Empir Software Eng manuscript No.
(will be inserted by the editor)

A Systematic Approach for Performance Assessment Using Process Mining

Industrial Experience Report

Simona Bernardi · Juan L. Domínguez · Abel Gómez · Christophe Joubert · José Merseguer · Diego Perez-Palacin · José I. Requeno · Alberto Romeu

Received: date / Accepted: date

Abstract In this work, we report our experience in the application of a methodology, based on process mining techniques, for the performance assessment of data-intensive software applications. The methodology is an original contribution of this work, while the system has been developed by PRODEVELOP and it is a customizable integrated port operations management system. The company is a medium-sized enterprise, located in Valencia (Spain), with high expertise in advanced geospatial technologies. The performance assessment has been carried out by a team composed by PRODEVELOP's engineers and researchers from the University of Zaragoza. The team worked within the DICE (Developing Data-Intensive Cloud Applications with Iterative Quality Enhancements) H2020 European project.

Keywords Performance · Process mining · Unified Modeling Language · Complex Event Processing · Stochastic Petri net

Simona Bernardi
Centro Universitario de la Defensa, AGM, Zaragoza (Spain)
E-mail: simonab@unizar.es

Juan Lucas Domínguez, Christophe Joubert, and Alberto Romeu
Prodevelop SL, Valencia (Spain)
E-mail: {jldominguez,cjoubert,aromeu}@prodevelop.es

Abel Gómez
Internet Interdisciplinary Institute, Universitat Oberta de Catalunya (Spain)
E-mail: agomezlla@uoc.edu

José Merseguer, Diego Perez-Palacin, and José Ignacio Requeno
Dpto. de Informática e Ingeniería de Sistemas, Universidad de Zaragoza (Spain)
E-mail: {jmerse,diegop,nrequeno}@unizar.es

1 Introduction

This paper is an industrial experience report, since it summarizes results regarding the application of a performance assessment methodology to a real software system, named Posidonia Operations [22]. The system, developed by Prodevelop, is a customizable Integrated Port Operations Management System. Prodevelop [27] is a SME (Small and Medium-sized Enterprise) that employs 80 engineers in Computer Science, Telecommunications, Cartography and Geodetics. Prodevelop is specialized in software development and advanced geospatial technologies.

Posidonia Operations [22] is a commercial product already deployed and operated by many different *port authorities* across Europe, Africa and Asia. It allows port authorities to optimize operational maritime activities related to the vessels flow within the service area of the port, integrating all the involved stakeholders and all the relevant information systems. Posidonia Operations is a data-intensive application implemented in Java. It processes streamed data from Automatic Identification System (AIS) receivers [5, 6], a system that gets vessels position and meta-data in real time. An AIS message is a binary encoded sentence that can be decoded into key-value objects. A Complex event processing (CEP) engine correlates the subscribed AIS messages in time and space by means of a set of geospatial rules to identify events produced by the vessels in the port.

Prodevelop is planning new deployments of Posidonia Operations for the upcoming years. However, several performance challenges have arisen that may jeopardise the software product. Consequently, the risk of producing a version of the product that fails on scalability and on the throughput of some services is real. The challenges have been collected in [15]. This paper reports the most relevant ones, which constitute our first objective, **O1**, *the performance assessment of* Posidonia Operations. This objective, according to [30], is our mean to carry out an *internal validation* of our research. In the following list, the objective is detailed according to the different stakeholders engaged within the software life-cycle:

O1.1 Software architects are concerned about the *scalability* of the product. Being a product already in production, performance has to be guaranteed under different velocity and volume of data to be processed (requirement *PO.2* in [15]).

O1.2 Software developers are interested in identifying possible *bottlenecks* in the processing of the CEP rules as well as in the AIS data parsing implementation (requirement *PO.5*).

O1.3 Quality engineers need to predict the impact of a business rule on the CEP *performance* in order to evaluate different alternative requirements (requirement *PO.1*).

The DICE project [13] is an asset for Prodevelop to address the aforementioned objective.

Moreover, researchers of the University of Zaragoza (Unizar), also involved in the project and experts in performance solutions, want to take profit from this experience in order to *offer a methodology for performance assessment easy to be applied by* SME *practitioners*. This constitutes the second objective, **O2**, of this paper, which means an *external validation* of our research according to [30]. The methodology will leverage mature and state-of-the-art techniques and tools. The latter is of great importance since our intention is to offer guidelines, assisted by tools, that can be followed by engineers without the need of researchers experts on performance evaluation.

The rest of the paper is organised as follows. Section 2 outlines the assessment methodology. Section 3 describes Posidonia Operations. Sections 4, 5 and 6 apply the steps of the methodology to Posidonia Operations. Section 7 discusses the benefits and limitations of the approach. Finally, Section 8 draws a conclusion and covers some related work. The paper also includes appendices, that detail input parameters and formal questions about Petri nets.

2 Assessment Methodology

Prodevelop's engineers and researchers of the Unizar made a team for addressing **O1**, i.e., the assessment of Posidonia Operations. The engineers, who had developed most of the current version of the system, contributed also with their knowledge in the problem domain. The researchers were specialists on the construction and evaluation of performance models. From the very beginning, the team considered the Unified Modeling Language (UML) as the lingua franca for communication between engineers and researchers. In fact, most of the system architecture had been already developed using UML. Hence, UML prevented the engineers to learn the performance language, i.e., Petri nets, while the researchers could learn details of the system without navigating code. Another important objective was the use of state of the art tools, that enable to automate the processes as much as possible. In this way, Prodevelop could eventually assess future versions of Posidonia Operations or even another products, not needing the help of performance experts. The result was the systematic approach summarized by the assessment methodology below, which fulfills **O2**, the second objective of this paper.

Methodology for Performance Assessment

Input: UML-MARTE performance scenario ($\mathcal{S}$), data logs ($\mathcal{L}$)
1: Get a normative model $\mathcal{N}$ from $\mathcal{S}$
2: Pre-process $\mathcal{L}$ to get event logs $\mathcal{EL}$
3: **repeat**
4: Filter $\mathcal{EL}$ (complete traces)
5: Check for conformance $\mathcal{N}$ and $\mathcal{EL}$
6: **until** fitness $\geq$ *threshold*
7: Enhance $\mathcal{S}$ with timing & probabilistic perspectives
8: Conduct sensitivity performance analysis: obtain $\mathcal{R}$
Output: Performance results ($\mathcal{R}$)

The **input** of the methodology consists of: 1) a UML-MARTE specification that represents a system performance scenario $\mathcal{S}$ and 2) data logs $\mathcal{L}$ of real execution traces of the system processes. The scenario can be modelled either with a Sequence Diagram (SD) or an Activity Diagram (AD). A Deployment Diagram (DD) can be also included to specify the software component allocation on computing nodes. Figures 1, 2 and 3 show, respectively, the deployment, sequence and activity diagrams of the case study and they will be described, in detail, in the next section. Observe that the UML diagrams are annotated using the MARTE profile [26] to specify performance input parameters – such as mean durations of activities, data stream arrival rates and probabilities of alternative steps execution – and the performance metrics of interest, according to the objectives of the analysis.

In **Step 1**, a formal model $\mathcal{N}$ (i.e., a Petri net model [8]) is automatically obtained from $\mathcal{S}$ using the DICE simulation tool [19]. This tool applies model-to-model (M2M) transformations [10, 23]. $\mathcal{N}$ represents a *normative* model since it is derived from the known behavioural specification of the system. Figures 4 and 5 depict the two Petri net models that are derived, respectively, from the parsing scenario of Figure 2 and the CEP scenario of Figure 3, considering the resource restrictions specified in the DD of Figure 1.

Step 2 consists in pre-processing the data logs $\mathcal{L}$ to convert them into the *event log* XES [37] standard format, where each execution trace is characterized by an ordered set of event occurrences together with their timestamps. The data logs of Posidonia Operations were collected in separate *comma separated values* (CSV) files and include the traces of two days of system execution, with a total of $12,331,320$ traces related to the first process (i.e., the parsing process) and $349,790$ traces related to the second one (i.e., the CEP process).

The aforementioned two steps are applied to the Posidonia Operations case study in Section 4.

In **Steps 3-6**, the event logs and the Petri net models are aligned using ProM [36] – a tool that provides support to a wide range of process mining techniques – in order to reach a fitness threshold (i.e., $threshold = 1$ means that all the traces in the log can be replayed in the model); the alignment may require several iterations and may improve the initial system scenario specification $\mathcal{S}$.

In **Step 7**, once the required fitness has been reached, the performance input parameters (e.g., mean time delays, arrival rates, execution step probabilities) of $\mathcal{S}$ are set to actual values, which are estimated using the event logs and the trace-driven simulator of the ProM tool.

Section 5 applies Steps 3-7.

Finally, **Step 8** produces the **output**. Then, $\mathcal{S}$ is used to conduct sensitivity analysis with the DICE simulation tool and to get performance results $\mathcal{R}$. This step is applied in Section 6.

3 The POSIDONIA Operations Case Study

In Posidonia Operations the vessel becomes the centre of the system, and all the actions and data are linked to the vessels through an integrated operator console that centralizes all the significant information coming from external sources and systems, like AIS, Radar, VTS, meteorology, communications, Port Management Systems, Port Community Systems, safety & security or cartography. The system has been designed to cover all the phases of a vessel: request, authorization, port approach, port enter, berthing and unberthing, berth change, anchoring and port leaving. It also fulfills port operations, including berth planning, coordination and register of pilots, tugs and moorers activities, vessel supplies and bunkering, wastes & disposal, incidents, repairs or port inner traffic.

A real time analysis engine based on spatial information can be configured to automatize relevant operational events like anchoring, berthing/unberthing, pilots and tugs operations, bunkering, enter and exit of areas like port service area, waypoints or inner harbour and port exit with pending requested anchoring. It processes streamed data from Automatic Identification System (AIS) receivers [5, 6], a system that gets vessels position and meta-data in real time. The encoding protocol of an AIS sentence can be found in [7]. To get data from an AIS Network, a TCP connection to the port AIS receiver is used. Once an AIS stream is parsed, it is published to a message queue as AIS messages for further processing: analysis, complex event processing (CEP), data integration, visualization, etc. An AIS message is a binary encoded sentence that can be decoded into key-value objects. Its size is usually under 100 bytes. In particular, the CEP goal is to correlate the subscribed AIS messages in time and space by means of a set of geospatial rules to identify events produced by the vessels in the port.

3.1 Performance goals

In the following, we recall the performance goals established in the Introduction.

O1.1 Being a product already in production, performance has to be guaranteed under different velocity and volume of data to be processed. For a single port, a velocity of about a hundred of AIS messages per second with a volume of about five million messages per day can be observed. These numbers may vary and can be multiplied by the number of ports managed by the product. Moreover, for a given *port authority*, several instances of the CEP engine would be needed, one for each port area. In this case, one of the challenges of the architects is related to the *scalability* of the product in terms of data processing, storage and analysis. In particular, architects are interested in evaluating the *performance* impact of changes in the data stream, in order to refactor the current architecture.

O1.2 Developers are interested in identifying possible *bottlenecks* in the processing of the CEP rules as well as in the AIS data parsing implementation.
O1.3 From the quality engineers point of view, it is important to predict the impact of additional geospatial business rules on the *CEP performance* in order to evaluate different alternative requirements.

According to the aforementioned performance challenges, we focused on the following core components of interest for performance analysis:

- The streaming processor – or AIS parser – that collects the data from the AIS receiver and parses it. An AIS parser consists of four sub-components: the *Parser*, the *Station Manager*, the *Station Processor*, and the *ParsingTask*;
- The message queue for subscribing/publishing data such as AIS messages or detected events; and
- The complex event processing (CEP) engine that subscribes to AIS messages, correlates them to identify events. A CEP component includes three sub-components: the *Subscriber AIS Sentence* (responsible for the AIS messages subscription), the *AIS Sencence Listener* and the *Stateful Knowledge Session* (in charge of AIS messages correlation and events identification).

The basic allocation of these components onto processing nodes is shown in the UML deployment diagram (DD) of Figure 1.

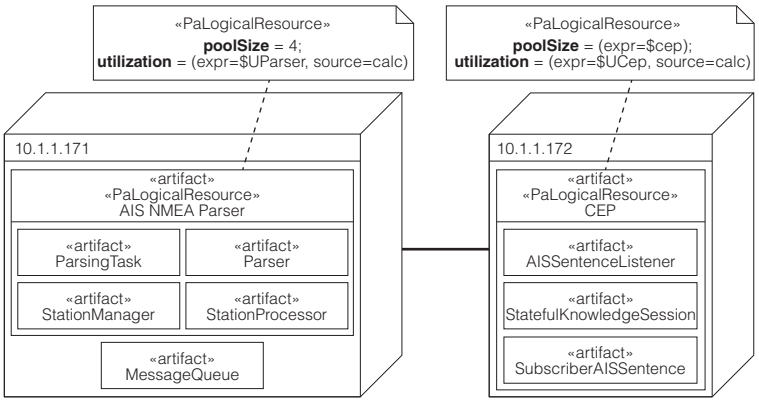


Fig. 1 UML Deployment Diagram.

3.2 System scenarios

The main parsing scenario, carried out by the AIS parser, is modelled by the UML sequence diagram of Figure 2. The data stream (*NMEAstream*) from the *AIS Receiver* is initially parsed by the *Parser* sub-component. Then, the produced AIS NMEA message is sent to the *Station Manager* that forwards it

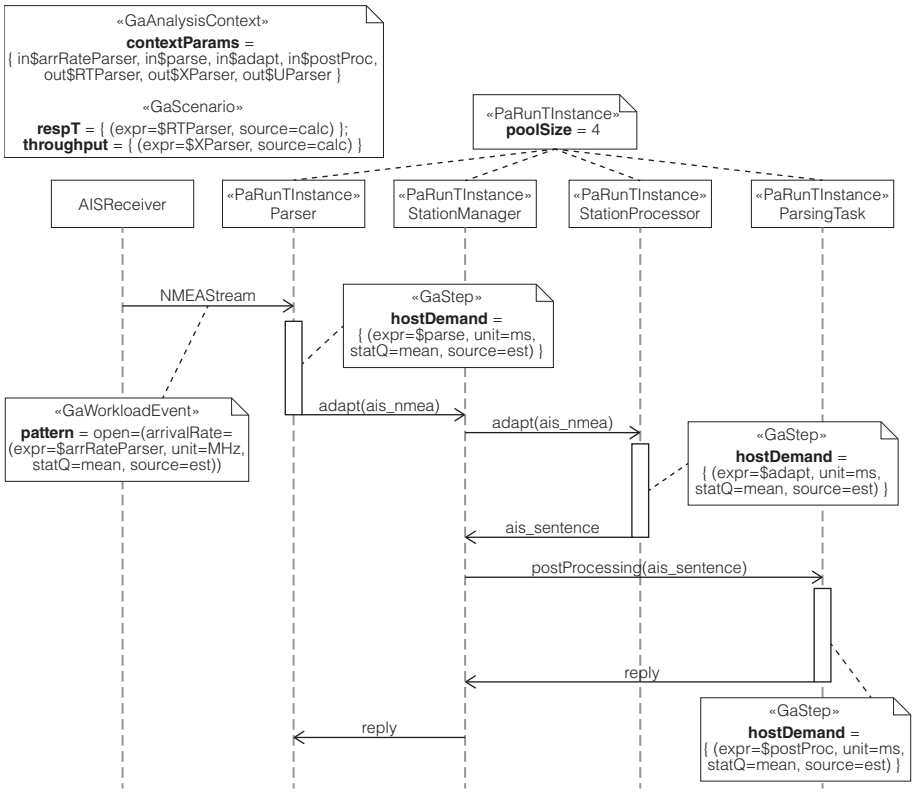


Fig. 2 AIS parser scenario.

to the *Station Processor* to convert it into a business object (*AIS Sentence*). The AIS sentence is successively post-processed by the *Parsing Task* to be published in the message queue, not explicitly represented in the diagram. The adaptation and post-processing steps are controlled by the *Station Manager*.

Figure 3 shows instead the main execution cycle of the CEP. Here, the AIS sentences – generated by the AIS parser – are analyzed in order to detect the events produced by the vessels that are of interest for a port authority (e.g., anchoring, docking, etc.). The *AIS Sentence Listener* sub-component is continuously monitoring the message queue and, when a new AIS sentence is produced, it starts a new execution cycle to handle it (*S_Handle Message*). In particular, the AIS sentence is added to the Knowledge Base (KB) repository and processed by the *Stateful Knowledge Session* sub-component. First, as a consequence of the new AIS sentence, the list of active rules is updated (*S_UpdateActiveFromZ*), a rule X consists of two parts, the activation condition and the body. Then, the *Stateful Knowledge Session* activates the rules whose condition is satisfied (*S_RuleX* and *E_RuleX*). The next step is to decide whether it is reasonable to execute the activated rules: if so, the rules are fired (*S_FireAllRules*) and the AIS events generated by their execution are

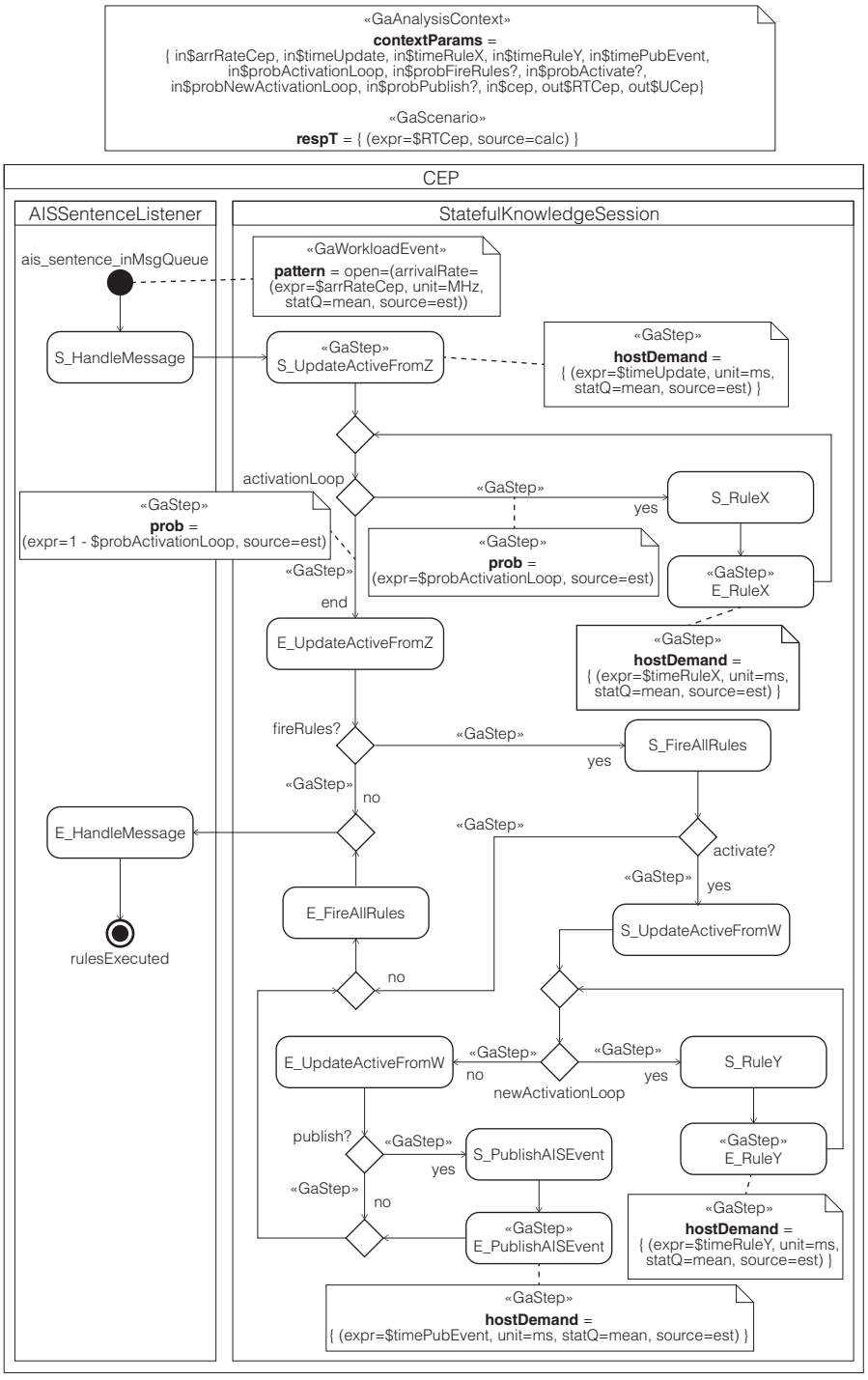


Fig. 3 CEP execution cycle.

added to the KB, otherwise the process terminates (*E_Handle Message*). The firing of a rule may cause the activation of other rules, then a new update is carried out (*S_UpdateActiveFromW*), where each new rule Y that satisfies the activation condition is added to the list (*S_RuleY* and *E_RuleY*). Finally, the *Stateful Knowledge Session* decides whether to publish an AIS event in the message queue (*S_PublishAISEvent* and *E_PublishAISEvent*).

3.3 Performance scenarios

The proposed methodology requires an initial UML-based specification of the system, that represents an execution scenario of interest. However, in order to obtain a performance model, as required by the **Step 1** of the methodology, the system scenarios need to be transformed into performance scenarios [34]. Basically, the UML diagrams need to be enriched with input parameters to characterize the workload, the timing and probabilistic specifications – such as the duration of the computation steps or the probability of alternative steps – and the performance metrics of interest – such as the response time and throughput of the scenario and the utilization of the resources.

To that end, we apply the standard MARTE profile (*Modeling and Analysis of Real-time and Embedded Systems*) [26], that enables the designer to specify performance parameters through UML extensions, i.e., stereotypes and tagged values. A key feature of MARTE is the framework for the specification of non functional properties (NFP) and the Value Specification Language (VSL). The former allows the modeller to define several properties of a performance parameter, such as the *source* to indicate whether it is an estimated value (i.e., *est*) or a value to be calculated (i.e., *calc*), or the type of statistical measure associated to it (e.g., a mean). The VSL enables the specification of variables and complex expressions according to a well-defined syntax.

In the UML diagrams of Figures 1, 2, and 3 we used the UML note symbol to show explicitly the tagged values associated to a given stereotyped model element. However, when a UML tool with profiling facilities is used, the stereotypes and tagged values can be easily set via the GUI. This is the case, for example, of the Eclipse Papyrus UML tool [35] which supports MARTE and has been used to define the UML specification of the case study.

In particular, the behavioral diagrams – SD or AD – are stereoyped with *GaAnalysisContext* and *GaScenario* to declare the variables used in the diagrams as input/output parameters and to specify the scenario-related performance metrics to be predicted, respectively. In MARTE, variables are defined by names prefixed by the dollar symbol. The workload characterization is specified by stereotyping the model element that represents the first step within the scenario with *GaWorkloadEvent*. When the scenario is modelled with an SD, such step is the first message – e.g., *NMEAstream* in Figure 2 – whereas in case of a scenario modelled with an AD, the first step is the initial node – e.g., *ais_sentence_in_MsgQueue* in Figure 3. Different types of workload are possible (e.g. open or closed): in the case study, both the parsing and the CEP

scenarios are characterized by an open workload, i.e., data stream, with a mean arrival rate. A step within a performance scenario (*GaStep*) may represent a computation step – i.e., an action execution specification or a message in an SD, an action in an AD – that requires a certain amount of time to be executed by the host, or a decision step – i.e., a transition outgoing from a decision node in an AD. Then, a mean duration is associated to computation steps (i.e., a VSL value or expression is assigned to the *hostDemand* tag, named tagged value) and a probability is associated to decision steps (i.e., the *prob* tagged value). Finally, the number of logical resources are annotated with the *poolSize* tagged value: in an SD-based scenario, the logical resources are the lifelines participating to the interaction (stereotyped with *PArunTinstance*) – e.g., the sub-components of the AIS Parser in Figure 2 – whereas in an AD-based scenario, they are the artifacts in the DD (stereotyped with *PaLogicalResource*) represented by the swimlanes in the AD – e.g., CEP in the Figures 1 and 3. In the DD, the resource-specific performance metrics, such as the resource utilization, can be also annotated.

4 Model Generation and Log Pre-processing

This section explains how to apply the first two steps of the methodology. The first step (**Step 1**) corresponds to the generation of normative models from the performance scenarios of Posidonia Operations. The second step (**Step 2**) corresponds to the pre-processing of the data logs, collected during the system execution, to get *event logs*, which will be analyzed using process mining techniques.

4.1 Automatic generation of normative models

The **Step 1** is carried out with the support of the DICE Simulation tool [19], that implements two model-to-model (M2M) transformation approaches from UML-MARTE specifications. One approach produces a Petri Net (PN) model from an SD scenario [10]. The other approach produces a PN model from an AD scenario [23]. In the case of Posidonia Operations, two performance scenarios were considered: the parsing process, modelled by the SD of Figure 2, and the CEP execution cycle, modelled by the AD of Figure 3. Therefore, two separate PN models are produced by the tool, one for each performance scenario.

Figure 4 depicts the PN as produced by the tool for the parsing scenario, we have only rearranged the PN places and transitions to facilitate the reading. Each vertical sequence of places-transitions represents an SD lifeline stereotyped with *PArunTinstance* (cf. Figure 2). The initial place of each sequence is marked with as many tokens as the *poolSize* tagged value associated to the corresponding SD lifeline. The transitions represent message event occurrences (send or receive) or action execution specifications. The horizontal

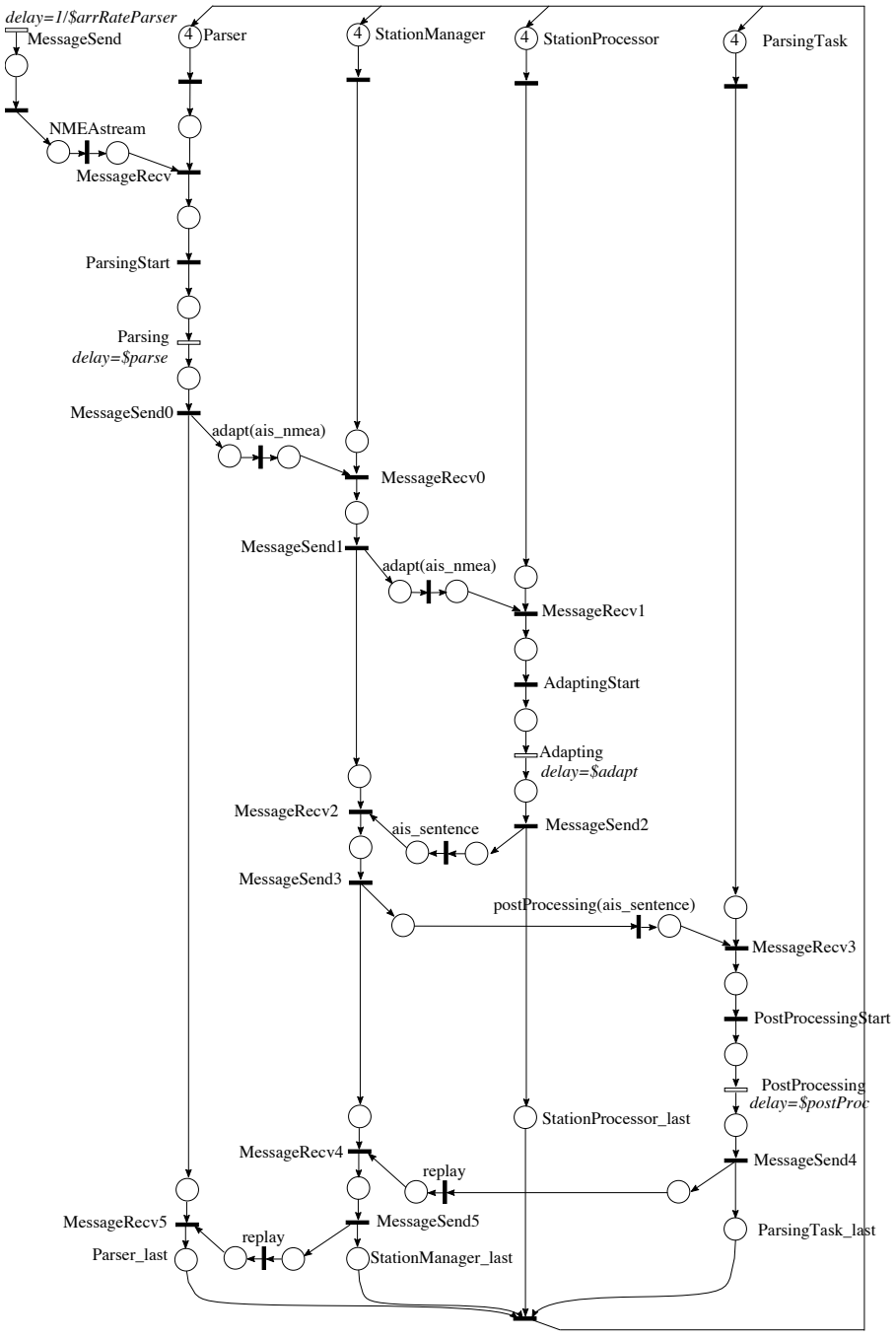


Fig. 4 Petri Net model of the Parsing scenario.

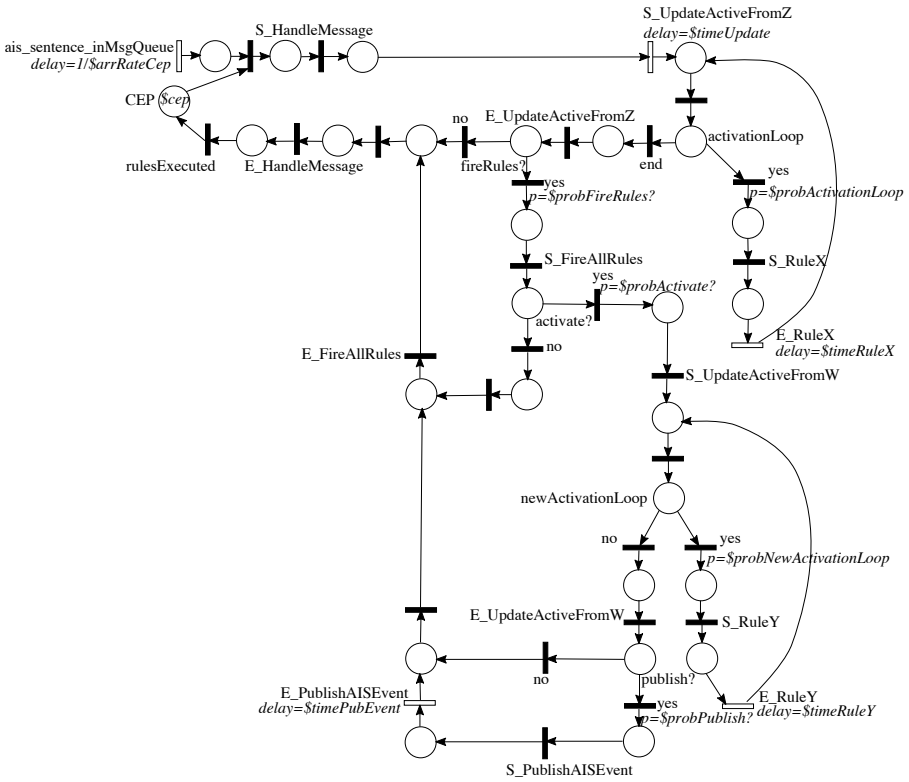


Fig. 5 Petri Net model of the CEP scenario.

sequences of place-transition-place represent instead messages exchanged between the lifelines – e.g., *adapt(ais_nmea)*. The action execution specifications or the messages stereotyped with *GaStep* are mapped to timed transitions, where the firing delay is set to the *hostDemand* tagged-value associated to the former. Finally, the PN includes a source transition (*MessageSend*, at the top-left of the figure) that models the open workload, where the firing delay is set to the inverse of the *arrivalRate* tagged-value associated to the first message (*NMEAstream*), sent by the actor *AISReceiver*, in the SD.

Figure 5 depicts the PN as produced by the tool for the CEP scenario, again, we have only rearranged the PN places and transitions for reading purposes. This M2M transformation also considers the logical resource restriction from the DD of Figure 1. The action nodes of the AD are mapped to PN transitions: those stereotyped with *GaStep* correspond to timed transitions, where the firing delay is set to the *hostDemand* tagged-value associated to the former. The decision nodes of the AD together with their outgoing transitions are mapped to free-choice subnets, where the weight of each conflicting PN transition is set to the *prob* tagged-value associated to the mapped outgoing transition. The initial and final nodes of the AD are mapped to timed and

immediate transitions, respectively. In particular, the timed transition is characterized by a firing delay equal to the inverse of the *arrivalRate* tagged-value associated to the initial node. Finally, the *CEP* logical resource – in the DD – is mapped to the corresponding place in the PN, where the initial marking is set to the *poolSize* tagged value associated to the former. In Figure 5, the initial marking of the place *CEP* is an input parameter.

4.2 Tool-assisted *log* pre-processing

Step 2 can be carried out concurrently with **Step 1** and its main goal is to get *event logs* $\mathcal{EL}$ from the data logs $\mathcal{L}$. Event logs are data logs collected during the system execution, where each event occurrence is characterized by at least the identifier of the process instance (the so-called *caseID*), the type of event (*eventID*) and the timestamp. This pre-processing phase produces event logs specified in CSV format, that are then fed to the ProM tool [36] and finally converted into the XES [37] standard format, ready to be analyzed in the further steps.

The data logs of Posidonia Operations, analyzed in this paper, consist of a set of six CSV files that contain the traces collected during, approximately, two days of system execution – June, 30th and August, 9th 2016. In particular, during the second day, the execution traces were collected under two different system conditions, approx. for a period of half-day each: normal and *forced rule*. In the latter case – that affects only the CEP scenario – the processing time of one of the CEP rules was manipulated to increase it, periodically, during the observation interval.

The execution traces of the parsing process correspond to the transformation of an NMEA message, from a unique AIS receiver, to an AIS sentence (cf. Figure 2), whereas the execution traces of the CEP process represent the handling of the AIS sentences by the CEP of the Valencia port (cf. Figure 3).

Each line of the data logs corresponds to an event occurrence, within an execution trace, and it is characterized by the following fields: the name of the parsing/CEP phase, a tag indicating either the starting or the ending of the phase, and finally the timestamp – in microseconds – from the latest restart of the node the application was running. Additionally, in the parsing process data logs, each line includes also the parsing thread.

The original data logs of Posidonia Operations have been transformed in event logs using scripts, where the *caseID* corresponds to either the NMEA message identifier (parsing process logs) or the AIS sentence identifier (CEP process logs) that is being processed, and the *eventID* indicates the starting or ending of a given phase.

Table 1 summarizes the overall statistics of Posidonia Operations event logs computed with the ProM tool: in particular, the date and the overall period of time of the logs (first column), the process of reference (second column) – i.e., the parsing scenario or the CEP scenario – the total number of execution traces (third column) and event occurrences (fourth column), the

Date - Period	Process	Cases	Events	Events per case		
				Min	Mean	Max
June 30, 2016 - 22:48:06	Parsing	5,728,367	18,120,165	1	3	8
August 9, 2016 - 11:32:21	Parsing	3,301,479	10,060,661	1	3	8
August 9, 2016 - 11:34:58	Parsing	3,301,474	10,060,213	1	3	8
June 30, 2016 - 22:47:24	CEP	168,687	5,312,354	4	31	3,316
August 9, 2016 - 11:33:52	CEP	90,642	2,792,622	4	31	2,938
August 9, 2016 - 11:36:09	CEP	90,641	2,793,450	4	31	37,718

Table 1 Summary of Posidonia Operations event logs.

minimum, average and maximum number of event occurrences per execution trace (the last three columns).

At a first glance, there are not important differences neither between the two dates nor between the two half-days of August in the logs of the parsing process. On the other hand, there is a high variability in the number of events per case in the logs of the CEP process, especially in the log that collects the execution traces under the *forced rule* condition assumption. Such raw statistics deserve further investigation in the next step.

5 Model Enhancement

This section explains how to apply **Steps 3–7** of the methodology. First, in subsection 5.1, **Steps 3–6** align the normative models and the event logs for the Posidonia Operations case study. Then, in subsection 5.2, **Step 7** uses a *trace-driven simulator* to estimate the input parameter values that are needed for performance analysis. Finally, **Step 7** validates the normative models, $\mathcal{N}$, and the already estimated input parameter values, using a statistical approach based on the computation of confidence intervals.

5.1 Alignment of models and event logs

The purpose of the alignment is threefold: 1) provide insight on the correctness of the system behavior represented by the normative models, 2) filter those execution traces that are significant from the performance evaluation point of view, and 3) discover new system behaviors from the logs that can improve the initial UML specification, considering the goals of the performance assessment.

The normative model is a formal representation of the system scenarios, while the event logs consist of sets of execution traces, where each trace describes a real concrete behavior. Therefore, the normative model is not exhaustive, since it considers a (sub-set of) system behaviors. Similarly, the event logs provide information about the running system, however they do not include a full description of the system behavior, but only those paths actually executed.

In **Step 4**, the *trace filtering* stage, only complete execution traces will be considered. That is, traces that correspond to the performance scenarios modelled with the initial UML specification. In the Posidonia Operations case

Date - Period	Process	Cases	Events	Events per case		
				Min	Mean	Max
June 30, 2016 - 22:48:06	Parsing	1,110,572	8,884,576	8	8	8
August 9, 2016 - 11:32:21	Parsing	576,284	4,610,272	8	8	8
August 9, 2016 - 11:34:58	Parsing	576,211	4,609,688	8	8	8
June 30, 2016 - 22:47:24	CEP	163,917	3,045,446	4	19	141
August 9, 2016 - 11:32:19	CEP	89,565	1,655,926	16	18	40
August 9, 2016 - 11:34:56	CEP	89,476	1,655,004	16	18	60

Table 2 Summary of Posidonia Operations event logs after filtering (complete traces).

study, the parsing scenario (Figure 2) starts with the reception of an NMEA stream, by the *Parser* component, and terminates when the latter receives a *reply* message. On the other hand, in the event logs of the parsing process, the first traces of each log include just one event (cf. Table 1, minimum event per case column) – this was due to the fact that the initial instant of data collection did not correspond to the beginning of a parsing process instance – and about 80% of the rest of the cases are partial execution traces. Such cases correspond actually to a no time-consuming alternative scenario – not explicitly modelled in the initial UML specification – where the *Parser* component is collecting information about a given vessel (e.g., the first action execution carried out by the *Parser*, in Figure 2) before producing the corresponding AIS sentence.

The CEP execution cycle (Figure 3) models the complete execution of the message handling by the *AIS Sentence Listener* component and, unlike the parsing scenario, includes several alternative scenarios. However, the event logs of the CEP include execution traces that represent behavior not modelled in the UML specification of Fig. 3. Such traces correspond to the removal of old AIS messages from the Knowledge Base (KB), that is not a scenario of interest considering the performance goals, and they are not statistically significatives (lower than 3% of the overall traces in the event logs).

Table 2 summarizes the overall statistics, gathered by **Step 4**, of the event logs after being filtered with the ProM tool. Observe that the complete traces of the parsing process now include exactly eight events and that the variability of the CEP traces has been drastically reduced.

In **Step 5**, the execution traces in the already filtered event logs are then "replayed" on the normative model using the *conformance checking* algorithm [3], that is implemented as plugin of the ProM tool [36]. The *conformance checking* provides feedback about the level of alignment of the normative model and the event logs. In particular, it enables both: 1) to discover anomalous traces, that may indicate either bugs in the implementation or flaws in the design specifications, and 2) to refine the behavior modeled in the UML specification.

The conformance checker of ProM requires a mapping between the transitions in the normative model $\mathcal{N}$ and the events in the logs. Table 3 summarizes the mapping for the PN model of the Parsing scenario (Figure 4). The table only shows those transitions that have a counterpart event in the logs, the rest

PN transition ($\mathcal{N}$)	Event ($\mathcal{EL}$)
ParsingStart	Parser_Parse_Start
MessageSend0	Parser_Parse_End
MessageRecv0	Station_Manager_Process_Start
AdaptingStart	Station_Processor_Process_Start
MessageSend2	Station_Processor_Process_End
PostProcessingStart	Parsing_Task_After_Station_Processed_Start
MessageSend4	Parsing_Task_After_Station_Processed_End
MessageSend5	Station_Manager_Process_Start

Table 3 Parsing scenario: mapping PN transitions and log events.

of the transitions are assumed not observable since they are not related to log events.

In the case of the CEP scenario, the mapping is more complex since there are transitions in the PN model that have correspondence with more than one event in the logs. This is because the initial UML specification abstracts from a specific rule activation and updating, and AIS event publication. Table 4 summarizes the mapping, where each transition type represents the starting and ending of the action (e.g., *RuleX* indicates the two transitions in the PN model, respectively *S_RuleX* and *E_RuleX*). Similarly, each event type represents the starting and ending of the event (e.g., *Simple_Dock_Start_Out* indicates the two events in the log, respectively *Simple_Dock_Start_Out_Start* and *Simple_Dock_Start_Out_End*) and the starting (ending) actions correspond to starting (ending) events. Like in the parsing scenario, there are transitions in the PN model of Figure 5 that have not counterpart events in the log, then they are assumed not observable.

The *conformance checker* returns a fitness score that estimates the alignment of the execution traces with respect to the normative model, e.g., a 100% of fitness means that all the execution traces in the log can be replayed in the normative model.

After filtering the event logs, the PN model of the parsing scenario presents a perfect fitness (i.e., 100%) and the PN model of the CEP presents a 99.999% fitness. The fitness is not perfect in the latter, since there are 24 out of 342,931 traces where the termination (starting) of an activation rule occurs at the beginning (end) of the cycle. Figure 6 represents one of such traces, which all belong to the forced rule log: the two mentioned events are emphasized in bold and correspond to the end/start of the forced rule (i.e., *Simple_Anchor_In*).

Finally, just one filtering iteration was carried out, **Step 6** of the methodology. It already enabled to reach a fitness of at least 99%, in both the parsing and CEP scenarios, that is an acceptable threshold when the final objective is to enhance the latter with timing and frequency information [4].

It is worth observing that the conformance checking allowed us also to refine the original specification of the CEP execution cycle, provided by the designers, that was at a higher abstraction level with respect to the final model (cf. Figures 7 and 3). Indeed, one of the features of the conformance checker implemented in ProM is the visualization of the log-model alignments

PN transition type ($\mathcal{N}$)	Event type ($\mathcal{EL}$)
HandleMessage	AIS_Sentence_Listener_Handle_Message
UpdateActiveFromZ	Update_Active_From_Insert_AIS_Geomdata
FireAllRules	Session_Fire_All_Rules
RuleX	Simple_Dock_Start_Out Simple_Dock_Stop Stop_Over_Out Stop_Over_In Simple_Anchor_In Simple_Anchor_Out
RuleY	Simple_Dock_Start_Out Simple_Dock_Stop Stop_Over_Out Stop_Over_In Simple_Anchor_In Simple_Anchor_Out
UpdateActiveFromW	Update_Active_From_Simple_Dock_Start_Out Update_Active_From_Simple_Dock_Stop Update_Active_From_Insert_Ship Update_Active_From_Simple_Anchor_In Update_Active_From_Simple_Anchor_Out
PublishAISEvent	Dock_Start_Out_Publish Dock_Stop_Publish Anchor_In_Publish Anchor_Out_Publish

Table 4 CEP execution cycle: mapping PN transitions and log events.

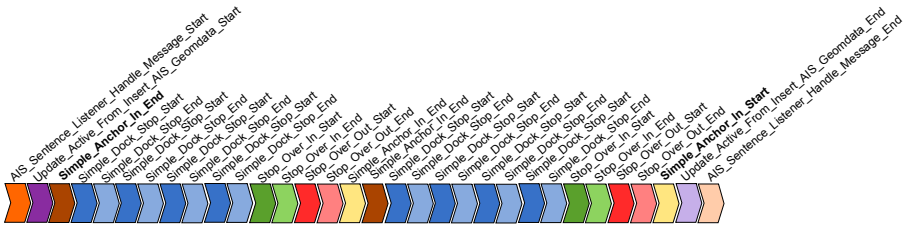


Fig. 6 CEP anomalous trace.

that shows, for each trace, the sequence of event occurrences characterized by different colors depending on the type of alignment with the model. In particular, the events in the traces that have not been mapped to transitions in the model can be also identified and provide useful information for the refinement. In the case of the CEP, one of the objectives of the performance analysis is to evaluate the cost associated to the rules so the events related to the rule activation and the AIS event publication have been explicitly modelled in the refined UML specification.

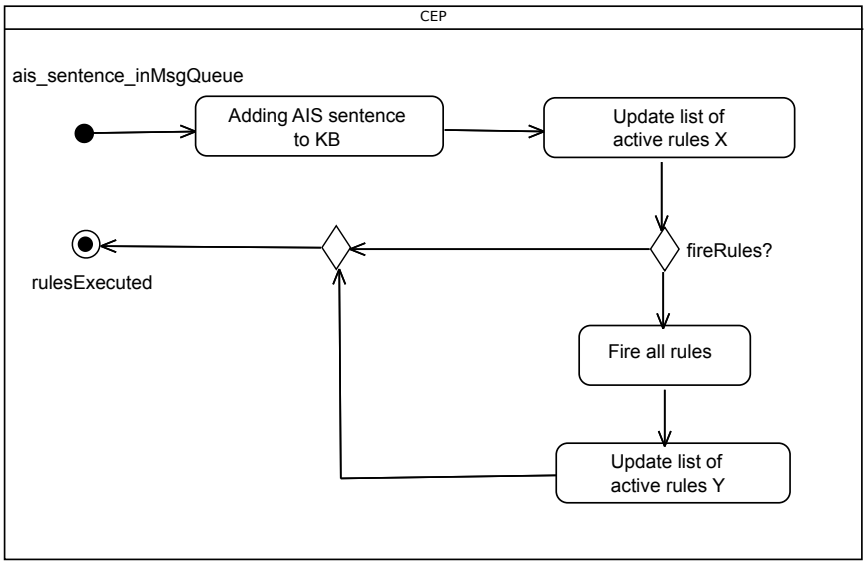


Fig. 7 CEP execution cycle (high level view).

5.2 Estimation of the input parameters

In **Step 7**, $\mathcal{S}$ is enhanced with timing and probabilistic perspectives. Indeed, the UML models of Figures 2 and 3 are characterized by several input parameters – i.e., workload arrival rates, step durations and step execution probabilities – that need to be set to actual values prior to carry out performance analysis in **Step 8**. Hence, we now obtain reference values by using the trace-driven simulator [3] of the ProM tool [36]. This tool "replays" the event logs (which had been filtered, in Step 4, to consider only complete traces) on the Petri Net models (automatically generated from the UML specifications, in Step 1). Then, it collects several statistics, such as the average, minimum, maximum and standard deviation of the time between the firing of two transitions, and the absolute firing frequency of transitions.

Similarly to the conformance checking algorithm, also the trace-driven simulator requires a mapping between transitions in $\mathcal{N}$ and events in $\mathcal{EL}$. The mapping already obtained has been reused (see Table 3 for the parsing scenario, and Table 4 for the CEP execution cycle).

Tables 5 and 6 show the input parameter values obtained, respectively, for the parsing and the CEP scenarios, by "replaying" the event logs of June (column $\mathcal{EL}_1$) and August (columns $\mathcal{EL}_2$ – normal condition – and $\mathcal{EL}_3$ – forced rule condition). In each table, the first column specifies a model element in the UML model $\mathcal{S}$, that corresponds to a transition in $\mathcal{N}$, and the second

column indicates the input parameter in the expression property of the VSL annotation in $\mathcal{S}$.

$\mathcal{S}$(SD)	Input parameter values			
First message	arrivalRate (1/ms)	$\mathcal{EL}_2$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
NMEA_stream	$arrRateParser	0.013529	0.013873	0.013817
Message	hostDemand (ms)	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
adapt(ais_nmea) to SM	$msgDelay1	0.177595	0.086282	0.092299
adapt(ais_nmea) to SP	$msgDelay2	0.160340	0.266934	0.271289
ais_sentence	$msgDelay2	0.160340	0.266934	0.271289
postProcessing(ais_sentence)	$msgDelay2	0.160340	0.266934	0.271289
replay to SM	$msgDelay2	0.160340	0.266934	0.271289
ActionExecution	hostDemand (ms)	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
parsing	$parse	0.433219	0.335267	0.363914
adapting	$adapt	0.331089	0.369738	0.372322
postProcessing	$postProc	0.891405	0.976893	1.030288

Table 5 Parsing scenario: setting input parameter values.

Two types of statistics, computed with the trace-driven simulator, have been considered to set the input parameter values:

- The average values that estimate the mean duration of steps (cf., the *mean* statistical qualifier set in the *hostDemand* tagged values in Figures 2 and 3), and
- The absolute frequency values that, once converted in relative frequencies values, enable to estimate the probabilities of step execution (cf., the *prob* tagged values in Figure 3).

Concerning the mean arrival rates (*arrivalRate* tagged values), they have been estimated by dividing the number of traces by the log period.

It is worth noting that the trace-driven simulation enables to refine the VSL annotations in the UML models, such as timing delays to execution steps that were initially assumed negligible.

In the CEP execution cycle, the rules activation in the first activation loop, the firing of the rules and the last step before the cycle termination are characterized by not negligible time delays (the light grey rows in Table 6). Similarly, in the parsing scenario, we discovered that there are delays between the action executions that were eventually associated to the messages (the light grey rows in Table 5). In particular, the overall delay of the interaction between the *Station Manager*, *Station Processor* and *Parsing Task* (cf. Figure 2) has been equally distributed to the exchanged messages.

Finally, the trace-driven simulator enables to compute statistics for event types in the logs that are mapped to a single transition of the Petri Net model, such as in the case of the CEP model (cf., Table 4). Appendix A includes detailed performance results concerning the single rules activation and updating, and the AIS events publication.

$\mathcal{S}$(AD)	Input parameter values			
Initial node	arrivalRate (1/ms)	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
ais_sentence_inMsgQueue	$arrRateCep	0.001998	0.002156	0.002146
Action	hostDemand (ms)	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
S_UpdateActiveFromZ	$timeUpdate	0.840358	1.021727	1.068320
E_RuleX	$timeRuleX	0.408430	0.387228	5.688869
E_UpdateActiveFromZ	$timeUpdate2	4.970570	4.833462	5.205351
E_FireAllRules	$timeFire	5.197355	4.781439	5.326718
E_RuleY	$timeRuleY	0.416560	0.336991	1.520000
E_publishAISEvent	$timePubEvent	2.310345	2.583333	4.115385
E_HandleMessage	$timeHandle	1.023043	0.827306	0.926749
Transition	prob	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
activationLoop–yes	$probActivationLoop	0.873800	0.873440	0.873476
activationLoop–end	1-$probActivationLoop	0.126200	0.126560	0.126524
fireRules?–yes	$probFireRules?	0.264607	0.250628	0.250664
fireRules?–no	1-$probFireRules?	0.735393	0.749372	0.749336
activate?–yes	$probActivate?	0.001916	0.003071	0.003541
activate?–no	1-$probActivate?	0.998084	0.996929	0.996459
newActivationLoop–yes	$probNewActivationLoop	0.872460	0.873973	0.872443
newActivationLoop–no	1-$probNewActivationLoop	0.127540	0.126027	0.127557
publish?–yes	$probPublish?	0.256637	0.130435	0.245283
publish?–no	1-$probPublish?	0.743363	0.869565	0.754717

Table 6 CEP execution cycle: setting input parameter values.

5.2.1 Validation of the normative models

The trace-driven simulation has enabled us to get statistical performance results. However, for performance prediction purposes, we will rely on stochastic models, concretely Generalized Stochastic Petri Nets (GSPN – see Appendix B). GSPN allows us to carry out sensitivity analysis by varying (some of) the input parameters, so to analyze the effects on the metrics of interest – e.g., the parsing or the CEP response times (cf., the *respT* tagged values in Figures 2 and 3).

Before conducting sensitivity performance analysis, the GSPN models need to be validated first [29]. To this aim we follow a statistical approach that, instead of using test hypothesis such as in [31], it is based on the computation of confidence intervals. The approach can be summarized as follows:

1. Choose, as a metric of reference, the scenario execution time.
2. Compute the 99% confidence interval of the average of the scenario execution time μ^s, obtained from the trace-driven simulation of the event logs.
3. Compute the mean of the scenario execution time μ^m with the event-driven simulation of the GSPN models, with the same confidence level.
4. Check whether μ^m falls in the confidence interval of μ^s.

In the parsing scenario, the metric of reference is the parsing processing time, whereas in the CEP scenario it corresponds to the CEP execution cycle time.

The trace driven simulation of the filtered event logs enables to compute the average μ^s and the standard deviation σ^s of the time between the first and the last event of the traces. In the case of the parsing scenario such events

correspond to the reception of the *NMEA stream* and the *reply* messages by the *Parser*, respectively (cf., Figure 2). Instead, in the CEP scenario the first and the last event correspond to the execution of the *S_HandleMessage* and the *E_HandleMessage* actions, respectively (cf., Figure 3). Therefore, we can compute a $100(1-\alpha)\%$ confidence interval for the estimated averages by applying the formula [25]:

$$\mu^s \pm \frac{z_{1-\alpha/2}\sigma^s}{\sqrt{N}} \tag{1}$$

where $z_{1-\alpha/2}$ is the $1-\frac{\alpha}{2}$ value of the standard normal distribution and N is the number of traces in the log.[1]

Parsing processing time (ms)			
Logs	Trace-driven simulation		Event-driven simulation
	μ^s	$\pm\frac{z_{0.995}\sigma^s}{\sqrt{N}}$	μ^m
$\mathcal{EL}_1$	2.474667	±0.007360	2.478704 ✓
$\mathcal{EL}_2$	2.835916	±0.007154	2.836631 ✓
$\mathcal{EL}_3$	2.943979	±0.026928	2.942335 ✓
CEP execution cycle time (ms)			
Logs	Trace-driven simulation		Event-driven simulation
	μ^s	$\pm\frac{z_{0.995}\sigma^s}{\sqrt{N}}$	μ^m
$\mathcal{EL}_1$	11.044730	±0.050825	11.044548 ✓
$\mathcal{EL}_2$	10.555474	±0.066901	10.557024 ✓
$\mathcal{EL}_3$	47.820008	±1.185844	47.855911 ✓

Table 7 Validation of the Parser and CEP models.

Table 7 shows, for each event log, the average (second column) and the 99% confidence interval (third column) of the parsing processing time and the CEP execution cycle time. Observe that the CEP execution cycle time increases considerably in the *forced rule* log, due to the manipulation of the processing time of the *Simple_Anchor_In* rule, while for the other logs the statistical results are similar.

The mean execution time μ^m of the parsing scenario and the CEP execution cycle has been computed using the DICE Simulation tool [19], considering a 99% confidence level. In particular, for each configuration of the input parameters in Tables 5 and 6, a simulation experiment has been conducted. For each experiment, the tool automatically generates a GSPN model – i.e., a Petri Net model enriched with the timing and probabilistic specification that are derived from the input parameter configuration – and launches the event-driven stochastic simulator of GreatSPN [16] with the GSPN model as input.

Finally, as shown in Table 7 (fourth column), the mean values (μ^m) fall in the confidence interval of the average values (μ^s), in all the experiments. Therefore, we can rely on the GSPN models produced by the tool.

[1] Observe that the normal distribution is considered instead of the Student's one since the sample size is large (i.e., $N \gg 30$).

6 Performance Analysis and Assessment

This section applies the **Step 8** of the methodology, that is carried out with the support of the DICE Simulation tool [19]. For the POSIDONIA Operations case study we conducted the performance analysis considering the three goals:

O1.1 The scalability of the product (in Subsection 6.1),
O1.2 The detection of bottlenecks (in Subsection 6.2), and
O1.3 The cost of a business rule (in Subsection 6.3).

The DICE Simulation tool enables to model the UML-MARTE specification, through a Papyrus [35] GUI, and to define the configuration of the simulation experiments, i.e., to set the values of the model and simulation parameters, via a proper GUI. In particular, the tool provides support to sensitivity analysis by allowing the user to assign a range of values to a (set of) model parameter(s), therefore a configuration may lead to a set of simulation experiments. Then, for each experiment, the tool automatically transforms the inputs into a GSPN model and launches the simulation to estimate the performance metrics, which were defined using VSL in the UML-MARTE specification. The simulation is eventually run by the GreatSPN [16] event-driven stochastic simulator. Finally, the performance results are synthetized by the DICE Simulation tool and presented to the user both in a textual and graphical formats.

The UML-to-GSPN transformation and the GSPN simulation steps are completely transparent to the user, therefore no knowledge on the GSPN formalism is needed to get the performance analysis. However, the user needs to interpret, in terms of the problem domain, the results produced by the tool.

$\mathcal{S}$(SD)	Input parameter values	
First message	arrivalRate (1/ms)	value
NMEA_stream	$arrRateParser	[0.01,1.7]
Message	hostDemand (ms)	value
adapt(ais_nmea) to SM	$msgDelay1	0.146399
adapt(ais_nmea) to SP ais_sentence postProcessing(ais_sentence) replay to SM	$msgDelay2	0.196756
ActionExecution	hostDemand (ms)	value
parsing adapting postProcessing	$parse $adapt $postProc	0.399755 0.344292 0.920610

Table 8 Parsing scenario: basic configuration parameters.

Concerning the model input parameters, we have considered a basic configuration where (see Tables 8, 9 and 10):

– There are four parser threads and there is one CEP resource (i.e., *$CEP* parameter in Table 10).

$\mathcal{S}$(AD)	Input parameter values	
Initial node	arrivalRate (1/ms)	value
ais_sentence_inMsgQueue	$arrRateCep	[0.002,0.168]
Action	hostDemand (ms)	value
S_UpdateActiveFromZ	$timeUpdate	0.904443
E_RuleX	$timeRuleX	0.400954
E_UpdateActiveFromZ	$timeUpdate2	4.922125
E_FireAllRules	$timeFire	5.057267
E_RuleY	$timeRuleY	0.380581
E_publishAISEvent	$timePubEvent	2.390244
E_HandleMessage	$timeHandle	0.953882
Transition	prob	value
activationLoop–yes	$probActivationLoop	0.873673
activationLoop–end	1-$probActivationLoop	0.126327
fireRules?–yes	$probFireRules?	0.259727
fireRules?–no	1-$probFireRules?	0.740273
activate?–yes	$probActivate?	0.002305
activate?–no	1-$probActivate?	0.997695
newActivationLoop–yes	$probNewActivationLoop	0.873144
newActivationLoop–no	1-$probNewActivationLoop	0.126856
publish?–yes	$probPublish?	0.2
publish?–no	1-$probPublish?	0.8

Table 9 CEP execution cycle: basic configuration parameters.

PALogicalResouce	poolSize	value
AIS NMEA Parser	–	4
CEP	$CEP	1

Table 10 Deployment: basic configuration.

- The mean *duration* of a step (i.e., action) is the weighted mean of the step average values obtained from the logs $\mathcal{EL}_1$ and $\mathcal{EL}_2$. The weight is given by the frequency of the step in the two logs.
- The *probability* of a step execution is the overall relative frequency of the step in the two logs $\mathcal{EL}_1$ and $\mathcal{EL}_2$.
- The parser and CEP *workloads* will range in intervals[2]. Tables 8 and 9 show the largest intervals considered for the arrival rate of the NMEA stream to the parsing process (i.e., *$arrRateParser*) and the AIS sentences to the CEP (i.e., *$arrRateCep*), respectively.

The statistical results obtained from the *forced rule* log (i.e., $\mathcal{EL}_3$) will be considered in the third goal (i.e., cost of a business rule). Finally, the confidence level of the simulation has been set to 99% for all the experiments, through the Simulation tool GUI.

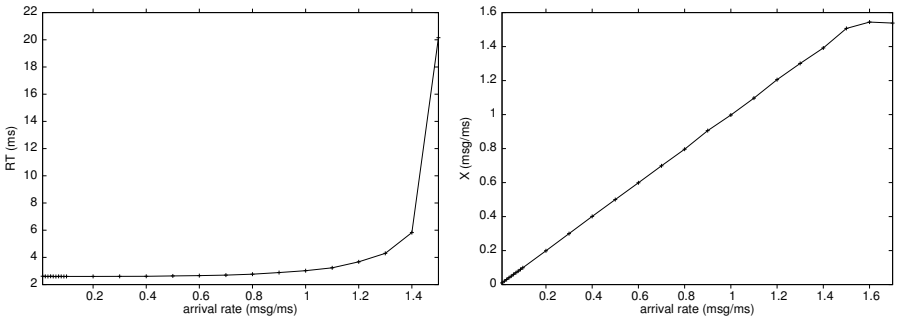
[2] Different intervals will be considered in the graphical representations, depending on the aim of the analysis.

6.1 Scalability of the product

From the point of view of the software architects, it is important to evaluate the impact on the performance of the changes in the data stream, both in the parsing and CEP scenarios in order to choose a deployment configuration (i.e., the number of logical resources to be assigned). The performance metrics of interest are the response time, for both the scenarios, and the throughput of the parsing process which is related to the workload of the CEP. The performance metrics have been explicitly declared in the two scenarios of Figures 2 and 3 with a UML-MARTE annotation (i.e., the *respT* and *throughput* tagged values associated to the *GaScenario* stereotyped diagrams).

Performance analysis. The analysis of the parsing process has been carried out considering the arrival rate of the NMEA stream (*$arrRateParser*) as varying parameter, (cf. Table 8). The minimum value of the sentivity interval corresponds to the weighted mean of the arrival rates computed from the two logs ($\mathcal{EL}_1$ and $\mathcal{EL}_1$), whereas the maximum has been set by considering a workload increase of two orders of magnitude.

Figure 8(A) plots the curve of the response time versus the NMEA stream arrival rate. The trend is basically constant until 0.7 msg/ms, i.e., its value ranges between 2.6 and 2.7 msg/ms, when the arrival rate is greater than 0.7 msg/ms the response time starts to increase. On the other hand – cf. Figure 8(B) – the parsing is carried out under stable conditions in the interval $[0.1, 1.6]$ msg/ms, i.e., the rate of generation of the AIS sentences is equal to the arrival rate, and the system becomes unstable when the arrival rate is greater than 1.6 msg/ms. Since the expected arrival rate is lower than such stability threshold, we can conclude that the parsing process with four threads is scalable.



(A) **(B)**

Fig. 8 Parser response time (A) and throughput (B).

In the analysis of the CEP scenario, the arrival rate of the AIS sentences (*$arrRateCep*) is one of the varying parameters. The other parameter is the number of CEP instances (*$CEP*). Indeed, we are interested to figure out

whether only one CEP is sufficient to process the workload related to a port or, instead, several instances of the CEP engine are needed to manage it. In particular, the arrival rate of AIS sentences to the CEP of the Valencia port is about the 24% of the throughput of the parsing process.

Let us consider the arrival rate of the NMEA stream to the parsing process in $[0.01, 0.7]$ msg/ms, where the response time of the parsing process is almost constant, as mentioned above. Therefore, for such interval, the arrival rate of the AIS sentences to the CEP of Valencia ranges in $[0.0024, 0.168]$ msg/ms. In the latter interval, the response time of one CEP increases considerably, as shown by the *1 CEP*-labelled curve in Figure 9. The figure also shows the trend of the CEP response time in case of five and seven instances of the CEP (labelled *5-7 CEP*), that remains basically constant ($\approx$ 10.8 ms) within the considered interval.

Assessment. From the analysis above, we assess **O1.1** as follows. In general, for a single a port, a workload of about 0.1 msg/ms can be observed, therefore it is necessary to configure a CEP with multiple instances in order to maintain the CEP response time between $[10, 11]$ msg/ms and, considering the two deployment configurations, the one with 5 CEP instances is sufficient to guarantee the response time requirement.

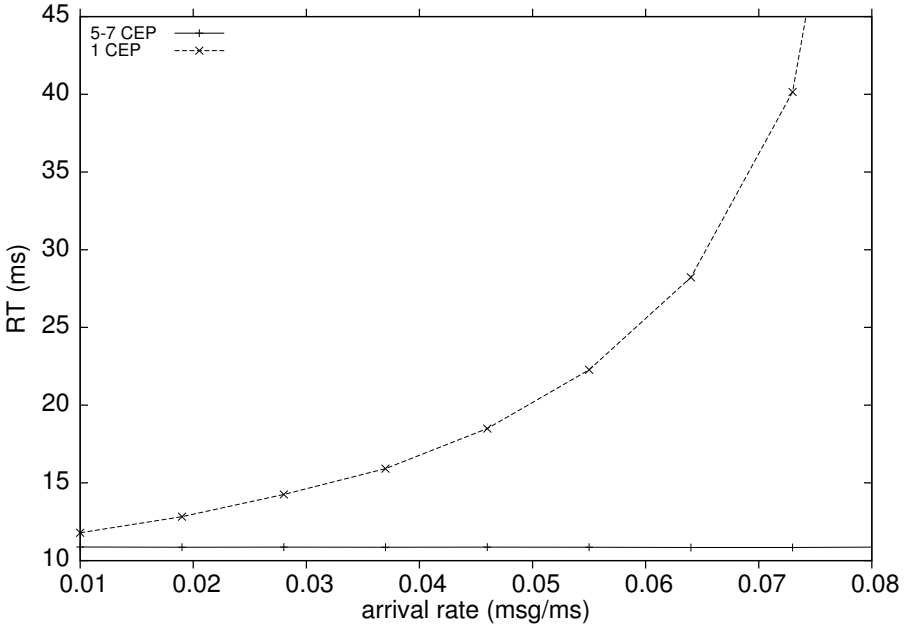


Fig. 9 CEP response time.

6.2 Detection of bottlenecks

We analysed the two scenarios with the same basic configuration and varying parameters considered in **O1.1**, whereas the performance metric of interest was the utilization of the logical resources assigned to the parsing and CEP processes (cf., the *utilization* tagged values associated to the *PAlogicalResource* artifacts in Figure 1). The utilization curves of the AIS NMEA Parser and the CEP are shown in Figure 10(A) and (B), respectively.

Assessment. From the analysis, we assessed **O1.2** as follows. The AIS NMEA Parser becomes saturated when the arrival rate of the data stream reaches 1.6 msg/ms, indeed this arrival rate is the stability threshold of the parsing process, as already observed for the throughput (cf., Figure 8(B)).

Considering the CEP, when the expected workload is about 0.1 msg/ms, just one instance of a CEP represents a bottleneck, indeed its utilization reaches 100% (Figure 10(B) – curve labelled *1 CEP*). This is not the case for a CEP with five or seven instances, where the utilization is 22% (curve labelled *5 CEP*) and 16% (curve labelled *7 CEP*), respectively.

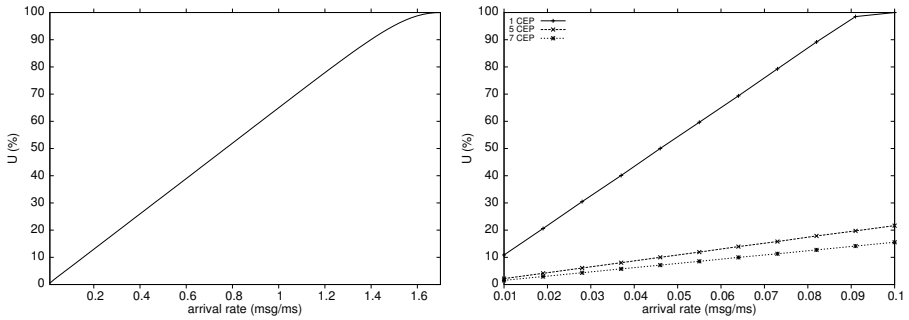


Fig. 10 AIS NMEA Parser (A) and CEP utilization (B).

6.3 Cost of a business rule

The experiments have been conducted to evaluate the scalability of the CEP scenario with respect to two different criteria: the number of rules to be processed and the mean execution time of the rule activation. In both cases, the performance metrics of reference are the response time and utilization, which have been already considered in the analysis of the two previous goals.

6.3.1 Number of rules

The analyzed logs collect the execution traces of a CEP with 6 different business rules (cf., Table 4, where first two rows indicate the activation of the

different rules which have been mapped to the type of actions *RuleX* and *RuleY*, respectively). However, POSIDONIA Operations is expected to manage an undetermined number of rules.

The implementation of a new rule impacts in several steps of the CEP execution cycle (Figure 3). In particular, for each new rule we assume a linear increment in:

1. The number of iterations of the two loops of rules activation (*activationLoop* and *newActivationLoop* decision nodes),
2. The probability of activating the second loop (*activate?* decision node), and
3. The probability of publicating new AIS events (*publish?* decision node).

Therefore, the varying parameters include the probabilities associated to such steps and to their complementary ones. Table 11 shows the configurations that have been set for the sensitivity analysis. The other varying parameters are the AIS sentences arrival rate (*$arrRateCep*) and the number of CEP instances (*$CEP*). The rest of the parameters have fixed values, according to the basic configuration in Table 9.

number of rules	6	7	8	9	10	11	12
$prob(New)ActivationLoop	0.873	0.888	0.899	0.908	0.916	0.922	0.928
1-$prob(New)ActivationLoop	0.127	0.112	0.101	0.092	0.084	0.078	0.172
$probActivate?	0.002	0.003	0.003	0.003	0.004	0.004	0.005
1-$probActivate?	0.998	0.997	0.997	0.997	0.996	0.996	0.995
$probPublish?	0.200	0.233	0.266	0.300	0.333	0.367	0.400
1-$probPublish?	0.800	0.767	0.734	0.700	0.667	0.633	0.600

Table 11 Number of rules sensitivity parameters

Figure 11 shows four plots of the CEP performance versus the number of rules (from 6 to 12) and the AIS sentences arrival rate. Since we are interested in analyzing the CEP under stable conditions, the interval of the arrival rate has been set considering a low utilization ($< 22\%$) of the CEP resources with 6 rules, which has been already analyzed in the performance goal **O1.2**. Therefore, the surfaces (A) and (B) that represent the response time and the utilization of the CEP with one instance, respectively, are plotted in the interval $[0.002, 0.02]$ msg/ms, whereas in the case of multiple instances the corresponding surfaces (C and D) are plotted in the interval $[0.01, 0.1]$ msg/ms.

Assessment. From this analysis, we assessed **O1.3** w.r.t. the number of rules as follows. The number of rules processed by the CEP affects both the response time and utilization. In the case of one CEP instance, the former (plot A) increases between 23% (arrival rate of 0.002 msg/ms, 11.043 ms for 6 rules and 13.551 ms for 12 rules) and 28% (arrival rate of 0.02 msg/ms, 13.064 ms for 6 rules and 16.755 ms for 12 rules) when the number of rules are doubled –

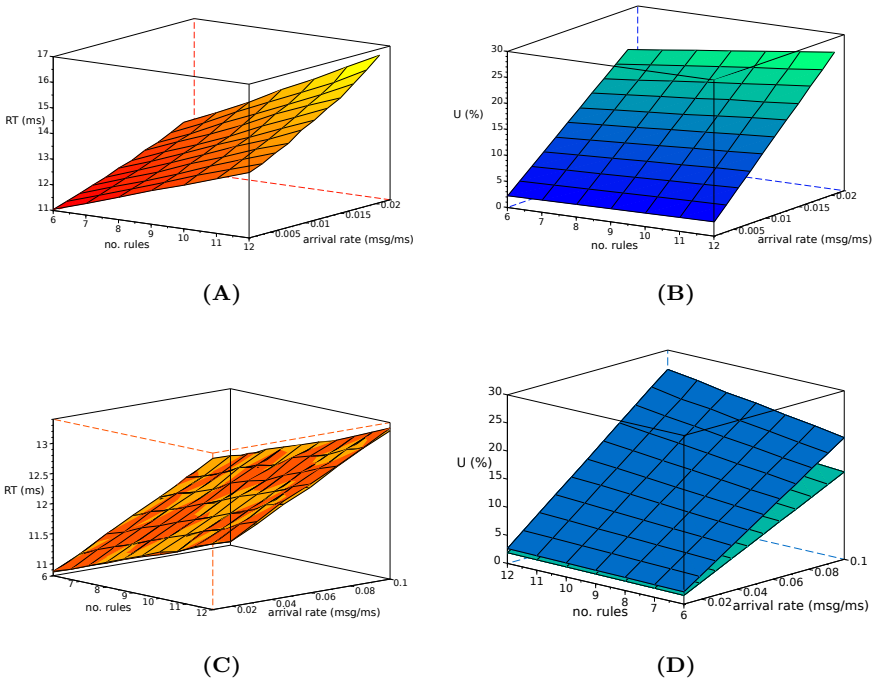


Fig. 11 CEP response time (left) and utilization (right) vs/ no. of rules and arrival rate

i.e., from 6 to 12 rules. Such increase changes according to the arrival rate of the AIS sentences, as it can be observed by the different slope of the surface when fixing two (different) arrival rates, e.g., the minimum and the maximum of the considered interval. The CEP utilization (plot B) also increases about $22 - 23\%$ by doubling the number of rules.

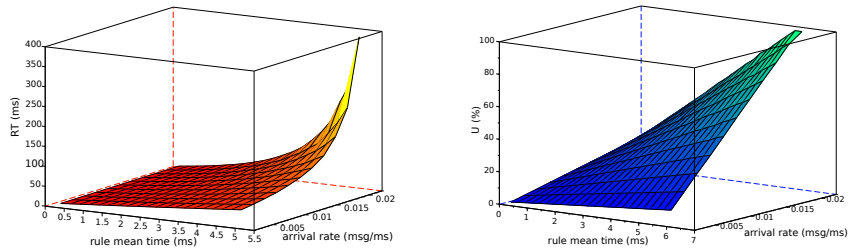
The plot C shows the trend of the response time in case of multiple instances (i.e., 5 and 7): the two surfaces, drawn with different color shading, basically coincide and, like the case of one CEP, the response time increases when the number of rules grows. However, the overall increment is lower than the one CEP case, since it is around the $22 - 23\%$ in the considered interval of arrival rate. On the other hand, the increase of the CEP utilization (plot D), when the number of rules is doubled is also around $22 - 23\%$. In the plot D, the two surfaces are distinguishable: the surface on the top corresponds to 5 instances – where the maximum CEP utilization of 26% is reached for 12 rules and an arrival rate of 0.1 msg/ms – whereas the bottom one corresponds to 7 instances – where the maximum is instead 19%.

6.3.2 Mean execution time of the rule activation

The mean execution time of the rule activation affects the CEP performance, as observed in the case of the *forced rule* log $\mathcal{EL}_3$ (cf., Table 6, the values of

the time parameters *$timeRuleX* and *$timeRuleY*), where the CEP execution time is about 3.5 times greater than the one estimated in the other two logs $\mathcal{EL}_1$ and $\mathcal{EL}_2$ (cf., Table 7).

In this set of experiments, we have focused on the first activation of the rules and the sensitivity analysis has been carried out by varying two parameters: the mean execution time of the first activation (*$timeRuleX* parameter) and the AIS message arrival rate. We assume one CEP instance and, for the rest of the parameters, the basic configuration of Table 9. Figure 12 shows the plots that represent the trend of the response time (A) and utilization (B) of the CEP versus the two varying parameters.



(A) **(B)**

Fig. 12 CEP response time (left) and utilization (right) vs/ rule mean time (ms) and arrival rate (msg/ms).

Assessment. From the analysis, we assessed **O1.3** w.r.t. the mean execution time of the rule activation as follows. When the mean execution time of the first activation increases from 0.4 to 5.15 ms, the slope of the surface (A) augments, revealing an exponential grow of the response time. Accordingly, also the utilization of the CEP increases (plot B) considerably and the CEP saturates for a mean execution time of 6.15 ms and an arrival rate of 0.02 msg/ms.

7 Discussion

The assessment of performance is a process of increasing importance in the industrial practice. For the case of our industrial partner, the success of next releases of the product strongly depends on the scalability. The work reported here has concluded the needs for the product to address such performance objectives. Moreover, the amount of work developed, during the year and a half that this experience has lasted, also allows us to report many other aspects of interest for practitioners. In the following, as suggested in [30], we discuss limitations of the results obtained, lessons learned and issues disclosed while

applying the assessment process, we also explain some of the consequences of all these matters.

7.1 Discussion of the research

The outcomes of an empirical research, as indicated by [30], should be interpreted from two perspectives: the *internal* and *external* validations.

Regarding the *internal* validation, it was established as the objective **O1** of the paper. In terms of our research, this means to analyse the usefulness of the performance assessment for PRODEVELOP. The assessment carried out in Section 6 has found the needs for ensuring the fulfilment of each sub-objective and it has determined the thresholds for improving each component of the architecture.

In particular, with respect to the scalability of the parsing process, we found that the NMEA stream arrival rate of 1.6msg/ms is the maximum threshold beyond which the system becomes unstable. Although this threshold has been never reached with the product that is currently operated – and then one instance of the AIS NMEA Parser component with four threads is sufficient to process the data stream – the software architects need to learn about it. Indeed, there are versions of the product under development that will monitorize new port areas, therefore they are expected to operate under higher NMEA stream workloads.

On the other hand, the assessment of the CEP revelead that the product in charge of managing a port, with just one instance of the CEP component, is not scalable under the expected workload (i.e., an AIS messages arrival rate of 0.1 msg/ms) and that multiple instances are needed. By estimating the cost of the business rules, the engineers can figure out finer solutions which consider – besides the multiple instances – also the number and mean execution times of the rules to be processed by each instance.

Summarising, the joint experience between engineers and researchers has been a success for clarifying where to focus the development efforts for the next releases of POSIDONIA Operations.

Regarding the *external* validation, it was addressed by **O2**, that established to *offer a methodology for performance assessment easy to be applied by SME practitioners*. In this paper, we have achieved to summarize the work carried out by the joint team of software engineers, from PRODEVELOP, and researchers, from UNIZAR, in eight steps that comprehend a substantial amount of models, data, analysis and discussions. Finally, we consider that **O2** has been achieved since each proposed step has been developed systematically, using software tools, an established methodology or a combination of both. Next paragraphs evaluate methodologies, whereas the tools are addressed in the next subsection.

Concerning UML, we can say that it is a mature language that has offered all the modelling features needed for our purposes. Moreover, it has been the vehicle for communication between the PRODEVELOP's engineers and the

Unizar researchers. Regarding the introduction of performance information in the UML models, we adopted MARTE since the DICE Simulation tool uses the VSL annotations. MARTE certainly also fulfilled all our needs. The combination of UML and MARTE was also advised by [20].

Process mining [1] provides methods for: *process discovery*, that is deriving process models from logs, *conformance checking*, that is checking the alignment of an existing/derived process model and logs, and *process enhancement*, that is enriching the process model through mining additional perspectives such as timing. The conformance checking and process enhancement techniques, applied in **Steps 3**–**7** of the methodology, require expertise in formal modelling; in particular, such techniques consider as input the Petri Net models obtained automatically from the UML-based specifications. This choice was mainly due to the unavailability of process mining tools that implement similar techniques at higher modelling levels, i.e., for UML activity or sequence diagrams. Nevertheless, the use of M2M transformations (**Step 1**), that keep track of the mapping between UML and Petri Net model elements, has reduced the effort of their application.

The log pre-processing (**Step 2**) to get *event logs* is the preliminary phase of the process mining and it is also crucial for the construction of a useful performance model in the proposed methodology. In particular, to produce event logs three characteristics need to be identified, i.e., the process instance identifier (*caseID*), the type of events (*eventID*), and the timestamp. Even though the definition of the *caseID* is not always straighforward, this step has been easy to carry out since the Prodevelop engineers have expertise in their products in operation and they have a clear vision of the objective of the performance assessment.

Finally, concerning the sensitivity performance analysis (**Step 8**), the critical point is the assignment of the range of values to the varying parameters in order to obtain *informational* results for the assessment of the performance objectives. Indeed, besides the setting of the minimum and maximum value of a given input parameter, also the number of values (or the increment step) in the range need to be fixed. In the performance analysis of Posidonia Operations, a constant increment step has been set for all the ranges of the sensitivity parameters since the performance behavior of the system is monotonic. However, in general, this may require several iterations of the simulation experiments.

7.2 Evaluation of the tools

The achievements above described regarding the assessment and the definition of a methodology are surely important. However, in our view, these would not be feasible for practitioners if experts are needed for applying the steps manually. Hence, most of our efforts, developing the methodology, have been focused on the selection of appropriate state-of-the-art tools and in the development of our own tool, also we cared the interaction of our tool with others, which were mature enough for relying on them instead of accomplish in-house de-

velopments. Finally, if some step could not be automated we tried to clearly explain how it should be applied, the subsection above has discussed issues at this regard concerning some steps of the process mining methodology.

In the following paragraphs we critically review the different tools advised by the methodology.

ProM tool. We have used the academic version of the ProM tool [28], although there exists a commercial one [17]. ProM tool includes a wide range of process mining techniques and it is constantly updated with new implementations from the process mining community. Within the methodology, it has been useful for the preliminary analysis of the event logs and the enhancement of the UML performance models. It also provides export functionalities, in different formats, of the results.

From the SME practitioners point of view, the main drawback of the tool is its usability, especially for what concerns the mapping of model elements and events in the log (in conformance checking and trace-driven simulation) and the presentation of the results. Another drawback refers to the limitation of the size of the logs that can be fed into and processed by the tool. Indeed, in the case study, we had to partition the event logs into sub-logs of smaller size ($\approx$ 250MB for the parser and $\approx$ 100MB for the CEP) with the inconvenience of having to spend more time in the synthesis of the results.

DICE Simulation tool. An important part of this work has been carried out using this tool [19]. The tool has been developed by the Unizar team in the context of the DICE project. The work on Posidonia Operations has helped a lot for improving our tool, in fact, some of its current features have been developed due to a need of the assessment, for example the graphical presentation of the analysis results, i.e., the computation of the metrics.

The Simulation tool is an Eclipse [18] plug-in [32] that complements the Papyrus modelling tool. So, the graphical modelling of the UML diagrams and the MARTE annotations are supported by Papyrus (see its evaluation next). Our plugin implements mature transformation approaches [23] and [10] that generate GSPN models from UML activity diagrams and sequence diagrams, respectively. The GSPN models are analysed by the GreatSPN tool (see also its evaluation next). However, our plugin provides a GUI, see Figure 13, that completely abstracts the user from GreatSPN. In fact, the latter can be installed elsewhere in a cloud, only the hostname/IP and port number need to be provided. Our tool processes the VSL and for example enables to assign complex VSL expressions to model input parameters. This feature is especially useful in the sensitivity analysis where the varying parameters are dependent (see our performance analysis when a different number of rules are considered). Another strong point of the tool is the support for the synthesis of the results and the automatic generation of 2D plots. Finally, we recognise as a practical limitation the number of varying parameters for sensitivity analysis.

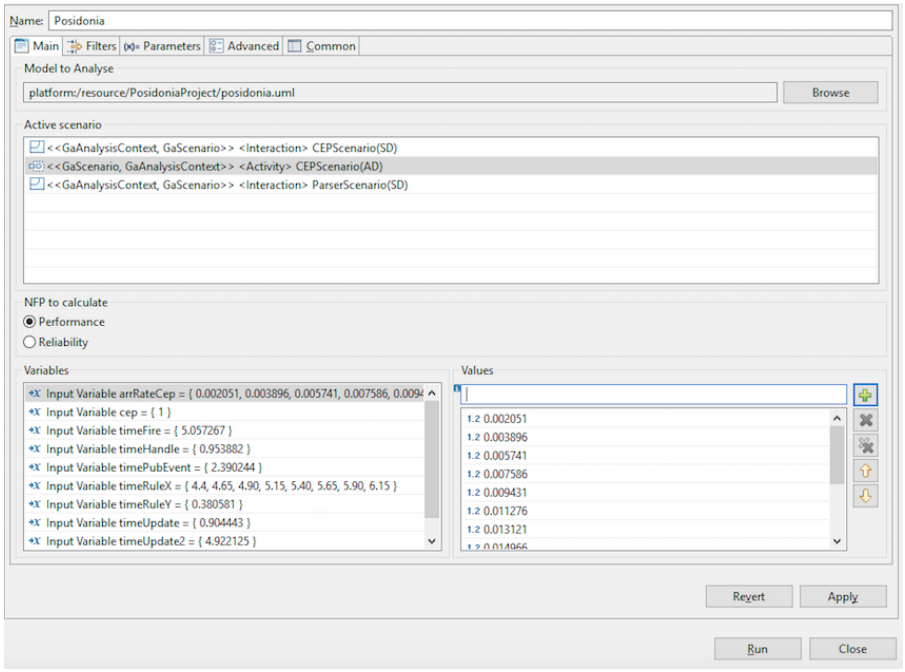


Fig. 13 Simulation tool: Setting input parameters.

Eclipse Papyrus is an UML 2.5 modeller that implements the MARTE profile. We have found two major drawbacks in Papyrus. First, the "learning curve", especially in the use of the stereotypes and tags. Indeed, it is somehow cumbersome the editing of the VSL expressions. Second, and probably the Papyrus' Achilles heel, refers to the usability of the sequence diagram modeller. Around 180 bugs have been reported, half of them are still open [12].

GreatSPN incorporates a mature event-driven simulator, which is successfully leveraged by our Simulation tool. This simulator is able to manage GSPNs with hundreds of places and transitions, which makes it a very useful tool for Petri net analysis.

8 Conclusion and Related Work

This paper has presented the activities carried out for the performance assessment of the Posidonia Operations system and the results achieved. The goal of the assessment was twofold: first, to find the actions needed for fulfilling the upcoming performance goals for the next versions of Posidonia Operations and, second, to propose and validate a methodology that can be applied by practitioners with tool support. The work has been developed by a joint team of software engineers, from Prodevelop, and researchers, from Unizar.

Detailed results of the performance analysis of the POSIDONIA Operations are available at the Zenodo open source repository: `https://doi.org/10.5281/zenodo.264286`.

Related work The assessment of non-functional (or quality) requirements of software systems is a well-established discipline due to a broad contribution of the software engineering community [9, 11, 33, 14]. Concerning specifications based on UML, most of the approaches define ad-hoc UML profiles to specify requirements and propose transformation methods to get formal models suitable for the analysis (such as Petri nets, queuing networks, fault trees or process algebras). Such efforts, especially the ones addressing performance and schedulability assessment, contributed to the definition of standard OMG UML profiles, concretely MARTE [26].

Although process mining mainly addresses the context of business processes, recent works apply such a discipline to also other domains such as software systems and, in particular, data-intensive software applications. In [24] a discovery technique is proposed that extracts constraint-based reference models from logs generated by a maritime safety and security system. The work also proposes an on-line monitoring technique to detect constraint violations of the reference model. The use of process mining for data-intensive applications is nowadays challenging, since there is a need of efficient and highly scalable techniques [2] to deal with event logs of several hundreds of gigabytes. Some contributions have been proposed to address this issue, such as [21] where a framework has been developed to enable the execution of Map Reduce-based process mining tasks.

A CEP input parameters: refined performance values

In this appendix, refined performance results from the trace-driven simulation of the CEP execution cycle model are provided. In particular, Table 12 shows the estimated mean durations of the CEP rules activation and AIS events publication for each event log.

Table 13 shows the relative frequencies of the rule updating and activation, and of the AIS events publication for each event log.

B Generalized Stochastic Petri Nets

In this appendix, we introduce the modelling formalism of Generalized Stochastic Petri Nets (GSPN).

A Generalized Stochastic Petri net (GSPN) [8] is a bipartite graph, formally defined as a 8-tuple $\mathcal{N} = (P, T, I, O, H, \Phi, W, M_0)$ where:

- P is the set of places,
- $T = T_i \cup T_t$ is the set of transitions, divided into *immediate* (T_i) and *timed* (T_t) transitions,
- $I, O, H : P \times T \to \mathbb{N}$ are, respectively, the input, output and inhibitor arc multiplicity functions,
- $\Phi : T \to \mathbb{N}$ assigns a priority to each transitions: timed transitions have zero priority, while immediate transitions have priority greater than zero,

Action ($\mathcal{S}$)	Event type ($\mathcal{EL}$)	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
		hostDemand (ms)		
E_RuleX	Simple_Dock_Start_Out	0.293333	0.560000	0.668000
	Simple_Dock_Stop	0.384433	0.393895	0.409126
	Stop_Over_Out	0.404627	0.389558	0.389563
	Stop_Over_In	0.400189	0.364337	0.389563
	Simple_Anchor_In	0.523323	0.390454	40.682050
	Simple_Anchor_Out	0.500000	0.500000	0.500000
E_RuleY	Simple_Dock_Start_Out	-	-	-
	Simple_Dock_Stop	0.415221	0.296957	0.344811
	Stop_Over_Out	0.362478	0.446087	0.333208
	Stop_Over_In	0.456283	0.365652	0.317642
	Simple_Anchor_In	0.442737	0.376279	9.965955
	Simple_Anchor_Out	-	-	-
E_PublishAISEvent	Dock_Start_Out_Publish	1.777778	2.170000	2.857143
	Dock_Stop_Publish	3.170000	3.000000	4.843077
	Anchor_In_Publish	1.833333	3.000000	4.000000
	Anchor_Out_Publish	1.000000	-	4.000000

Table 12 Mean durations for rule activation and AIS event publication

- $W : T \to \mathbb{R}$ assigns to each immediate transition a weight, and to each timed transition a firing time delay. The firing time delay is the mean value of the negative exponential distribution,
- $M_0 : P \to \mathbb{N}$ assigns the initial number of tokens to each place.

Figures 4 and 5 show the graphical representation of two GSPN models, where places are depicted as circles, immediate transition as thin black bars and timed ones as thick white bars.

Transitions of a GSPN model represent actions or events in the modelled system, whereas places represent pre- and post-conditions for the actions/events occurrence. The dynamic of a GSPN model is governed by the concession, enabling and firing rules of transitions in a marking $M : P \to \mathbb{N}$, which is reachable from the initial marking M_0 due to the firing of a sequence of transitions.

A transition *has concession* in a marking M, when its input places contain at least as many tokens as the corresponding arc multiplicities and its inhibitor places contain less tokens than the corresponding arc multiplicities.

A transition is *enabled* in a marking M iff it has concession in M and its priority is greater or equal to the one of the transitions t' having also concession in M.

Consequently, only transitions of the same priority level can be enabled in a marking. A transition t, enabled in marking M, may fire then leading to a new marking M', according to the equation:

$$M'(p) = M(p) + O(p,t) - I(p,t), p \in P.$$

Acknowledgements This work has been supported by the European Commission under the H2020 Research and Innovation program [DICE, Grant Agreement No. 644869], the Spanish Ministry of Economy and Competitiveness [ref. CyCriSec-TIN2014-58457-R], and the Aragonese Government [ref. T94, DIStributed COmputation (DISCO)].

References

1. Van der Aalst W (2011) Process Mining - Discovery, Conformance and Enhancement of Business Processes. Springer, DOI 10.1007/978-3-642-19345-3, URL `http://dx.doi.org/10.1007/978-3-642-19345-3`

Action ($\mathcal{S}$)	Event type ($\mathcal{EL}$)	$\mathcal{EL}_1$	$\mathcal{EL}_2$	$\mathcal{EL}_3$
		Relative frequencies		
E_RuleX	Simple_Dock_Start_Out	0.000146	0.000100	0.000167
	Simple_Dock_Stop	3.998353	3.996427	3.997944
	Stop_Over_Out	0.999597	0.999107	0.999486
	Stop_Over_In	0.999597	0.999107	0.999486
	Simple_Anchor_In	0.926237	0.906615	0.906489
	Simple_Anchor_Out	0.000024	0.000045	0.000067
E_RuleY	Simple_Dock_Start_Out	-	-	-
	Simple_Dock_Stop	0.002757	0.004109	0.004739
	Stop_Over_Out	0.000689	0.001027	0.001185
	Stop_Over_In	0.000689	0.001027	0.001185
	Simple_Anchor_In	0.000580	0.000960	0.000995
	Simple_Anchor_Out	-	-	-
E_PublishAISEvent	Dock_Start_Out_Publish	0.000055	0.000067	0.000078
	Dock_Stop_Publish	0.000073	0.000045	0.000145
	Anchor_In_Publish	0.000037	0.000022	0.000022
	Anchor_Out_Publish	0.000012	-	0.000045
S_UpdateActiveFromW	Update_Active_From_Simple_Dock_Start_Out	0.000055	0.000045	0.000078
	Update_Active_From_Simple_Dock_Stop	0.000073	0.000067	0.000145
	Update_Active_From_Insert_Ship	0.000512	0.000893	0.000894
	Update_Active_From_Simple_Anchor_In	0.000037	0.000045	0.000022
	Update_Active_From_Simple_Anchor_Out	0.000012	-	0.000045

Table 13 Relative frequencies for rule updating and activation, and AIS event publication

2. Van der Aalst W (2014) Business Intelligence: Third European Summer School, eBISS 2013, Dagstuhl Castle, Germany, July 7-12, 2013, Tutorial Lectures, Springer International Publishing, chap Process Mining in the Large: A Tutorial, pp 33–76
3. Van der Aalst W, Adriansyah A, van Dongen B (2012) Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 2(2):182–192
4. Adriansyah A, Buijs J (2012) Mining process performance from event logs: the BPI Challenge 2012 case study. BPM reports, no. 1215 – Url: `http://repository.tue.nl/91892e24-93b6-4b38-bae5-e4394e171bf7`

5. AISencoding (2012) Automatic Identification System - Encoding Guide. Url: http://www.uscg.mil/hq/cg5/TVNCOE/Documents/links/AIS.EncodingGuide.pdf, accessed 04/29/2016
6. AISreceiver (2016) Installation and Quick Reference Guide - SRL-200/G AIS Receiver. Url: http://www.comarsystems.com/brochures/Installation%20SLR200G%20Guide%20.pdf, accessed 04/29/2016
7. AIVM/AIVDO (2015) AIVDM/AIVDO protocol decoding. Url: http://catb.org/gpsd/AIVDM.html, accessed 04/29/2016
8. Ajmone-Marsan M, Balbo G, Conte G, Donatelli S, Franceschinis G (1994) Modelling with Generalized Stochastic Petri Nets, 1st edn. John Wiley & Sons, Inc., New York, NY, USA
9. Balsamo S, Di Marco A, Inverardi P, Simeoni M (2004) Model-based performance prediction in software development: A survey. IEEE Trans Softw Eng 30(5):295–310
10. Bernardi S, Campos J, Merseguer J (2011) Timing-Failure Risk Assessment of UML Design Using Time Petri Net Bound Techniques. IEEE Trans Industrial Informatics 7(1):90–104
11. Bernardi S, Merseguer J, Petriu DC (2012) Dependability modeling and analysis of software systems specified with uml. ACM Comput Surv 45(1):1–48
12. Bugzilla (2017) Papyrus Bug List. Url: https://bugs.eclipse.org/bugs/buglist.cgi?bug_status=UNCONFIRMED&bug_status=NEW&bug_status=ASSIGNED&bug_status=REOPENED&bug_status=VERIFIED&classification=Modeling&component=Diagram&f1=short_desc&f2=short_desc&o1=substring&o2=substring&product=Papyrus&query_format=advanced&v1=Sequence&v2=Diagram – last accessed Jan. 2017
13. Casale G, Ardagna D, Artac M, Barbier F, Nitto ED, Henry A, Iuhasz G, Joubert C, Merseguer J, Munteanu VI, Perez JF, Petcu D, Rossi M, Sheridan C, Spais I, Vladuic D (2015) DICE: quality-driven development of data-intensive cloud applications. In: 7th IEEE/ACM International Workshop on Modeling in Software Engineering, MiSE 2015, Florence, Italy, May 16-17, 2015, pp 78–83
14. Cortellessa V, Marco AD, Inverardi P (2011) Model-Based Software Performance Analysis, 1st edn. Springer Publishing Company, Incorporated
15. DICE-D1.2 (2015) Requirement Specification – Companion Document. Url: https://www.dice-h2020.eu/deliverables/
16. Dipartimento di informatica, Università di Torino (2016) GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. http://www.di.unito.it/~greatspn/index.html - accessed 01/05/2016.
17. Fluxicon (2017) Fluxicon process mining for professionals. Url: https://fluxicon.com/disco/
18. Foundation TE (2017) Eclipse - The Eclipse Foundation open source community website. URL: https://eclipse.org/, last accessed Jan. 2017
19. Gómez A, Joubert C, Merseguer J (2016) A Tool for Assessing Performance Requirements of Data-Intensive Applications. In: Proc. of the XXIV National Conference of Concurrency and Distributed Systems (JCDS 2016), pp 159–169, Url: https://github.com/dice-project/DICE-Simulation, accessed 01/23/2017
20. Gómez-Martínez E, Gonzalez-Cabero R, Merseguer J (2014) Performance assessment of an architecture with adaptative interfaces for people with special needs. Empirical Software Engineering 19(6):1967–2018, DOI 10.1007/s10664-013-9297-1, URL http://dx.doi.org/10.1007/s10664-013-9297-1
21. Hernández S, van Zelst SJ, Ezpeleta J, van der Aalst W (2015) Handling big(ger) logs: Connecting ProM 6 to Apache Hadoop. In: Proc. of the BPM Demo Session 2015, CEUR-WS.org, vol 1418, pp 80–84
22. Joubert C, Montesinos M, Sanz J (2014) A comprehensive port operations management system. ERCIM News 2014(97), Url: http://ercim-news.ercim.eu/en97/special/a-comprehensive-port-operations-management-system
23. López-Grao J, Merseguer J, Campos J (2004) From UML activity diagrams to Stochastic Petri nets: application to software performance engineering. In: Proceedings of the Fourth International Workshop on Software and Performance, WOSP 2004, Redwood Shores, California, USA, January 14-16, 2004, pp 25–36

24. Maggi FM, Mooij AJ, Van der Aalst W (2013) Analyzing vessel behavior using process mining. In: van de Laar P, Tretmans J, Borth M (eds) Situation Awareness with Systems of Systems, Springer, pp 133–148, DOI 10.1007/978-1-4614-6230-9
25. NIST/SEMATECH (2013) e-Handbook of Statistical Methods. http://www.itl.nist.gov/div898/handbook/. Accessed 10/15/2016.
26. OMG (2011) UML Profile for MARTE: Modeling and Analysis of Real-time Embedded Systems, Version 1.1. Tech. rep., Object Management Group, Url: http://www.omg.org/spec/MARTE/1.1/
27. Prodevelop (1993) Prodevelop- Integrating Tech. Url: https://www.prodevelop.es/en
28. ProM Tools (2017) ProM Tools. Url: http://www.promtools.org/doku.php
29. Robinson S (1997) Simulation Model Verification and Validation: Increasing the Users' Confidence. In: Proceedings of the 29th Conference on Winter Simulation, IEEE Computer Society, WSC '97, pp 53–59
30. Runeson P, Höst M (2008) Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14(2):131, DOI 10.1007/s10664-008-9102-8, URL http://dx.doi.org/10.1007/s10664-008-9102-8
31. Sargent R (2010) A new statistical procedure for validation of simulation and stochastic models. Tech. rep., Department of Electrical Engineering and Computer Science, Syracuse University, technical report SYR-EECS-2010-06
32. Shavor S, D'Anjou J, Fairbrother S, Kehn D, Kellerman J, McCarthy P (2003) The Java Developer's Guide to Eclipse. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA
33. Smith CU (1990) Performance Engineering of Software Systems, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA
34. Smith CU, Williams LG (2002) Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA
35. The Eclipse Foundation (2016) Papyrus. Url: https://eclipse.org/papyrus/
36. Van Dongen BF, et al (2005) The ProM framework: A new era in process mining tool support. In: Applications and Theory of Petri Nets 2005, Springer, pp 444–454
37. XES (2016) Extensible Event Stream. IEEE Task FOrce on Process Mining, [Online; accessed 18-April-2016]