Performance Modelling of Intelligent Transportation Systems: Experience Report

Lorenzo Pagliari
Gran Sasso Science Institute, Italy
lorenzo.pagliari@gssi.it

Mirko D'Angelo
Linnaeus University, Sweden
mirko.dangelo@lnu.se

Mauro Caporuscio
Linnaeus University, Sweden
mauro.caporuscio@lnu.se

Raffaela Mirandola
Politecnico di Milano, Italy
raffaela.mirandola@polimi.it

Catia Trubiani
Gran Sasso Science Institute, Italy
catia.trubiani@gssi.it

ABSTRACT

Modern information systems connecting software, physical systems and people, are usually characterized by high dynamism. These dynamics introduce uncertainties, which in turn may harm the quality of systems and lead to incomplete, inaccurate, and unreliable results. To deal with this issue, in this paper we report our incremental experience on the usage of different performance modelling notations while analyzing Intelligent Transportation Systems. More specifically, Queueing Networks and Petri Nets have been adopted and interesting insights are derived.

CCS CONCEPTS

• **Software and its engineering → Software performance**.

KEYWORDS

Intelligent Transportation Systems; Petri Nets;
Model-based Performance Analysis.

ACM Reference Format:
Lorenzo Pagliari, Mirko D'Angelo, Mauro Caporuscio, Raffaela Mirandola, and Catia Trubiani. 2021. Performance Modelling of Intelligent Transportation Systems: Experience Report. In *Companion of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21 Companion), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3447545.3451205

1 INTRODUCTION

Digitalization of industry, by many considered as the fourth industrial revolution, is changing the competitive landscape in several business domains. The connectivity between software and physical systems opens up for new innovative business or mission critical services responsible for a vast part of the value chain. Indeed, modern information systems (e.g., intelligent transportation systems, smart power grids, network infrastructures and robotics) usually connect software, physical systems, and people [16], who either interact with or are part of the system itself.

ICPE '21 Companion, April 19–23, 2021, Virtual Event, France

ACM ISBN 978-1-4503-8331-8/21/04…$15.00
https://doi.org/10.1145/3447545.3451205

Such modern systems are usually characterized by high dynamism, as participating and interacting entities are heterogeneous and autonomous, and unexpected and uncontrolled conditions may arise within the environment. These dynamics introduce uncertainties, which in turn may harm the quality of system and lead to incomplete, inaccurate, and unreliable results [5]. Managing uncertainty is then crucial to operate modern and complex interacting systems and satisfy their quality requirements.

To this end, it is key to devise an engineering approach for modern systems that guides their development taking into account quality aspects under uncertain conditions. Indeed, in this paper we focus on a specific aspect: investigating how to validate the quality of different design decisions before putting the system into operation, e.g., in terms of specific quality indicators, such as the system response time, resource utilization, service throughput, etc. The parametric analysis of the system under investigation through the use of analytic models and the sensitivity analysis on some of their key parameters allow us to take into account some of the system' uncertainty aspects.

Classical engineering approaches for quality engineering may be not suitable to deal with complex interacting systems. In this paper, we investigate the usage of different performance modelling notations, i.e., Queueing Networks [10] and Petri Nets [20], and their analysis results. We make use of an Intelligent Transportation System (ITS) running example, which is used to show the challenges related to the software performance engineering domain. ITS envisions autonomous cars adapting their behavior to improve the performance (i.e., travel time) of special vehicles crossing a road section. We build on the work presented in [15] and we show how to define the performance models and what to extract from their analysis results, with the goal to either filter out or deepen particular solutions – e.g., the ones achieving poor performances. We discuss then the model-based performance analysis results obtained with the different formalisms highlighting their strengths and weaknesses in this application context.

The rest of this paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce the ITS running example. Section 4 illustrates the performance modelling, and Section 5 explains the role of model-based performance analysis results, and briefly discusses threats to validity. Section 6 concludes the paper with hints for future research.

2 RELATED WORK

During the system engineering life-cycle it is fundamental to analyze the behavior of the system under investigation. In particular, it is of key relevance to understand how the designed software alternatives impact the Quality-of-Service (QoS) requirements. Two types of analysis can be performed: one which is driven by analytical models, and one emulating the actual system behavior through simulation. The former is typically performed at design time and aims at quantifying as early as possible the QoS characteristics of the systems with analytical and/or QoS-based analysis techniques [1]. The latter is usually used when the resulting system behavior is too complex to be captured by theoretical techniques and more detailed models of the system are introduced to get meaningful QoS-based results. In the following we present the state-of-the-art in the direction of QoS-based design analysis.

For the majority of modern systems a satisfactory and omni-comprehensive analysis is highly impractical or even impossible to perform at design time. In fact, in this stage, the software engineer has to verify a complex system with respect to a set of requirements, and there is often no need to consider the precise structure of the system and the details of its elements [11]. When the QoS requirements are not tied to the concrete behavior/execution of the system, high-level QoS models can be selected to preliminarily assess the designed system. Analytical models can be adopted in this phase depending on the specific domain of the system under investigation and the type of QoS requirements to validate. A large body of analysis techniques have been used in the literature to deal with QoS-based validation – e.g., well-known analytical QoS models adopted at design-time are Queuing Networks [9], Petri Nets [14] and Stochastic Automata Networks [22].

In the following, we focus on static-based approaches related to the domain we are considering, i.e., intelligent transport systems. Stankova et. al. [21] deal with the problem of freeway traffic density estimation for a jump Markov linear model by using a method based on particle filtering. For the sake of simplicity, similarly to our running example of Section 3, their study has been performed for a freeway segment without on-ramps and off-ramps sections. Khazaei et. al. [8] propose an analytical technique based on an approximate Markov chain model for performance evaluation of a cloud computing center. Di Febbraro et al. [4] uses hybrid Petri nets to model and solve the problem of coordinating several traffic lights with the aim of improving the performance of some classes of special vehicles crossing a road section. Finally, Mokdad et al. [13] conduct the performance evaluation of a mobile IP reservation protocol by using stochastic automata networks.

3 CASE STUDY

In this section we present a running example on a concrete case study that is purposely simple but not simplistic. The presented scenario, despite its simplicity, encompasses the characteristics of complexity and dynamicity typical of smart CPS. The case study belongs to the transport domain, specifically to the so called Intelligent Transportation System (ITS). This type of environment is defined *intelligent* because it is supported by some form of intelligence, i.e., control logic added to the system itself or to its components, in order to optimize its overall quality, achieve particular goals in certain circumstances or fulfil specific requirements.

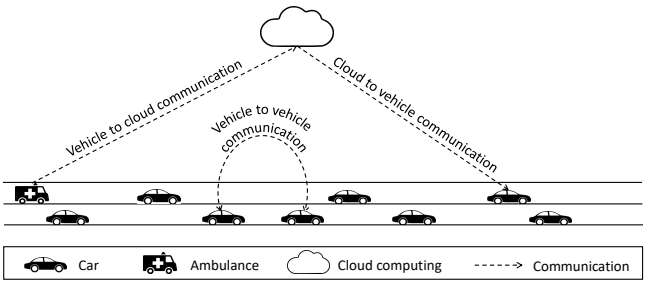


Figure 1: Intelligent Transportation System scenario

Figure 1 illustrates an ITS scenario for a particular road section. For the sake of simplicity, similarly to [21], our case study considers a freeway segment without on-ramps and off-ramps sections. The system comprises *normal vehicles* (i.e., cars), *special vehicles* (i.e., ambulances), a wireless communication network, and a remote *cloud* computing infrastructure. The entities are able to communicate with each other by exchanging data (see the arrows) using different communication media (e.g., 5g network) and different strategies (e.g., vehicle to vehicle communication, vehicle to cloud communication). All the vehicles inside the systems are autonomous, thus can sense the environment (i.e., messages sent by other vehicles) and be programmed to move without human intervention.

The requirement of the system is to minimize the travel time of special vehicles crossing the road section. More specifically, the ITS should satisfy the following global requirement:

r0: *The system shall maximize the traffic flow that allows special vehicles to travel the road (by encountering as less traffic as possible) and reach their destination soon.*

This requirement has to be fulfilled in any road configuration and traffic flow conditions until congestion. In fact, after road saturation, a travel time analysis is pointless as it is impossible for special vehicles to find space and overcome other vehicles.

Starting from **r0**, to assess the performance of the system, another requirement is derived, specifically **r1**: *Special vehicles shall not be hindered by regular vehicles*

We are interested in analyzing the traffic flow dynamics and congestion points in different road configurations. The analysis is focused on how the behavior of regular vehicles affects special vehicles in different traffic conditions.

To this end, a new requirement (sub-requirement of **r1**) is defined for the travel time of special vehicles, i.e., **r1.1**: *The average Travel Time for Special vehicles (TTS) has to be:* $lower\_bound \leq TTS \leq optimum + (optimum * 50\%)$. This expresses that the travel time for special vehicles shall not be less than its lower bound, defined in terms of real world constraints, i.e., regulated by the physic law: vehicles with a specific speed can not have a travel time less than $lower\_bound = \frac{roadlength}{speed}$, i.e., the optimal value.

Moreover, in order to be acceptable, the travel time for special vehicles can suffer of a delay that is at maximum equal to the 50% of the optimal time. This requirement is derived by observing at

the relation between the travel time and traffic delay, as a function of the flow rate, for generating unsaturated traffic conditions [19].

4 PERFORMANCE MODELLING

As outlined in [15], QN models offer a good first set of measurements for performance analysis, but the modeling of more complex state dependent behaviors is not straightforward. Therefore, we consider a different formalism, namely PetriNets (PN) [12][17] as a target analysis model.

The Petri Net formalism is characterised by three major elements: *tokens*, *places* and *transitions*. A *token* is an abstract element that can represent everything, from an abstract object (e.g., signals, jobs, requests etc.) to a real one (e.g., physical actors, real entities, etc.). Besides the standard Petri Net formalism, there is a variant called *coloured* PN [6, 7] in which tokens belong to classes and multiple classes can coexist in the same model. The peculiarity is that tokens can be handled differently, depending on their class. Also, token with classes can be seen as carrying information and it is possible to assign different characteristics to each class. Moreover, a coloured Petri Net will evolve accordingly to the tokens classes allowing to define more complex behaviours, specially class and state dependant.

A *place* is a container of tokens. It is an abstract entity that can represent different things similarly to tokens, from an abstract operational state to a physical working station of an implant or a web server. A place can have a capacity limit that bounds the number of tokens it can contain, from one to infinity. In the case of a coloured PN, it is possible to specify a capacity limit for each class. Places are always connected to transitions in input, output or both and a place is never connected directly to another place.

A *transition* is the most important and tricky element of a Petri Net model. A transition is always connected to places and never directly to another transition. This component represents events, actions, everything that lets the system evolve over time. Its major function is to evolve the system by executing, or not, the function it is supposed to represent. In order to do so, it has two sets of rules or pre-conditions that: (i) *enable* or (ii) *inhibit* the transition execution. These two sets or rules are defined on the amount of tokens that are present in the places connected to the transition on both input and output, and these two sets should be mutual exclusive. However, inhibiting conditions are checked first and then the enabling ones. If no inhibiting conditions are satisfied, then the transition can execute the enabling rule that is satisfied, if any. If multiple enabling rules are enabled at the same time, one among them is randomly chosen. When finally a transition fires, it consumes and creates an amount of tokens respectively from the input and output places based on what the chosen enabling condition has defined. Inhibit conditions do not consume or create tokens, they just prevent the transition from executing. Moreover, a transition can execute instantaneously or it can take a certain amount of time. The time a transition takes to execute can be defined by the transition itself regardless the tokens' class or can be defined for each tokens' class. All these PN aspects allow to model more complex and state-dependent behaviours that queueing networks are not capable of.

To model the ITS, we consider the road as a grid where each cell represents a portion of the road which, in the base case, is entirely occupied by a vehicle. Each vehicle moves with a speed abstracted in *number of cells per time unit*; it can change lane if there is more than one cell, and only if there is enough safe space to do so.

The model in Figure 2 represents a road section of a two lane road of a certain length. With the same assumptions made in [15], the road has a total length of *road_length* and is divided in a grid where each cell has a length of $\Delta X = 7.5$ meters. Therefore, to model an entire lane $k = \lceil \frac{road\_length}{\Delta X} \rceil$ cells are necessary. The number of lanes, two in this case, is identified by the variable i. Therefore, a couple *place-transition* represents a road segment ΔX and a chain of k couples *place-transition* represents a lane. Putting i lanes side by side defines the road we want to model.

Each lane segment ΔX has a place $P_{i,k}$ and a transition $T_{i,k}$, where i identifies the lane and k the so called position in the lane. Both indices start from zero and increase respectively from left to right and from top to bottom. Any place $P_{i,k}$ in the road can host only one token, regardless of the class. A lane has transition $T_{i,k}$ that allows or denies vehicle movements forward on the same lane. We insert a further transition $T_{OTk,k+1}$, called overtake transition, between the lanes to model changing lane manoeuvres and overtakes. An overtake transition is connected to both places k and $k+1$ of both lanes. Any transition enables a movement only if at least one destination place is empty, so there is no possibility to have two or more vehicles overlapping in the same road section. This avoids possible collisions. Moreover, the overtake transition enables changing lane manoeuvres to a vehicle only if the following condition is satisfied when it has finished crossing its road section: the next place, on the same lane, in which it has to go is still occupied by the preceding vehicle. In other words, a vehicle can change lane only when it is necessary, as dictated by logic.

In our case study, there are two types of vehicles and therefore we adopt a coloured PN with two classes of tokens. The vehicles flow is arriving into the system with some rate λ that is randomly routed to one of the lanes and distinguished by token class. Each new vehicle is routed in the first available free space among the $P_{i,0}$. Besides that, thanks to coloured PN formalism, we define an execution time for each transition in the model that is class dependant. Therefore every transition, when it fires, it will take a certain amount of time dependant to which vehicle is in the relative place. In order to cover some variance in the driving style, each transition execution time is randomly chosen from an exponential distribution with mean value the ideal velocity of the relative vehicle class.

In our PN model we implement a concept of *safety*, thanks to the tokens limit of one we defined on places where a lane transition $T_{i,k}$ is enable/disable just form the status of the current place $P_{i,k}$ and the next one $P_{i,k+1}$. However, to cover more complex safety aspects, we insert some sort of safety criteria when changing lane manoeuvres occur. As in the real life, drivers do not change lane any time they want but only when it assumes that the manoeuvre is secure to be done and this is done usually by looking if there is any upcoming vehicle on the destination lane. To this aim we use the enabling/inhibiting conditions of overtake transitions to do exactly so. An overtake transition $T_{OTk,k+1}$ is connected to both the destination and source places on both lanes, therefore it is aware of

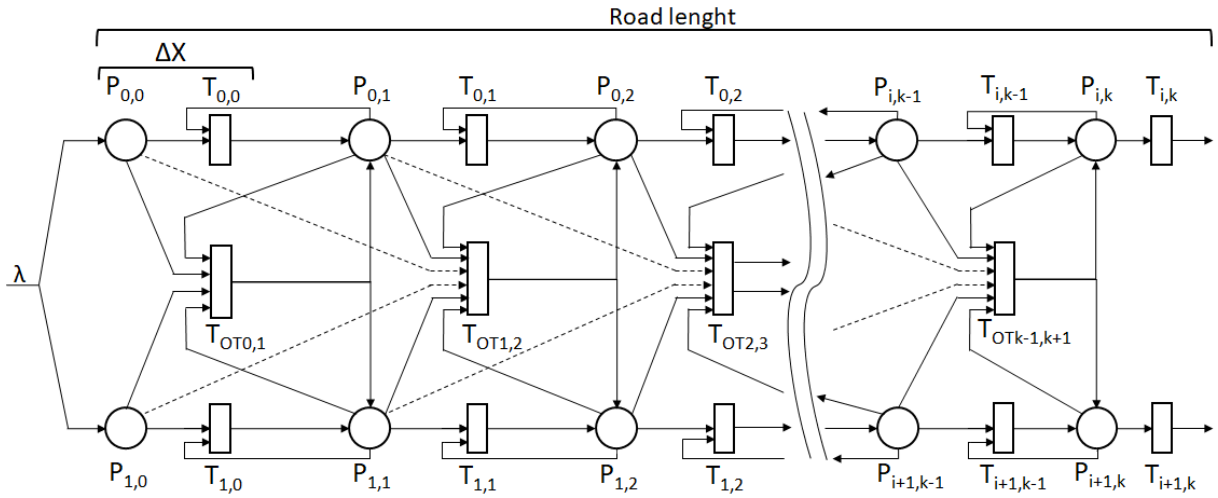


Figure 2: PetriNet architectural model of a road

the current state of four places. In order for a vehicle in place $P_{i,k}$ to change lane the conditions to enable the manoeuvre are two: (i) the next place on the same lane still has to be occupied and (ii) the places on the destination lane, i.e., $P_{i\pm1,k+1}$ and in $P_{i\pm1,k}$, must be empty. In other words, a change lane is enabled only if there is a slow vehicle in front and there are no upcoming vehicles on the other lane. This safety criteria takes into consideration two places on the destination lane, i.e., $P_{i\pm1,k+1}$ and in $P_{i\pm1,k}$, and therefore we called it *two look-ahead* (*2LA*) safety rule. We define also a more secure criterion that requires to take into consideration in the overtake enabling choice also the $k-1$ place status. Hence, also the $k-1$ place has to be empty meaning that in order to change lane there has to be even more free space on the destination lane. From a PN modelling point of view this translates into an additional edge connecting $P_{i,k-1}$ on both lanes in input to $T_{OTk,k+1}$ (see the dashed edges in Figure 2), besides the ones that connect the $T_{OTk,k+1}$ to the $P_{i,k1}$ and $P_{i,k+1}$ already present. Since this second safety criteria considered three places, it is has been called *three look-ahead* (*3LA*) safety criterion. In Figure 2 we represent only the 2LA for sake of clarity; but an example of 3LA is reported with the dashed input edges to the overtake transitions. It is straightforward to understand that this safety rule can be easily extended and generalised to a *n look-ahead* where *n* places are taken into account in the enabling/inhibiting conditions.

5 PERFORMANCE ANALYSIS

In this section we briefly report our previous experience with the analysis of the QN model that was proposed in [15] (see Section 5.1), then we show the benefit of adopting the PN model (see Section 5.2). Finally, we discuss the differences found between these two performance modelling abstractions, along with the open challenges raised when comparing these set of analysis results.

5.1 Previous analysis with the QN model

The QN model has been proposed in [15]. It includes a routing rule able to model overtake situations. The adopted rule is the ,*Join the Shortest Queue* (JSQ), i.e., a vehicle that finds a slower car in front of it can chose to move in one empty adjacent queue, if any. The same rule is adopted for the routing of new vehicles.

The model is solved by means of the JMT tool [2], and results are reported in Figure 3. The average travel time of the ambulances asymptotically reaches the average travel time of the cars with an increase of traffic flow, because the ambulance is not able to perform overtakes with a fully congested road. With respect to the requirements defined in Section 3, we can state that **r0** is partially satisfied, indeed it does not allow a clear identification of a congestion point; **r1.1** is satisfied until 5k vehicles per hour, then the presence of congestion makes the results not precise enough.

However, even if the requirements seem satisfied, the observed behavior highligth a problem. We expected to find two break points between low and high traffic jams: (*i*) when the traffic density starts affecting the ambulance travel time, and (*ii*) when the ambulance travel time becomes asymptotic to the cars travel time, due to the high traffic jam. Unfortunately, the graph depicted in Figure 3 does not show break points or an asymptotic trend in the ambulances travel time's curve. The policy of JSQ is a rough emulation of an overtake policy that takes into consideration the state of the queue when looking for a free spot. However, this policy does not guarantee safety in case a vehicle decides to overtake another one. This leads to a situation where any vehicle is able to overtake in any traffic condition, also in a fully congested road. We deduce that this is the reason why the curves of Figure 3 show no evident break point or asymptotic behavior. To further investigate the performance characteristics of ITS, we proceed with the analysis of the PN model presented in Section 4.

5.2 Incremental knowledge with the PN model

The model is simulated with two different safety rules in overtaking, called *2LA* and *3LA*, which check two or three positions ahead, respectively, before engaging an overtaking step. In both cases, due to the complexity of the model, the road has two lanes only.

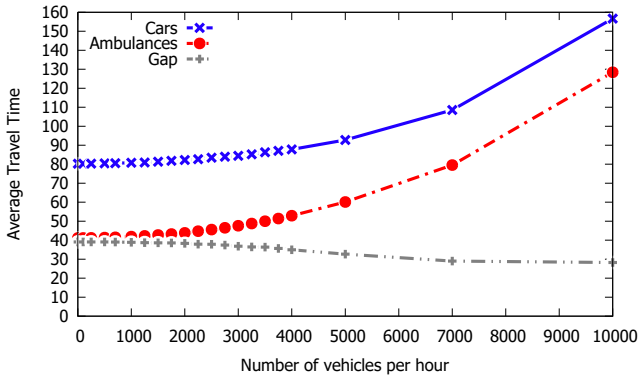
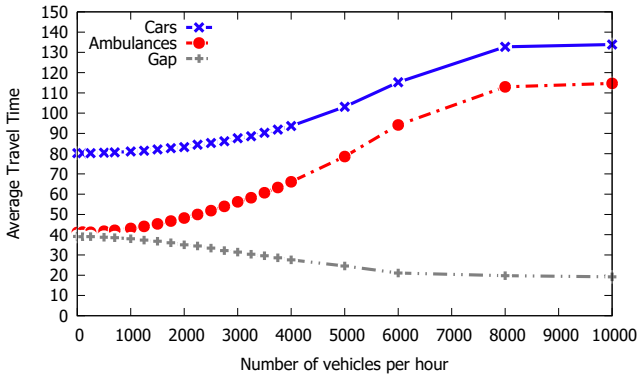
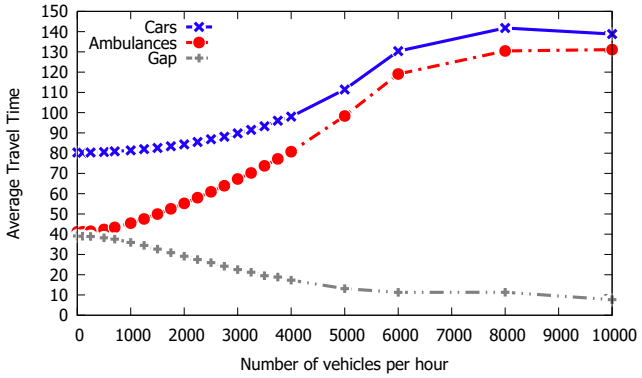


Figure 3: QN simulation [15]



(a) 2 Look-Ahead Overtake



(b) 3 Look-Ahead Overtake

Figure 4: Different overtake strategies: PetriNet results

The results of the simulation are reported in Figure 4. Requirements **r0** and **r1.1** are satisfied also in this case and we observe a more realistic behavior. We can notice that for high traffic densities (i.e., from 4k/5k vehicles per hour and more) the average travel time of the ambulances is asymptotic and very close to the one experienced by cars. Moreover, we can identify break points in the ambulance trace, specially with the 3 Look-Ahead configuration, and this is also highlighted by the gap that has an increasing trend towards zero. We are aware that the safety criteria push the jamming effect denying the change lane manoeuvre more often as in real life.

It is worth noticing that we are simulating a model that always evolves at its maximum without human reaction. In fact when a transition fires, it moves a token at its maximum speed, even in a congested traffic scenario. This is due to the absence of global variables that are system-dependant, it is an intrinsic problem of our model. What causes the delays is either the increasing road density but also the exponentially distributed firing time of transitions.

The local crossing speed of each road section is independently exponentially distributed but has no dependency or relation with the global traffic density. This implies that any vehicle, independently from the traffic flow or density, when moving to any subsequent free road section it goes at full speed. The PN model lacks a relation between the vehicles' local road section mean velocity and the global road density, that is what happens in a real traffic situation. As the traffic increases, the vehicles slow down their velocity instead of keep driving to their maximum speed and afterwards break to avoid collisions. A traffic density dependency on the vehicles speed gives back an early congestion point and an early delay on the average travel time of ambulances, from low traffic flows. However, this PN analysis fits into a test case trying to verify the requirement 1.1. In fact, for the issues just discussed we need to push the traffic flow way more than feasible for the road in order to see congestion effects, congestion points and saturation. For both the graphs in Figure 4, the average travel time of ambulances increases with the traffic flow until it reaches the saturation point, around 6k vehicle/hour, when it becomes asymptotic to the cars' travel time. Higher the safety rule is, less reckless the ambulance is, resulting in a smaller gap between the two travel time curves.

Discussion. From Figures 3 and 4 it is worth noting that QN and PN abstractions provide different trends for model-based performance analysis results. In fact, our case study shows that PN are able to point out a congestion that is instead missed using QN. This is relevant in our opinion since we are interested to understand which model abstraction is more suitable for the scope of identifying a congestion status. Summarizing, the results highlight that with PN performance formalism we are able to obtain a quality improvement with respect to the results obtained with the QN model.

With respect to queueing networks, PN provides more accurate results. However, PN maintains the problem of the velocity. Tokens move at their maximum velocity at any time, regardless to the traffic density, and that is not accurate. What is lacking is a way to insert a dependency in the vehicle velocity distribution on the global status of the traffic flow, density and congestion. In this way, when the traffic pumps up, the vehicles automatically slow down as it happens in real traffic jam scenarios.

The analyses performed with QN and PN have also pointed out that it is difficult to model and assess traffic environments where the agents can move freely. Besides, neither PN or QN are able to model message passing or any kind of runtime behavior change. Hence, for this complex system, there is the need to look for other modelling paradigms including (co-)simulations environments [18].

Threats to validity. There are some potential threats to the validity of the qualitative and quantitative assessment. The development of a performance model for the Intelligent Transportation System case study can be considered as a first step and it provides only a limited set of simplified features. In fact, when dealing with complex systems, such as the ones we are investigating, models tend to become very complex and intractable. For instance, the Petri Net model described in this paper consists of 435 KLOC and is solved in the order of hours. This task is difficult to be performed and needs experienced engineers able to link the granularity of the models to the expected results that such models are able to produce.

Different models and (ad-hoc) tools help tackling the complexity of systems. However, this brings the inherent difficulty of linking together different formalisms and aspects. To smooth this threat, we plan to use pivot languages like CSM [24] and KLAPER [3] that reduce the number of transformations and maintain some common abstract model supporting the reasoning and the traceability of both models and analysis results [23].

6 CONCLUSION

In this paper we have presented the results of an experience report whose aim was to investigate the applicability of performance models like QN and PN for the performance analysis of systems characterized by high dynamism and complex interactions. To this end, we have considered an intelligent transport system as case study and we have compared the results obtained applying a general QN model and a more detailed PN model. Our experience highlights that performance modelling and analysis of complex systems is challenging, as many cross-cutting concerns must be jointly accounted during the development process. On the other hand, our experimental results highlight some key aspects: (*i*) QN and PN analyses, despite their limitations, can be useful for a first understanding of the system behavior, and can represent a baseline for further analyses and comparisons; (*ii*) some congestion behaviors that are not easily captured by QN models can be expressed using more complex formalism, like coloured PN; (*iii*) QN and PN models are able to give significant insights only for a limited set of aspects; (*iv*) the need of more powerful models able to capture the whole system behavior and provide performance results in short time.

As future work, considering the PN formalism, we aim at applying the proposed performance models in real-world industrial settings. To this end, we plan to adopt Petri Nets in different domains, by conducting a controlled experiment with engineers and practitioners from industrial partners. Such application will be used to evaluate the effectiveness of Petri Nets and derive meaningful descriptive statistics. From a more general view point, we intend to investigate multi-modeling approaches able to deal with different aspects of the system in a combined way.

REFERENCES

[1] Radu Calinescu, Lars Grunske, Marta Z. Kwiatkowska, Raffaela Mirandola, and Giordano Tamburrelli. 2011. Dynamic QoS Management and Optimization in Service-Based Systems. *IEEE Trans. Software Eng.* 37, 3 (2011), 387–409.

[2] Giuliano Casale and Giuseppe Serazzi. 2011. Quantitative system evaluation with Java modeling tools. In *Proceedings of the International Conference on Performance Engineering*. 449–454.

[3] Andrea Ciancone, Mauro Luigi Drago, Antonio Filieri, Vincenzo Grassi, Heiko Koziolek, and Raffaela Mirandola. 2014. The KlaperSuite framework for model-driven reliability analysis of component-based systems. *Software and System Modeling* 13, 4 (2014), 1269–1290.

[4] A. Di Febbraro, D. Giglio, and N. Sacco. 2004. Urban traffic control structure based on hybrid Petri nets. *IEEE Transactions on Intelligent Transportation Systems* 5, 4 (2004), 224–237. https://doi.org/10.1109/TITS.2004.838180

[5] David Garlan. 2010. Software engineering in an uncertain world. In *Proceedings of the International Workshop on Future of software engineering research*. 125–128.

[6] Kurt Jensen. 2013. *Coloured Petri nets: basic concepts, analysis methods and practical use*. Vol. 1. Springer Science & Business Media.

[7] Kurt Jensen and Grzegorz Rozenberg. 2012. *High-level Petri nets: theory and application*. Springer Science & Business Media.

[8] H. Khazaei, J. Misic, and V. B. Misic. 2012. Performance Analysis of Cloud Computing Centers Using M/G/m/m+r Queuing Systems. *IEEE Transactions on Parallel and Distributed Systems* 23, 5 (2012), 936–943. https://doi.org/10.1109/TPDS.2011.199

[9] P. Kuehn. 1979. Approximate Analysis of General Queuing Networks by Decomposition. *IEEE Transactions on Communications* 27, 1 (1979), 113–126. https://doi.org/10.1109/TCOM.1979.1094270

[10] E. D. Lazowska, J. Zahorjan, G. Scott Graham, and K. C. Sevcik. 1984. *Computer System Analysis Using Queueing Network Models*. Prentice-Hall.

[11] Massimo Marchiori. 1998. Light Analysis of Complex Systems. In *Proceedings of the ACM Symposium on Applied Computing* (Atlanta, Georgia, USA). ACM, New York, NY, USA, 18–22. https://doi.org/10.1145/330560.330564

[12] Marco Ajmone Marsan, Gianfranco Balbo, Gianni Conte, Susanna Donatelli, and Giuliana Franceschinis. 1994. *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, Inc.

[13] L. Mokdad and J. B. Othman. 2003. Performance evaluation of MIR (mobile IP reservation protocol) based on stochastic automata networks. In *Proceedings of the International Conference on Semiannual Vehicular Technology*, Vol. 3. 1778–1782. https://doi.org/10.1109/VETECS.2003.1207129

[14] T. Murata. 1989. Petri nets: Properties, analysis and applications. *Proc. IEEE* 77, 4 (1989), 541–580. https://doi.org/10.1109/5.24143

[15] Lorenzo Pagliari, Mirko D'Angelo, Mauro Caporuscio, Raffaela Mirandola, and Catia Trubiani. 2019. To what extent formal methods are applicable for performance analysis of smart cyber-physical systems?. In *Proceedings of the European Conference on Software Architecture, ECSA Companion Volume 2*, Laurence Duchien, Anne Koziolek, Raffaela Mirandola, Elena Maria Navarro Martínez, Clément Quinton, Riccardo Scandariato, Patrizia Scandurra, Catia Trubiani, and Danny Weyns (Eds.). 139–144.

[16] Ragunathan Rajkumar, Insup Lee, Lui Sha, and John Stankovic. 2010. Cyber-physical systems: the next computing revolution. In *Proceedings of the International Conference on Design Automation*. 731–736.

[17] Wolfgang Reisig. 2012. *Petri nets: an introduction*. Vol. 4. Springer Science & Business Media.

[18] James A Rowson. 1994. Hardware/Software Co-Simulation. In *Proceedings of International Conference on Design Automation*, Vol. 94. 6–10.

[19] Hemant Kumar Sharma, Mansha Swami, and BajrangLal Swami. 2012. Speed-flow analysis for interrupted oversaturated traffic flow with heterogeneous structure for urban roads. *International Journal for Traffic and Transport Engineering* 2, 2 (2012), 142–152.

[20] C.U. Smith and L.G. Williams. 2002. *Performance Solutions: A Practical Guide to Creating Responsive, Scalable Software*. Addison-Wesley.

[21] Kateřina Staňková and Bart De Schutter. 2010. On freeway traffic density estimation for a jump Markov linear model based on Daganzo's cell transmission model. In *Proceedings of the International Conference on Intelligent Transportation Systems*. 13–18.

[22] M. A. L. Thathachar and P. S. Sastry. 2002. Varieties of learning automata: an overview. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 32, 6 (2002), 711–722. https://doi.org/10.1109/TSMCB.2002.1049606

[23] Catia Trubiani, Achraf Ghabi, and Alexander Egyed. 2017. Exploiting traceability uncertainty between software architectural models and extra-functional results. *Journal of Systems and Software* 125 (2017), 15–34.

[24] C. Murray Woodside, Dorina C. Petriu, José Merseguer, Dorin Bogdan Petriu, and Mohammad Alhaj. 2014. Transformation challenges: from software models to performance models. *Software and System Modeling* 13, 4 (2014), 1529–1552.