# A systematic approach for performance evaluation using process mining: the POSIDONIA Operations case study

Simona Bernardi
Centro Universitario de la Defensa, AGM (Spain)
simonab@unizar.es

José Ignacio Requeno
Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza (Spain)
nrequeno@unizar.es

Christophe Joubert, Alberto Romeu
Prodevelop SL (Spain)
cjoubert@prodevelop.es
aromeu@prodevelop.es

## ABSTRACT

Modelling plays an important role in the development of software applications, in particular for the assessment of non-functional requirements such as performance. The value of a model depends on the level of alignment with the reality.

In this paper, we propose a systematic approach to get a performance model that is a *good* representation of the system under analysis. From an UML-based system design we get automatically a *normative* Petri net model, which formally represents the system supposed behavior, by applying model-to-model (M2M) transformation techniques. Then, a conformance checking technique is iteratively applied to align -from the qualitative point of view- the normative model and the data log until the required fitness threshold is not reached. Finally, a trace-driven simulation technique is used to enrich the aligned model with timing specification from the data log, then obtaining the performance Generalized Stochastic Petri Net (GSPN) model.

The proposed approach has been applied to a customizable Integrated Port Operations Management System, POSIDONIA Operations, where the performance model has been used to analyze the scalability of the product considering different deployment configurations.

## Keywords

Model-driven transformation, process mining, trace and log analysis, performance, Generalized Stochastic Petri Nets (GSPN), Unified Modeling Language (UML), data-intensive application.

## 1. INTRODUCTION

Modelling plays an important role in the development of software applications, in particular for the assessment of non-functional requirements -such as performance, reliability, safety and security. Essentially, two types of models can be used: informal and formal. The former (e.g., Unified Modeling Language) are used by the software engineers for requirement/design/test specification, while the latter (e.g. Petri nets) are more suitable for the analysis.



Regardless of the type, the value of a model depends on level of alignment with the reality. For example, it is useless and risky to conduct simulation experiments to analyze the scalability of the system using a model that represents an idealized version of the actual behavior, since -based on the analysis results- incorrect design decisions could be made.

Process mining [24] is a relatively new discipline that builds on model-driven approaches and data mining. It aims at providing methods, techniques and tools for the construction of models aligned with the reality, considering system execution traces (i.e., logs). Although lots of process mining techniques have been proposed and several tools are available, their usage still requires expertise in formal modelling and analysis. Therefore, they cannot be considered as straightforward solutions from the perpective of software engineers, who are usually more acquainted with informal models.

In this paper, we propose a systematic approach to get a *good* performance model, where the quality of the model is based on the fitness estimation of the model with the system data log. We assume to have an UML-based design, that includes the specification of a (set of) system execution scenario(s) and the resource restrictions, and the data log, that includes a set of execution traces obtained from either testing or production environments. By applying model-to-model (M2M) transformation techniques, we get automatically a Petri net model from the UML specification, namely a *normative* model which formally represents the system supposed behavior. Then, a conformance checking technique is iteratively applied to align the normative model and the data log from the qualitative point of view, (i.e., that is considering only the event occurrences and neglecting the timing information) until the required fitness threshold is not reached. Finally, a trace-driven simulation technique is used to enrich the aligned model with timing specification from the data log, then obtaining the performance model, i.e., a Generalized Stochastic Petri Net (GSPN) model [19].

The proposed approach has been applied to a customizable Integrated Port Operations Management System, POSIDONIA Operations, one of the data-intensive applications selected as demonstrator of the DICE project [10].

The paper is organized as follows: Section 2 considers the related work. Section 3 introduces the Posidonia case study. Section 4 describes the approach overview. Sections 5 and 6 describe in detail the application of the conformance checking and trace-driven simulation techniques, respectively, on the case study. Finally, Section 7 provides conclusion and future work.

## 2. RELATED WORK

Model-driven engineering aimed at assessing non-functional (or

quality) requirements of software systems is a well-established discipline due to the broad contribution provided by the software engineering community in the last decade [5, 7, 16]. Concerning UML-based software specification, most of the approaches define ad-hoc UML profiles to specify non-functional properties (NFPs) and propose transformation methods to get formal models (such as Petri nets, queuing networks, fault trees, process algebras, etc.) suitable for the analysis. Such efforts, especially the ones addressing performance and schedulability assessment, contributed to the definition of standard OMG UML profiles, such as MARTE [21].

In this paper, we rely on the transformation approach [17] that generates a GSPN model from UML Activity Diagrams (AD). The approach has been enhanced to consider resource restrictions and the MARTE performance extensions, and has been implemented in the DICE-simulation tool [13].

On the other hand, process mining [24] mainly 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 data input for process mining may come from different sources and need to be pre-processed to get *event logs*, i.e., structured data related to ordered events occurring within a process.

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 [18] 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 [1] to deal with event logs of several hundreds of gigabytes. Some contributions have been proposed to address this issue, such as [14] where a framework has been developed to enable the execution of Map Reduce-based process mining tasks.

Knowing if the runtime behaviour of a system is compliant with the original design specification needs a deep and exhaustive exploration of all the potential behaviors. One of the goals of this paper is the detection of deviations between the execution traces and the normative model indicating errors. We apply conformance checking for this purpose, however other approaches have been proposed that rely on equivalence checking. Equivalence checking is a formal process for proving that two representations -usually expressed as Labelled Transition Systems (LTSs)- of a system exhibit exactly the same functionality. The comparison of two LTSs requires the definition of an equivalence relation between the states of the models or between the trace (language) of actions that they may execute. Different kind of equivalence relations are provided in the literature depending on whether they focus on models or traces [6]: strong [22], branching [26], observational [20], tau [12] or safety equivalence [8] for models, and trace or weak trace equivalence [9] for languages.

# 3. THE POSIDONIA CASE STUDY

POSIDONIA Operations [15] is a customizable Integrated Port Operations Management System that allows a port to optimize the operational maritime activities related to the vessel flow within the service area of the port, integrating all the involved stakeholders and all the relevant information systems.

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, cartography, etc.

POSIDONIA Operations is 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 fulfils port operations, including berth planning, coordination and register of pilots, tugs and moorers activities, vessel supplies and bunkering, wastes & disposal, incidents, repairs, port inner traffic, etc.

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, port exit with pending requested anchoring, etc.

## 3.1 Architecture

POSIDONIA Operations is a data-intensive application (DIA) implemented in Java. It processes streamed data from Automatic Identification System (AIS) receivers [2, 3], a system that gets vessels position and meta-data in real time. The encoding protocol of an AIS sentence can be found in [4].

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 for further processing: analysis, complex event processing, 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. Velocity and volume of data depends on the number of parallel AIS streams to be processed.

The core components of interest for performance analysis are: 1) a streaming processor -or AIS parser- that collects the data from the AIS receiver and parses it (Figure 1 shows the main scenario); 2) a message queue for subscribing/publishing data such as AIS messages or detected events; and 3) a complex event processing (CEP) engine that subscribes to AIS messages and correlates them in time and space to identify events.

## 3.2 Test Scenarios

POSIDONIA Operations is a commercial product already deployed and operated in several port authorities. An example of deployment of the AIS parser of Balearic Islands and the CEP of one port area (Palma) is given in Figure 2.

Being a product already in production, we have to ensure its performance under different velocity and volume of data to be processed. For a single area of a port, a velocity of about hundred 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 port areas managed by the product for a given Port Authority. Several instances of the CEP would then need to be instantiated, one for each port area. In this case, one of the challenges is related to the scalability of the product in terms of data processing, storage and analysis.

# 4. APPROACH OVERVIEW

We aim at deriving a formal model, concretely a GSPN model, amenable to be used for performance predictions. To that end, we propose the systematic approach summarized by the Algorithm 1.

The input specification consists of: 1) a UML-based design that includes a (set of) Activity Diagram(s) $\mathcal{AD}$, which represent the execution process(es) of a data-intensive application -such as the

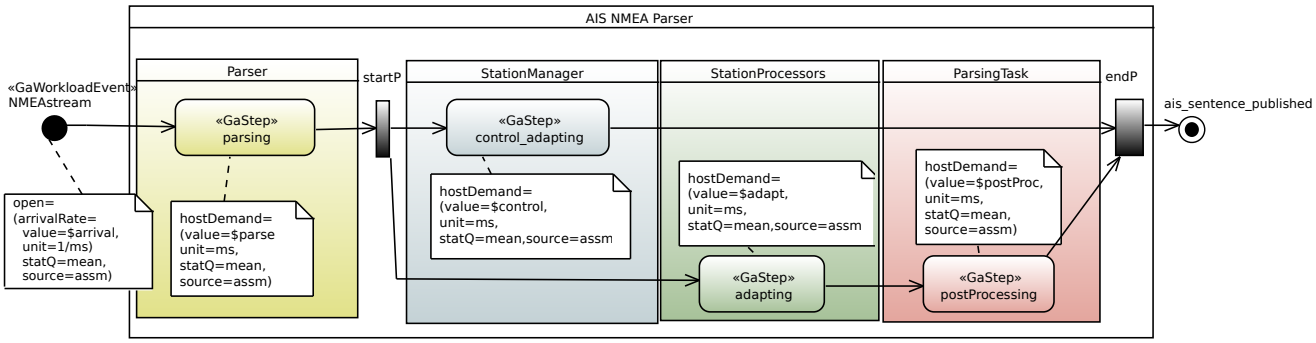



**Figure 1: Parsing scenario.**

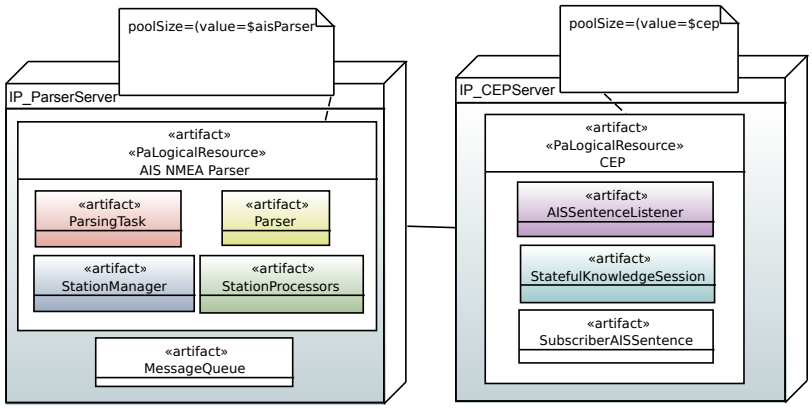



**Figure 2: Deployment.**

**Algorithm 1** Approach

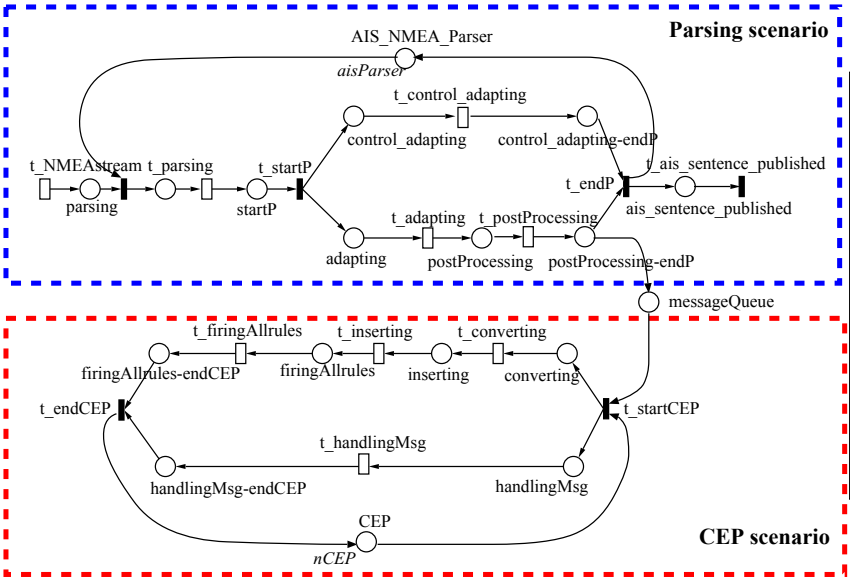
**Input:** UML design ($\mathcal{AD}$, $DD$), data log ($\mathcal{L}$)
**Output:** Performance model ($\mathcal{GSPN}$) & results ($\mathcal{R}$)
1: Get a normative model $\mathcal{N}$ from $\mathcal{AD}$
2: Pre-process data log to get event log $\mathcal{EL}$
3: **repeat**
4: Filter $\mathcal{EL}$
5: Check for conformance $\mathcal{N}$ and $\mathcal{EL}$
6: **until** fitness $\geq thres$
7: Enhance $\mathcal{N}$ with timing perspective: $\mathcal{GSPN}$
8: Performance analysis with $\mathcal{GSPN}$: $\mathcal{R}$

parsing scenario of POSIDONIA shown in Figure 1- and a Deployment Diagram $\mathcal{DD}$, which specifies the software component allocation on computing nodes, such as the DD of Figure 2; and 2) the data log $\mathcal{L}$ which include a set of process execution traces.

In the first step (Step 1), a Petri Net model $\mathcal{N}$ is automatically derived from each $\mathcal{AD}$ and the $\mathcal{DD}$ via a model-to-model (M2M) transformation [17], using the DICE simulation tool [13]. This Petri Net model represents the *normative* model since it is derived from the known behavioural specification of the system. Observe that the AD in Figure 1 is annotated using the MARTE profile; in particular, input parameters are assigned to the mean durations of the action steps (i.e., *hostDemand* tagged-values) and to the data stream arrival rate (i.e., *arrivalRate* tagged-value). Figure 3 shows the two Petri net subnets, in the dotted rectangles, that are derived from the parsing scenario of Figure 1 and a CEP scenario[1], considering the (logical) resource restrictions specified in the DD of Figure 2 (*poolSize* tagged-values).

The next step (Step 2) consists in pre-processing the data log $\mathcal{L}$ to convert it into the *event log* XES [27] standard format, where each execution trace is characterized by an ordered set of event occurrences together with their timestamps. The data logs of POSIDONIA were collected in separate `.csv` files, 4 files related to the parsing process -one for each parser thread- with a mean number of 69920 traces, and one single file related to the CEP process with a total of 56698 traces. Each parsing trace represents the transformation of an NMEA message -from the AIS receiver of the Balearic Islands- into an AIS sentence and includes 8 event occurrences, which correspond to the start and completion of each action modelled in the AD of Figure 1. Each trace of the CEP process represents the message handling by the CEP of the Palma port. In the original log, the traces contain several event occurrences including the activation and firing of single rules. The pre-processing step produces an event log where only the event occurrences which cor-

[1]The Activity Diagram representing the CEP scenario has been omitted due to space limitation.




| *Transition name* | *Time parameter* | *Value(ms)* |
|---|---|---|
| t_NMEAstream | 1/arrival | 0.022(1/ms) |
| t_parsing | parse | 15.780 |
| t_control_adapting | control | 0.725 |
| t_adapting | adapt | 0.093 |
| t_postProcessing | postProc | 0.237 |
| t_handlingMsg | handle | 136.800 |
| t_converting | convert | 0.160 |
| t_inserting | insert | 12.800 |
| t_firingAllrules | fire | 102.340 |
| *Place name* | *Marking parameter* | *Value* |
| AIS_NMEA_Parser | aisParser | 4 |
| CEP | cep | 1 |

**Figure 3: GSPN model.**

respond to the start and completion of the actions modelled in the AD are considered in order to align it with the latter. Each trace of the CEP, finally includes between 4 and 8 event occurrences; indeed, 5.5% of the traces are partial execution of the CEP scenario.

In the Steps 3-6, the event log and the Petri net model are aligned using the ProM tool [25] in order to reach a fitness threshold (e.g., $thres = 1$ means a 100% alignment between the event log and the model); the alignment may require several iterations. The next Section 5 describes such steps using the case study.

Once the required fitness is reached, then the Petri Net model is enriched with timing specification (Step 7) to get a $\mathcal{GSPN}$ model. In particular, the parameters associated to the timed transitions of the GSPN model are set to values which are estimated using the event log and the trace-driven simulator of the ProM tool. Finally, the GSPN model is used to carry out performance analysis (Step 8). In particular, in the case study the objective of the analysis is to evaluate the scalability of the application. Such steps are detailed in Section 6.

## 5. CONFORMANCE CHECKING

A normative model $\mathcal{N}$ is an abstract representation of the system that includes a set of behavior scenarios, while an event log $\mathcal{EL}$ consists of a set of execution traces each one describing a possible behavior. The normative model does not consider all the possible system behaviors. Similarly, the event log provides information about the running system, but it is not guaranteed that it includes a full description of all the combinations of the environment.

In Steps 3-6 (Algorithm 1), the normative model (without temporal information yet) and the event log are aligned in order to assess whether the normative model is a *good* representative of the event log. To this aim, the execution traces in the event log are replayed on the normative model using the conformance checking algorithm [23], implemented as plugin in the ProM tool [25]. The ProM plugin requires each event of $\mathcal{EL}$ to be mapped to the corresponding transition in $\mathcal{N}$ as well as an initial and final marking to be set for $\mathcal{N}$.

The ProM plugin returns a fitness score that estimates the conformance of the execution traces with respect to the normative model (Step 5). We worked with the default configuration to compute the fitness, that penalizes mismatches with a unitary cost per miss when moving a token on the model but not in the log, or vice versa.

First, we analyzed the normative model of the AIS NMEA Parser (Figure 3, top subnet), where the initial transition *t_NMEAstream* and the immediate ones were considered not observable in the conformance checking because they didn't have a direct correspondence with the events in the log. The mean fitness score of the non-filtered event logs produced by the four parser threads was 0.998, meaning that almost all the traces in the logs can be replayed on the model.

In this case, the application of a filter (Step 4) removed erroneous traces. Indeed, by analyzing the event logs, we found that some traces presented an anomalous initial (or end) event. On average, the 0.025% of the traces (around 18 sequences per parser) started with a wrong event, and the 0.016% of the traces (around 11 sequences per parser) finished in a wrong one. For instance, Figure 4 shows a trace where the *ParsingTask.PostProcessing* starts the execution before the task *Parser.Parsing*. Finally, the fitness score for filtered logs was 1.0.

The second normative model to check for conformance was the CEP scenario (Figure 3, bottom subnet). In this case, the fitness score of the non-filtered event log produced by the CEP was 0.98. By analyzing the event log, we detected some incomplete traces: only the initial part of the CEP scenario was executed (i.e., conversion and handling of the message) but not the insertion and firing of the rules. These traces represented a 5.5% of the log (3065 sequences). The fitness score rose up to 1.0 by removing such incomplete traces.

Finally, with conformance checking we found some situations where the order of certain events of a trace were inverted with respect to the corresponding transitions of the normative model. This happened occasionally in case of events labeled with the same timestamp, since the conformance checker has not enough information for inferring the exact succession. The traces that fall in this last category were not considered as incorrect traces.


**Figure 4: Example of erroneous trace.**

# 6. PERFORMANCE ENHANCEMENT

In order to carry out performance analysis, we need to enrich the aligned normative model with timing specifications, then obtaining a GSPN model (Step 7 of the Algorithm 1). To that end, we use the trace-driven simulator of the ProM tool [25] that replays the event log and computes the mean and standard deviation of the transitions firing times. The table in Figure 3 (right side) shows the mean values obtained for the timed transitions of the two GSPN subnets of POSIDONIA (left side), which resulted by the analysis of the filtered logs (removing less frequent traces - Step 4).

The mean values of the parsing scenario were calculated as the average execution time of the four parser threads. The arrival rate of the data stream (*t_NMEAstream*) was estimated according to the timestamps of the parsing starting events in the logs. All the transition firing times were characterized by the negative exponential distribution, since the standard deviations were similar to the mean values.

Concerning the CEP scenario, transition firing time *t_firingAll rules* had a lower standard deviation value (63.09 ms) and it was approximated by an Erlang distribution with $k = 3$ steps. The rest of transition firing times were all exponentially distributed.

We validaded separately the two GSPN subnets by considering four parser threads and a single CEP (last two rows of the table in Figure 3), and the mean processing time as performance metric of reference. We used both the analytical solver and the event driven simulator of GreatSPN [11]. The relative error of the mean processing time of the parsing with respect to the one inferred by the logs was less than $1\%$, while the relative error of the mean processing time of the CEP was around $10\%$. This is probably due to the abstraction level of the normative model, where the activation and firing of single CEP business rules are not explicitly represented.

Once the GSPN model is validated, it can be used for performance analysis (Step 8 - Algorithm 1). In the case of POSIDONIA, we used the complete GSPN model -representing the parsing and the CEP scenarios, and the message queue- to study the scalability of the system under different assumptions on the deployment environment.

| Parsers (threads) | CEP | Mean arrival time (msg/sec.) |
|---|---|---|
| 1 (4) | 1 | 5 |
| 1 (4) | 1 | 7 |
| 1 (4) | 1 | 8 |
| 1 (4) | 5 | 40 |
| 1 (4) | 7 | 15 |

**Table 1: Deployment basic configurations.**

Table 1 shows the deployment basic configurations that represent the normal situation of one parser (with 4 threads) and one CEP for each geographical area. The number of parsers is related to the number of AIS receptors; each port authority provides the access to a unique AIS receptor. Instead, the number of CEPs depends on the number of ports managed by the port authority, i.e., one CEP for each managed port. The mean arrival time of the data stream ranges between 5 and 40 messages per second. Such values may change between seasons (summer/winter) and daytime (light/night), and can be multiplied up to a factor of 10.

Figure 5 shows some performance results obtained via simulation with the GreatSPN tool[2]. The curve on the left shows the parser utilization vs/ the mean arrival time of the data stream: the resource becomes saturated when the data stream rises to 280 msg/sec. The curves on the right show the trend of the mean processing time of the CEP for the three different values vs/ the mean arrival time. For one CEP, the system is not stable for the considered message workload: indeed the size of the message queue increases, the CEP utilization is $100\%$ and the processing time ranges in $[138, 153]$ ms. In the other two configurations, the processing time is maintained in the range $[136, 139]$ ms. and there is not significant difference between using five or seven CEPs.

# 7. CONCLUSION

We have proposed a systematic approach to get a performance model by applying M2M transformation and process mining techniques. The former enables to automatically obtain a *normative* model from UML-based design specifications, while the latter is used to get a performance model that is a good representation of the system under analysis, by considering the data logs from system executions. The approach has been exemplified with the POSIDONIA Operations case study, where the performance model has been used to analyze the scalability of the product considering different deployment configurations.

The size of the POSIDONIA event logs, analyzed in this work, is of the order of 100 megabytes. As future work, we aim at investigating the efficiency and scalability of the current available process mining techniques when *big data* logs are considered. On the other hand, the case study deserves an in-depth analysis since the developers are interested also in other performance and reliability issues, such as to figure out the impact of a new CEP business rule on the quality metrics.

## Acknowledgment

Special thanks to Juan Lucas Domínguez and Abel Gómez for the enlightenments about, respectively, the case study and the DICE-simulation tool. 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. T27, DIStributed COmputation (DISCO)].

# 8. REFERENCES

[1] W. M. Aalst. *Business Intelligence: Third European Summer School, eBISS 2013, Dagstuhl Castle, Germany, July 7-12, 2013, Tutorial Lectures*, chapter Process Mining in the Large: A Tutorial, pages 33–76. Springer International Publishing, 2014.

[2] Automatic Identification System - Encoding Guide, 2012. `http://www.uscg.mil/hq/cg5/TVNCOE/`

[2] A $99\%$ confidence level and a $3\%$ accuracy were set for all the experiments.

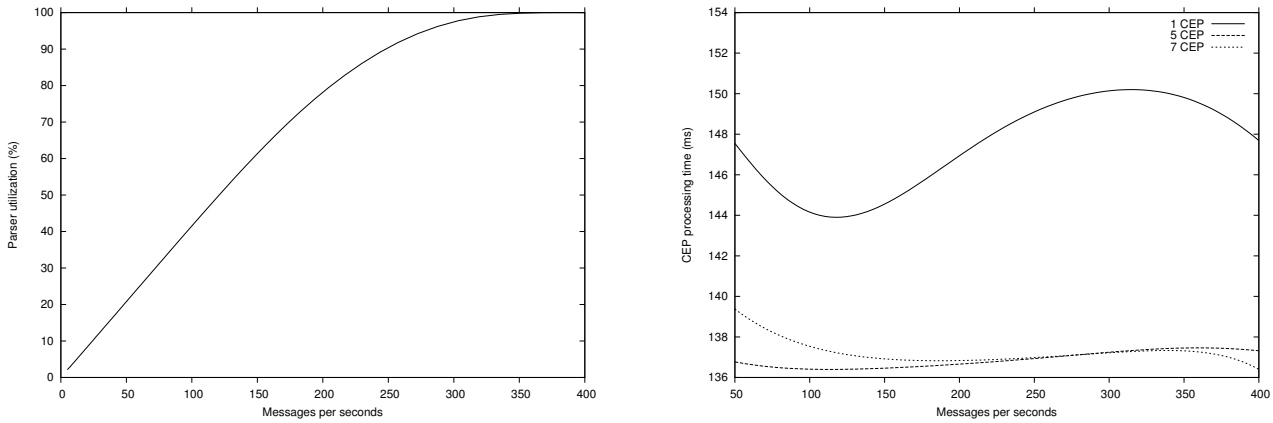



**Figure 5: Performance results.**


Documents/links/AIS.EncodingGuide.pdf, accessed 04/29/2016.

[3] *Installation and Quick Reference Guide - SRL-200/G AIS Receiver*. http://www.comarsystems.com/brochures/Installation%20SLR200G%20Guide%20.pdf, accessed 04/29/2016.

[4] *AIVDM/AIVDO protocol decoding*, 2015. http://catb.org/gpsd/AIVDM.html, accessed 04/29/2016.

[5] S. Balsamo, A. Di Marco, P. Inverardi, and M. Simeoni. Model-based performance prediction in software development: A survey. *IEEE Trans. Softw. Eng.*, 30(5):295–310, May 2004.

[6] D. Bergamini, N. Descoubes, C. Joubert, and R. Mateescu. BISIMULATOR: A modular tool for on-the-fly equivalence checking. In *Proceedings of TACAS 2005, held within ETAPS 2005, Edinburgh, UK, April 4-8*, pages 581–585, 2005.

[7] S. Bernardi, J. Merseguer, and D. C. Petriu. Dependability modeling and analysis of software systems specified with uml. *ACM Comput. Surv.*, 45(1):1–48, Dec. 2012.

[8] A. Bouajjani, J.-C. Fernandez, S. Graf, C. Rodriguez, and J. Sifakis. Safety for branching time semantics. In *Automata, Languages and Programming*, pages 76–92. Springer, 1991.

[9] S. D. Brookes, C. A. Hoare, and A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM (JACM)*, 31(3):560–599, 1984.

[10] G. Casale et al. Dice: Quality-driven development of data-intensive cloud applications. In *Proceedings of MiSE'15*, pages 78–83, Piscataway, NJ, USA, 2015. IEEE Press.

[11] Dipartimento di informatica, Università di Torino. GRaphical Editor and Analyzer for Timed and Stochastic Petri Nets. http://www.di.unito.it/~greatspn/index.html - accessed 01/05/2016.

[12] J.-C. Fernandez and L. Mounier. "on the fly" verification of behavioural equivalences and preorders. In *Computer Aided Verification*, pages 181–191. Springer, 1991.

[13] A. Gómez. DICE simulation tools - initial version (v1.1). Technical report, DICE Project no. 644869 - H2020, February 2016. http://www.dice-h2020.eu/deliverables/.

[14] S. Hernández, S. J. van Zelst, J. Ezpeleta, and W. M. P. van der Aalst. Handling big(ger) logs: Connecting ProM 6 to Apache Hadoop. In *Proc. of the BPM Demo Session 2015*, volume 1418, pages 80–84. CEUR-WS.org, 2015.

[15] C. Joubert, M. Montesinos, and J. Sanz. A comprehensive port operations management system. *ERCIM News*, 2014(97), 2014.

[16] J. Jürjens. *Secure Systems Development with UML*. Springer-Verlag, Berlin, Heidelberg, 2010.

[17] J. P. López-Grao, J. Merseguer, and J. Campos. From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering. *SIGSOFT Softw. Eng. Notes*, 29(1):25–36, Jan. 2004.

[18] F. M. Maggi, A. J. Mooij, and W. M. P. van der Aalst. Analyzing vessel behavior using process mining. In P. van de Laar, J. Tretmans, and M. Borth, editors, *Situation Awareness with Systems of Systems*, pages 133–148. Springer, 2013.

[19] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[20] R. Milner. *Communication and concurrency*, volume 84. Prentice hall New York etc., 1989.

[21] OMG. *A UML profile for Modeling and Analysis of Real Time Embedded Systems (MARTE)*. Object Management Group, 2011. Document formal/11-06-02.

[22] D. Park. *Concurrency and automata on infinite sequences*. Springer, 1981.

[23] W. Van der Aalst, A. Adriansyah, and B. van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[24] W. M. P. van der Aalst. *Process Mining - Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011.

[25] B. F. Van Dongen et al. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005*, pages 444–454. Springer, 2005.

[26] R. J. Van Glabbeek and W. P. Weijland. Branching time and abstraction in bisimulation semantics. *Journal of the ACM (JACM)*, 43(3):555–600, 1996.

[27] Extensible Event Stream. IEEE Task FOrce on Process Mining, [Online; accessed 18-April-2016].