Discrete-event Markov Chains Used in 2016-2020 Research Papers Published in Leading Software Engineering Venues

We identified the CORE2020 (https://www.core.edu.au/conference-portal) rank A* software engineering journals and conferences, and we searched these venues and the *Journal of Systems and Software* for all the research papers that made use of discrete-event Markov models within the last five complete years (2016-2020).

The five venues used in our study are:

1. IEEE Transactions of Software Engineering (TSE)
2. ACM Transactions on Software Engineering Methodology (TOSEM)
3. Journal of Systems and Software (JSS)
4. International Conference on Software Engineering (ICSE)
5. ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)

In total we identified the 10 research papers shown in Table 1. From these 10 papers, we eliminated two papers:

1. We eliminated the first paper[1] because it describes research on probabilistic models with a semantics based on DTMCs, but contains no DTMC models given explicitly.
2. We eliminated the second paper[2] because its content did not allow us to determine the size of the DTMCs used. However, we note that one of the DTMCs mentioned in this paper is for a Tele Assistance System[3] that other projects (e.g., https://www.cs.york.ac.uk/tasp/FACT/, https://www.cs.york.ac.uk/tasp/VERACITY/) model using a DTMC with 10 or 11 states.

Table 1: Number of papers using discrete-event Markov chains

	Year					
Venue	2016	2017	2018	2019	2020	Total
TSE	1		1		1	3
TOSEM						0
JSS	1			3		4
ICSE	1		1			2
ESEC/FSE			1			1

[1] Maurice H. ter Beek, Axel Legay, Alberto Lluch-Lafuente, and Andrea Vandin. "A framework for quantitative modeling and analysis of highly (re)configurable systems." *IEEE Transactions on Software Engineering* **46**(3): 321-345, 2020.
[2] Cámara, Javier, David Garlan, and Bradley Schmerl. "Synthesizing tradeoff spaces with quantitative guarantees for families of software systems." *Journal of Systems and Software* **152**:33-49, 2019.
[3] D. Weyns, R. Calinescu, Tele assistance: A self-adaptive service-based system exemplar, in: 10th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2015, IEEE Computer Society, 2015. doi:10.1109/SEAMS.2015.27.

For each of the remaining 8 papers, we analysed the size (number of states) of the DTMC models used to evaluate the approach(es) described in the paper. The result of this analysis is summarised in Table 2.

The aim of our study was to examine the suitability of using DTMC models comprising 10 states and 17 states for the evaluation of a new verification method for software systems. Therefore, the analysis reported in Table 2 examines whether research papers published in leading software engineering venues use DTMC models:

(a) with 10 or fewer states;
(b) with between 11 and 17 states;
(c) and/or with more than 17 states.

Table 2: Size (number of states) of the DTMC models used in the identified papers

	DTMC(s) used to evaluate the approach(es) introduced in the paper		
Research paper	**DTMC(s) with up to 10 states**	**DTMC(s) with between 11 and 17 states**	**DTMC(s) with over 17 states**
Su, Guoxin, David S. Rosenblum, and Giordano Tamburrelli. "Reliability of run-time quality-of-service evaluation using parametric model checking." *38th International Conference on Software Engineering*, 2016.	- DTMC with 10 states		
Wang X, Sun J, Chen Z, Zhang P, Wang J, Lin Y. "Towards optimal concolic testing." *40th International Conference on Software Engineering*, 2018.	- DTMC with 5 states[4] - DTMC with 10 states[4]	- DTMC with 15 states[4]	- DTMC with 20 states[4]
Nakagawa, Hiroyuki, Hiromu Toyama, and Tatsuhiro Tsuchiya. "Expression caching for runtime verification based on parameterized probabilistic models." *Journal of Systems and Software* **156**:300-311, 2019.	- DTMC with 9 states	- DTMCs with 11, 14 and 17 states[5]	- DTMCs with up to 20-59 states[5]
Guoxin Su, Yuan Feng, Taolue Chen and David S. Rosenblum. “Asymptotic Perturbation Bounds for Probabilistic Model Checking with Empirically Determined Probability Parameters.” I*EEE Transactions on Software Engineering* **42**(7):623-639, 2016.	- DTMC with 7 states		- DTMC with 25 states[6] - DTMC with 6642 states[7]

[4] These DTMCs are randomly generated to represent abstractions of programs.
[5] These are abstract DTMCs that do not model a real system.
[6] This is a DTMC model of the PageRank algorithm.
[7] This is a DTMC model of an imperfect NAND gate.

Radu Calinescu, Danny Weyns; Simos Gerasimou, Muhammad Usman Iftikhar, Ibrahim Habli, and Tim Kelly. "Engineering trustworthy self-adaptive software with dynamic assurance cases." *IEEE Transactions on Software Engineering* **44**(11):1039-1069, 2018.		- DTMC with 17 states	
Humaira Afzal, Muhammad Rafiq Mufti, Irfan Awan, Muhammad Yousaf. "Performance analysis of radio spectrum for cognitive radio wireless networks using discrete time Markov chain." *Journal of Systems and Software* **151**:1-7, 2019.		- DTMC with 15 states	
Franco, Joao M., Francisco Correia, Raul Barbosa, Mário Zenha-Rela, Bradley Schmerl, and David Garlan. "Improving self-adaptation planning through software architecture-based stochastic modeling." *Journal of Systems and Software* **115**:42-60, 2016.	- DTMC with 6 states - DTMC with 8 states		- DTMCs with over 100 states[8]
Yamilet R. Serrano Llerena, Marcel Böhme, Marc Brünink, Guoxin Su and David S. Rosenblum. "Verifying the long-run behavior of probabilistic system models in the presence of uncertainty." *26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018.		- DTMC with 17 states	- DTMC with 46 states[9]

[8] These DTMCs model a system where multiple servers are placed in parallel, such that one of n servers is chosen probabilistically. The larger system (and model) is obtained by placing more servers in parallel, increasing the size of the model to 100 states.
[9] This is a DTMC model of the `wc` word/byte counter function from the GNU Coreutils library.