challenges, set in the cyber

range, are designed to test the participants' technical skills. These ranging from exercises that require them to recognise attacks in progress and implement appropriate remedial actions, to tests that assess participants' ability to keep services operational in the event of an attack. In order to test processes and train technical personnel, exercises are needed to consider the procedure as a whole. This would test, for instance, the effectiveness of the procedure, communication between different teams, the technical skills just described, and the ability of teams to faithfully follow the predefined procedure. With this thesis, we propose a method for extending traditional cybersecurity exercises. The goal is to implement a system that takes a graphical representation of a procedure as input and uses it as a model to track and evaluate the teams' progress in reproducing it. We want the system to be interoperable, providing real-time and automated tracking and evaluation throughout the course of the exercise. To the best of our knowledge, there has been no previous research into building a solution for this application.

1.2 Contributions

The contributions of this thesis are summarized as follows.

- A novel methodology that extends the capabilities of cyber ranges and the security exercises they host by enabling the definition of processes that participants must follow during the execution.
- A solution that tracks in real-time whether teams adhere to these processes, integrating the scoring system with metrics to evaluate their performance.
- A comprehensive test of the approach was conducted in a realistic scenario by executing an incident response exercise in a cyber range organised by Leonardo SpA.

1.3 Overview of the Thesis

In this section we describe the structure of the thesis. Chapter 2 covers the background knowledge needed to understand the topics covered in the following chapters. Chapter 3 presents the state of the art of the topics in this thesis. Chapter 4 shows the architecture we have chosen for each component of our application. Chapter 5 shows the implementation of each module using pseudo code when possible. Chapter 6 shows the case study we used to test the functionality of our application. The last Chapter 7 shows conclusions about our thesis and possible future work.

Chapter 2

Background

2.1 Process Mining

Process Mining is the area within the research field that deals with the analysis of event data that several information systems generate during the execution of processes. Process mining specifically uses event log data to generate process models, which can be used to discover, compare or enhance a given process. In this field, specialized algorithms are applied to event log data, to identify trends, patterns and details of how a process unfolds. In this topic, event logs play a fundamental role. They are log files that contain detailed information about the events that occur within a system or computer application. These events can include user actions, system operations, errors, alerts, transactions, and other types of activities recorded by the application or operating system. Each event in such a log refers to an activity (i.e., a well-defined step in some process) and is related to a particular case (i.e., a process instance).

Process Mining consists of three distinct sets of functions including discovery, conformance checking, and enhancement [Van12].

- **Discovery:** takes as input an event log and produces a process model without using any priori information. This is probably the most common Process Mining function, and it is used to understand the performance of processes, revealing bottlenecks and other areas for improvement;
- **Conformance checking:** compares event data, that the process execution produces,

to process models that define their normative or descriptive behavior, identifying any deviations from the intended model;

- **Enhancement:** extends or improves an existing process model using information about the actual process found in the event log.

In our thesis we will focus on the topic of conformance checking. In the state-of-the-art to interpret the results of this type of analysis are used four different metrics.

- **Fitness:** measures a model's ability to properly describe the behavior found in an event log and is often assessed by how accurately the model can replay the traces in the event log. A model has perfect fitness if all traces in the log can be replayed by the model from beginning to end. Often fitness is described by a number between 0 (very poor fitness) and 1 (perfect fitness);
- **Precision:** measures the extent to which the behavior allowed by the model remains sufficiently close to the behavior present in the event log. Precision conformance checking techniques determine superfluous activities and connections to quantify precision. A model that is not precise is "underfitting." Underfitting is the problem that the model over-generalizes the example behavior in the log;
- **Simplicity:** measures how complex a model is. This is often explained with the Occam's razor principle. It says, the simpler the model is, the better it is;
- **Generalization:** measures how much the process model represents the process paths in a generic way or only the behavior observed in the data. A model that does not generalize sufficiently is "overfitting." Overfitting occurs when a model is generated memorizing event log data and not generalizing its behavior.

2.2 Business Process Modeling Notation

Business Process Modeling Notation (BPMN) is a type of flowchart that models the steps of a planned process from beginning to end, developed by the Business Process Management Initiative (BPMI). The BPMN 1.0 specification was released to the public in May 2004. Its goal was to provide a notation easily understood by all business users: from the analysts who design processes, to the developers who implement them, to the managers who monitor them. The BPMN creates a standardized bridge between business process design and implementation by visualizing in detail the activities and information flows required to complete a process. It was created to represent business processes, but is now used in many other areas, such as process mining [Whi04]. BPMN uses these four types of elements for business process diagrams.

- **Flow object:** include 3 different symbol classes:
 - **Event:** an event that starts, modifies, or completes a process. Event types include message, timer, error, compensation, signal, cancel, escalation, link, and others. They are represented by circles containing other symbols based on the type of event. Based on their function, these are classified as "throwing" (which triggers an activity) or "catching" (which are generated by an activity);
 - **Activity:** a particular activity or operation performed by a person or system. It is represented by a rectangle with rounded corners. They can get more detailed with subprocesses, loops, compensations, and multiple instances;
 - **Gateway:** decision point that can shape the path based on conditions or events. They are represented as diamonds. They can be exclusive or inclusive, parallel, complex, or data- or event-based;
- **Connecting objects:** are lines that connect BPMN flow objects. There are three different types: sequence flows, message flows, and associations;
- **Swimlanes:** are used to organize aspects of a process in a BPMN diagram. Swimlanes visually group objects into lanes, with each aspect of the process added to

a separate lane. These elements can be arranged either horizontally or vertically. Swimlanes not only organize activities into separate categories, they can reveal delays, inefficiencies, and the workers responsible for each step in a process;

- **Artifacts:** represent information relevant to the model but not to individual elements within the process. The three artifact types are annotations, groups, and data objects that can be used in a BPMN diagram. All three are used to augment and describe a BPMN process.

Chapter 3

Related Work

The aim of this chapter is to describe the state of the art with regard to the main topics covered in this thesis. To do this, we present two projects that share certain topics and objectives with us. During this presentation we will analyse the differences and advantages of our work.

3.1 Scalable and automated Evaluation of Blue Team cyber posture in Cyber Ranges

Bianchi et al. [BBS24] propose a framework using Blue and Red Team reports and known databases to automate the evaluation of Blue Team results during cybersecurity exercises set in cyber ranges. The automatic evaluation pipeline is divided into four phases:

1. collection of reports from Blue and Red Teams;
2. definition of Reference/Response Graphs from reports;
3. automatic evaluation of multiple intermediate scores from defined graphs;
4. computation of the final score and the Cyber Posture.

During the first phase, the Blue and Red teams' reports are collected. The report structure proposed in this project is based on the components of the MITRE ATT&CK matrix, a

database containing the knowledge gathered by the security community on tactics, techniques and procedures used by attackers, together with the corresponding possible mitigation measures and detections. In these reports, the White team can assign weight to a specific field if they want it to be considered more during the Blue team’s evaluation. Having compiled the reports, in the second phase the scoring system uses them to create the Reference Graph and Response Graph using a variant of the ADTree algorithm. The first graph is assembled using the red team report and the ATT&CK database. Similarly, the second one uses the Blue team report. The first graph will be used as a reference to evaluate the work of the Blue team. Once the graphs are ready they are compared using the breadth-first method by assigning the correct weight to the nodes in the graph that the Blue team has correctly identified and deleting the others. The Blue team’s final score is an average of some intermediate scores that take into account the Blue team’s ability to recognise the attack strategy and implement the right mitigation techniques. From the information gathered, it is possible to define the Blue team’s cyber posture, which is a general indication of the capabilities demonstrated during the exercise.

This project shares with ours the objective of automating the correction of cybersecurity exercises through the creation of a document used as a template for the evaluation of participants’ work. Our thesis additionally aims for a broader set of exercises to be evaluated and does so in real time by also providing a tracking of the teams’ progress. In addition, it succeeds in evaluating the coordinated work of several collaborating teams.

3.2 A Framework for the Evaluation of Trainee Performance in Cyber Range Exercises

Andreolini et al. [And+20] propose a framework for the evaluation of the performance achieved by trainees involved in cybersecurity exercises implemented in modern cyber ranges. In particular, this system does not merely indicate whether or not the challenge was completed but aims to provide a measure of performance. The tool provides an initial set of uniform, standardized events at different granularities that represent the most common actions performed by a trainee during an exercise. These events are used to construct

an oriented graph representing the procedure performed by the participant. Vertices of the graph represent intermediate states reached by the trainee during an exercise, while edges represent the actions he or she performs to move from one state to the next. A graph representing the ideal actions to be performed during the exercise is provided by the exercise organizer. The two graphs are compared and the matching actions or states of the system are marked as correct, while a dummy node is added to the participant's graph for the remainder. Finally, scoring is done using the symmetric difference between the two graphs to identify the correct, missing, and incorrect actions. Scoring is done taking into account the closeness between the two graphs and how quickly the participant achieved the goal.

This project shares some objectives, even if more specific, and some methodological choices with ours, but differs completely in terms of implementation. Our thesis can be used to evaluate a wider range of exercises by assessing aspects not considered by this project such as collaboration between different teams. The evaluation is also different as in our case we consider the order of operations, the teams that are assigned to carry it out and the deadlines that can be set for each individual operation. The latter factor is due to a different exercise target.

Chapter 4

Methodology

Summary

The objective of our thesis is to create an application that can track and evaluate in real time the operations performed by teams in a system that can be real or simulated. For the evaluation, our system calculates the congruence of the procedure carried out by the teams, whose operations are being monitored, with a procedure used as a model of which a graphical representation is provided. To ensure that our tool is interoperable, we have defined interfaces that use common structures and data. In this way, any system implementing these interfaces can exploit our application regardless of its nature. The application can thus be used for example for the evaluation of cybersecurity challenges in a simulated environment, but thanks to the tracking function it can also be used to monitor the status of the system.

Logical Architecture

In this section, we present the logical architecture of our application. Figure 4.1 shows the main components with their respective inputs and outputs data.

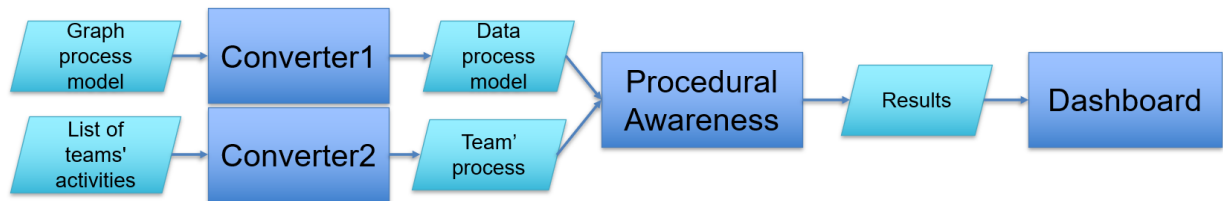


Figure 4.1: Logical Architecture

The first two elements are the converters. These modules are responsible for processing the data supplied as input by the user using our application. In the first case, the input data is a graph that defines the procedure which must be reproduced by the teams. The converter transforms the graph into structured data, retaining all the information it contains. In the second case, you need to provide the activities carried out by the teams as input to the converter. The second converter then creates an ordered list of activities describing the procedure performed. The data thus created is the element used to track and evaluate the work of the teams. The two elements just described prepare the inputs for the main component of our application, which is the Procedural Awareness module. This component takes the descriptions of the two procedures, the one used as a model and the one performed by the teams, compares them and derives an evaluation based on the congruence and tracking of the teams' progress. These results are sent to a dashboard for real-time visualisation. On the latter, a graphical representation of the procedures with results and real-time progress by teams can be displayed. Thanks to a colour system, it is possible to distinguish activities performed, those performed late, those performed in disorder or those not performed and those performed by wrong team.

Architecture's design

In this section, we shall analyse each component of our architecture, explaining the functionalities the element implements, the inputs it requires and the outputs it provides. For each point, we will explain the choices made and the reason for them. The Figure 4.2 shows an ArchiMate model to describe the application layer of our thesis.

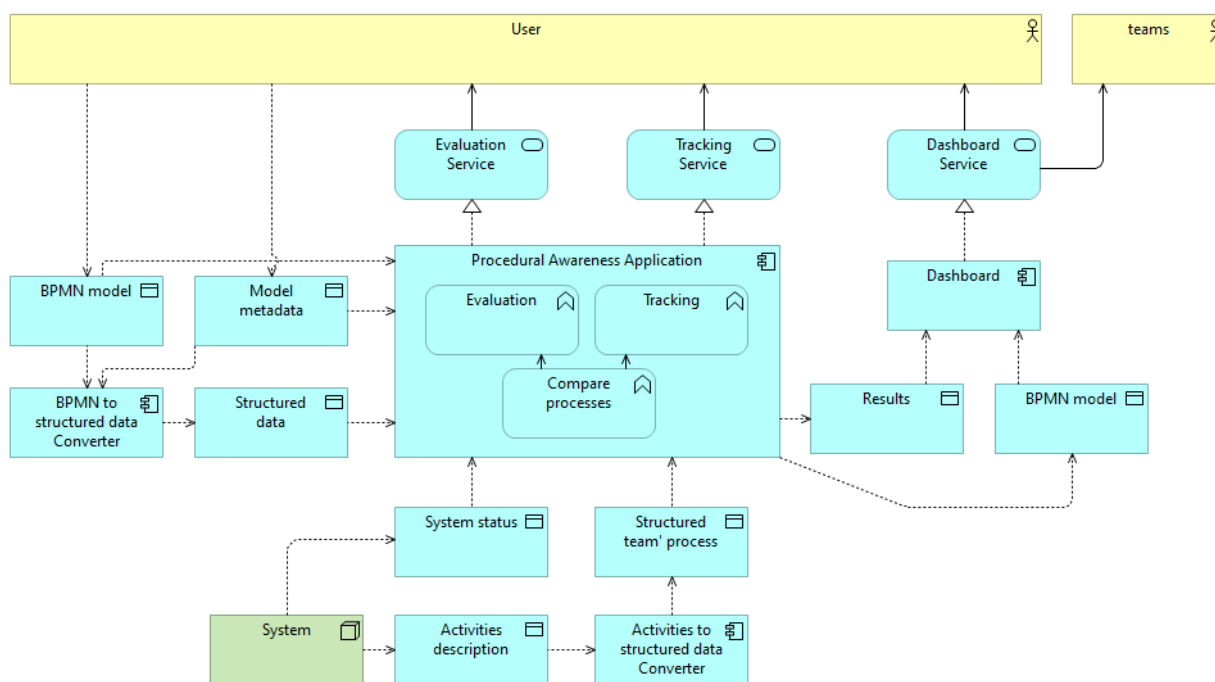


Figure 4.2: Application architecture design

4.0.1 BPMN to structured data converter

This module is responsible for mapping the process description from the graphical form to structured data. The conversion is necessary because the comparison in the Procedural Awareness module is done at the structural level, not at the graphical one. No information must be lost during the conversion and it must be organized in such a way that the Procedural Awareness module can exploit it in real time.

The converter has two types of input:

- **BPMN model:** is a sBPMN graph. This graph represents the procedure that will be used as a model during the evaluation process. For this reason, it is essential that the representation be clear and effective;
- **Model metadata:** is a structured document. This document contains additional attributes to the activities in the procedure. These attributes are associated with the activity to which they relate and will be exploited during the evaluation phase.

We made some assumptions to regulate the structure of the BPMN graph. The first concerns its elements. These types of graphs have more than 50 possible elements but on average a graph contains only 9 different constructs [MR13]. Since our goal is to have a clear and simple representation we decided to limit the types that can be used. We selected eight that are the most used elements to represent a scenario like ours [MR13; CT12], in which it is necessary to describe a process that has a name and involves several participants. The Figure 4.3 shows the elements we have selected, explains their meaning from a procedural point of view and gives a graphic representation of them.








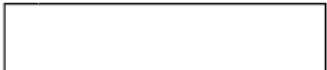
Element	Behavior	Visual
Sequence flow	The sequence flow denotes the execution order of elements in the process model.	
Task	The task represents an activity of the procedure	
Start event	Start event represents the instant of time when the procedure begins	
End event	End event represents the instant of time when the procedure ends	
Exclusive gateway	Element that causes the execution flow to split or join . Must be completed with a condition. Is used to create alternative process flows. It is called exclusive since only one of the possible choices can be followed.	
Parallel gateway	Element that causes the execution stream to split or merge. Is used to create parallel paths without evaluating any conditions.	
Pool	The Pool defines the boundaries of a process. All activities, events, gateways, and sequence flows within the Pool are part of the process that Pool represents.	
Lane	Lane defines the boundaries of a team. All activities in the lane must be completed by that team	

Figure 4.3: List of components that the user can use to construct the BPMN graph

Having defined the set of elements that can be used, we shifted our attention to defining a structure for the graph. The BPMN standard imposes no such requirement. In order to implement a reliable converter, we decided to use the well-structured graph definition defined by [Bie19].

Well-structured BPMN model : a BPMN model is well-structured if every node with multiple outgoing arcs (split) or multiple incoming arcs (join) is either a parallel- or exclusive gateway. Furthermore for every split construct in the model there is a corresponding join construct of the same type (parallel/exclusive), and vice versa,

such that the fragment of the model between split and join forms a single-entry-single-exit (SESE) process component.

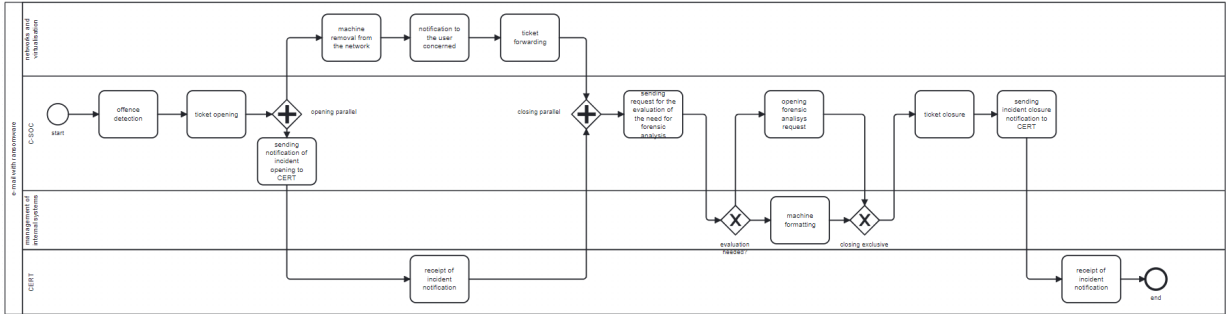


Figure 4.4: Example of a BPMN graph which meets the definition of well-structured.

With the inputs discussed above and thanks to the rules just described, the converter outputs structured data containing the information extracted from the graph enriched with attributes from the structured file. In order for the Procedural Awareness module to correctly interpret this output, we have defined a precise structure of it. The converter divides the graph into fragments. These fragments are composed of consecutive activities that are not divided by exclusive gateways, later we will understand the reason for this choice. In the case of parallel gateways, however, the fragment is not interrupted. By doing so, the output data will contain the entirety of the graph but divided into fragments.

4.0.2 Activities to structured data converter

This module is used to create the structured file representing the description of the procedure followed by the teams. The expected input is a series of events. Each event represents an activity of the procedure and must contain information such as the name of the activity, the name of the team that performed it, the time at which it was performed and any threshold value. The information is grouped and organised in a single structured file. The file thus created represents the output of the converter and is the step-by-step description of the procedure performed by the teams with all the attributes required for evaluation. Passed as input to the Procedural Awareness module, it is used to track and evaluate the teams' progress.

4.0.3 Procedural Awareness

This is the core of our application and performs three main functionalities:

1. It understands the right path to take in the procedure graph based on the state of the system. In this way it builds the model in real time;
2. Track teams' progress in following the procedure;
3. Evaluates the procedure carried out by the teams based on its congruence with the process model.

The inputs required by this module are:

- **BPMN model:** is the same graph passed to the first converter. It is passed as input only because the module must later send it to the dashboard but without making any changes;
- **Model metadata:** file containing additional data related to the model. This is the same file used by the first converter. From this file will be extracted information needed to calculate the evaluation;
- **Structured data:** is the output of the first converter. It contains the list of fragments into which the model is divided;
- **System status:** messages containing events coming from the system being monitored. These messages describe changes in the state of the system we use to update the model path;
- **Structured team' procedure:** is the output of the second converter. Is a structured file containing the ordered list of activities performed by teams. For each activity should be given name, timestamp, team that performed it, and the value to compare with the threshold if present.

The first functionality is necessary because the path to be followed in the graph is not always known a priori. In the presence of a specific element, the exclusive gateway, the

path becomes deterministic. This means that in those cases it is necessary to make a choice to find the right path based on the state of the system being used. The Procedural Awareness component monitors the state of the system waiting for events to update it. These events are the input that we call System status. When an event arrives this module use it to understand the right path to follow and updates the model accordingly. The other two functionalities are based on conformance checking carried out between the model of the procedure we just discussed and the procedure performed by the teams. To understand on what basis the evaluation is carried out, we must first explain what information is provided by the model and its metadata. The graphical model defines for each activity its order and the team that is assigned to perform it. The second establishes for each activity a deadline and a threshold value. This value, which may or may not be present, is an integer that may be a minimum or a maximum that is associated with a specific activity. During conformance checking operations, this value is used to assess whether or not teams have reached the required threshold while performing that activity.

The conformance checking evaluation is calculated on the basis of five metrics defined as follows:

- **Activity metric:** activities not performed in the procedure or performed but not required;
- **Order metric:** procedure activities in disorder;
- **Time metric:** late activities in relation to assigned deadlines;
- **Team metric:** activities performed by the wrong team;
- **Threshold metric:** activities that do not meet the threshold;

The algorithm used for conformance checking creates two models of our procedure. The first includes all the activities that are to be performed, the team that is to perform them and their order. The second is the time model. This template contains all the deadlines for each activity. The information is saved in the templates in the form of constraints. In order to perform the conformance checking, the models are compared with

the same information concerning the procedure performed by the teams. The result of this comparison in both cases is a list of constraints not met by the procedure. We then used these results to calculate the value of certain metrics not automatically calculated by the algorithm. Finally, all the results obtained are used to update the score. The data obtained in this way is output in a list. The output lists, for each evaluation metric, all activities that do not meet that specific metric. This list also contains the updated scores of the two teams.

4.0.4 Dashboard

This component is used to display the results obtained from the main module in real time. It shows the names of the teams with their scores and the BPMN graph of the model with the activities coloured according to the results obtained from conformance checking. For the creation of the graph, the module receives as input the results of the Procedural Awareness module and the BPMN graph of the model. The results are lists, one for each metric to be displayed, containing the names of the activities that satisfied the metric under examination. Each metric is then associated with a colour, with which the activity is coloured in the BPMN, to allow a simple and effective understanding of progress. Thus, thanks to the colours, it is immediate to recognise activities that have not been performed or activities that have been performed late. As we said, however, the Procedural Awareness module also allows progress to be tracked. To visualise this, only those activities on which you have information are coloured on the graph, leaving the others blank. Therefore, in order to understand where the teams are, it is sufficient to find the last coloured activity knowing that the following ones have not yet been tackled by them.

Chapter 5

Architecture implementation

In this section, we describe the implementation of each component by showing the technologies we have chosen. Using pseudocode whenever possible, we will show the operating logic of some elements.

5.0.1 BPMN to structured data converter

For the description of the implementation of this module, we start by talking about the inputs:

- **BPMN model:** as already mentioned several times, the graph representing the procedure must be in BPMN format. For reasons discussed in the previous chapter, the graph provided as input must follow the structure we defined;
- **Model metadata:** this data must be structured. We have chosen the json format for its simplicity to be parsed, because it is supported by many programming languages and also because it is easily readable and writable by a human being. The first key in this file represents the list of activities in the procedure. For each activity, name, timestamp, team and then two additional attributes, called add-on, such as timer and threshold are listed. Both of these last two attributes have a field indicating whether they are active or not and if they are, they have indications of their value. The deadline indicated the reference activity for the start of the timer, which can be the start event but also any activity in the procedure. The priority is also indicated,

the higher this value the more points will be deducted if this activity is late. The threshold, on the other hand, indicated the value and whether it should be a minimum or maximum limit. An example of an activity with the characteristics just described is included below.

```
{
  "concept:name": "sending incident closure notification to CERT",
  "time:timestamp": "2024-03-12T11:40:00.000+01:00",
  "team": "C-SOC",
  "addon": {
    "timer": {
      "active": "True",
      "start_activity": "start",
      "priority": 1
    },
    "threshold": {
      "active": "True",
      "value": 30,
      "segno": "min"
    }
  }
}
```

In addition to the activity key, there are other keys that give information about the teams participating in the exercise, the starting score, the starting time, and any points that must be taken off for each activity that meets a metric. There is also a key called “interval_in_second” that defines the time interval that must elapse between two successive starts of the function responsible for tracking and evaluating the procedure.

```

"starting_point": 1000,
"point_for_missing_activity": 10,
"point_for_messy_activity": 10,
"point_for_wrongteam_activity": 10,
"point_for_threshold": 10,
"point_for_completed_activity": 10,
"point_for_late_activity": 10,
"teams": [
    "network and visualisation",
    "C-SOC",
    "management of internal system",
    "CERT"
],
"interval_in_seconds": 300,
"start_time": "2024-03-12T08:40:00.000+01:00",
"end_time_in_minutes": 10

```

The purpose of this converter, as already mentioned, is to create a non-graphical description of the procedure represented in the BPMN. We use a table format such as dataframe to organise this data. This is because it is compatible with the library we use for conformance checking and because it is suitable for creating a list of elements with a series of attributes each. If the graph represented only a series of activities, without gateways, it would be sufficient to create a dataframe that lists them in order. With the presence of the exclusive gateways, which require a choice from the various available, this solution is not sufficient. We therefore decided to divide the graph into fragments. These are created from the start event and include all subsequent activities until an exclusive gateway is found. At this point the fragment is interrupted and from this point subsequent fragments will start, one for each possible path into which the procedure is divided by the exclusive gateway. These fragments in turn will stop at the next exclusive gateway or when the end event is reached.

The result of this process is a series of fragments describing all possible paths in the graph. The last problem to be solved is to find the method of choosing the correct path based on the condition of the gateway and the state of the system. In the Procedural Awareness section, we will explain how we solved this problem. Parallel gateways also need to be managed, as they also involve flow splitting. We have defined that in parallel branches, order is only considered for activities in the same branch, because for them the order is specified. Between different branches in parallel, the order is not specified, so it does not have to be taken into account in the score calculation. So in the division into fragments, the first converter puts branches in series in parallel by marking each activity with a specific attribute, so that the rules just discussed can be taken into account during evaluation and tracking.

We now use pseudocode 1 to describe how the converter works. As a first step, the program loads the two inputs just described. After that, a five-step process begins to create the dataframe list.

- **Step 1:** scrolls the entire graph following the arrows and for each gateway it encounters, whether parallel or exclusive, it adds it to a list indicating its type as an attribute. The start event is also added. This list represents all the elements from which the temporary fragments will start;
- **Step 2:** scrolls through the list created during Step 1 and for each element creates a fragment that includes all the following activities until another gateway or the end event is reached. At the end of this step we have a list of fragments starting from a gateway, or the start event, and ending at the first gateway they encounter in the BPMN, or the end event;
- **Step 3:** in this step, fragments containing a parallel gateway, which divides the execution flow, are selected from those created in Step 2. At this point, all fragments starting from the same parallel gateway are put in series. This is done because, as we explained earlier, the parallel gateway does not impose a choice;
- **Step 4:** during this step, the fragments divided by parallel gateways are concate-

nated. This is done because the parallel gateways do not imply a choice during execution so where having saved the information on parallel activities, an action done in the previous step, these are no longer needed.

- **Step 5:** with this step, the exclusive gateways that merge the previous flows are eliminated. The algorithm selects each fragment preceding the gateway and concatenates it individually with the next one.

The result of this algorithm is a list of dataframe. Each dataframe represents a fragment and each row represents an activity. This data is written to a CSV file that will be read by the Procedural Awareness module.

Algorithm 1 BPMN to json conversion algorithm

```
1: INPUT file BPMN
2: INPUT file JSON
3: for all arrow in graph do ▷ Step 1
4:   if arrow start from a gateway then
5:     append gateway to gateway_list
6:   end if
7: end for
8: for all gateway in gateway_list do ▷ Step 2
9:   if next element is end_Event then
10:    stop fragment
11:    append fragment to fragment_list
12:  end if
13:  if next element is a gateway then
14:    stop fragment
15:    append fragment to fragment_list
16:  end if
17:  if next element is not a gateway then
18:    append element to fragment
19:  end if
20: end for
21: for all parallel_gateway in list_parallel_gateway do ▷ Step 3
22:   select parallel_fragment
23:   concatenate parallel_fragment
24: end for
25: for all parallel_gateway in list_parallel_gateway do ▷ Step 4
26:   select previous fragment
27:   select next fragment
28:   concatenate fragments
29: end for
30: for all exclusive_gateway in list_exclusive_gateway do ▷ Step 5
31:   if exclusive_gateway merge fragment then
32:     individually concatenate fragment before gateway with the one after it (in different fragments)
33:   end if
34: end for
35: OUTPUT file DataFrame
```

Let us clarify all the steps just described with an example. Usually within each task there is a description of the task, which also acts as an indentifying name. In this case, to simplify the explanation, we have numbered the tasks from 1 to 15.

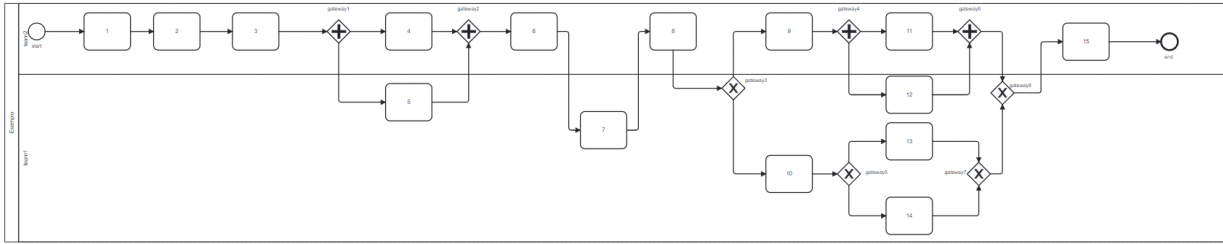


Figure 5.1: Example of BPMN graph.

Let us therefore go through the five steps explained above, indicating for each the type of data created:

1. List created with the elements from which the provisional fragments originate: [start, gateway1, gateway2, gateway3, gateway4, gateway5, gateway8];
2. Provisional fragments created from the elements listed in step 1: [[start, 1, 2, 3], [4], [5], [6, 7, 8], [9], [10], [11], [12], [13], [14], [15,end]];
3. Fragments corresponding to parallel branches are put in series: [[start, 1, 2, 3], [4, 5], [6, 7, 8], [9], [10], [11, 12], [13], [14], [15, end]];
4. Concatenate fragments divided by parallel gateways: [[start, 1, 2, 3, 4, 5, 6, 7, 8], [9, 11, 12], [10], [13], [14], [15, end]]
5. Finally, concatenates the preceding fragments to the exclusive gateway that merges the streams with the next one: [[start, 1, 2, 3, 4, 5, 6, 7, 8], [9, 11, 12, 15, end] [10], [13, 15, end], [14, 15, end]].

As we can see from the last fragment list in Step 5, the first fragment starts at start and stops at the first exclusive gateway found. At this point there are several possible paths to the end event. The fragments in Step 5 list all possibilities. We will see later how our system understands the right path to take and thus which is the right fragment to chain to the first one.

5.0.2 Activities to structured data converter

As mentioned above, this module receives as input events representing the individual activities carried out by the teams. In order for our application to be independent of the system with which it interacts, we decided to use a nats server for receiving events. Nats is a messaging system used in distributed systems to enable communication between services. In our case, the converter connects to the server by subscribing to the topic passed to it by the user as input. It then waits for the system to send messages representing events to that topic. An example of the necessary description of the event is shown below.

```
["offence detection", "2023-05-25T11:05:00.000+01:00", "C-SOC", 10]
```

For each event received, the converter writes the data in dataframe format and adds it to the XES file that it saves on the file system.

```
<trace>
  <string key="concept:name" value="exercise1" />
  <event>
    <string key="concept:name" value="start" />
    <date key="time:timestamp" value="2023-05-25T11:00:00+01:00" />
    <string key="team" value="nan" />
    <string key="case_id" value="exercise" />
    <float key="threshold" value="nan" />
  </event>
  <event>
    <string key="concept:name" value="machine removal from the network" />
    <date key="time:timestamp" value="2023-05-25T11:05:00+01:00" />
    <string key="team" value="C-SOC" />
    <string key="case_id" value="exercise" />
    <float key="threshold" value="10.0" />
  </event>
</trace>
```

The code above shows the trace in the XES file that is generated after the event is received. The start event is added automatically by extracting the start time of the exercise from the json file describing the model metadata. The second event contains all the information received in the message shown above. So what happens is that for each event received, this file representing the procedure performed by the users is updated. Every interval time the central module of our application will read this updated file to track and evaluate the progress of the teams.

5.0.3 Procedural Awareness

We begin the presentation by listing the inputs of this module, not mentioning the BPMN model and model metadata already described over and over again:

- **Structured data:** this is the output of the first converter. The Procedural Awareness module reads this data by loading a CSV file. This file contains the list of dataframes representing the fragments into which the BPMN graph is divided. This list will be used by the module to get the necessary information to build in real time the model of the procedure to be followed;
- **System status:** as we will explain later in our example, these are messages posted on a topic exposed by a nats server. These messages are the information necessary for the module to recognise the right path to take in the procedure according to the conditions in the exclusive gateways;
- **Structured team' procedure:** this is the output of the second converter. It represents the ordered list of activities that the teams performed to follow the procedure. This data is saved in an XES file, read by the module we are talking about and parsed to obtain a dataframe.

This module as we shall later see from the pseudocode is divided into two parallel tasks:

- **Update the model:** is responsible for the construction of the model procedure in real time. By receiving events relating to the state of the system, it is able to figure

out the right path to follow in the procedure. Once the path has been found, it selects the fragment that describes it and concatenate it to the model built up to that point. This mechanism is made possible by the use of a nats server. Whenever a choice needs to be made in the graph, the task we are talking about subscribes to a specific topic on the server. When the system state relating to the choice changes, a message is written in that topic, which the programme will use to make the choice. This task ends when the end event is reached, which means that the end of the graph has been reached and the model is complete;

- **Tracking and Evaluation task:** is the one responsible for calculating tracking and evaluation. Every time interval, defined by the user through the model metadata file, starts the task that reads the output of the second converter, the procedure description of the teams, calculates the tracking and evaluation results and through a websocket sends them to the dashboard.

The second task described is the main element of our application. It uses conformance checking to complete its task of tracking and evaluating processes. For this task, we decided to use the open-source Python library called PM4PY[Ber19]. This library provides us with all the necessary functionality to perform conformance checking between a procedure, used as a model, and the procedure we want to evaluate, which in our case is the one describing the activities carried out by the teams. We present the main functions, which are part of the library, used and their purpose:

- **discover_declare:** this function, given as input an event log or a dataframe, creates a declarative model describing the procedure defined by this data. The model is expressed in the form of a dictionary and consists of constraints that describe the behaviour of the procedure. In our case, these constraints describe the activities that are to be performed and the order in which they are to be executed;
- **discover_temporal_profile:** this function given as input an event log or dataframe create a declarative model that lists all the deadlines that must be met in carrying out the activities that are part of the procedure. The model is expressed in the form

of a dictionary and lists a series of constraints that define the maximum time that must elapse between one activity and all others.

- **conformance_declare**: with this function we perform conformance checking between the model, created with the `discover_declare` function, and the data representing the procedure performed by the teams. This function as output provides the list of constraints that are not met by the procedure and an evaluation regarding the Fitness metric. To perform the conformance checking we used only the constraints that are not respected thus being able to determine the evaluation method;
- **conformance_temporal_profile**: with this function the time model is compared with the timestamps of the procedure activities. In this way we can verify that the activities are completed on time. By having all the maximum time intervals that must elapse between activities, we can calculate whether the activities are late not only with respect to the start event but with respect to any activity in the procedure.

We now use pseudocode 2 to describe the execution flow of the module under consideration. First, the CSV file corresponding to the list of fragments into which the BPMN graph was encoded by the first converter is loaded. The five phases into which the code is divided are described below:

- **Step 1**: starts the two asynchronous tasks and waits for them to finish;
- **Step 2**: function responsible for building the model in real time. If the model is empty, it inserts the fragment comprising the start event as the first one. Otherwise, for each exclusive gateway that splits the flow it subscribes to a topic on the nats server waiting for a message indicating which is the right branch to take. Once the right path is found, the corresponding fragment is selected and concatenated to the model created up to that point. This process continues until the end event is reached;
- **Step 3**: it creates the websocket server which is used to exchange information with the dashboard. For each connection received, it can handle more than one connection, call the function described in the next step;

- **Step 4:** call every interval time the function responsible for tracking and evaluating the procedure. This interval is a user-defined value and represents the time between one team progress evaluation and the next;
- **Step 5:** first, it loads the file describing the procedure performed by the teams up to that point. It does conformance checking between this procedure and the model. The model is not considered in its entirety, but is cut off at the last activity performed by the teams. If we considered it in its entirety all the activities not yet performed by the users because they have not yet been taken into account would be considered as not performed and thus unfairly deducted points. Using the same reasoning, the time model is calculated. At this point the lists corresponding to missing and late activities are created by extracting these data from the conformance checking result. Then still using this data but in this case reworking it, the activities performed by the wrong teams, those performed correctly and those with incorrect add-ons are calculated. Finally the results obtained are sent to the dashboard.

The output of this module is a dictionary that has as keys the metrics described above and as values the activities that meet the metrics. In addition, there are the real-time scores of the teams. This output is created at each interval time defined by the user. In order to share this data, this module uses the websocket server created at Step 3. The dashboard connects as a client to this server. Whenever new evaluation and tracking results are ready, thanks to this channel, they are sent to the dashboard.

Algorithm 2 Procedural Awareness module

```
1: INPUT file CSV

2: MAIN ▷ Step 1
3:   start task to update the model
4:   start task that creates websocket server
5:   await the end of the tasks
6: END MAIN

7: UPDATE MODEL ▷ Step 2
8: if the model is empty then
9:   append to the model the fragment with start event
10: else
11:   for exclusive divergent gateway in list exclusive divergent gateway do
12:     connect to NATS server
13:     await information about the branch to be taken
14:     append to the model the chosen fragment
15:   end for
16: end if
17: END UPDATE MODEL

18: START SERVER ▷ Step 3
19: create server on localhost port 8765
20: for every connection start loop function
21: END START SERVER

22: LOOP ▷ Step 4
23: call function that does tracking and evaluation
24: sleep interval time
25: END LOOP

26: TRACKING AND EVALUATION CALCULATION ▷ Step 5
27: load dataframe of team operations to be tracked and evaluated
28: performs conformance checking between model and team operations with PM4PY
29: calculate temporal profile with PM4PY
30: extracts missing activities from result of conformance checking
31: extracts messy activities from result of conformance checking
32: calculates activities carried out by wrong team
33: calculates correctly performed activities
34: calculate add-ons
35: send via websocket the results to the dashboard
36: END TRACKING AND EVALUATION CALCULATION
37: OUTPUT evaluation of team performance
```

5.0.4 Dashboard

To create the dashboard we decided to use the Express framework for Node.js [Fou23]. This choice is due to the flexibility and ease of use of this framework. Given the need to show a BPMN graph on the dashboard, it was necessary to install the bpmn-js library. This library thanks to the bpmn-modeler module allowed us to display and modify the chart in real time. To receive the information from the Procedural Awareness module we created a websocket that connects to the websocket server created by the module. By doing so every time the Procedural Awareness module has new results available it sends them in the form of a json string to the dashboard. The latter for each message received updates the displayed page by rewriting the teams' scores and updating the colours of the graph following the new results.

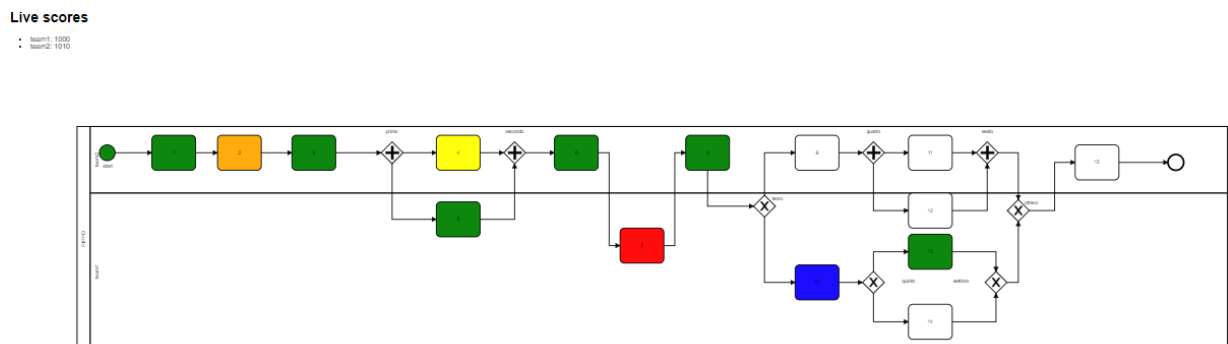


Figure 5.2: Example of a Dashboard.

We use the Figure 5.2 to explain how to interpret the information available through the dashboard. At the top left is the list of teams with their real-time scores. In the middle is the BPMN graph used as a template for the procedure. Below we list the meaning of each colour:

- **Green:** activities carried out correctly by the right team within the specified time-frame;
- **Yellow:** activities carried out late;

- **Red:** activities not carried out;
- **Orange:** activities performed not in the correct order;
- **Blue:** activities carried out by the wrong team;
- **White:** activities have not yet been executed but are not overdue.

We use activity 15 to explain the concept of tracking. Looking at the graph, we understand that the teams have reached activity [13]. Since the activity [15] is white, we know that it has not yet been carried out, but that it is not late. These two pieces of information allow us to understand where the teams are in the procedure.

In contrast to [15] the activities [9,11,12,14] are in branches of the graph that are not to be executed. We can glean this information from the fact that the alternative activities to the latter listed are coloured green, indicating that they have been correctly executed.

Chapter 6

Experimental Evaluation

6.1 Case Study Overview

This chapter presents the case study we decided to use to test our application. It is a security challenge organised by Leonardo SpA for external companies, which for privacy reasons we shall call Authority One and Authority Two. For the execution of this exercise, a scenario within a cyber range was created by Leonardo SpA. This scenario has the structure shown in 6.1.

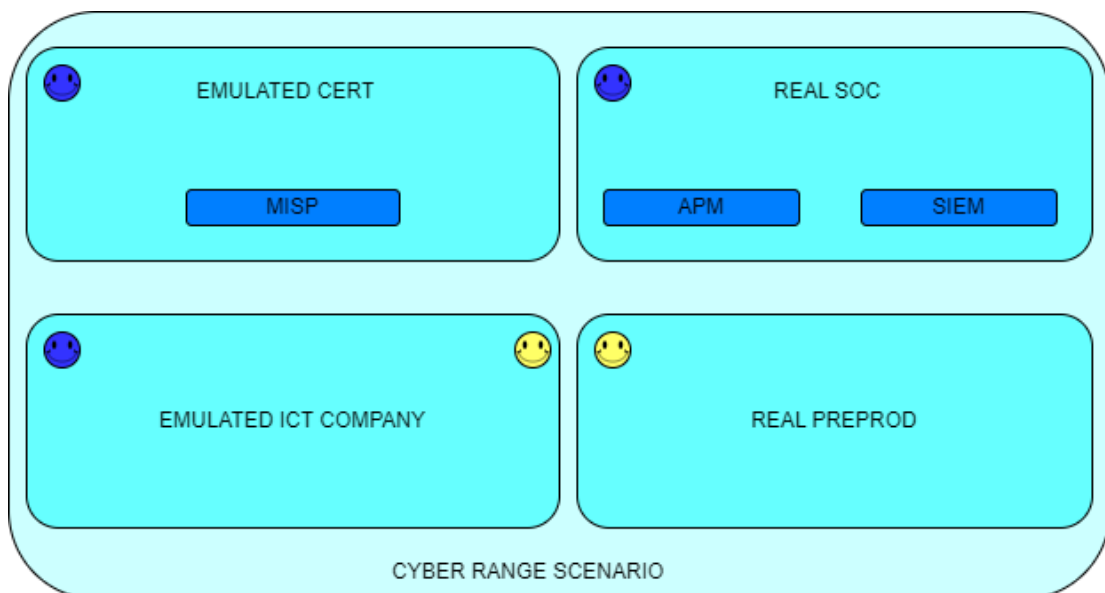


Figure 6.1: Cybersecurity exercise scenario.

The objective of the exercise is to simulate the entire operational cyber threat management scenario to verify and analyse the prevention and response capabilities of the actors involved. The simulation consists of five phases:

1. the emulated Computer Emergency Response Team (CERT) receives security bulletins, which are official communications that contains the procedures for handling known incidents and other information like known vulnerabilities, warnings about new threats or information about security updates. In this phase, participants must be able to extract from the bulletins the Indicators of Compromise (IoCs) that will be uploaded to the CERT's Malware Information Sharing Platform (MISP);
2. the Cyber-Security Operation Center (C-SOC) receives the IoCs from the CERT. These IoCs are stored in the MISP node of the C-SOC and are used for the definition of the onboarding rules in the Security Information and Event Management (SIEM). The created rules will be used for anomaly detection in the monitored systems;
3. simulated automatic attacks are launched towards the ICT company and preprod by the red team;
4. thanks to the onboarding rules, the C-SOC detects attacks and carries out the incident management procedure. At the end of the procedure, all attack data is collected and sent to the CERT;
5. the CERT shares information about the attack with other CERTs or other institutions such as National Cybersecurity Agency (ACN).

The 6.2 shows the steps just described.

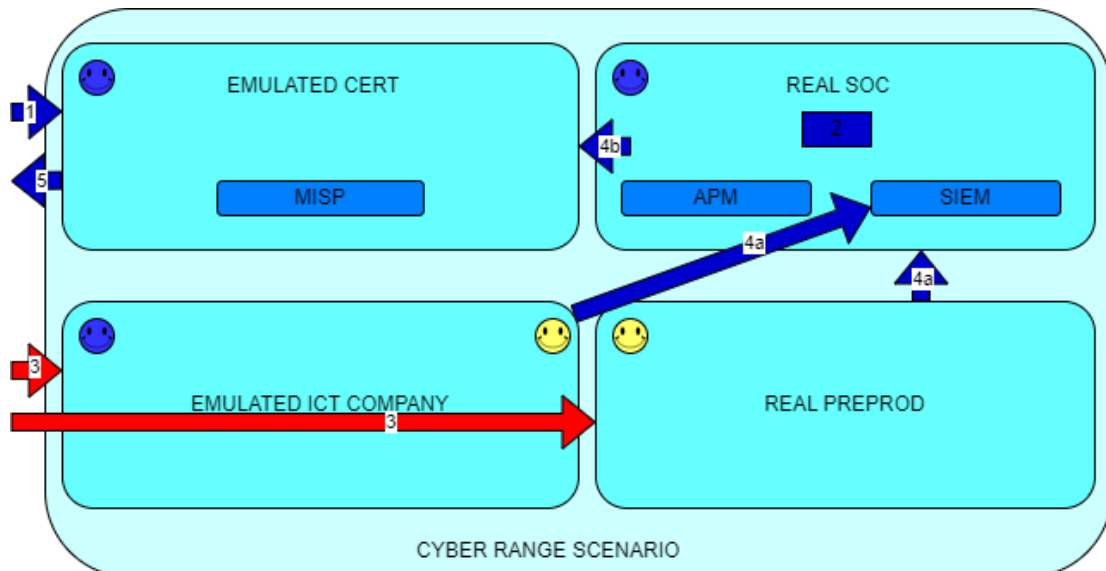


Figure 6.2: Cybersecurity exercise scenario with representation of planned actions.

Four different teams participated in the exercise, which we present in the list below:

- **Red:** the organisers of the exercise are part of it. They know the planned attacks and also the actions to be taken to mitigate the effects of the attacks.
- **Blue:** are the people for whom the exercise was created. They are not aware of the planned attacks and their ability to perform their assigned tasks is assessed during the exercise.
- **Yellow:** consists of people simulating ordinary operational activities. They serve to create noise during the exercise.
- **Green:** is composed of technical support personnel needed in the event of problems not foreseen in the exercise scenario.

In the figure, the arrows indicating the activities to be performed are coloured according to the team that has to perform them. There are also smilies indicating the area of competence of certain teams.

6.2 Case study implementation

To test our work, we took the data from the challenge performed by Leonardo SpA and corrected the attack management part with our application. We selected two blue teams, which we will call Blue One and Blue Two, assigned to the task of handling the incident. The two teams were divided into four subgroups each in charge of a different part of the simulated environment. We took the logs of their actions in response to the attack, reconstructed step by step the procedure they performed, and thanks to the graph in Figure 6.3 used as a model, we traced and evaluated their actions.

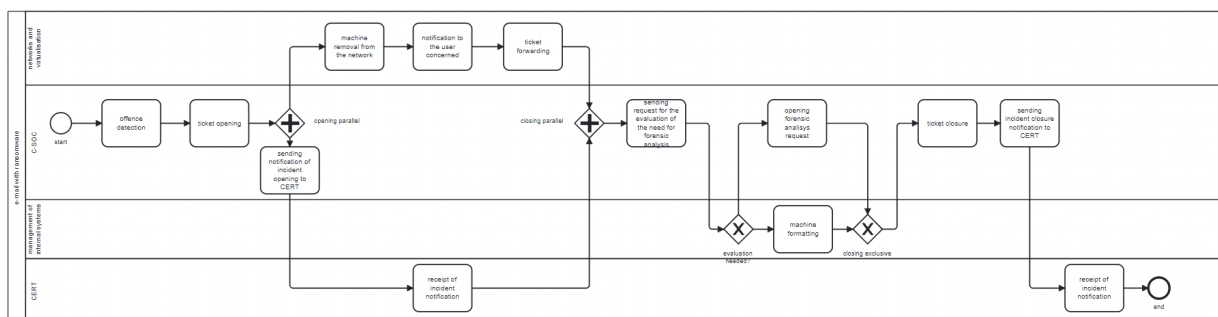


Figure 6.3: Model of the response procedure to a ransomware attack.

The attack executed during the exercise is a ransomware one. In this type of attack, malware is grafted onto the system to be attacked. This malware is able to encrypt certain data in the system or block access to a network or device. The attacker then demands a ransom to restore the system to its original state. In our case, the malware was sent as an e-mail attachment. Downloading and opening the attachment allowed the malware to infect the system.

6.2.1 BPMN to structured data converter

The choice of taking a BPMN format graphic as input is due to the fact that it lends itself to the representation of cybersecurity and other procedures. In fact, in the example case, the procedures contained in the security bulletins are in this format, such as the one

shown in the Figure 6.3. The second input of this converter is the json file containing the model's metadata. Using the instructions in the exercise, we compiled it with the necessary information.

The output of this module is the division of the graph used as a model into fragments. We list all the steps performed by our algorithm to arrive at the final output:

1. list of the elements from which the provisional fragments originate:[start,opening parallel, closing parallel, evaluation needed?, closing exclusive];
2. provisional fragments created from the elements listed in step 1:[[offence detection, ticket opening], [machine removal from the network, notification to the user concerned, ticket forwarding],[sending notification of incident opening to CERT, receipt of incident notification],[sending request for the evaluation of the need for forensic analysis],[opening forensic analysis request],[machine formatting],[ticket closure, sending incident closure notification to CERT, receipt of incident notification]];
3. fragments corresponding to parallel branches are put in series: [[offence detection, ticket opening], [machine removal from the network, notification to the user concerned, ticket forwarding, sending notification of incident opening to CERT, receipt of incident notification], [sending request for the evaluation of the need for forensic analysis],[opening forensic analysis request],[machine formatting],[ticket closure, sending incident closure notification to CERT, receipt of incident notification]];
4. concatenate fragments divided by parallel gateways: [[offence detection, ticket opening, machine removal from the network, notification to the user concerned, ticket forwarding, sending notification of incident opening to CERT, receipt of incident notification, sending request for the evaluation of the need for forensic analysis],[opening forensic analysis request],[machine formatting],[ticket closure, sending incident closure notification to CERT, receipt of incident notification]];
5. Finally, concatenates the preceding fragments to the exclusive gateway that merges the streams with the next one: [[offence detection, ticket opening, machine removal

from the network, notification to the user concerned, ticket forwarding, sending notification of incident opening to CERT, receipt of incident notification, sending request for the evaluation of the need for forensic analysis], [opening forensic analysis request, ticket closure, sending incident closure notification to CERT, receipt of incident notification], [machine formatting, ticket closure, sending incident closure notification to CERT, receipt of incident notification]].

We can see that step 5, which shows the final fragments that constitute the output of the converter, contains 3 lists. The first contains all the activities until the exclusive gateway is reached. The others show the two possible paths to the final event.

6.2.2 Activities to structured data converter

To test this component since we did not have the cyber range used in the exercise but only the logs describing user activities, we created a script that would simulate the system to be monitored. For each log describing an activity of the procedure performed by the user, the script sent a message to the NATS server. An example of a message is as follows:

```
["machine removal from the network", "2023-05-25T11:20:00.000+01:00", "C-SOC"]
```

For each message received, the converter extracts the information contained and uses it to write the XES file describing the procedure carried out by the teams.

6.2.3 Procedural Awareness

This module extracts from model metadata the information that it needs. In this case, the starting score is 1000 points and the time interval is 5 minutes. For the sake of simplicity each mistake is penalised with 10 points. The same value is added for each task successfully performed in the right order with the set time. So every 5 minutes this module reads the XES file describing the procedure carried out by the teams and tracks and evaluates it. The table below shows the scores of the two teams after 30, 60, 90 and 120 minutes.

networks and virtualisation	1000
C-SOC	1030
management of internal systems	1000
CERT	1010

Table 6.1: Blue One after 30 minutes

networks and virtualisation	1010
C-SOC	1050
management of internal systems	1000
CERT	1010

Table 6.3: Blue One after 60 minutes

networks and virtualisation	1010
C-SOC	1060
management of internal systems	1000
CERT	1010

Table 6.5: Blue One after 90 minutes

networks and virtualisation	1010
C-SOC	1050
management of internal systems	1000
CERT	1020

Table 6.7: Blue One after 120 minutes

networks and virtualisation	1010
C-SOC	970
management of internal systems	1000
CERT	1010

Table 6.2: Blue Two after 30 minutes

networks and virtualisation	1010
C-SOC	960
management of internal systems	1000
CERT	1010

Table 6.4: Blue Two after 60 minutes

networks and virtualisation	1010
C-SOC	950
management of internal systems	1000
CERT	1010

Table 6.6: Blue Two after 90 minutes

networks and virtualisation	1010
C-SOC	950
management of internal systems	1000
CERT	1020

Table 6.8: Blue Two after 120 minutes

6.2.4 Dashboard

At this point, we can use the dashboard to show the results of the real-time tracking of the teams' progress.

Figures 6.4, 6.5, 6.6 and 6.7 refer to the procedure carried out by Team Blue One.

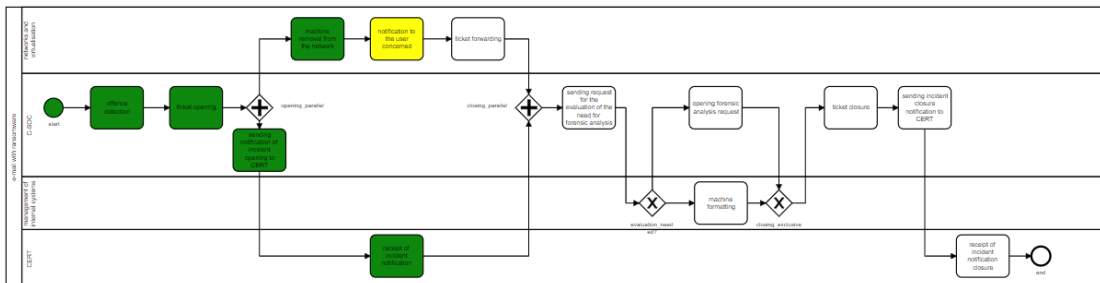


Figure 6.4: Progress status after 30 minutes by the Blue One team.

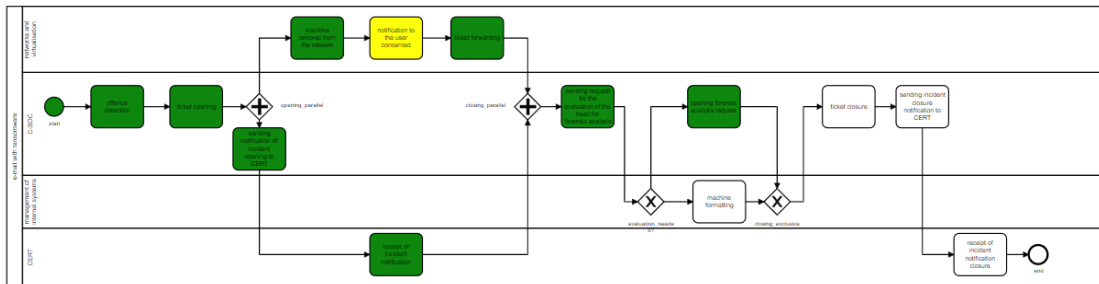


Figure 6.5: Progress status after 60 minutes by the Blue One team.

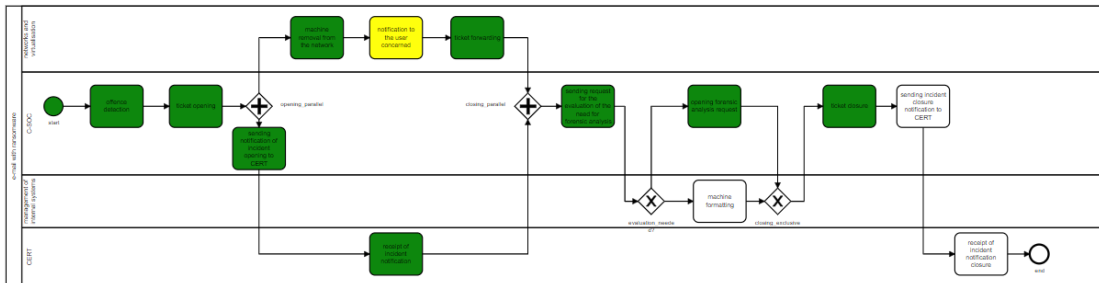


Figure 6.6: Progress status after 90 minutes by the Blue One team.

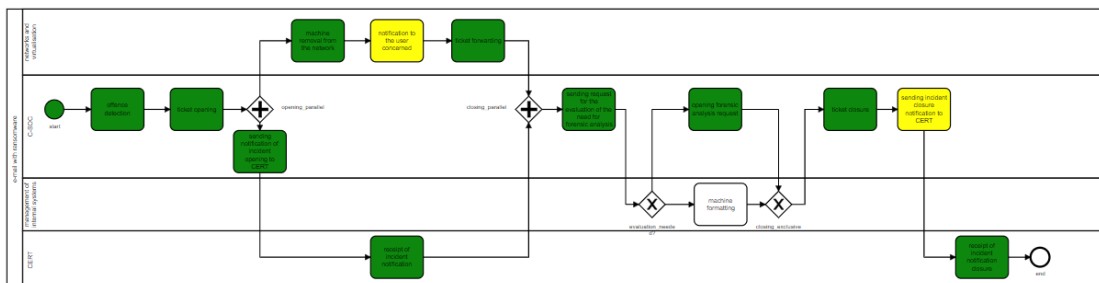


Figure 6.7: Progress status after 120 minutes by the Blue One team.

Instead, Figures 6.8,6.9,6.10 and 6.11 refer to the procedure carried out by Team Blue Two.

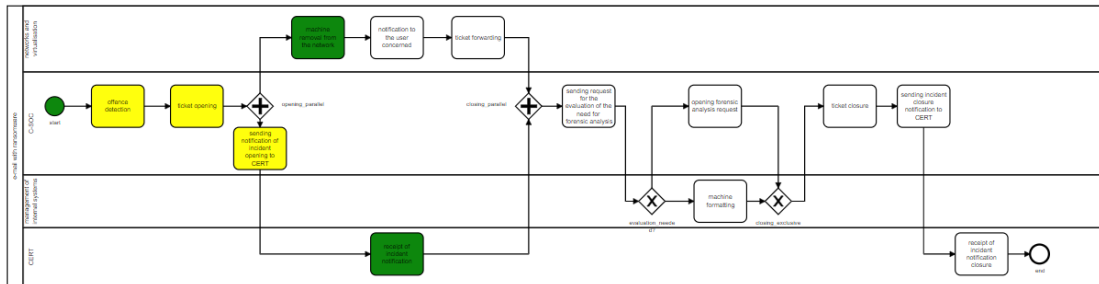


Figure 6.8: Progress status after 30 minutes by the Blue One team.

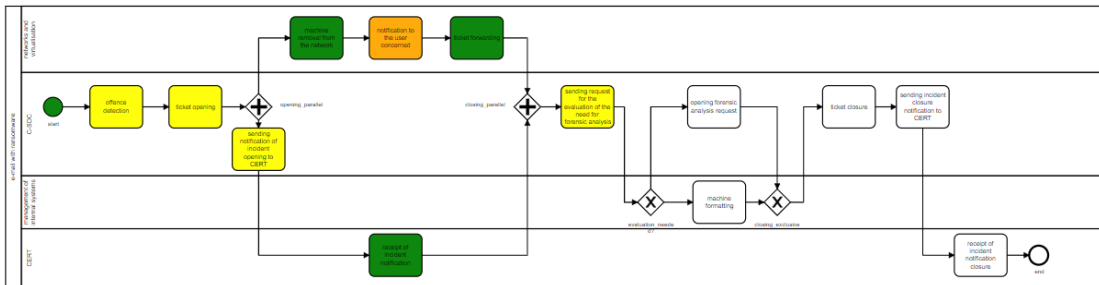


Figure 6.9: Progress status after 60 minutes by the Blue One team.

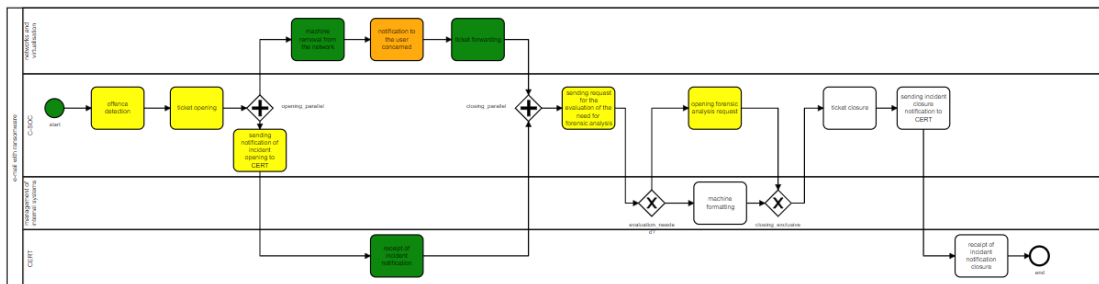


Figure 6.10: Progress status after 90 minutes by the Blue One team.

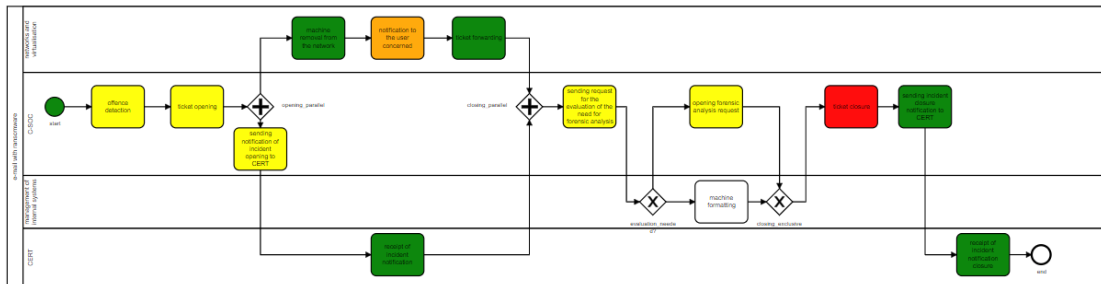


Figure 6.11: Progress status after 120 minutes by the Blue One team.

6.3 Discussion

The purpose of implementing the case study is to check whether the objectives set at the beginning of the thesis have been met. The first challenge was to take as input a graphical representation of the procedure that our system uses as a model. This was successfully done during the case study. However, we modified its representation so that its structure corresponded to the one we defined in our architecture. One consequence of the definition 4.0.1 is that every time the procedure execution flow is split by a gateway, there must be a corresponding gateway to bring the flows together. The original graph is shown in the Figure 6.12. It was necessary to add the second parallel gateway in order to use the BPMN graph.

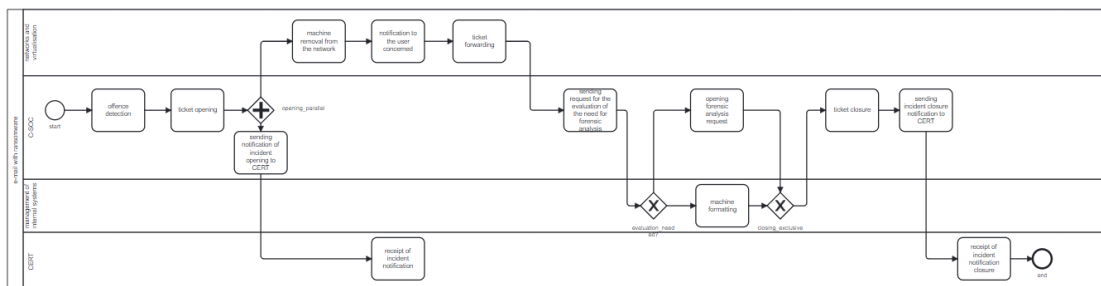


Figure 6.12: Original BPMN graph used during the exercise.

The function test of the second converter was carried out using a script simulating the cyber range used during the original exercise. This was due to the fact that the cyber range used for the exercise was not available. However, this fact proved the interoperability of our

application. The real-time construction of the procedure carried out by the users was performed correctly.

After processing the input, we tested the functioning of the module responsible for tracking and evaluating the procedure. As we can see from the scores in 6.2.3, the application was able to distinguish all four groups into which the two teams, Blue One and Blue Two, are divided and assign the corresponding score to each of them. From the images in 6.2.4 we can see how this module was able to track the teams' progress.

One of the main objectives of this thesis is to simplify and speed up the correction of security challenge. Only a few tens of minutes were needed to carry out the due processing of the inputs. Everything else as planned was performed by our application, thus having the results available in real time. The execution of this case study showed that all the objectives set at the beginning of the thesis were met.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

During the presentation of our thesis, we talked about the design and development of an application that can track and evaluate the ability of a cyber incident response team to follow a predefined procedure. This thesis is necessitated by the ever-increasing need to educate and train technical personnel to test and reproduce complex procedure on computer systems. This software aims to extend security exercises and provide a method to simplify their correction. As it is fully automatic and in real time, it allows the organisers of the exercises to reduce the number of personnel responsible for the evaluation. Another special feature is that it is independent of the monitored system because a NATS server is used for the communication of team activities. In this way, any system capable of sending messages to a NATS server can be used as a scenario for the exercise, be it simulated or real. During the case study, we reproduced a security exercise organised by Leonardo SpA. This exercise confirmed what was described earlier. The preparation of the necessary inputs takes a few tens of minutes, depending on the familiarity with the type of data. As explained in detail, the correction step is automatic and does not require any action on the part of the user of our application. The dashboard proved to be clear and effective.

7.2 Future Works

The implemented case study gave rise to some ideas for future work. An advantage of our architecture is that each module is autonomous, the only elements that bind them are the input and output formats. This allows future work to be designed around a single module. The results calculated by the main module of our application at this time are only used for display on the dashboard. Given the precision with which the results are output, a future development could be to use them for the creation of evaluation reports. This document could contain the errors made by a team and associate corrections with them. Another idea could be to enhance the error handler. Since conformance checking is based on string comparison, a typo can cause errors in the tracking or evaluation of the procedure. The last future work we propose concerns the first converter described. In our implementation we have limited the number of elements that can be used to represent the procedure in the BPMN. This is because the elements we have chosen seem sufficient to clearly describe the procedure. If this set of elements is considered to be limited, the converter responsible for this task can be updated.

Bibliography

- [Whi04] Stephen A White. “Introduction to BPMN”. In: *Ibm Cooperation* 2.0 (2004), p. 0.
- [CT12] Michele Chinosi and Alberto Trombetta. “BPMN: An introduction to the standard”. In: *Computer Standards & Interfaces* 34.1 (2012), pp. 124–134.
- [Van12] Wil Van Der Aalst. “Process mining: Overview and opportunities”. In: *ACM Transactions on Management Information Systems (TMIS)* 3.2 (2012), pp. 1–17.
- [MR13] Michael zur Muehlen and Jan Recker. “How much language is enough? Theoretical and practical use of the business process modeling notation”. In: *Seminal Contributions to Information Systems Engineering: 25 Years of CAiSE* (2013), pp. 429–443.
- [Cam14] Camunda. *bpmn.io*. 2014. URL: <https://bpmn.io/>.
- [Gov18] Queensland Government. *Incident Management Guidelines*. Settembre 2018. 2018. URL: <https://www.forgov.qld.gov.au/information-and-communication-technology/qgea-policies-standards-and-guidelines/incident-management-guideline>.
- [Ber19] Alessandro Berti. *PM4Py API reference*. 2019. URL: <https://processintelligence.solutions/static/api/2.7.11/api.html>.
- [Bie19] B de Bie. “Visual conformance checking using BPMN”. PhD thesis. Master’s thesis, Eindhoven University of Technology, 2019. 1, 12, 13, 15, 2019.
- [And+20] Mauro Andreolini et al. “A framework for the evaluation of trainee performance in cyber range exercises”. In: *Mobile Networks and Applications* 25 (2020), pp. 236–247.
- [Nic22] Maciej Barelkowski Nico Rehwaldt. *bpmn-js library*. 2022. URL: <https://github.com/bpmn-io/bpmn-js>.
- [ACN23] ACN. *LINEE GUIDA PER LA REALIZZAZIONE DI CSIRT*. 2023. URL: https://www.acn.gov.it/portale/documents/d/guest/acn_linee_guida_csirt.

- [Fou23] OpenJS Foundation. *Express documentation*. 2023. URL: <https://expressjs.com/en/5x/api.html>.
- [BBS24] Federica Bianchi, Enrico Bassetti, and Angelo Spognardi. “Scalable and automated Evaluation of Blue Team cyber posture in Cyber Ranges”. In: *Proceedings of the 39th ACM/SIGAPP Symposium on Applied Computing*. 2024, pp. 1539–1541.
- [Clu24] Clusit. *Rapporto clusit 2024*. 2024. URL: <https://clusit.it/rapporto-clusit/>.
- [Cor24] Francesco Corona. *Cert e Csirt questi sconosciuti, tutti i passaggi della cybersecurity italiana*. 2024. URL: <https://www.agendadigitale.eu/cultura-digitale/il-futuro-del-lavoro-con-lia-cosa-ci-dice-linkedin/>.

Listing of acronyms

CSIRT Computer Security Incident Response Team

NIS Network and Information Security

ACN National Cybersecurity Agency

CERT Computer Emergency Response Team

SIEM Security Information and Event Management

BPMN Business Process Model and Notation

BPMI Business Process Management Initiative

CSV Comma-Separated Values

XES eXtensible Event Stream

IoC Indicators of Compromise

MISP Malware Information Sharing Platform

C-SOC Cyber-Security Operation Center

ICT Information and Communication Technology