Log2model: Inferring Behavioral Models from Log Data

Kasper S. Luckow*
*†Carnegie Mellon University Silicon Valley,
Mountain View, CA, USA

Corina S. Păsăreanu†
†NASA Ames Research Center,
Mountain View, CA, USA

***Abstract*—We present LOG2MODEL, an approach, supported by a tool, that builds behavioral models from log data. The logged data consists of time series encoding the values of the states of a system observed at discrete time steps. The models generated are Discrete-Time Markov Chains with states and transitions representing the values recorded in the log. The models contain key information that can be visualized and analyzed with respect to safety, delays, throughput etc, using off-the-shelf model checkers such as PRISM. The analysis results can be further used by users or automated tools to monitor and alter the system behavior. We present the architecture of LOG2MODEL and its application in the context of autonomous operations in the airspace domain.**

I. Introduction

This work was done in the context of a greater effort that investigates assurance methods to produce reliable software for surface and airspace operations in various applications at NASA. Our applications range from SAFETUGS [14], an application of self-driving vehicle technology to the problem of towing aircraft at busy airports with autonomous vehicles called "tugs", to AUTORESOLVER, a sophisticated automated air-traffic conflict resolution system [6], and Unmanned Aerial Systems (UASs) used in various missions, to name a few. Typically such systems operate in uncertain environments but have critical probabilistic requirements. For example the probability of an unmanned aerial vehicle turning too fast should be less than 10^{-6}. We are developing probabilistic analysis techniques [7], [13], [8] to complement traditional simulation for increased assurance.

As part of this effort, we present LOG2MODEL, an approach to building probabilistic models systematically from telemetry data, log files and simulation data available from previous or similar usages (missions) of the software under analysis. LOG2MODEL has been implemented as an open-source project (Apache License Version 2.0) and is available from GitHub: https://github.com/ksluckow/log2model.

The generated models can be visualized and analyzed with respect to quantitative and qualitative properties using off-the-shelf model checkers, such as PRISM [10] and UPPAAL [11]. Through these models, the developers can gain a deep understanding of how a system behaves throughout a mission, or several missions, without dealing with the intricacies of the implementation, hardware dependencies or the outside, uncertain, physical environment. Furthermore, the models can be used in a *predictive way*. For example, for the SAFETUGS application, which studies the use of autonomous vehicles (i.e. tugs) for safe and efficient towing of airplanes in an airport, one can use the models derived from the data recording the surface operations in an airport to optimize the decisions in the tug dispatcher with the goal of minimizing delays in taxiing and avoiding congestions.

While our focus is on safety critical applications used in surface or aerial operations, LOG2MODEL can be used (with minimal modifications) in many other applications where there is a need of analyzing interactive software under different probabilistic usage profiles that accounts for external interactions, both with the user and with external resources such as remote components.

II. Overview

The goal of LOG2MODEL is to build "behavioral models" of complex software used in surface or aerial operations. These models contain key information that enable analysis with respect to safety, delays, throughput and other properties.

LOG2MODEL uses log data (time series) for building models. The log data is discretely sampled, in some cases many times per second resulting in large amounts of data. Therefore, it is necessary to select a resolution to allow for more realistic state transitions and to prevent state space explosion.

The states of the model then represent "abstractions" of the states reported in the log file and transitions in the model correspond to the time steps in the log file. The abstraction is defined by the user and depends on the properties of interest and on the application domain. To help with this task LOG2MODEL already provides a common abstraction, namely a "range" abstraction, which partitions the values in the log file in user-specified intervals. For instance, an abstract state of a vehicle's position, can be a (2D or 3D) cell obtained by decomposition of the operational space into a grid.

LOG2MODEL also has a component based on univariate temporal event detection and event classification for finding *likely events* that in turn can be used for *mining* these state definitions. The event detection component is based on the ideas of the moving average control chart variant [17]. LOG2MODEL provides different moving average methods, such as a standard, rectangular (i.e. unweighted) average, and triangular smoothing that uses, e.g., an exponential weight distribution to the elements.

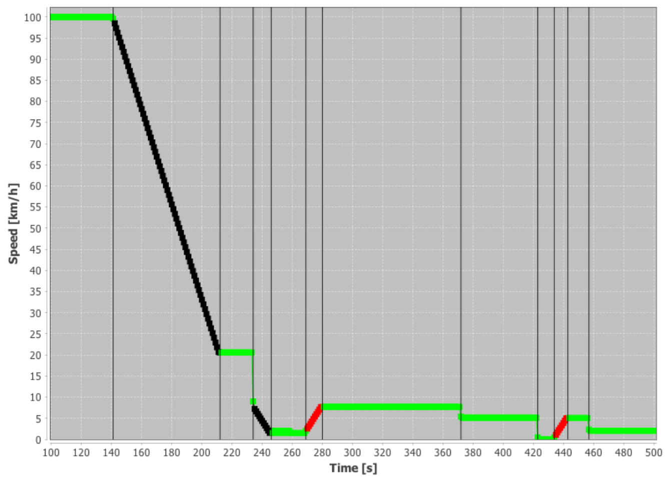

Fig. 1: Mining events for the rate-of-change feature of the speed component of a SafeTug when $k = 3$.

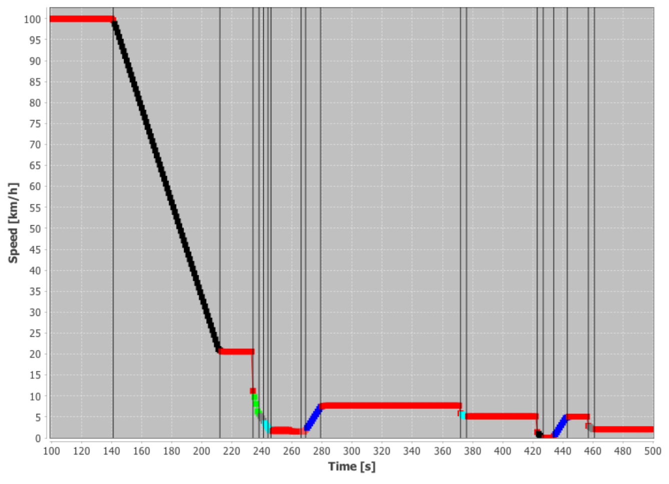

Fig. 2: Mining events for the rate-of-change feature of the speed component of a SafeTug when $k = 6$.

From the relevant feature extracted from the time series, a prediction model is computed. It predicts the value f_t of the feature at time t based on the sample mean, $\bar{x}$, and sample standard deviation, s, from the window $f_{t-c}, .., f_{t-1}$ of size c considered in the moving average. The range $[\bar{x} - ns; \bar{x} + ns]$ is then the expected values for f_t. The user specifies the constants c and n. If the control limits are exceeded, f_t is the occurrence of a new event. The parameter n controls in some sense the sensitivity with which changes in the feature should be regarded as the occurrence of a new event or not.

Let $e_1, e_2, .., e_m$ denote m events found at time $t_{e_1}, t_{e_2}, .., t_{e_m}$. Each interval $[0; t_{e_1}), [t_{e_1}; t_{e_2}), .., [t_{e_m}; t_{end}]$ denotes the time interval to which the event applies—t_{end} is the time for the last log entry. For each interval, the average value of the feature is computed and classified with k-means clustering to group similar events into broader classes. Each class can be used as a state definition subsequently. In the future, we plan to experiment with other event detection algorithms, e.g., CUSUM [15] and support multivariate data.

We have, among others, used this component to process autonomous vehicle log data for mining state definitions corresponding to acceleration, deceleration and cruising from the rate-of-change feature of the speed component. As an example, using $c = 3$, $n = 3$, and $k = 3$, LOG2MODEL processed 8 MB log file of SAFETUGS activity in a couple of seconds and visualized the time series for the speed component along with the identified events. The output is shown in Figure 1.

Vertical black lines denote the occurrence of an event, and the colored segments of the plot denote the event classes found with k-means clustering of the event intervals. Green, red, and black segments can be interpreted as time intervals in which the tug is moving with constant speed, accelerating, and decelerating, respectively. More event classes can be found by increasing k. For example, with $k = 6$, LOG2MODEL identifies—in addition to the same intervals for constant speed and acceleration—various levels of deceleration happening during the activity of the tug as shown in Figure 2.

Using the state definitions—e.g. using range abstractions or based on the event classes found with temporal event detection—-LOG2MODEL generates an intermediate model from the log that describes a finite state machine with labels on transitions. This step is performed by processing each log entry, extracting the fields of interest, and then mapping them to the abstract state definitions. Transitions connect the abstract states and are labelled with counts according to the frequency with which that transition is observed in the log data.

As an example, assume that the cells of a 2x2 grid abstraction are used as abstract states for the positions of the tugs in the SAFETUGS application. The intermediate model will have four states corresponding to the quadrants of the grid. Transitions between those states are generated based on the movement of the tugs on the grid: if a tug starts in quadrant 1 (i.e. time 0) and keeps moving in this quadrant for 10 time units after which (at time 11) it moves to quadrant 2, then the state representing to quadrant 1 has a transition to itself with count 10, and a transition to the state representing quadrant 2 with a count of 1.

From the intermediate model, models are generated for the input language of supported model checkers. For example, the intermediate representation readily allows us to capture the semantics of Discrete-Time Markov Chains (DTMCs). Informally, the states and transitions of the intermediate model are directly translated to states and transitions in the DTMC. Transition probabilities are computed as the count of a transition of the sum of the transition counts.

For DTMCs, LOG2MODEL can currently generate models for the PRISM and UPPAAL model checkers. Visualization of these models is achieved by a translation to the DOT graph description language. Through PRISMs reactive modules, the DTMCs can be directly encoded. The PRISM models allow reasoning on Probabilistic Computation Tree Logic (PCTL) properties. UPPAAL does not directly support DTMCs, and it uses a continuous time model. When producing these, we simulate a discrete time model by restricting transitions with equalities with integer constants on the clock variables. UPPAAL enables reasoning on Timed Computation Tree Logic (TCTL) properties—the transition weights (probabilities) are used for the statistical model checking extension (UPPAAL-SMC) [5] when sampling paths.

For DTMCs, the probability distribution for a state is estimated by computing the ratio between the counts for each outgoing transition and the total count of the transitions exiting the state; this corresponds to the maximum likelihood estimator for the probability distribution at that state.

A. Architecture

Figure 3 provides a description of LOG2MODEL, which we implemented in Java. It is comprised of three main components: `Log Parser`, `Intermediate Model Generator`, and `Model Generator`. The modularity along with the design approach that provides several extension points, facilitates easy experimentation with pre-processing and abstraction techniques and translations to different modeling formalisms. The `Log Parser` processes the input log, and uses (pre-defined or mined) state definitions and the transition function to generate the intermediate model. Informally, a state is an assignment of values to model variables, and transitions can have attributes attached which are used for encoding the probabilistic component of DTMCs. Supporting other output formats is a matter of translating the intermediate model to that format. LOG2MODEL provides various facilities and APIs for easing this translation. LOG2MODEL currently performs single-threaded processing of log files.

III. APPLICATIONS

For SAFETUGS, we have used simulation data from surface operations at Dallas International Airport while for UAS, we have the data from several missions (e.g. Sierra). We also have logs produced from AUTORESOLVER (also a NextGen component for mid-air separation assurance). These systems have in common that they pertain to the operation of vehicles (ground and air); each entry contain information about the state of the vehicle, e.g., its (2D or 3D) position, heading, etc. We will use the positional component of the logs obtained from SAFETUGS to demonstrate the use of LOG2MODEL (see [14] for an overview of the SAFETUGS architecture).

For SAFETUGS, LOG2MODEL builds models with different grid abstractions, which divides the airport surface into cells and keeps "count" of vehicles in each of them—the counts of the grid elements at a specific discrete time constitute a state in the model. Figure 4 shows a 4x4 grid abstraction overlaid the airport surface.

Decomposing the airfield into a coarse (resp. fine) grid is likely to yield a less (resp. more) precise model at the expense of a smaller (resp. bigger) state space.

Figure 5 shows the DTMC produced with a 2x2 grid configuration.

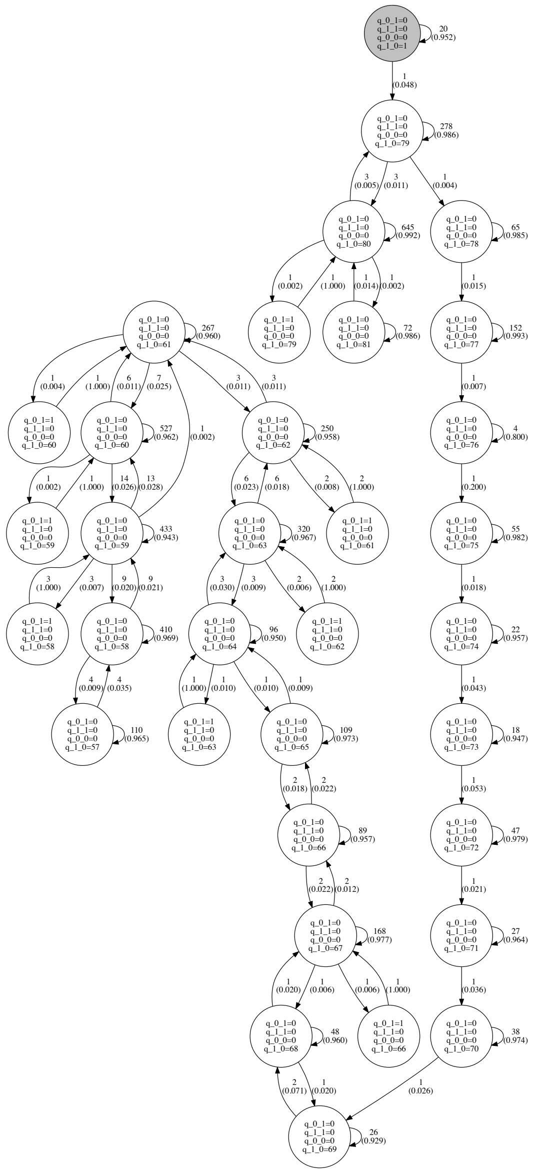

Fig. 5: DTMC inferred from example SAFETUGS data. States are represented by the number of vehicles in each cell of a 2x2 grid projected on the airport surface.

LOG2MODEL can build either a single model for the whole operations or several vehicle-centric models, capturing their individual operations, together with the interactions with other vehicles in its immediate vicinity. The advantage of this latter approach is that the models are smaller and therefore easier to build, while at the same time being more precise.

a) ***Analysis:*** For a SAFETUGS log recording more than 30 tugs and airplanes each second, for 70 minutes of activity, we analyzed 123 MB of data in less than $5s$ on a standard 2015 Macbook. The generated models have 75 states (10×2 grid abstraction) and 96 states (4×4 grid abstraction), respectively. The DTMC models were analyzed with PRISM, with respect to the safety and quantitative properties:

- $\mathbf{P < 0.6}\ [\mathbf{F < 50} \quad \mathbf{c_{\langle x,y \rangle} < 30}]$: *The probability that less than 30 tugs/airplanes are present in cell* $\langle x, y \rangle$

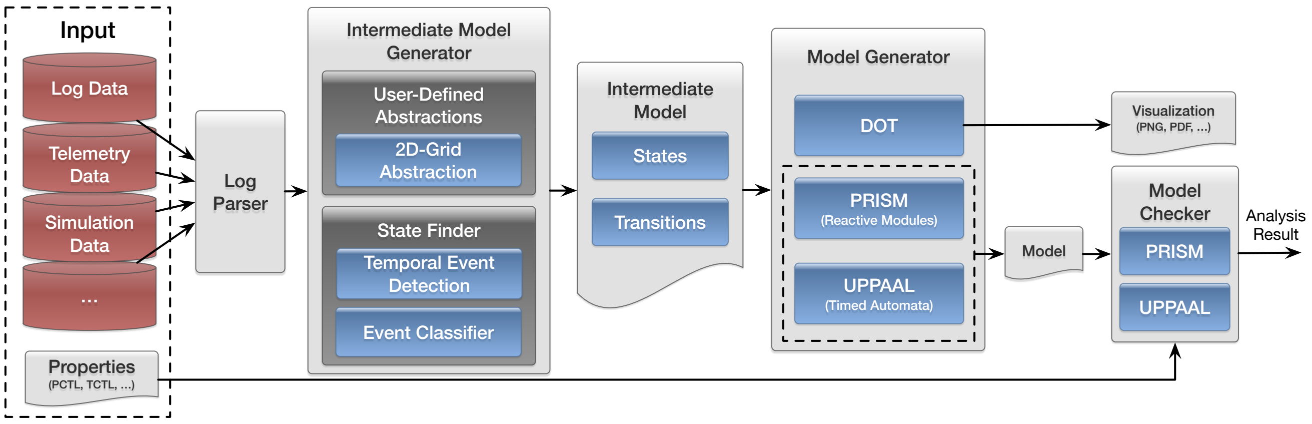

Fig. 3: LOG2MODEL architecture overview.

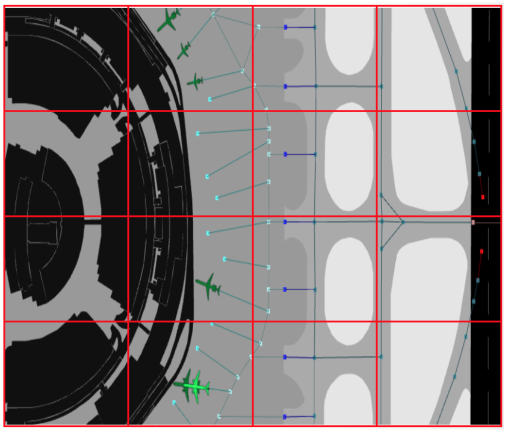

Fig. 4: Grid projected on airport surface. Each cell constitute an abstract position of the aircraft and tugs.

TABLE I: Results for different grid sizes for multiple vehicles (row Multiple vehicles) and single vehicles (rows Single vehicle, V$_*$) models. *ID* is the vehicle ID; $Tr_\#$ ($St_\#$) is the number of transitions (states) in the generated model; P is the probability for property $\mathbf{P} =?\ [\mathbf{F} < \mathbf{3}\quad \mathbf{q}_{\langle \mathbf{0},\mathbf{1}\rangle} > \mathbf{0}]$; and $Time$ is the analysis time.

Grid	Vehicle configuration	*ID*	$Tr_\#$	$St_\#$	P	$Time$
2×2	Multiple vehicles	—	257	115	1.00	4s
	Single vehicle, V_{min}	#19	5	3	0.0138	3s
	Single vehicle, V_{max}	#10	11	4	0.00825	2s
4×4	Multiple vehicles	—	546	288	1.00	15s
	Single vehicle, V_{min}	#19	13	7	0.0138	3s
	Single vehicle, V_{max}	#29	31	13	0.00	3s
8×8	Multiple vehicles	—	949	549	1.00	16s
	Single vehicle, V_{min}	#19	23	12	0.0138	2s
	Single vehicle, V_{max}	#24	85	38	0.00	3s
10×10	Multiple vehicles	—	1143	688	0.00	16s
	Single vehicle, V_{min}	#19	33	17	0.00	1s
	Single vehicle, V_{max}	#24	103	45	0.00	3s

within the first 50 time units (seconds), is less than 0.6.

- $\mathbf{P} =?\ [\mathbf{F} < \mathbf{300}\quad \mathbf{c}_{\langle \mathbf{x},\mathbf{y}\rangle} > \mathbf{33}]$: *What is the probability that more than 33 tugs/airplanes are present in cell $\langle x, y\rangle$ within the first 300 time units (seconds)?*

The analyses take less than a second, respectively returning *true* and 0.1058.

Table I illustrates how grid sizes affect analysis results and state space of the model. For the single vehicle models, min (max) denote the minimum (maximum) number of transitions in the model. Putting the individual models in parallel yields a complete system. For each model, we analyzed the example property $\mathbf{P} =?\ [\mathbf{F} < \mathbf{3}\quad \mathbf{c}_{\langle \mathbf{0},\mathbf{1}\rangle} > \mathbf{0}]$, i.e. *What is the probability of reaching the grid cell represented by state $c_{\langle 0,1\rangle}$ within 3 discrete time steps?*

Analyzing properties on the multi-vehicle model takes longer than for the single-vehicle models. Also note that for vehicle #19, different grid sizes yield the same probability.

IV. Related Work

There is a large body of work dedicated to the automatic inference of software behavioral models automatically from execution logs [3], [12]. Many model inference algorithms [2], [9] have been proposed in recent work. CSight [2] infers models in the form of communicating finite state machines from logs of the behavior of concurrent systems. The closest to ours is the approach of Ghezzi et al. [9] which mines probabilistic models from user-intensive web applications. In the case of web applications the log file is a sequence of requests of Web resources issued by clients. While so far the application domain for LOG2MODEL has been different, LOG2MODEL's parser can be easily extended to accommodate this kind of log data. Also relevant is the work on formal analysis of log files [1] which studies the application of run-time verification technology to telemetry data.

Further our work is related to the area of Process Mining [19], which addresses inferring behavioral process models. In this context, analyzing plane trajectories is addressed in [4], however the inferred models do not contain any probabilistic or timing information. Another recent work addresses probabilities of taking paths [18], but only in a very limited context, where the inferred models are acyclic, while more general models are addressed in [16] and are based on Petri Nets and thus quite different from ours.

V. Conclusions and Future Work

We have described our ongoing efforts in developing a model-inference component that supports analysis of qualitative and quantitative properties. For future work, we plan to improve on the automated event detection by accounting for the multivariate case. We also plan to investigate methods for converting from discrete-time into continuous-time models for timing analysis. To bootstrap the model inference, we would also like to use random testing. Finally, we plan to investigate further the use of the models for predictive analysis and in an on-line monitoring setting, where the models are built and refined during operation.

Acknowledgment

This material is based on work partially sponsored by NSF under grants CCF-1319858 and CCF- 1549161

References

[1] H. Barringer, A. Groce, K. Havelund, and M. H. Smith. Formal analysis of log files. *JACIC*, pages 365–390, 2010.

[2] I. Beschastnikh, Y. Brun, M. D. Ernst, and A. Krishnamurthy. Inferring models of concurrent systems from logs of their behavior with CSight. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 468–479, New York, NY, USA, 2014. ACM.

[3] A. W. Biermann and J. A. Feldmann. On the synthesis of finite-state machines from samples of their behavior. *IEEE Transactions on Computers*, 100(6):592–597, 1972.

[4] C. Cabanillas, C. Di Ciccio, J. Mendling, and A. Baumgrass. Predictive task monitoring for business processes. In *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, pages 424–432, 2014.

[5] A. David, K. G. Larsen, A. Legay, M. Mikučionis, and Z. Wang. Time for statistical model checking of real-time systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification*, CAV'11, pages 349–355, Berlin, Heidelberg, 2011. Springer-Verlag.

[6] H. Erzberger, T. A. Lauderdale, and Y. cheng Chu. Automated conflict resolution, arrival management and weather avoidance for atm. In *Proceedings of the 27th International Congress of the Aeronautical Sciences*, 2010.

[7] A. Filieri, C. S. Păsăreanu, and W. Visser. Reliability analysis in Symbolic Pathfinder. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pages 622–631, Piscataway, NJ, USA, 2013. IEEE Press.

[8] A. Filieri, C. S. Păsăreanu, W. Visser, and J. Geldenhuys. Statistical symbolic execution with informed sampling. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 437–448, New York, NY, USA, 2014. ACM.

[9] C. Ghezzi, M. Pezzè, M. Sama, and G. Tamburrelli. Mining behavior models from user-intensive web applications. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 277–287, New York, NY, USA, 2014. ACM.

[10] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

[11] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1-2):134–152, 1997.

[12] D. Lorenzoli, L. Mariani, and M. Pezzè. Automatic generation of software behavioral models. In *30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, pages 501–510, 2008.

[13] K. Luckow, C. S. Păsăreanu, M. B. Dwyer, A. Filieri, and W. Visser. Exact and approximate probabilistic symbolic execution for nondeterministic programs. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, pages 575–586, New York, NY, USA, 2014. ACM.

[14] R. Morris, C. Pasareanu, K. Luckow, W. Malik, H. Ma, T. K. Kumar, and S. Koenig. Planning, scheduling and monitoring for airport surface operations. In *AAAI Workshops*, 2016.

[15] E. S. PAGE. Continuous inspection schemes. *Biometrika*, 41(1-2):100–115, 1954.

[16] A. Rogge-Solti and G. Kasneci. Temporal anomaly detection in business processes. In *Business Process Management - 12th International Conference, BPM 2014, Haifa, Israel, September 7-11, 2014. Proceedings*, pages 234–249, 2014.

[17] W. A. Shewart. *Economic control of Quality of Manufactured Product*. Van Nostrand Reinhold Co., New York, 1931.

[18] N. R. T. P. van Beest, M. Dumas, L. García-Bañuelos, and M. L. Rosa. Log delta analysis: Interpretable differencing of business process event logs. In *Business Process Management - 13th International Conference, BPM 2015, Innsbruck, Austria, August 31 - September 3, 2015, Proceedings*, pages 386–405, 2015.

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