An Analysis Pathway for the Quantitative Evaluation of Public Transport Systems

Andrea Vandin\textsuperscript{1}, Mirco Tribastone\textsuperscript{1}, and Stephen Gilmore\textsuperscript{2}

\textsuperscript{1} Electronics and Computer Science, University of Southampton
\textsuperscript{2} Laboratory for Foundations of Computer Science, University of Edinburgh

Abstract. We consider the problem of evaluating quantitative service-level agreements in public services such as transportation systems. We describe the integration of quantitative analysis tools for data fitting, model generation, simulation, and statistical model-checking, creating an analysis pathway leading from system measurement data to verification results. We apply our pathway to the problem of determining whether public bus systems are delivering an appropriate quality of service as required by regulators. We exercise the pathway on service data obtained from Lothian Buses about the arrival and departure times of their buses on key bus routes through the city of Edinburgh. Although we include only that example in the present paper, our methods are sufficiently general to apply to other transport systems and other cities.

1 Introduction

Modern public transport systems are richly instrumented. The vehicles in a modern bus fleet are equipped with accurate GPS receivers, Wi-Fi, and on-board communications, allowing them to report their location for purposes such as fleet management and arrival-time prediction. High-frequency, high-resolution location data streams back from the vehicles in the fleet to be consumed by the predictive models used in real-time bus tracking systems.

We live in a data-hungry world. Users of public transport systems now expect to be able to access live data about arrival times, transit connections, service disruptions, and many other types of status updates and reports at almost every stage of their journey. Studies suggest that providing real-time information on bus journeys and arrival times in this way encourages greater use of buses [1] with beneficial effects for the bus service. In contrast, when use of buses decreases, transport experts suggest that this aggravates existing problems such as outdated routes, bunching of vehicles, and insufficient provision of greenways or bus priority lanes. Each of these problems makes operating the bus service more difficult. Bus timetables become less dependable, new passengers are discouraged from using the bus service due to bad publicity, which leads inevitably to budget cuts that further accelerate the decline of the service.

Service regulators are no less data-hungry than passengers, requiring transport operators to report service-level statistics and key performance indicators which are used to assess the service delivered in practice against regulatory requirements on the quality of service expected. Many of these regulatory requirements relate to punctuality of buses,
defined in terms of the percentage of buses which depart within the window of tolerance around the timetabled departure time; and reliability of buses, defined in terms of the number of miles planned and the number of miles operated. The terms schedule adherence or on-time performance are also used to refer to the degree of success of a transportation service running to the published timetable.

With the aim of helping service providers to be able to work with models which can be used to analyse and predict on-time performance, we have connected a set of modelling and analysis tools into an analysis pathway, starting from system measurement data, going through data fitting, model generation, simulation and statistical model-checking to compute verification results which are of significance both to service providers and to regulatory authorities.

The steps of the analysis pathway, depicted in Figure 1, are as follows:

1. Data is harvested from a bus tracking system to compile an empirical cumulative distribution function data set of recorded journey times for each stage of the bus journey. In this paper, we generate inputs to the system using the BusTracker automatic vehicle tracking system developed by the City of Edinburgh council and Lothian Buses [2].
2. The software tool HyperStar [3] is used to fit phase-type distributions to the data sets.
3. A phase-type distribution enables a Markovian representation of journey times which can be encoded in high-level formalisms such as stochastic process algebras. In particular, we use the Bus Kernel model generator (BusKer), a Java application which consumes the phase-type distribution parameters computed by HyperStar and generates a formal model of the bus journey expressed in the Bio-PEPA stochastic process algebra [4]. In addition, the BusKer tool generates an expression in MultiQuaTEx, the query language supported by the MultiVeStA statistical model-checker [5]. This is used to formally express queries on service-level agreements about the bus route under study.
5. MultiVeStA is hooked to the simulation engine of the Bio-PEPA Eclipse Plugin, consuming individual simulation events to evaluate the automatically generated MultiQuaTEx expressions. It produces as its results plots of the related quantitative properties.

We are devoting more than the usual amount of effort to ensuring that our tools are user-friendly and easy-to-use. This is because we want our software tools to be used “in-house” by service providers because only then can service providers retain control over access to their own proprietary data about their service provision. With respect to ease-of-use in particular, making model parameterisation simpler is a crucial step in making models re-usable. Because vehicle occupancy fluctuates according to the seasons, with the consequence that buses spend more or less time at bus stops boarding passengers, it is essential to be able to re-parameterise and re-run models for different data sets from different months of the year.

It is also necessary to be able to re-run an analysis based on historical measurement data if timetables change, or the key definitions used in the evaluation of regulatory
requirements change. Evidently, a high degree of automation in the process is essential, hence our interest in an analysis pathway.

**Related work.** We are not aware of other toolchains based on formal methods for the quantitative analysis of public transportation systems. The same bus system is studied in [7], from which we inherited the data-set acquisition and its fitting to phase-type distributions. Differently from our approach, in [7] different software tools are individually used to perform distinct analyses of the scenario. For example, the Traviando [8] post-mortem simulation trace analyser is fed with precomputed simulation traces of a Bio-PEPA model similar to ours, and the probabilistic model checker PRISM [9] is used to analyse a corresponding model defined in the PRISM’s input language.

More generally, our approach takes inspiration from generative programming techniques [10], in that we aim at automatic generation of possibly large stochastic process algebra models (our target language) from more compact higher-level descriptions (i.e., the timetable representation and the model parameters).

The generation of MultiQuaTEx expressions fits well with the literature on higher-level specification patterns for temporal logic formulae [11]. Temporal logics, the common property specification languages of model checkers, are not in widespread use in industry, as they require a high degree of mathematical maturity and experience in formal language theory. Furthermore, most system requirements are written in natural language, and often contain ambiguities which make it difficult to accurately formalise them in any temporal logic. In an attempt to ease the use of temporal logic, [11] gives a pattern-based classification for capturing requirements, paving the way for semi-automated methodologies for the generation of inputs to model checking tools. From a general perspective, in this work we fix the property patterns of interest, and completely hide property generation and evaluation to the end user.

**Paper structure.** Section 2 motivates our reasons for constructing a stochastic model of the problem. Section 3 describes the analysis problem in greater detail and presents the key definitions used in the paper. Section 4 describes how measurement data is transformed into model parameters to initiate the analysis which is undertaken. Section 5 describes the software tools in the analysis pathway. Section 6 presents the software analyser which combines these disparate tools. Section 7 presents our analysis in terms of the key definitions of the paper. Conclusions are presented in Section 8.
2 The importance of modelling

We are working in a context where we have an existing operational instrumented system which is gathering data on its service provision. However, instead of working directly with the data we will construct a high-level stochastic model of the data, using Erlang distributions with a number of phases and an exponentially-distributed rate to describe a journey between two timing points. The timing points are those bus stops which are named in the published timetable for the route.

We work with a stochastic model instead of working with the data directly because, importantly, we are not concerned with detecting post-hoc violations of the regulations from measurement data. Rather, we are trying to estimate the likelihood of future violations of the regulations in journeys which are similar to those which we have seen, although not identical to them. For this reason we generalise from the data to a stochastic process which describes the data well in a precise sense statistically.

Measurement data only records particular historical events: it does not generalise. For example, if our collected observations tell us that a bus journey can take five, six, eight, or nine minutes it is reasonable to assume that it can also take seven minutes, although this is not actually recorded in the data. Generalising from data like this is the act of abstraction which is at the heart of modelling. Models have many other strengths.

– Models are intellectual tools for understanding systems. They can be understood by service operators and used to communicate with regulators or other stakeholders.
– Models impose order on data, shaping it to become information which can be used in making decisions about how systems are modified.
– Models are concise and can be easily compared. In contrast, data is verbose and difficult to compare.
– Models are high-level and structured. Data is low-level and unstructured.
– Models are scalable. The number of phases in the stochastic description of the journey can be easily modified in order to explore the effect of different routes. Adding more phases corresponds to lengthening the route; removing phases corresponds to shortening it. Data is not scalable in this way.
– Models are tuneable. Rates can be easily adjusted in order to explore the effect of increased congestion on the routes or the effect of changes in the speed limit on parts of a route. Data is not tuneable in this way.
– Models are editable in a way which data is not. We can predict the effect of planned engineering works on journey times by using measurement data which incorporates the effect of previous engineering works and scaling it to fit if needed.

Because measurement data consists of a finite number of observations we know that there is additional possible behaviour which we have not seen. Stochastic modelling is a powerful reasoning tool allowing us to estimate the likelihood of values which we have not seen based on the frequency of occurrences of those values which we have seen. Conclusions drawn solely from the data would be misleading in that we would be led to believe that some combinations of events were impossible when in fact they are only relatively unlikely.

Finally, in moving from the data to the stochastic model we only need to ensure that we have identified a suitable stochastic process to represent the data. In Section 4 we will explain the use of the Kolmogorov-Smirnov statistical test to ensure this.
3 The analysis problem

The notion of punctuality which we are considering here is defined in terms of the concept of a “window of tolerance” around the departure times advertised in the timetable. Perhaps not very surprisingly, this notion differs between different operators and different countries, for instance:

- According to Transport for London, a bus is considered to be on time if it departs between two minutes and 30 seconds early and five minutes late [12].
- In England outside London, a bus is considered to be on time if it departs between one minute early and five minutes, 59 seconds late [13].
- In Scotland, according to the definitions reported in the Scottish Government’s *Bus Punctuality Improvement Partnerships* report, a bus is considered to be on time if it departs between one minute early and five minutes late [14].

Each region has a definition of on-time in terms of the window of tolerance but clearly when comparing the quality of service in one region with the quality of service in another it is necessary to be able to re-evaluate the service delivered historically against the definitions used by another.

Our problem is to generate a mathematical model which allows us to analyse the following properties, for each bus stop advertised in a timetable.

**P1.** The average time of departure from the bus stop.

**P2.** The average distance of the departure time from the timetabled time.

**P3.** The probability that a bus departs on time.

**P4.** The probability of an early departure.

**P5.** The probability of a late departure.

Since the window of tolerance is asymmetric with respect to the timetable, property P2 is formally defined as the expected value of the absolute value of the difference between the time of departure and the respective timetabled time. Note that properties P3–P5 clearly depend on the notion of punctuality adopted.

In this paper we focus on a particular bus route. Specifically, we consider the Lothian Buses #31 bus on its journey from North Bridge in Edinburgh’s city centre to Bonnyrigg Toll in the south, passing through the Cameron Toll and Lasswade Road timing points. The same bus route has been studied in [7], as discussed in Section 1. Table 1 shows its timetable, where the departure time from North Bridge is taken as the reference time 0.

<table>
<thead>
<tr>
<th>Timing point</th>
<th>Code</th>
<th>Timetable (in minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>North Bridge</td>
<td>NB</td>
<td>0</td>
</tr>
<tr>
<td>Cameron Toll</td>
<td>CT</td>
<td>16</td>
</tr>
<tr>
<td>Lasswade Road</td>
<td>LR</td>
<td>24</td>
</tr>
<tr>
<td>Bonnyrigg Toll</td>
<td>BT</td>
<td>34</td>
</tr>
</tbody>
</table>

Table 1. Timetable for the #31 bus operated by Lothian Buses in Edinburgh.
4 From measurement data to model parameters

We now turn our attention to how model parameters are found for a BusKer input.

4.1 The BusTracker data

The raw data which is the input to the pathway is a dataset compiling measured journey times between timing points, forming an empirical distribution over the journey times. This data set incorporates the unpredictable effects of many different types of delays which the service can experience, due to traffic congestion and competition with other buses for access to bus stops. The data is obtained from the passenger waiting time website for Lothian Buses [2]. We collected raw data from this website by scripting, and wrote the data to a file for post-processing. This data is available from the QUANTICOL website at http://www.quanticol.eu. Post-processing identified departure events in the data, and computed journey times between timing points, compiling an empirical distribution of journey times.

4.2 HyperStar

Phase-type distributions are a class of probability distributions formally defined as the time to absorption of a continuous-time Markov chain (CTMC). They are very popular in the performance evaluation community because they can approximate, with arbitrary precision, generally-distributed events by means of appropriate stages (or phases) of independent exponential distributions [15]. Concretely, this allows a modeller to accurately describe general systems exhibiting nonexponential distributions using a Markov chain as the underlying mathematical formalism. An Erlang distribution, hereafter denoted by $Erl(k, \lambda)$, is a special case of a series of $k > 0$ exponential phases, each with mean duration given by $1/\lambda$, with $\lambda > 0$. The mean duration of the distribution is $k/\lambda$. It is particularly useful for modelling activities with low variance — in the limit $k \to \infty$ it behaves deterministically. It has been found in [7] to approximate bus journey times well. For this reason, our current implementation supports Erlang distributions only, although an extension to general phase-type distributions is possible.

Given a set of observed durations, the problem is to find the parameters of a phase-type distribution that fits them most appropriately (according to some criterion of optimality). For an Erlang distribution, this amounts to finding the values of the parameters $k$ and $\lambda$ that completely characterise it. For this, we use HyperStar, a new software tool released in 2013 [3] to convert our empirical distribution to an analytic one.

4.3 The Kolmogorov-Smirnov test

The Kolmogorov-Smirnov test can be used to quantify the distance between an empirical distribution function and a cumulative distribution function. The test can be used to answer the question whether the data would be thought to have come from the specified distribution. We applied this test to the empirical data and the Erlang distributions returned by HyperStar. The null hypothesis was accepted with credible test statistics and critical values in all three cases meaning that the Erlang distributions are suitable stochastic process descriptions of the data.
5 The analysis pathway

In this section we describe in more detail the modelling tools and formal languages of our analysis pathway, as well as their integration.

5.1 BusKer

The Bus Kernel model generator (BusKer) is a Java command-line application that takes as input the specification of the window of tolerance (parameters $\text{maxAdvance}$ and $\text{maxDelay}$, respectively) and a BusKer specification, i.e. a comma-separated representation of the timetable and the Erlang distribution for the time to reach the next timetabled bus stop. For instance, in this paper we will consider the parameter fitting used in [7] for the route in Table 1, which yields the following BusKer specification:

# Timing point, Code, Timetable, $k$, $\lambda$
North Bridge, NB, 0, 0.105, 6.47
Cameron Toll, CT, 16, 83, 8.79
Lasswade Road, LR, 24, 98, 10.54
Bonnyrigg Toll, BT, 34, --, --

As a result, BusKer generates the inputs for the next two steps of our analysis pathway: a Bio-PEPA model of the bus service, and the MultiQuaTEx expression analysed by MultiVeStA to state the quality of the studied bus service with respect to the provided window of tolerance.

5.2 Bio-PEPA

Although designed for application to modelling problems in biological systems, Bio-PEPA has been effectively applied to problems as diverse as crowd dynamics [16], emergency egress [17] and swarm robotics [18]. Here, we use it because it is a stochastic process algebra with an underlying CTMC semantics; as such it is possible to encode phase-type distributions in Bio-PEPA. Furthermore, it is implemented by a software tool, the BioPEPA Eclipse Plugin, which supports stochastic simulation in a way that is easily consumable by MultiVeStA. Referring the reader to [4] for the complete formal account, we will use the following simplified BusKer specification to briefly overview the language:

# Timing point, Code, Timetable, $k$, $\lambda$
North Bridge, NB, 0, 3, 0.19
Cameron Toll, CT, 16, --, --

BusKer will generate the specification shown in Listing 1.1. The model concerns the five species $\text{NB}_1$, $\text{NB}_2$, $\text{NB}_3$, $\text{CT}_1$ and $\text{DepaFromNb}$, representing the number of buses in North Bridge ($\text{NB}_1$), those in the first ($\text{NB}_2$) and second ($\text{NB}_3$) part of the journey from North Bridge to Cameron Toll, and the number of buses at Cameron Toll.
// Definitions of rate functions
// Functions for North Bridge -> Cameron Toll (3 phases)
NBtoCT_1 = [0.19 * NB_1];
NBtoCT_2 = [0.19 * NB_2];
NBtoCT_ARRIVED = [0.19 * NB_3];
// Definitions of processes
// Processes for North Bridge -> Cameron Toll (3 phases)
NB_1 = NBtoCT_1 << ;
NB_2 = NBtoCT_1>> + NBtoCT_2 << ;
NB_3 = NBtoCT_2>> + NBtoCT_ARRIVED<< ;
// Cameron Toll is the final stop.
CT_1 = NBtoCT_ARRIVED>> ;
// State observations
DepFromNB = NBtoCT_1>> ;
// Initial configuration of the system (one bus in North Bridge)
NB_1[1] << NB_2[0] >> NB_3[0] <<<
CT_1[0] >>
DepFromNB[0]

Listing 1.1. The Bio-PEPA model generated by BusKer for the scenario of (2)

(CT_1). Finally, DepFromNB is an observer process used to count the number of departures from North Bridge. Lines 16–18 provide the initial system configuration: one bus is in North Bridge, while all the other populations are set to 0. A reaction prefix such as NBtoCT_1<< in a species definition (e.g. NB_1 = NBtoCT_1<< at line 8) causes the population count of that species (NB_1) to decrease by one when the reaction NBtoCT_1 occurs. In particular, line 3 specifies that the reaction NBtoCT_1 occurs with a rate obtained by multiplying the constant 0.19 with the population count of the species NB_1. In our model we follow the journey of a single prototypical bus, so this product in the rate expression acts as a switch, allowing the reaction to fire when a bus is present and preventing it from firing at other times (because the rate evaluates to 0 when a bus is not present). Similar to this is the case of the reaction prefix NBtoCT_1>>, the only difference being that in this case the involved population counts increase by one. For example, line 14 specifies that the population of the species DepFromNB increases by one whenever the reaction NBtoCT_1 occurs, making DepFromNB a de facto counter for the departures of buses from North Bridge. In contrast, line 10 specifies that the population of the species NB_3, i.e. the buses in the second part of the journey from North Bridge to Cameron Toll, increases by one whenever a bus moves from the first to the second part of the journey (NBtoCT_2>>), and decreases by one whenever a bus arrives at Cameron Toll (NBtoCT_ARRIVED<<).

The Bio-PEPA model built from the input to BusKer is a statistically-plausible stochastic model of the journey of a prototypical bus travelling from the first to the last specified bus stops, using the Erlang parameters learnt from the measurement data which has been processed by HyperStar. Clearly, the predictive power of this model depends crucially on the quality and scope of the data supplied to HyperStar. Because it is ultimately learnt from data, the model will incorporate the effects of contention for bus stops with other buses serving the same route, and, for good or ill, it will incorporate the influence of any atypical events (e.g. unusually long delays) which occurred during the measurement period.
5.3 MultiVeStA

MultiVeStA [5] is a recently-developed Java-based distributed statistical model checker which allows its users to enrich existing discrete event simulators with automated and statistical analysis capabilities. The analysis algorithms of MultiVeStA do not depend on the underlying simulation engine: MultiVeStA only makes the assumption that multiple discrete event simulations can be performed on the input model. The tool has been used to reason about collision-avoidance robots [19], volunteer clouds [20] and crowd-steering [21] scenarios.

MultiVeStA comes with a property specification language, MultiQuaTEx, which makes it possible for users to express and evaluate many properties over the same simulated path. In contrast to Continuous Stochastic Logic [22, 23] and Probabilistic Computation Tree Logic [24] commonly used in probabilistic and statistical model checking, MultiQuaTEx allows users to define their own parametric recursive temporal operators within the logic itself, and to query real-typed properties, rather than just probabilities. In particular, with MultiQuaTEx we can express all the properties listed in Section 3.

Listing 1.2 provides a MultiQuaTEx expression to estimate the expected departure times from each bus stop of interest (property P1) using the BusKer specification (1). Lines 5–6 specify the three expected values to be estimated, i.e. the departure times from North Bridge, Cameron Toll and Lasswade Road. Lines 1–4 specify a parametric recursive temporal operator which returns, for each simulation, the departure time of the bus from the bus stop specified as the parameter. This is iteratively evaluated by performing steps of simulations (triggered by the operator #) until the guard of the if statement is satisfied, i.e. until a departure occurs from the selected bus stop. Intuitively, as discussed in Section 5.2, the Bio-PEPA model generated by BusKer counts the departures from each bus stop by defining observer processes DepsFromNB, DepsFromCT and DepsFromLR whose populations are incremented every time the corresponding event happens. Finally, we note that MultiVeStA can access information about the current state of the simulation with s.rval(observation), where observation can be the current simulated time (i.e. time), or the current population of a species (e.g. "DepsFromNB").
Fig. 2. Plot generated by TBA for the specification presented in Equation (1), and the (1.5) window of tolerance.

6 Tool chaining: The Bus Analyzer

The last three tools of our analysis pathway, highlighted in Figure 1, have been integrated in a single tool called TBA: The Bus Analyzer. TBA hides from the user the steps involved in the generation of the Bio-PEPA model and of the MultiQuaTEX expression, as well as the invocation of MultiVeStA. TBA can be downloaded, together with our BusKer specification (1), from the Tools section of the QUANTICOL web-site at http://www.quanticol.eu/.

A first clear advantage brought by TBA is the automation of the analysis phase, as the user only has to execute the command

```
java -jar TBA.jar busker scenario.busker maxAdv maxDelay [servers]
```

(3)

where `scenario.busker` is a file containing a BusKer specification, and `maxAdv` and `maxDelay` specify the required window of tolerance (in minutes). The optional parameter `servers` gives the degree of parallelism to automatically distribute independent simulations across CPU cores.

TBA evaluates properties P1–P5. The results are provided to the user via a GUI consisting of an interactive scatter plot containing a point for each studied property, and
// Definitions of rate functions
// Functions for North Bridge -> Cameron Toll (105 phases)
NBtoCT_1 = [6.47 * NB_1];
...
NBtoCT_104 = [6.47 * NB_104];
NBtoCT_ARRIVED = [6.47 * NB_105];
// Functions for Cameron Toll -> Lasswade Road (83 phases)
CTtoLR_1 = [8.79 * CT_1];
...
CTtoLR_82 = [8.79 * CT_82];
CTtoLR_ARRIVED = [8.79 * CT_83];
// Functions for Lasswade Road -> Bonnyrigg Toll (98 phases)
LRtoBT_1 = [10.54 * LR_1];
...
LRtoBT_97 = [10.54 * LR_97];
LRtoBT_ARRIVED = [10.54 * LR_98];

// Definitions of processes
// Processes for North Bridge -> Cameron Toll (105 phases)
NB_1 = NBtoCT_1 << ;
NB_2 = NBtoCT_1 >> + NBtoCT_2 << ;
...
NB_104 = NBtoCT_103 >> + NBtoCT_104 << ;
NB_105 = NBtoCT_104 >> + NBtoCT_ARRIVED << ;
// Processes for Cameron Toll -> Lasswade Road (83 phases)
CT_1 = NBtoCT_ARRIVED >> + CTtoLR_1 << ;
CT_2 = CTtoLR_1 >> + CTtoLR_2 << ;
...
CT_82 = CTtoLR_81 >> + CTtoLR_82 << ;
CT_83 = CTtoLR_82 >> + CTtoLR_ARRIVED << ;
// Processes for Lasswade Road -> Bonnyrigg Toll (98 phases)
LR_1 = CTtoLR_ARRIVED >> + LRtoBT_1 << ;
LR_2 = LRtoBT_1 >> + LRtoBT_2 << ;
...
LR_97 = LRtoBT_96 >> + LRtoBT_97 << ;
LR_98 = LRtoBT_97 >> + LRtoBT_ARRIVED << ;
// Bonnyrigg Toll is the final stop.
BT_1 = LRtoBT_ARRIVED >> ;
// State observations
DepsFromNB = NBtoCT_1 >> ; DepsFromCT = CTtoLR_1 >> ; DepsFromLR = LRtoBT_1 >> ;
// Initial configuration of the system (one bus in North Bridge)
NB_1[1] <<= ... <<= NB_105[0] <<=
CT_1[0] <<= ... <<= CT_83[0] <<=
LR_1[0] <<= ... <<= LR_98[0] <<= BT_1[0] <<=
DepsFromNB[0] <<= DepsFromCT[0] <<= DepsFromLR[0]

Listing 1.3. The Bio-PEPA model generated by BusKer for input Equation (1)

are also stored on disk. For example, the interactive plot allows the modeller to hide some properties, to apply zooming or rescaling operations, to change the considered boundaries, and to save the plot as a picture. Figure 2 depicts the plot obtained for the BusKer specification (1) when considering the Scottish window of tolerance, i.e., maxAdv=1 and maxDelay=5. A discussion of the analysis is provided in Section 7. In the remainder of this section we focus on the usability and accessibility advantages provided by chaining the three tools.

Clearly, given that TBA hides the generation of the model and of the property, as well as their analysis, the user is not required to learn the two formal languages, nor to use their related tools. Furthermore, for realistic bus scenarios the generated Bio-PEPA models and MultiQuaTeX expressions tend to be large and thus error-prone to write down manually. For example, the Bio-PEPA model generated by TBA for our scenario
1 // Static part of the expression: the parametric temporal operators
2 // Probabilities of departing on time, too early or too late
3 DepartedOnTime(depsFromBusStop, timeTabledDep, maxAdv, maxDelay) =
4 if { s.rval(depsFromBusStop) == 1.0 } then CheckIfDepOnTime(depsFromBusStop, timeTabledDep, maxAdv, maxDelay)
5 else # DepartedOnTime(depsFromBusStop, timeTabledDep, maxAdv, maxDelay)
6 fi;
7
8 CheckIfDepOnTime(depsFromBusStop, timeTabledDep, maxAdv, maxDelay) =
9 if { timeTabledDep - s.rval("time") > maxAdv }
10 then 0.0
11 else if { s.rval("time") - timeTabledDep > maxDelay }
12 then 0.0 else 1.0
13 fi;
14
15 DepartedTooEarly(depsFromBusStop, timeTabledDep, maxAdv) = // like DepartedOnTime
16 DepartedTooLate(depsFromBusStop, timeTabledDep, maxDelay) = // like DepartedOnTime
17 // Expected departure time
18 DepartureTime(depsFromBusStop) = // as in Listing 1.2
19 // Expected deviation from the timetabled departure time
20 DistanceFromTimeTable(depsFromBusStop, timeTabledDep) =
21 if { s.rval(depsFromBusStop) == 1.0 } then ComputeDistanceFromTimeTable(depsFromBusStop, timeTabledDep)
22 else # DistanceFromTimeTable(depsFromBusStop, timeTabledDep)
23 fi;
24
25 ComputeDistanceFromTimeTable(depsFromBusStop, timeTabledDep) =
26 if { timeTabledDep > s.rval("time") } then timeTabledDep - s.rval("time")
27 else s.rval("time") - timeTabledDep
28 fi;
29
30 // Static part of the expression: the 15 properties to be estimated
31 eval E[ DepartureTime("DepsFromNB") ];
32 eval E[ DistanceFromTimeTable("DepsFromNB",0.0)];
33 eval E[ DepartedOnTime("DepsFromNB",0.0,1.0,5.0)];
34 eval E[ DepartedTooEarly("DepsFromNB",0.0,0.1,0.0)];
35 eval E[ DepartedTooLate("DepsFromNB",0.0,0.5,0.0)];
36 // same eval clauses for "DepsFromCT", and "24.0" rather than 0.0
37
38 Listing 1.4. The MultiQuaTEx expression generated by BusKer

is almost 900 lines long, as sketched in Listing 1.3. This is due to the the fact that the journeys between bus stops are modelled using Erlang distributions with many phases, and each phase is associated with a distinct species (hence at least a line in the source code). More specifically, Listing 1.3 can be divided in four parts: lines 1–16 define the rates with which the modelled prototypical bus moves, lines 17–36 define the processes specifying the bus’s stochastic behaviour, lines 37–38 define the state observations of interest, while lines 39–44 specify the initial configuration of the system. The third section only depends on the number of considered bus stops, while, as depicted by the ellipsis, the other ones also depend on the number of phases of the provided BusKer specification.

The MultiQuaTEx expression generated by BusKer for our scenario is a fixed length for any window of tolerance. It is sketched in Listing 1.4 for the Scottish window of tolerance. Overall it evaluates fifteen properties, i.e., P1–P5 for each of the three bus stops. Lines 1–28 define the parametric recursive temporal operators which specify how to compute such properties, whereas lines 29–36 list the fifteen properties to be estimated. For each simulation, each temporal operator observes the bus stop provided as a parameter, specifically: DepartedOnTime, DepartedTooEarly and DepartedTooLate return
Table 2. Analysis results for the #31 bus operated by Lothian Buses in Edinburgh

<table>
<thead>
<tr>
<th></th>
<th>North Bridge</th>
<th>Cameron Toll</th>
<th>Lasswade Road</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>0.32</td>
<td>16.42</td>
<td>25.84</td>
</tr>
<tr>
<td>P2</td>
<td>0.32</td>
<td>1.28</td>
<td>2.13</td>
</tr>
<tr>
<td>P3</td>
<td>1.00</td>
<td>0.81</td>
<td>0.88</td>
</tr>
<tr>
<td>P4</td>
<td>0.00</td>
<td>0.19</td>
<td>0.04</td>
</tr>
<tr>
<td>P5</td>
<td>0.00</td>
<td>0.00</td>
<td>0.06</td>
</tr>
</tbody>
</table>

1 if the bus departed on time, too early, or too late, respectively. DepartureTime returns
the departure time of the bus, while DistanceFromTimeTable returns the absolute
value of the difference between the actual departure time and the timetabled one. That
expression does not depend on the number of phases of the BusKer specification, but
only on the number of timetabled bus stops. In particular, the expression can be divided
in a static part, which is given once, for any possible input specification, and a dynamic
one, which instead depends on the input specification. Thanks to their parametrisation,
the temporal operators (lines 1–27) do not depend on the input specification, and are
thus the static part of the expression. Lines 28–35 are the dynamic part of the expres-
sion, as five eval clauses instantiated with the timetabled departures and the window
of tolerance are needed for each bus stop considered.

7 Analysis of the scenario

In this section we present the analysis of our scenario using TBA. The results for the
Scottish window of tolerance [14] are summarised in Table 2. We fixed $\alpha = 0.05$ for
all properties, $\delta = 0.2$ for those regarding the expected departure times and deviations
from the timetable, and $\delta = 0.05$ for the probabilities. It was necessary to perform 1860
simulations to attain this confidence interval for all the 15 studied properties, requiring
less than 10 seconds in total, thus without requiring to resort to MultiVeStA’s capability
of distributing simulations.

These results suggest that buses tend to lose adherence with respect to the timetable
while performing the route. This effect is also observed in practice: the variance of
departure times is seen to increase along the route. However, this does not necessarily
Correspond to a degradation of the quality of service, as a greater deviation from the
timetable generated by delayed departures may correspond to a better quality of service
than a smaller deviation generated by anticipated departures.

In order to have further insights into the quality of the studied #31 bus service,
the last three rows of Table 2 provide the probabilities that a bus departs on time, too
early, or too late from each bus stop. Consistent with the slight deviation found from the
timetable, we have that buses always depart on time from the North Bridge stop. Then,
buses tend to perform the route from North Bridge to Cameron Toll too quickly, causing
early departures in 20% of cases. The quality of service improves at Lasswade Road,
where only 12% of departures are outside the window of tolerance. This may seem to
contradict the results about the deviation from the timetable, as we found that at the
Lasswade Roll time point there is a greater deviation from the timetable with respect to that at Cameron Toll. However, this is explained by noticing that our analysis tells us that the deviations from the timetable are mainly caused by anticipated departures at Cameron Toll, and by delayed departures at Lasswade Road. In fact, we first of all notice that the expected departure time is 0.42 minutes greater than the timetabled one at Cameron Toll, and 1.84 at Lasswade Road.

Furthermore, we have early departures in 20% of cases and no late departures at Cameron Toll. Instead, at Lasswade Road we have early departures in only 4% of cases, and late departures in 6% of cases. In conclusion, we find that buses tend to spend more time than is scheduled in performing the journey from Cameron Toll to Lasswade Road, thus absorbing the effect of earlier departures from Cameron Toll, leading to a halved percentage of departures there outside the window of tolerance with respect to Cameron Toll.

It is worthwhile to note that analysing the quality of service with respect to other windows of tolerance only requires launching the command (3) with different parameters. For example, Table 3 compares the results using the Scottish window of tolerance (SC), and the English one (EN), the latter obtained by setting parameters maxAdv=1 and maxDelay=5.59. Not surprisingly, the table depicts a slightly better quality of service for the same data when considering the looser English window of tolerance rather than the stricter Scottish one.

### 8 Conclusions

In this paper we have presented an analysis pathway for the quantitative evaluation of service-level agreements for public transportation systems. Although we discussed a concrete application focussing on a specific bus route in a specific city, our approach is more general and it can in principle be applied to other transportation systems publishing timetabled departure times.

The methodology which we have proposed here requires the availability of the raw data from a bus tracking system. At first sight, it might have seemed that the properties of interest could have been calculated directly from measurement data. However, data sets are necessarily incomplete and working from the data provides less coverage of the full range of the system behaviour and hence delivers fewer insights than are obtained when working with a stochastic process abstraction of the data.

In addition, only (automatically generated) models can assist service providers and regulatory authorities in evaluating what-if scenarios, e.g., understanding the impact of
changes along a route on the offered quality of service. In this respect, the measurements are crucial to calibrate the model with realistic parameters, which can be changed by the modeller (by simply manipulating the compact BusKer specification) in order to study how the properties would be affected. For instance, regulators could determine how proposals to amend the notion of punctuality might impact on a provider’s capability to satisfy the regulations on services.

As discussed, the model involves a single route only, hence the measurements already incorporate effects of contention such as those due to multiple buses sharing the same route, and multiple routes sharing segments of the road. Developing a model where such effects are captured explicitly is an interesting line of future work, as is extending our analysis pathway to such a scenario.

Acknowledgements This work is supported by the EU project QUANTICOL, 600708. The Bio-PEPA Eclipse Plugin software can be obtained from www.biopepa.org. The MultiVeStA statistical analysis tool is available from code.google.com/p/multivesta/. The authors thank Allan Clark and Ludovica Luisa Vissat for the work in [7] which provided the model parameters for the present paper. The authors thank the anonymous reviewers for suggestions which helped us to improve the paper.

References


