Modelling and evaluation of a Control Room application

Elvio Gilberto Amparore[1], Susanna Donatelli[1], and Elisa Landini[2]

[1] Dipartimento di Informatica, Università di Torino, Torino, Italy
{amparore, donatelli}@di.unito.it
[2] RE:Lab S.r.l. , Reggio Emilia, Italy
elisa.landini@re-lab.it

Abstract. This application paper describes the study of a control room system that has been performed inside the EU Artemis project HoliDes. The control room object of the study is for an Italian operator in gas energy distribution. Customers call the control room of the energy operator to signal malfunctioning of gas distribution and/or of gas apparatus. Upon a call the control room operators assign a technician delegated to physically reach the intervention site and make it, in first place, secure, and, in second place, back to normal operating condition. Because of the safety issues inherently associated with the gas distribution, the Italian Regulatory Authority for Electricity Gas and Water has set a service level agreement (SLA) requirement that states that an operator should reach the client site in less than 60 minutes in 95% of the times.
This paper describes the Petri net models that have been used to assess what is the load of calls that can be dealt with without violating the SLA, and what type of conditions make the system in a critical state. Petri nets considered are colored stochastic Petri Nets with and without deterministic and generally distributed transitions. In modelling terms the main issue that has been faced is that of adequately represents the geographical distribution of calls and technicians, while the main issue for the computation of the the performance indicator has been the SLA assessment, that requires a passage-time computation, an index that is not widely available in Petri net tools.

1 Introduction

Energy network surveillance systems are important services for the maintenance of safety-critical infrastructures. In this paper we consider the case of a utility company operating in Italy which, among other businesses, controls a part of the gas distribution network. Problems in this network are treated with great urgency, since a gas leak could easily result in explosions and other dangerous outcomes. For this reason, the Italian Regulatory Authority for Electricity Gas and Water (AEEG) requires that company treating gas networks should have an Emergency Call Center (ECC) available h24 subject to the additional constraint that at least 95% of the time a client reports a problem, a company technician should be on site in less than 1 hour. The company addresses this requirement by deploying an extensive network of company technicians distributed all over the service area, and keeps track of each intervention in a detailed log, for inspections and for planning the human resource allocation.

This cases study stems from our work with the utility company and RE:Lab inside the EU-funded Artemis project HoliDes. RE:Lab is an italian SME, located in Reggio Emilia devoted to human and machine interfaces. The project main aim is to study adaptive behaviour of cooperative systems, and in the desire to make the assignment of technicians more adaptive to the environment. For the HoliDes project RE:Lab has developed a prototype interface for the management of the intervention list by the field technicians. Technicians are equipped with a hand-held device with a GPS sensor, which helps the control room in selecting the nearest (free) technician to intervene. We have worked with RE:Lab to understand and support their choice of the nearest technician, and to identify the critical load over which it is not possible to respect the AEEG constraint without calling additional technicians from other areas.

Modelling in control rooms: literature We have found limited literature that address the problem of complying strict SLA requirement for emergency call centers. The work in [20] provides a good overview of the typical problematics that emergency control planning needs to address, and the amount of support a simulation tool can provide. The work in [17] addresses the problem of determine the critical load conditions of a emergency call center by using computer simulations, using a multi-agent system. The work in [24] considers the performance and availability measures for an emergency call center with a look at the cost optimization, but no need for strict SLA requirement is present.

In this paper we present a model for the evaluation of the AEEG SLA based on Generalized Stochastic Petri Nets (GSPN) [22] and Stochastic Well-formed Nets (SWN) [13]: GSPN are P/T Petri nets in which transition duration is either immediate or exponentially distributed, while SWN are their colored counter-part. We shall also use Deterministic Stochastic Petri nets (DSPN) [1] and the stochastic logic CSL^{TA} [16]. Often we shall use the non-Markovian extensions of GSPN and SWN in which distributions can assume any shape.

The paper is organized as follows: Section 2 states the problem being addressed: the control room functioning and the objective of this study; Section 3 discusses the modelling problems encountered and how they have been solved; Section 4 presents a first model of the control room, that is the basis for the definition , in Section 5, of the setting of the CSL computation. In Section 6 the SLA is studied using a model checker for the CSL^{TA} stochastic logic, while Section 7 addresses the problem of identifying the critical load conditions under which the system cannot guarantee the SLA, done using stochastic simulations. Section 8 concludes the paper.

Reproducibility. All models used for the analysis are available at `www.di.unito.it/~amparore/ATPN17-models.zip`. The data from the gas energy operator that have been used to set the timing of the models are instead not available as they contain sensible data.

Acknowledgements. This research was supported by the Artemis EU project HoliDes (grant no. 332933). The goal of the HoliDes project is the improvement of adaptive systems where human and machines operate together to guarantee critical goals.

2 The Control Room problem

This paper presents an application case study centered around the problem of human resource allocation of an utility company. The company manages the gas distribution network, and has many logical units (*areas*) for each cluster of municipalities. The company furtherly subdivides an area into *zones*. Each area has a dedicated control room that takes care of the incoming calls made by the customers, which mainly concern critical problems related to the gas network (malfunctions, leakages, gas odour in the air, etc...). Commercial assistance is done by a separate call center, to reduce the load. We mainly focus on the Reggio Emilia district, which groups together 42 municipalities, and has an area of 2291 km^2 and a population of 531K inhabitants. That area is subdivided into four zones: North, East, West and South.

Control room description. There are one or more operators in the control room that deal with incoming calls. Sometimes, calls are not related to gas problem, and are therefore diverted to the other call center. When a client calls the control room to report a gas problem, the control room operator opens a ticket and transmits the assistance request to a group of company technicians (mainly plumbers) to reach the client site and inspect the problem. Each area has an area supervisor (reference technician) that receives the control room transmission and either intervenes directly, or redirects the request to another company technician. Each task of dealing with the identified problem is split in two subtasks: a **securing subtask** followed by a **repairing subtask**. The first one consists in reaching the client site and removing the direct cause of the problem (like closing a valve to avoid a gas leakage). An idle technician assigned to a securing task has to reach the location, analyse the problem and make the site secure. Typically, the securing task is done quickly. The second task is the actual fixing, and may require longer times. The second one may be missing if there is no repair to be done, or if the repair can be postponed. Usually, if there are other urgent calls, the repair task is assigned to a separate technician, and is done later.

Service Level Agreement. The period of time from the instant in which the call is answered until a technician reaches the location for which the call has been placed is named *intervention time*. The Italian Regulatory Authority for Electricity Gas and Water requires that in 95% of the cases the intervention time is less than 1 hour, otherwise the company will incur in a fine. The company takes accurate logs of each ticket, with timings from the client call to the closing of the intervention.

Task assignment policy. In the current assignment protocol, technicians are pre-assigned to zones (and we can speak of *his zone*), but they may occasionally intervene on problems in other zones. Therefore, a ticket of a problem in a zone is preferentially assigned to the zone technician. We call this *assignZone* policy. If the zone technician is already busy when a new call arrives, then the call is diverted to the area responsible that may decide to assign the ticket to a technician of another zone in the same area (but not outside the boundary of the area) The policy of calling a technician to intervene in a zone different from his zone is subject to a different consideration: When the travelling time is reasonable, we call the assignment policy *assignNextTo*, otherwise we call it *assignNonConvenient*. Since the area is quite large it is not always possible to ensure the 1 hour requirement by allocating any technician to arbitrary calls. The policy has been modified recently by the company RE:Lab with to support of precise localization of the tehnicians using GPS-tracking hand-held devices. The devices cover the transmission protocol using a dedicated GUI.

The problem being verified. The goal of the studied system is to first define a model whose behaviour fits the empirical data provided by the company logs. Given the model, it is then of interest to determine how the system behaves in critical situations: what is the maximum load capacity of the system can support with a given, fixed amount, of technicians? What conditions determine the possibility to respect the SLA?

We performed this analysis together with RE:Lab that was in charge of building the new technician GUIs over the smart phones. The data that we had available for model validation was limited (8 weeks of operations with recorded info like intervention site, time of arrival on the call, time of arrival on-site) and we did not have access to the record of the contracts, to know the geographical distribution of the Clients.

3 Modelling issues

In this section we describe the model of the interventions planned by the control room. We use a continuous stochastic approach, with Generalized Stochastic Petri Nets (GSPN) [22]. In particular, we are interested in making distinctions between tokens of the net, i.e. we use Stochastic Well-formed Nets (SWN) [13] as the modeling formalism. All nets have been created and solved with the GreatSPN tool [3], unless otherwise indicated.

The system is decomposed into multiple logical blocks:

CG: calls' generation. A subsystem of the calls that reach the control room (calls distribution in time and in the geographical space)

CR: Control room. A subsystem of the control room operators, that receive the client calls and open new tickets.

TP: Technician assignment protocol. A subsystem of the protocol that assigns technicians to open tickets

TA: Technician activity. A subsystem modelling the activity of each technician, i.e. reaching the target site and securing the gas distribution network.

Since we have had assurance from RE:Lab colleagues that the control room is never a bottleneck of the system, the model of the **CR** subsystem is reduced to a minimum and we assume that all call lead to an assignment of a technician while in reality many calls are inappropriate and are discarded. Since the goal of the study is to compute the SLA satisfaction, and since SLA satisfaction is computed only for calls that lead to an intervention, assuming that all calls lead to a technician intervention is adequate. Considering the simplicity of the **CR** subsystem, its behaviour has been included into the model of the **CG** subsystem.

Modelling by subsystems is usually a good approach, but there are issues that are common to the whole model, in particular it is necessary to decide whether or not *to manage the identities* of the clients and of the technicians and how to manage the *physical locations* of the calls.

Modelling clients. In the SLA objective of the study there is no notion of specific client. An example of SLA of this type is instead: “calls within a week from the same client should be dealt with by the same technician”. An identification of the clients may be needed to be able to compute the time of intervention, which is central in the notion of the considered SLA, but, although the model should be built taking into consideration the performance indicators we need to compute, changing the model to favour the computation of the performance indicators is delegated to a later stage (section 5), as we believe it is important to distinguish the model from the model modified so as to favour the computation of the indices. The first model therefore will consider clients as black tokens. The only relevant information is the location of the intervention (site) required by a call, as discussed next.

Modelling the geographical aspects. Calls may come from any client in the area, and the issue is the level of abstraction at which to model the call sites. Since an area is split into zones a first model is to abstract the geographical coordinates of the call sites and to identify the call site with their zone. Another possibility is to consider municipalities. We shall discuss this issue in more details when describing the **CG** subsystem.

Modelling technicians. None of the objectives of the study involves the single technician, there is no requirement to make comparison among technicians, so technicians, as calls, should be identified by their position in the area, with the same level of details as that of the location sites.

Timing of the model. All times are expressed in minutes. Timings are based on distributions fitted from real data, as specified in Section 4.

4 A first model

Figure 1 depicts two possible models for the **CG** subsystem. The Figure is a screenshot of the GreatSPN GUI[3] with which all the models used in this paper have been created and through which all numerical results have been obtained. In the model canvas there is a color class Z that represents the site locations (the location in which there is a failure that requires a technician intervention): the choice of the location sites is very coarse, and we use the 4 geographical zones,as discussed before, to model the geographical origin of the calls.

The simple model on the left generates the calls through the exponential transition transition *Simple_calls*, which is of single server type. Each generated call gets the color of a location site in a non deterministic manner, as, at each firing, a value for the variable r of color class Z is chosen and a token of that color is deposited in place *Simple_OpenRequests*. In the solution process this non-determinisms results in equal probability.

This model has three peculiarities: it has no explicit model of the control room operators (which is consistent with the information we had that the bottleneck of the system are the technicians and not the control room operators), it assumes an infinite population of calls (as transition *Simple_calls* has no input place) which may make the performance evaluation impractical, and there is no way to generate a load of calls that are not uniformly distributed over the four zones that represent our geographical space.

These limitations are lifted in the “Complete model” of Figure 1. There is a pool of clients (place *Clients*) initialized with the parameter *N*, that arrive as soon as there is an operator available (place *Operators* initialized with *Op* operators) to answer the call (immediate transition *call*). As we shall see later when the full model will be composed, place *Clients* will be used to close the model and to generate a finite state space. Calls receive a geographical identity through the four immediate transitions that transform a generic call (neutral token in place *Dispatch*) into a call from a geographical zone. The weights associated to the four immediate transitions are what allows to model calls from certain geographical areas as being more probable than others: in the model a call from the *South* zone is almost three times less probable than a call from the other zones, as we shall later see. This difference indeed accounts for the actual difference in population in the four zones. Remember that in GSPN/SWN the modeller does not assigns probabilities to immediate transitions, but weights, that are then normalized to compute the probabilities of the choices out of the vanishing states of the reachability graph.

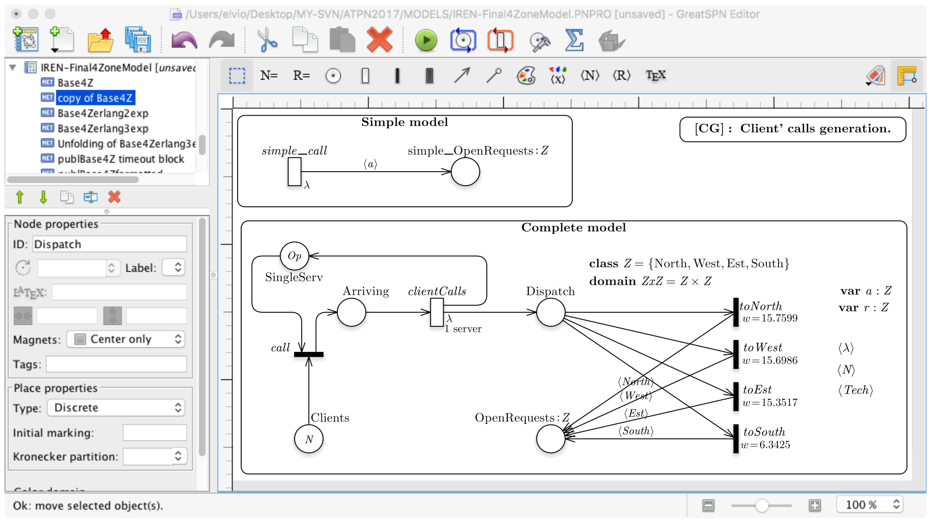


Fig. 1. The first model of the **CG** component.

Figure 2 depicts the model of the **TP** subsystem. The technicians (place *IdleTechs*) are identified upon their geographical location (class *Z*). The assignment protocol in use before the HoliDes project had one technician assigned to each geographical zone, which would lead to an initial marking of *All* for place *IdleTechs*, meaning that, in the initial state, there is one technician per zone.

Since the current protocol assigns to each call the closest available technician, reader may doubt how this info can be included in the model. Should we have made a model that includes continuous variables so as to represent the distance of a technician from an intervention site? Or should we model the GPS coordinates of a technician and how these coordinates evolves while he/she is travelling? Since the law with which the coordinates change while travelling depends on many conditions that change from time to time (like traffic and weather conditions), we believe that a very detailed modelling approach will not help, since we would model very detailed aspects for which there is no available evolution law. We have therefore taken a discretization approach, in which we, again, consider only the zones, and we define which zones are close to each other and which ones are not. The distinction between convenient and non-convenient assignment is based on the travelling times, which will be explained later.

To define closeness we have checked the distance among the middle points of the four zones, as explained in the comment to Figure 5 and we have observed that the only pair of zones that cannot be considered close one to the other are *North* and *South*, which lead to three levels of closeness: technician and site in the same zone, in a "next to" zone and in a "non convenient" zone, implemented, in Figure 2 by the three immediate transitions *assignSameZone*, *assignNextTo*, and *assignNonConvenient*. The three transitions have decreasing priorities, so if there is a technician in the same zone as that of the call (tokens of the same color in places *IdleTechs* and *OpenRequests*) *assignSameZone* fires; if there are not, transition *assignNextTo*, which has the next smaller priority, fires. If even this transition is not enable, transition *assignNonConvenient* fires. Note that the output places differ, since if the technician is assigned to the same zone the travelling time will be significantly smaller than that of the other two cases.

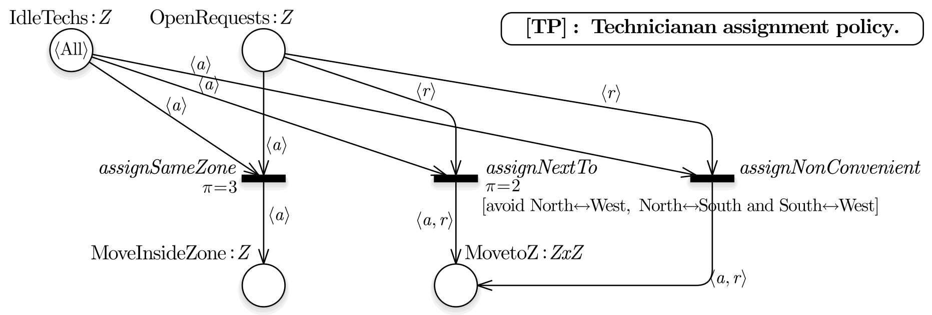


Fig. 2. The first model of the **TP** component.

Figure 3 depicts the model of the **TA** subsystem: from the time a call is assigned to a technician and he/she starts travelling, to the end of the intervention. The travelling time model consider two cases: travelling among zones, or travelling inside a single zone. The set of 6, mutually exclusive, exponential transitions on the right in the SWN of Figure 3 models the time it takes to reach the target zone *r* from the current zone *a* of the technician. The rate of these transitions corresponds to the inverse of the average times reported in the tables contained in Figure 5. Similarly, the set of four transitions *move*_$*$ on the left model the traveling times inside a zone.

Figure 4 depicts the composition of the three models by superposition over places of equal names. The composition is achieved by using the composition facility of Great-

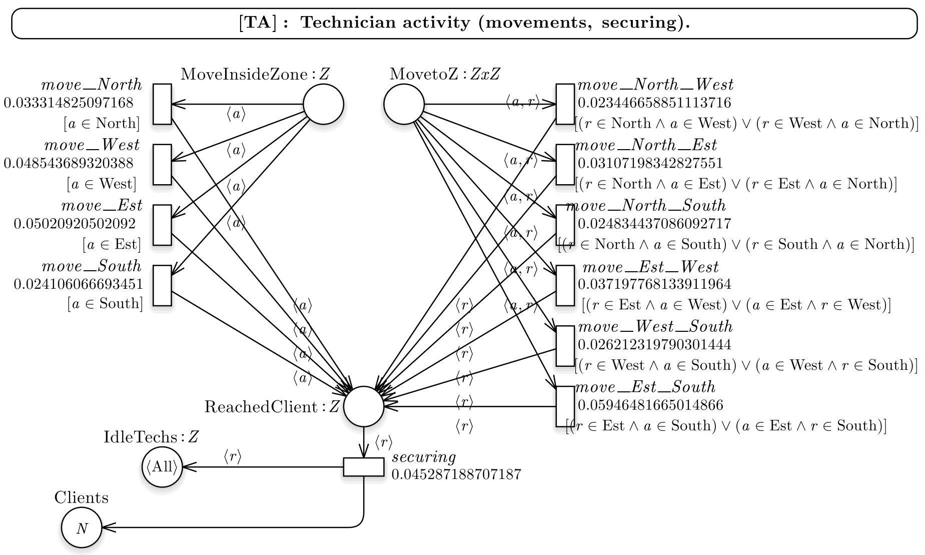


Fig. 3. The first model of the **TA** component.

SPN (the Σ button in the GUI depicted in Figure 1), which is a pairwise composition based on places and/or transition superposition, as described in [12]. The model has been then slightly modified by hand to make it easier to read.

In the composed model it is immediate to observe that there is a pool of N unidentified clients going around in the model. Closing the model to generate a finite state space is a rather standard technique in performance evaluation, but it is important to make sure that N is big enough. Since what we want to model is an arrival process with inter-arrival time equal to the inverse of λ, the value of N has to be large enough to guarantee that there is a very low probability that transition *clientCalls* is not enabled because of lack of tokens in the *Clients*. Moreover we have fixed $Op = 1$, therefore setting to 1 the initial marking of the *SingleServer* place. This choice is based on what has been reported by the company (control room operators are never a bottleneck of the system), moreover the data that we have available to estimate the arrival rate do not distinguish the information flux of the different operators, but we can only observe the inter-arrival time of calls at the control room. Note that at this point the *Clients* place could have been directly connected to the single server transition *clientCalls*, but keeping place *SingleServ* does not increase the tangible state space and makes explicit the single server policy. This will be particularly useful in more detailed models in which clients are identified.

Another observation is that there is no specific queue associated to the calls that wait for an idle operator. This is the easiest choice since GSPN and SWN do not have queueing places, although extension in that sense have been defined [11]. By not keeping the queue, and due to the presence of priorities over the three immediate transitions

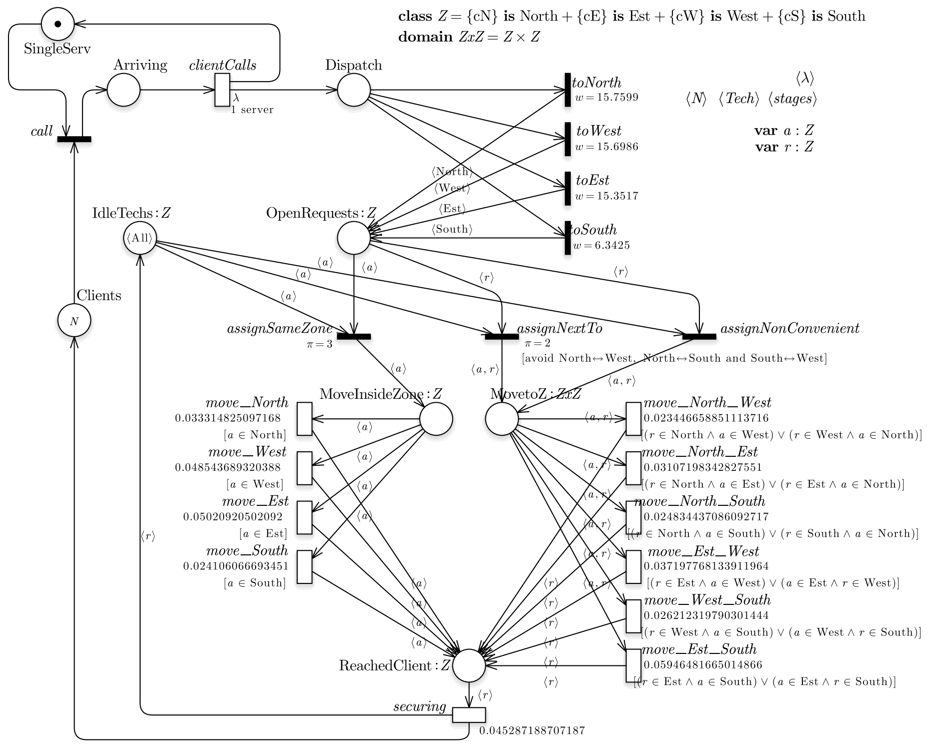
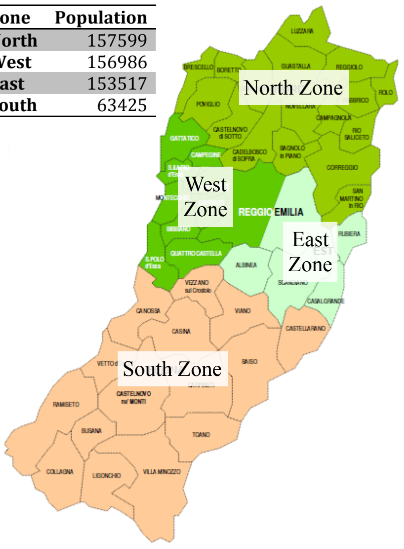


Fig. 4. The first model of the control room.

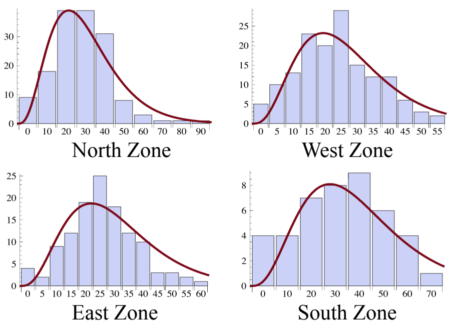
in the SWN of Figure 2, in the model we tend to assign "same zone" technicians more frequently than in reality.

Figure 5 shows the data that we have used to set the model's parameter. The zone considered is the area of Reggio Emilia. The main town, Reggio Emilia, concentrates around 1/3 of the population (531K inhabitants) in 1/10 of the area. The colour on the map reflects the aggregation of municipalities into zones, with the amount of population in each zone reported in the upper table. These data have been used to set the weights of the immediate transitions of the subnet in Figure 1: since we have no data on the amount of clients per municipality (which is considered a sensible information by the company not to be disclosed to third parties) we have assumed that the clients are spread in the zone proportionally to the number of inhabitants. The time to move inside an area has been computed as the time to move among two municipalities chosen among the most populated municipalities and among the pairs that where not at the extreme opposite points of the zone. The data have been computed through the Google Maps API, by using a script, so as to be able to try different choices. Another possibility would be to make an exhaustive search and do the weighted sum (based on population) of the distance among any two pairs of municipalities. The average time taken for the travelling time inside a zone (intra-zone travel) is reported in the lower table. Times are

Zone	Population
North	157599
West	156986
East	153517
South	63425



Fitting of traveling time distributions inside each zone from company data. Erlang-3 distribution is used.



Average traveling times (inter/intra zones) used in the model.

	North	West	East	South
North	30.02	42.65	32.18	40.27
West	42.65	20.60	26.88	38.15
East	32.18	26.88	19.90	16.82
South	40.27	38.15	16.82	41.5

Fig. 5. Data for model parametrization.

expressed in minutes. To compute the travelling time among two zones a similar criteria has been applied but the pairs of municipalities considered have to be in different zones.

Figure 5 reports also the travelling time inside a zone computed from the available data. The records in the company logs report the indication of the destination of the travel (the intervention site), but not the location from which the technician started from. Not surprisingly the experimental data give a longer travelling time than the one computed using Google Maps, which assumes that the technician is already somewhere in the area. Of course estimating the average travelling time is not enough, as also the distribution (or at least the variance) of this value may have a significant impact. Figure 5 shows the four bar-charts of the travelling times computed on the available

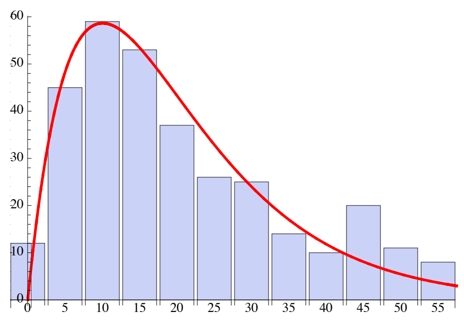


Fig. 6. Bar chart of the securing time obtained from company logs.

data, while the smooth curve is an Erlang-3 shape with average time equal to the one computed using Google Maps API.

Figure 6 shows the histogram of the securing times, obtained from the available 320 samples extracted again from the company logs. The overlayed red curve shows an Erlang distribution with 2 stages that fits the sampled data. The obtained distribution is used for the *securing* distribution in the model. All fittings have been done using the Mathematica tool.

5 SLA computation

The *passage time distribution* is a specific type of performance index which is particularly useful when reasoning about properties related with SLA or safety requirements. For our SLA we do not need to have the full distribution, but simply the probability that passage time is greater than one hour, that is to say the probability that the system violates the SLA.

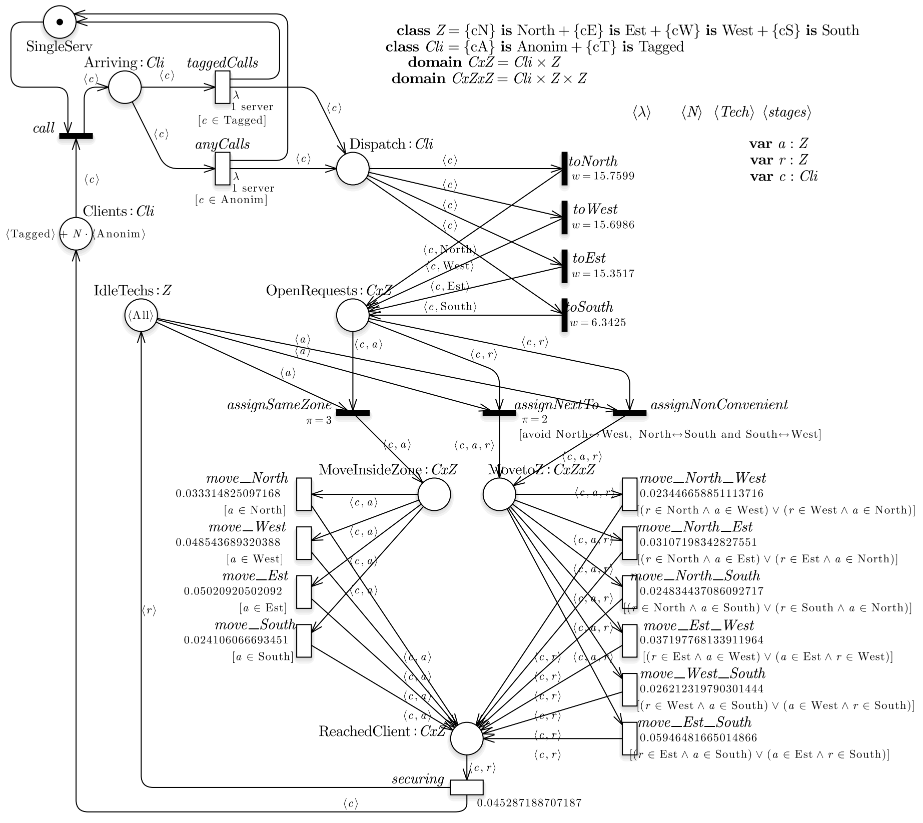
The definition of the SLA includes a passage time (60 minutes from when a call is received by the operator, until a technician arrives on-site), and the computation of a passage time requires to identify a start and an ending conditions: the passage time is accumulated from the time the start condition gets true until the ending condition gets true. In our system the SLA is based on the time it takes from the end of the call with control room application until a technician reaches the location, therefore the start and end conditions are respectively defined as the events: a client enters in place *OpenRequest* and the same client reaches place *StartSecuring*. In addition, we have to ensure that the time is taken *for the same client*.

A passage time measure specification for CTMC is usually based on the definition of entry, goal and forbidden states: the distribution of the time required to reach a goal state from any entry state without hitting any forbidden state can be computed with different methods and tools [21, 15]. This typically requires the (automatic) manipulation of either the CTMC or of the high level model used to generate the CTMC, often through the synchronization with an automaton that specifies the behaviours to be taken into account, as for Extended Stochastic Probes (XSP [14], operating on PEPA models [19]) and Path Automata (PA, operating on Stochastic Activity Networks [23]).

Computing passage time of an entity in a Petri net requires to be able to identify that entity (typically called "a client") and its evolution in the net, and to measure the time required for the identified entity to go from one state to another of the net, or from one transition firing to another. In GSPN terms this often leads to the need to follow a specific token through the net. Since the identification of the flux of clients is trivial in our model, to distinguish one client from the other we can simply define a color class *Cli* with N distinguished colors c_i and change the color domain of all places that carry a client so as to account for the change (places *Clients, Arriving, Dispatch, OpenRequests, AssignedLocal, AssignedMove, StartSecuring*). The resulting net is shown in Figure 7.

The passage time identification and computation is not trivial if the token to be followed may go through places containing other tokens, since they are indistinguishable and/or if the flux of tokens in the net is not clearly identifiable: this aspect has been tackled in the literature by introducing a formalism extension called Tagged GSPNs [8], for which a specific language for passage time specification has been later developed (Probe Automata [5]) to ease the task of defining the TGSPN conditions for entry, goal, and forbidden states. In Tagged GSPN the user specifies the initial and final condition of the "passage" and the system takes care of coloring the net so as to identify the client, and to associate to the client a subnet that represents the client evolution during the "passage", based on P-semiflows computation.

In our model, we do not need forbidden states (states that abort the computation of the passage time if a certain condition happen), and the notion of client is simple. Following the intuition behind tagged GSPN we can also simplify the definition of the color class *Cli*: If all clients behave the same, then it is enough to follow a generic one, so the color class *Cli* can be defined as having only two colors: anonymous client (cA) and one identified, tagged, client (cT). The net is then initialized with $N+1$ clients, N anonymous and one tagged. Figure 7 shows the resulting net.



class Z = {cN} is North + {cE} is Est + {cW} is West + {cS} is South
class Cli = {cA} is Anonim + {cT} is Tagged
domain CxZ = Cli × Z
domain CxZxZ = Cli × Z × Z
⟨λ⟩ ⟨N⟩ ⟨Tech⟩ ⟨stages⟩
var a : Z
var r : Z
var c : Cli
SingleServ
Arriving : Cli
taggedCalls
λ 1 server [c ∈ Tagged]
anyCalls
λ 1 server [c ∈ Anonim]
call
Clients : Cli
⟨Tagged⟩ + N · ⟨Anonim⟩
Dispatch : Cli
toNorth w = 15.7599
toWest w = 15.6986
toEst w = 15.3517
toSouth w = 6.3425
OpenRequests : CxZ
IdleTechs : Z
⟨All⟩
assignSameZone π = 3
assignNextTo π = 2
assignNonConvenient
[avoid North↔West, North↔South and South↔West]
MoveInsideZone : CxZ
MovetoZ : CxZxZ
move_North 0.033314825097168 [a ∈ North]
move_West 0.048543689320388 [a ∈ West]
move_Est 0.05020920502092 [a ∈ Est]
move_South 0.024106066693451 [a ∈ South]
move_North_West 0.023446658851113716 [(r ∈ North ∧ a ∈ West) ∨ (r ∈ West ∧ a ∈ North)]
move_North_Est 0.03107198342827551 [(r ∈ North ∧ a ∈ Est) ∨ (r ∈ Est ∧ a ∈ North)]
move_North_South 0.024834437086092717 [(r ∈ North ∧ a ∈ South) ∨ (r ∈ South ∧ a ∈ North)]
move_Est_West 0.037197768133911964 [(r ∈ Est ∧ a ∈ West) ∨ (a ∈ Est ∧ r ∈ West)]
move_West_South 0.026212319790301444 [(r ∈ West ∧ a ∈ South) ∨ (a ∈ West ∧ r ∈ South)]
move_Est_South 0.05946481665014866 [(r ∈ Est ∧ a ∈ South) ∨ (a ∈ Est ∧ r ∈ South)]
ReachedClient : CxZ
securing 0.045287188707187
⟨c⟩ ⟨a⟩ ⟨r⟩ ⟨c, a⟩ ⟨c, r⟩ ⟨c, a, r⟩ ⟨c, North⟩ ⟨c, West⟩ ⟨c, Est⟩ ⟨c, South⟩

Fig. 7. The model of the system for passage time computation

This allows, in the next subsections, to present two different approaches for the SLA computation. We recall that the SLA is satisfied if

[SLA:] 95% of the calls that require an intervention observe the intervention time (from client call to the arrival of a technician) in within 1 hour.

which implies that the computation required is not a passage time distribution, but it is enough to compute the probability that the passage time is less than 60 minutes, although, as usual, the distribution is more informative. In the following we shall present two different approaches: in Section 6 the SLA is computed as a path property in the stochastic logic CSL$^{\mathrm{TA}}$, while in Section 7 the net is modified and the SLA satisfaction is based on standard performance evaluation features (throughput of transitions) on the modified model.

6 Use of CSLTA for SLA computation

Stochastic logics for CTMCs define formulas of the type $Prob_{\bowtie\alpha}(\varphi)$ that are satisfied by a CTMC if the probability of the set of executions of the CTMC that satisfy the constraints φ is $\bowtie \alpha$. The most well-known of the stochastic logics is CSL [7], that, in the desire to limit the cost of the model-checking, trades simplicity of φ for efficiency, resulting in a logic that is not adequate for passage time computation [4].

The CSL$^{\mathrm{TA}}$ [16] logics has been introduced to give more flexibility to the modeler, as formula takes the form of $Prob_{\bowtie\alpha}(\mathscr{A})$, where $\mathscr{A}$ is a one-clock timed-automaton. CSL$^{\mathrm{TA}}$ model-checker is part of the GreatSPN suite. The model-checking is based on numerical solution and requires the solution of a Markov Regenerative Proces (MRP). An efficient solution algorithm for this type of MRPs has been developed [6].

The relationship between passage time and stochastic logics has been investigated in [4], in the context of HASL (Hybrid Automata Stochastic Logic) [10]. HASL allows very complicated properties to be expressed as $Prob_{\bowtie\alpha}(H)$, where H is an Hybrid Automata. The complexity of the path specification makes the underlying stochastic process impossible to solve in numerical form, and simulation is the only viable option. Cosmos [9] is a simulation-based model-checker for HASL that operates on systems specified as Petri nets. Following the approach in [4] we use a stochastic logic to compute the passage times. We have used CSL$^{\mathrm{TA}}$: the reason why CSL$^{\mathrm{TA}}$ has been preferred over HASL is that CSL$^{\mathrm{TA}}$ is part of the GreatSPN suite and *allows an exact numerical solution*, while HASL properties can only be checked through simulation.

Figure 6 shows the Timed Automaton used to compute the SLA: it consists of 5 locations, including the initial location l_3, a success final location l_4 and a failure final location l_2. Locations are labelled with atomic propositions, and edges are labelled with action sets. In GreatSPN the DTA is defined in parametric form; when the DTA is used to model check a CTMC generated by a GSPN, the DTA actions are instantiated on transitions names and the atomic proposition are instantiated on boolean formulas over the Petri net markings.

Following the ideas in [4] the DTA of Figure 6 accepts, at the beginning (while in location l_3) all the GSPN behaviours (*Act* stays for "any transition firing", there is no constraint on the value of the clock x, and the atomic proposition associated to l_3 is

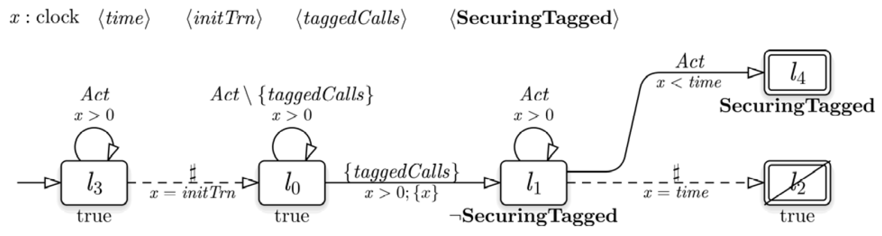


Fig. 8. The timed automata for the CSL^{TA} computation of passage time

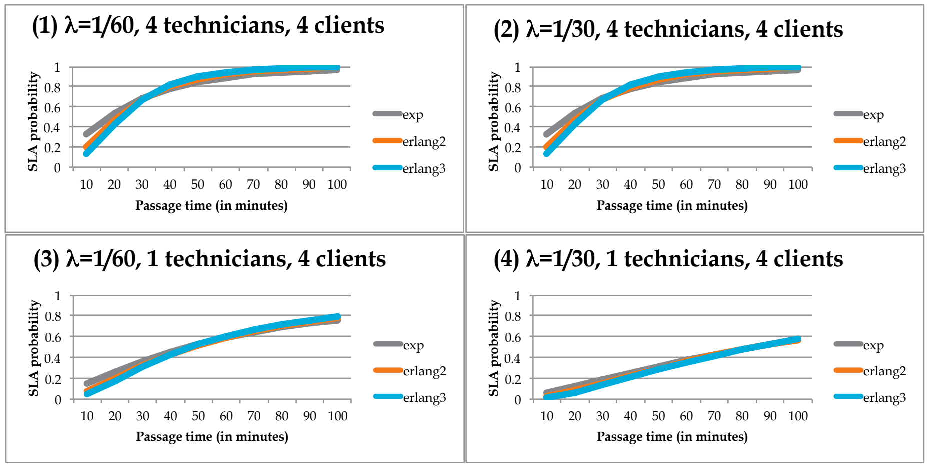
true). When the DTA clock reaches *initTrn* the DTA keeps accepting anything until a tagged call takes place (action *taggedCalls*). At this point the clock is reset and the DTA moves to location l_4 and starts observing until the condition **SecuringTagged** does not hold. When the condition **SecuringTagged** becomes true before time *time*, the DTA moves to location l_4 and accepts the path. If instead the time boundary *time* is exceeded, the DTA is forced to move to the failure location l_2.

To check the SLA, we apply CSL^{TA} model-checking by binding the DTA parameters to the net elements. Note that the model we have defined is an SWN, while CSL^{TA} is defined for CTMCs, and the implementation in GreatSPN assumes that the CTMC is generated from a GSPN. Therefore the net of Figure 7 has been unfolded[3] and the parameters of the DTA have been instantiated over the marking of the GSPN resulting from the unfolding It was shown in [4] that if the transient behaviour is long enough (and this depends on the value of *initTrn*) so that the CTMC is in steady state when the simulator reaches time *initTrn*, then the model checking of the DTA of Figure corresponds to verify the SLA in steady state.

Figure 9 depicts the distribution of the passage time in four different situations: charts (1) and (2) consider four technician and a load of incoming call generated from a population of $N = 4$ clients and a duration of 60 (left chart) and 30 (right chart). Charts (3) and (4) depicts the same situations, but with a single technician. As from the data available from the company, the situation of (1) is considered the normal condition (4 technicians available, about 1 client call per area per hour). The lower charts show instead a situation of congestion, when the number of incoming calls cannot be handled by a single technician without breaking the SLA.

On each chart there are three passage time distributions for three models: the “exp” curves refer to the case where all SWN transitions are either immediate or exponential, the “Erlang-2” (Erlang-3) labelled curve are obtained by expanding the exponentially distributed transitions of travelling time and security activity in 2 (3) exponential stages twice (three times) as fast as the exponential transition they substitute. This approach, which is rather standard [2] allows to generate the CTMC while using distributions (like Erlang-2 and Erlang-3) that have a significantly lower variance than not exponential.

[3] The unfolding facility is available through the GUI (open box icon on the toolbar), a facility that has been implemented for the participation at the Petri net Model-Checking Contest, while the definition of the parameters in terms of GSPN elements is not automatically supported.



Query CSLTA:
P PassageTime(initTrn=1000, time=x |
taggedCalls=taggedCalls |
SecuringTagged=#StartSecuring>0)

SLA Probability at 60 minutes:

	case (1)	case (2)	case (3)	case (4)
using exp	0.8921	0.8910	0.5882	0.3754
using erlang2	0.9270	0.9258	0.5947	0.3600
using erlang3	0.9469	0.9458	0.6023	0.3562

Fig. 9. SLA computation with CSL$^{\mathrm{TA}}$.

Note that, by not explicitly considering a FIFO queue, when N increases the probability of having one or more clients waiting in place *Dispatch* increases as well, which increases the probability that an idle technician gets immediately assigned to a new client and that the technician will find a client request for the same zone where he/she is currently located. This phenomenon tends to make a heavily loaded system to work a bit better than it should be.

The charts show also the impact of using Erlang-3 distributions (which actually fit quite well the experimental data) instead of using exponential transitions. Lower variance of the Erlang distribution makes the system closer to the SLA requirement.

The model-checking of CSL$^{\mathrm{TA}}$ allows to exactly compute the probability of exceeding the 60 minutes, but it suffers the standard limitation of state space explosion, worsened by the fact that the stochastic process to be solved is an MRP. The largest state space covered in this test were of about 1.7×10^6 states. Despite big improvements in memory and space on MRP solution [18], it is difficult to solve MRP's with more than a few million states, which in our case limits the exact solution at instances of the SWN of Figure 7 with 4 clients, 4 technicians and the Erlang-3 expansion (that, obviously, significantly increases the state space).

6.1 Summary of CSL$^{\mathrm{TA}}$ results and lesson learned

Lesson learned - case study. From the results of the analytical solution of CSL$^{\mathrm{TA}}$ we can observe that 4 technicians are a minimal number to sustain the load of calls.

Lesson learned – modelling and tools. Thanks to unfolding and stage expansion we have been able to model-check CSL^{TA} for an SPN that includes also non exponential transitions.

7 Use of timeouts and DSPN for SLA computation

Since the exact model checking of CSL^{TA} is limited to small systems, then the only viable solution for larger systems is simulation. In this context simulation can take two forms: use a model-checker based on simulation (what is now called "statistical model-checking") or modify the model so as to compute the SLA with a generic SWN simulator. We have previous experience in using Cosmos[9], the tool for statistical model-checking of the HASL logic, but the performance indicator for this case study does not require the power of a complex simulator like Cosmos, that has to take into account the full power of hybrid automata, moreover the use of another tool would have required to translate the net from the GreatSPN format to the Cosmos one.

We took the second approach: the net has been modified so that the SLA satisfaction can be computed based on transition throughputs. Since the SLA requires to compute the percentage of calls for which the time period from the time of the call until the time the technician reaches the site is less or equal to 60 minutes, it is enough to set a timeout of 60 minutes whenever a call enters the *OpenRequests* (which is the same time at which the transitions *anyCalls* and *taggedCalls* fire) and to check whether the associated technician reaches the site before or after the timeout has expired.

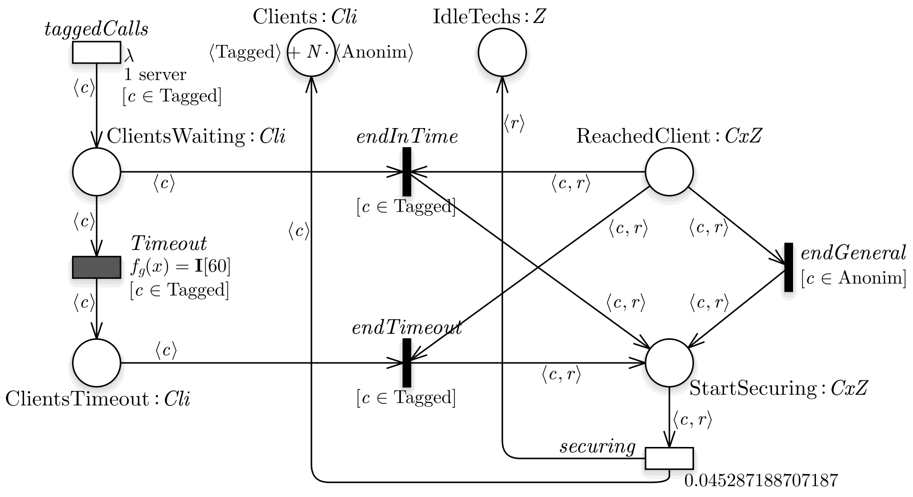


Fig. 10. The model of the system for passage time computation with DSPNs.

Figure 10 shows the modification to the model of Figure 7. Note that a timeout is set only when transition *taggedCalls* fires, since, as already done for the CSL^{TA} solution, we know (from, for ex. [8] that observing a single selected client is enough for correctly computing the SLA. When the technician r associated to the tagged client c reaches the intervention site (a token of color $\langle c, Tagged \rangle$ reaches place *ReachedClient*) the net immediately fires either transition *endInTime* (if the deterministic transition *Timeout* has not fired yet) or transition *endTimeout* (if the deterministic transition *Timeout* has already fired, meaning that the deadline of 60 minutes was violated). The probability of the SLA is then computed as the ratio between the throughput of transition *endInTime* and the sum of the throughput of transitions *endInTime* and *endTimeout*.

Note that, since there is a single tagged client, in any one state there is at most a single deterministic transition enabled, therefore the model of Figure 10, or, better, its unfolding, is a DSPN model [1]. Using the DSPN solver of GreatSPN we could check that the two techniques for SLA computation (CSL^{TA} formula or model modification to include a timeout) are equivalent or not.

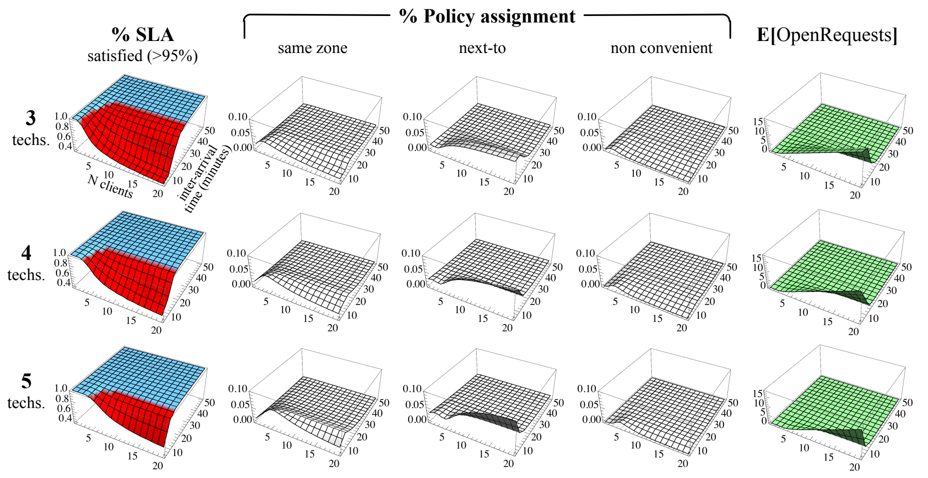


Fig. 11. Results space obtained by running multiple simulations.

Figure 11 shows the result space of the model (for the Erlang-3 case) . Since the solution technique is simulation we could introduce Erlang-3 as distribution for SWN transitions. The net was not unfolded, as the GreatSPN simulator for SWN was used, which already includes a number of common activity.

Each point in the charts corresponds to a simulation run of the SWN simulator. 600 separate simulations were run in batch to generate the result space. We have used a batch method with 95% confidence and 10% accuracy. Simulations end when all the default performance indicators (mean number of tokens in places and throughput of transitions) satisfy the confidence and accuracy requirements. Each row of Figure 11 shows the results for 3, 4 and 5 technicians in the system. Each 3D plot depicts the results for a varying number of clients in the system (1 to 20), and for different inter-arrival times of the client' calls (from 5 to 60 minutes). The first column is the probability of respecting the SLA requirement: the red zone stays below the 95% requirement, while the blue zone stays above. The second, third and fourth columns show the probability of each technicians to have to move between far zones (non-convenient), near zones (next-to) or to remain in the same zone when a new ticket is assigned. Finally, the last column shows the number of open requests in the system, which measures the system stability (high inter-arrival and not-enough technicians will end up in a divergent system).

Lessons learned – case study. The plot shows that for the average client call rate (about 1 call every hour), the system is well beyond the red zone of the SLA plot, i.e. it is expected to maintain the SLA requirement. The critical inter-arrival time (which may happen in a crisis situation) obviously depends on the number of available technicians. With 3, 4 and 5 technicians, it is about 26, 18 and 13 minutes of calls inter-arrival times, respectively. This information may be used to design a pro-active policy, where for instance if the last N client calls have arrived with a 20 minute time delay, and the company has currently 4 technicians, it should start alerting another technician (not currently in service) to be ready for any incoming task.

Lessons learned – modelling and tools. The simulation proved quite effective, but the analysis would have greatly profited by the possibility of specifying different accuracies for different performance indicators and by the possibility of computing the accuracy of simple formulas (like the ratio among throughputs that we use to compute the SLA).

Another aspect that was important is the simulator ability to compute the throughput also of immediate transitions: this is a feature that is not always available but that is very convenient to have.

We have also observed that the availability of a DSPN numerical solver allowed to check whether the model in Figure 10 and that in Figure 7 lead to the same SLA computation (a comparison difficult to do if only simulation results are available). The DSPN solver was also used to check that the setting of the parameter *initTrn* of the timed automaton that defines the SLA, that identifies the time to reach steady state, is correctly set.

8 Conclusions and perspectives

In this work we have developed a model of a utility company control room, based on real data provided by the company. The goal was studying the critical conditions that make the system in use unable to fulfil a required SLA. Using histograms of the available data, we observed that most quantities involved can be fitted accurately using Erlang distributions, thus making the model easy to specify using standard stochastic Petri nets.

The problem of addressing the high variability of the geographical positions is only considered in a discretized form. By using a different technique (using the actual map of the area) it would be possible to simulate the movement in a more accurate way. However, since we are mostly interested in the average behaviour, a discretized average distribution of the travelling time should provide relatively accurate measures.

Another aspect that we have not considered is the modelling of assignment policies different from the one currently in use by the company. For instance, it is not clear if, under critical conditions, it is better to not move technicians out of their zone (since they would spend more time travelling). This is left to further investigations.

References

1. Ajmone Marsan, M., Chiola, G.: On Petri nets with deterministic and exponentially distributed firing times. In: Advances in Petri Nets. Lecture Notes in Computer Science, vol. 266/1987, pp. 132–145. Springer Berlin / Heidelberg (1987)
2. Ajmone Marsan, M., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM Transactions on Computer Systems 2, 93–122 (May 1984), `http://doi.acm.org/10.1145/190.191`
3. Amparore, E.G.: A New GreatSPN GUI for GSPN Editing and CSLTA Model Checking. In: Norman, G., Sanders, W. (eds.) Quantitative Evaluation of Systems, Lecture Notes in Computer Science, vol. 8657, pp. 170–173. Springer International Publishing (2014)
4. Amparore, E.G., Barbot, B., Beccuti, M., Donatelli, S., Franceschinis, G.: Simulation-based verification of hybrid automata stochastic logic formulas for stochastic symmetric nets. In: Proceedings of the 1st ACM SIGSIM Conference on Principles of Advanced Discrete Simulation. pp. 253–264. ACM (2013)
5. Amparore, E.G., Beccuti, M., Donatelli, S., Franceschinis, G.: Probe automata for passage time specification. In: Proceedings of the 2011 Eighth International Conference on Quantitative Evaluation of SysTems. pp. 101–110. QEST 2011, IEEE Computer Society, Washington, DC, USA (2011)
6. Amparore, E.G., Donatelli, S.: Optimal Aggregation of Components for the Solution of Markov Regenerative Processes, pp. 19–34. Springer International Publishing, Cham (2016)
7. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.: Model-checking continuous-time Markov chains. ACM Transactions on Computational Logic 1(1), 162–170 (2000)
8. Balbo, G., Beccuti, M., De Pierro, M., Franceschinis, G.: First Passage Time Computation in Tagged GSPNs with Queue Places. The Computer Journal 54, 653–673 (2010), first published online July 22, 2010.
9. Ballarini, P., Djafri, H., Duflot, M., Haddad, S., Pekergin, N.: COSMOS: a statistical model checker for the hybrid automata stochastic logic. In: Proceedings of the 8th International Conference on Quantitative Evaluation of Systems (QEST'11). pp. 143–144. IEEE Computer Society Press, Aachen, Germany (Sep 2011)
10. Ballarini, P., Djafri, H., Duflot, M., Haddad, S., Pekergin, N.: HASL: An expressive language for statistical verification of stochastic models. In: Proceedings of the 5th International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'11). pp. 306–315. Cachan, France (May 2011)
11. Bause, F., Buchholz, P.: Queueing petri nets with product form solution. Performance Evaluation 32(4), 265–299 (1998)
12. Bernardi, S., Donatelli, S., Horváth, A.: Implementing compositionality for stochastic petri nets. International Journal on Software Tools for Technology Transfer (STTT) 3(4), 417–430 (2001)
13. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: Stochastic well-formed colored nets and symmetric modeling applications. IEEE Transactions on Computers 42(11), 1343–1360 (1993)
14. Clark, A., Gilmore, S.: State-aware performance analysis with extended stochastic probes. In: Proceedings of the 5th European Performance Engineering Workshop. pp. 125–140. EPEW '08, Springer-Verlag, Berlin, Heidelberg (2008)

15. Dingle, N.J., Harrison, P.G., Knottenbelt, W.J.: HYDRA: HYpergraph-based Distributed Response-time Analyser. In: International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2003). pp. 215–219 (June 2003)
16. Donatelli, S., Haddad, S., Sproston, J.: Model checking timed and stochastic properties with CSL^{TA}. IEEE Transactions on Software Engineering 35(2), 224–240 (2009)
17. Dugdale, J., Pavard, J., Soubie, B.: A pragmatic development of a computer simulation of an emergency call center. Designing cooperative systems: the use of theories and models pp. 241–256 (2000)
18. German, R.: Iterative analysis of Markov regenerative models. Performance Evaluation 44, 51–72 (April 2001)
19. Hillston, J.: Compositional markovian modelling using a process algebra. In: Computations with Markov chains, pp. 177–196. Springer (1995)
20. Jain, S., McLean, C.: Simulation for emergency response: a framework for modeling and simulation for emergency response. In: Proceedings of the 35th conference on Winter simulation: driving innovation. pp. 1068–1076. Winter Simulation Conference (2003)
21. Kulkarni, V.G.: Modeling and analysis of stochastic systems. Chapman & Hall Ltd., London, UK (1995)
22. Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with generalized stochastic Petri nets. John Wiley & Sons, Inc. (1994)
23. Obal, W.D., II, Sanders, W.H.: State-space support for path-based reward variables (1998)
24. de QV Lima, M.A., Maciel, P.R., Silva, B., Guimarães, A.P.: Performability evaluation of emergency call center. Performance Evaluation 80, 27–42 (2014)