Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 20 Issue 1 Version 1.0 Year 2020
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Performance Evaluation of Software using Formal Methods

By Akinsola, Jide E. T., Kuyoro, Afolashade, O., Adeagbo, Moruf A. & Awoseyi, Ayomikun A.
First Technical University

Abstract- Formal Methods (FMs) can be used in varied areas of applications and to solve critical and fundamental problems of Performance Evaluation (PE). Modelling and analysis techniques can be used for both system and software performance evaluation. The functional features and performance properties of modern software used for performance evaluation has become so intertwined.

Traditional models and methods for performance evaluation has been studied widely which culminated into the modern models and methods for system and software engineering evaluation such as formal methods. Techniques have transcended from functionality to performance modeling and analysis. Formal models help in identifying faulty reasoning far earlier than in traditional design; and formal specification has proved useful even on already existing software and systems. Formal approach eliminates ambiguity. The basic and final goal of the performance evaluation technique is to come to a conclusion, whether the software and system are working in a good condition or satisfactorily.

Keywords: formal methods, performance evaluation, performance modeling, software performance evaluation, machine learning, markov chains, queuing networks.

GJCST-C Classification: *D.2.m*

PERFORMANCEEVALUATIONOFSOFTWAREUSINGFORMALMETHODS

Performance Evaluation of Software using Formal Methods

Akinsola, Jide E. T. [α], Kuyoro, Afolashade, O. [σ], Adeagbo, Moruf A. [ρ] & Awoseyi, Ayomikun A. [Ѡ]

Abstract- Formal Methods (FMs) can be used in varied areas of applications and to solve critical and fundamental problems of Performance Evaluation (PE). Modelling and analysis techniques can be used for both system and software performance evaluation. The functional features and performance properties of modern software used for performance evaluation has become so intertwined.

Traditional models and methods for performance evaluation has been studied widely which culminated into the modern models and methods for system and software engineering evaluation such as formal methods. Techniques have transcended from functionality to performance modelling and analysis. Formal models help in identifying faulty reasoning far earlier than in traditional design; and formal specification has proved useful even on already existing software and systems. Formal approach eliminates ambiguity. The basic and final goal of the performance evaluation technique is to come to a conclusion, whether the software and system are working in a good condition or satisfactorily.

Formal methods (FM) or Formal Techniques (FT) for performance evaluation include formalisms for performance modeling (which are Markov chains, queuing networks, stochastic Petri nets, and stochastic process algebras), equivalence checking and model checking, efficient solution techniques, and software performance engineering. Modeling consists of five classes: requirements, activities, connectors, performers, and resources.

The paper focuses on formal methods for performance evaluation using formal modeling with emphasis on Modeled System, Markov Chains, Queuing Networks, Generalized Stochastic Petri Nets, Stochastic Process Algebras, Markovian Behavioral Equivalences and Software Performance Engineering (SPE) in relation tofunctional features and performance properties.

Keywords: formal methods, performance evaluation, performance modeling, software performance evaluation, machine learning, markov chains, queuing networks.

I. Introduction

The term Formal Methods (FM) refers to the use of mathematical modelling, calculation and prediction in the specification, design, analysis and assurance of computer systems and software. The reason it is called formal methods rather than mathematical modelling of software is to highlight the character of the mathematics involved (Rushby, 1995).

According to Wikipedia, the use of formal methods for software and hardware design is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical analyses can contribute to the reliability and robustness of a design.

Formal methods (FM) or Formal Techniques (FT) for performance evaluation include formalisms for performance modeling (Markov chains, queuing networks, stochastic Petri nets, and stochastic process algebras), equivalence checking and model checking, efficient solution techniques, and software performance engineering (Bernardo & Hillston, 2007). Collins (1998), opined that formal methods are techniques used to model complex systems as mathematical entities. By building a mathematically rigorous model of a complex system, it is possible to verify the system's properties in a more thorough fashion than empirical testing.

System engineers can inspect the modeled system architecture to determine whether it is acceptable, but few formal methods exist to aid in the performance of this task (Rodano & Giammarcob, 2013). In a safety critical system, ambiguity can be extremely dangerous, and one of the primary benefits of the formal approach is the elimination of ambiguity (Kling, 1994).

Modelling is one of the ways used in presenting performance evaluation. Heuristics can be applied in determining the good characteristics for performance evaluation. Formal methods can be applied to identify the characteristics of a good system architecture using logical notations. Formal method is the fast approach to identify possible problems in any software architectural design (Rodano & Giammarcob, 2013).

Performance evaluation gives a measure of the service delivered by a system (Jean-Yves & Boudec, 2010) and performance is one of the most important non-functional aspects of any (hardware or software) system. Performance evaluation comprises of certain techniques such as direct measurements using test-beds, analytical or simulation modeling which can be applied to existing or envisioned systems like computer systems, communication networks, algorithms and protocols (Jain, 1991). The basic and final goal of the performance evaluation concept is to come to a conclusion, whether the software and system are

Author α ρ Ѡ: Department of Mathematics and Computer Sciences, First Technical University, Ibadan, Oyo State, Nigeria. e-mails: akinsolajet@gmail.com, adedegy@gmail.com, awoseyiayomikun@gmail.com

Author σ: Department of Computer Science, Babcock University, Ilishan-Remo, Ogun State, Nigeria. e-mail: afolashadeng@gmail.com

working in a good condition or satisfactorily. This is can be achieved with formal modelling techniques.

Datamining is the discovery of “models” fordata (Leskovec, Rajaraman & Ullman, 2014). According to Anwaar, Junaid, Raihan, Arjuna, Andrej & Jon (2016), datamining normally denotes the automation of pattern discovery and prediction from huge volumes of data using Machine Learning (ML) techniques. Datamining can also be used to denote an Online Analytical Processing (OLAP) or Structured Query Language (SQL) queries that entails retrospectively searching a large data base for a specific query. There has been upsurge in availability of information and device connectivity have brought about increase in application of machine learning (which is a sub-domain Artificial Intelligence (AI) in diverse areas (Akinsola, Awodele, Idowu & Kuyoro, 2020). These areas include applications of Machine Learning (ML) in performance evaluation and verification of software. ML requires application of algorithms for model building using performance metrics. Every performance metric must be considered holistically before choosing an optimal algorithm for predictive analytics (Akinsola, Awodele, Idowu & Kuyoro, 2020).

Formal method axioms can be used in structural evaluation of a software model especially data mining model. The relationships among the various elements of data mining software you be used to evaluate its effectiveness in terms of performance. Formal methods can be used for testing the realization of the entire software against its specification as well as connections between components in order to determine its interoperability.

Characteristics heuristics natural language axioms. The axioms symbolizes syntactic checks that can be used in software performance evaluation. Transformation of axioms into formal language notation is essential in performance evaluation of data mining software. CORE and Innoslate are some of the software engineering tools for software performance evaluation.

The quality of any software for performance evaluation has three sets of factors which are functionality, engineering, and adaptability. They are also referred to as exterior quality, interior quality and future quality respectively. Formal method functionality features are the exterior qualities such as Correctness, Reliability. Usability and Integrity. The engineering features are Efficiency, Testability, Documentation and Structure while the adaptability features are Flexibility, Reusability and Maintainability

II. Literature Review

Axioms are statements that we cannot deny without using them in our denial. Axioms are the foundation of all knowledge. When they are well constructed, the transformation of axioms into formal language notation can be a veritable tool in performance evaluation of data mining software. Formal methods axioms can be used in structural evaluation of a software model especially data mining model.

CORE I is used for analyzing the axioms. Innoslate is a web-based system modeling tool that is based on the Lifecycle Modeling Language (LML)

Model consists of five classes: requirements, activities, connectors, performers, and resources. Resources are data or information that is produced and/or consumed by the system. An activity is an element that transforms inputs into outputs (inputs and outputs are both resources). Performers carry out activities, and physical or logical relationships between performers are known as connectors. Requirements are written specifications for the system (Giammarco, 2012)

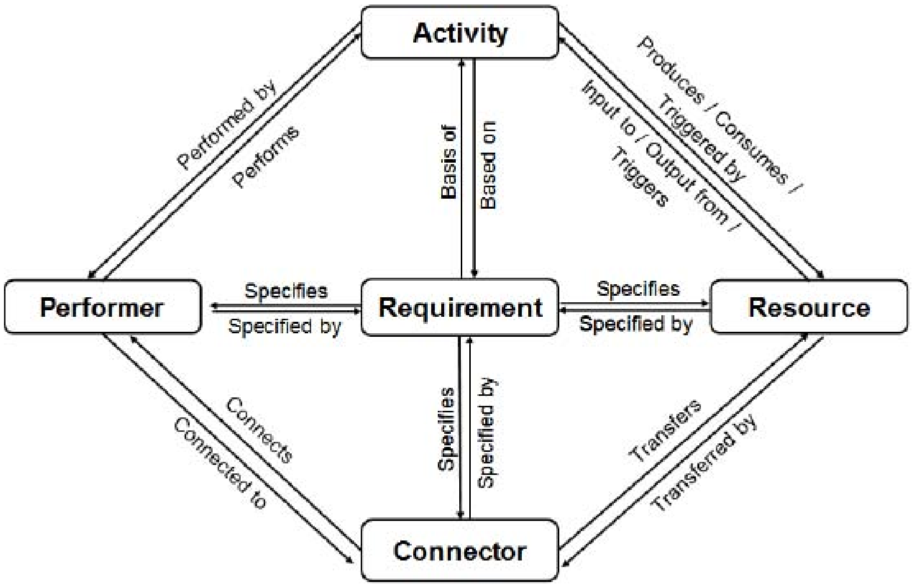

Figure 1: Class / Relationship Diagram of Software Model (Source: Giammarco, 2012)

The axioms for evaluating a modeled software architecture are categorized into five groups: Decomposition, Requirements Traceability, Activity Performance, Input/Output and Connection.

Markov chains have become an accepted technique for modeling a great variety of situations. Formal methods in computer science as a prominent approach to the rigorous design of computer, communication and software systems. Markov chains, the fundamental performance modeling formalism in use since the early 1900s. The success that has accompanied queuing modeling has largely eliminated the need to set up and solve global balance equations numerically. However, as models become more complex, it is becoming increasing evident that there is place for numerical analysis methods in the modelers' toolbox (Stewart, 2007).

Queuing Networks (QNs) have been proved to be a powerful and versatile tool for system performance evaluation and prediction. Queuing networks, a class of stochastic models extensively applied to represent and analyze resource-sharing systems such as communication and computer systems. Product-form queuing networks, allows for defining efficient algorithms to evaluate average performance measures. The main computational algorithms for QNs have been integrated in various software tools for performance modelling and analysis that include user friendly interfaces based on different languages to take into account the particular field of application, e.g., computer networks, computer systems. Basic queuing systems have been defined in queuing theory and applied to analyze congestion systems (Balsamo & Marin, 2007).

Generalized Stochastic Petri Nets (GSPNs), a modeling formalism that can be conveniently used both for the functional verification of complex models of discrete-event dynamic software and systems as well as for their performance and reliability evaluation. The automatic construction of the probabilistic models that underlie the dynamic behaviors of these nets rely on a set of results that derive from the theory of untimed Petri Nets. Petri nets are a powerful tool for the description and the analysis of systems that exhibit concurrency, synchronization and conflicts. There is general consensus that the only means of successfully dealing with large models is to keep them simple by using a "divide and conquer" approach in which the solution of the entire model is constructed on the basis of the solutions of its individual components (Balbo, 2007).

Process algebras emerged as a modelling technique for the functional analysis of concurrent systems approximately twenty years ago. Over the last 17 years there have been several attempts to take advantage of the attractive features of this modelling paradigm within the field of performance evaluation. Stochastic Process Algebras (SPA) were first proposed as a tool. Stochastic process algebras and their use in performance modeling, with a focus on the PEPA formalism is highly efficient for evaluation. The compositional modeling capabilities of the formalism and the tools available to support Markov-chain based analysis are good for formal models building (Clark, Gilmore, Hillston, & Tribastone, 2007).

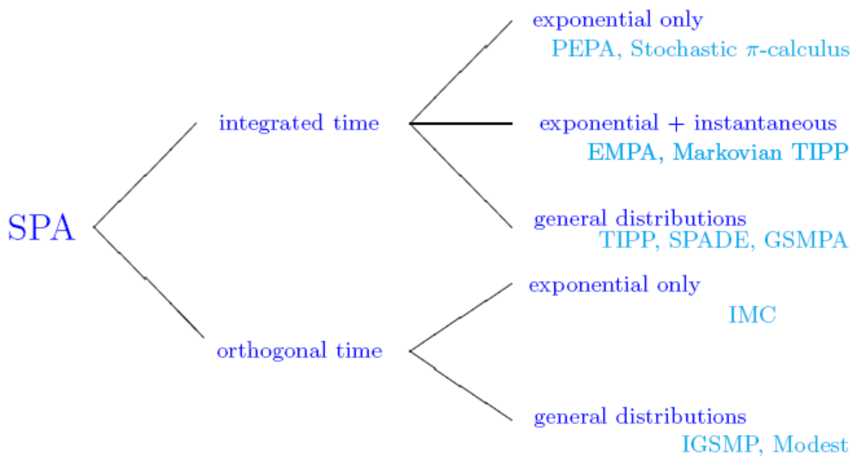

Figure 2: Classification of the stochastic process algebras (Source: Clarke et al., 2007)

The formality of the process algebra approach allows assigning of a precise meaning to every language expression. This implies that once we have a language description of a given system its behavior can be deduced automatically (Clarke et al., 2007)

Performance-oriented notations provide the designer with the capability of building performance aware system models, which can be used in the early development stages to predict the satisfy ability of certain performance requirements as well as to choose among alternative designs on the basis of their expected Quality of Service (QoS) guarantees. Markovian behavioral equivalences with respect to a number of criteria such as their discriminating power, the exactness of the Markov-chain-level aggregations they induce, the achievement of the congruence property, the existence of sound and complete axiomatizations, the existence of logical characterizations, and the existence of efficient verification algorithms can provide satisfactory analysis with respect to certain criteria such as exact aggregation, congruence property , sound and complete axiomatization, logical characteristics and verification complexity (Bernardo, 2007).

Probability is an important component in the design and analysis of software and hardware systems. In distributed algorithms electronic coin tossing is used as a symmetry breaker and as a means to derive efficient algorithms, Model checking for both discrete-time and continuous-time Markov chains, which deals with algorithms for verifying them against specifications written in probabilistic extensions of temporal logic, including quantitative properties with rewards supports probabilistic modeling such as Probabilistic Symbolic Model (PRISM) checker (Kwiatkowska, Norman & Parker, 2007).

Software performance engineering (SPE) is a systematic, quantitative approach to constructing software systems that meet performance requirements. SPE provides an engineering approach to performance, avoiding the extremes of performance-driven development and "fix-it-later." SPE uses model predictions to evaluate trade-offs in software functions, hardware size, quality of results, and resource requirements. Two SPE models provide the quantitative data for SPE: the software execution model and the system execution model. The software execution model represents key facets of software execution behavior. The model solution quantifies the computer resource requirements for each performance scenario. The system execution model represents computer system resources with a network of queues and servers. The model combines the performance scenarios and quantifies overall resource utilization and consequent response times of each scenario (Smith, 2007).

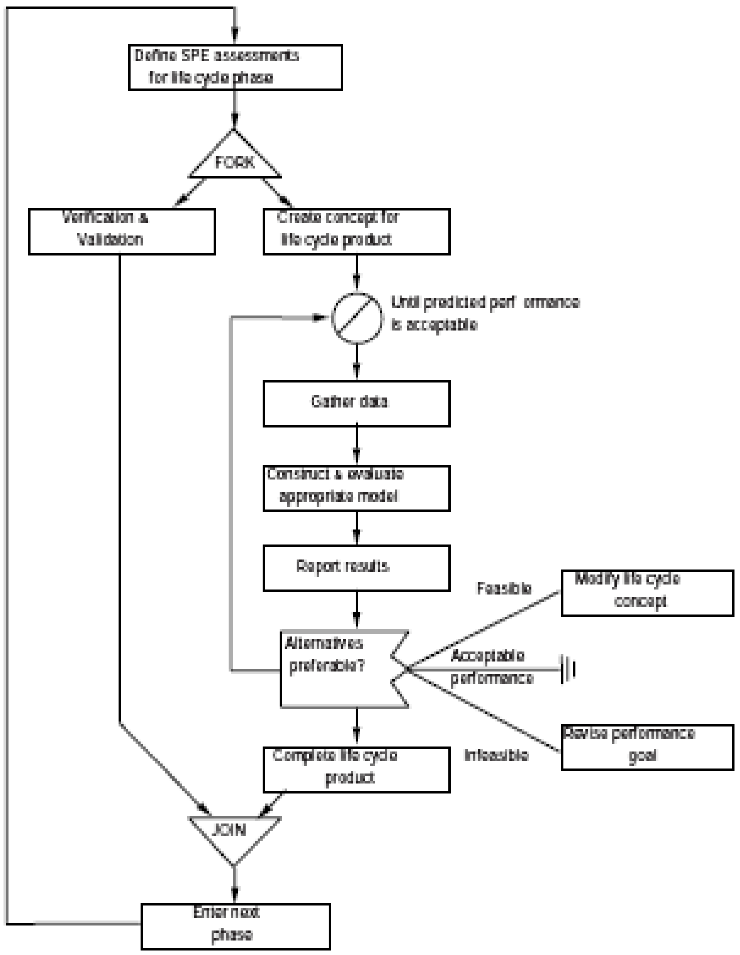
Define SPE assessments for life cycle phase
FORK
Verification & Validation
Create concept for life cycle product
Until predicted perf ormance is acceptable
Gather data
Construct & evaluate appropriate model
Report results
Feasible
Modify life cycle concept
Alternatives preferable?
Acceptable performance
Infeasible
Revise performance goal
Complete life cycle product
JOIN
Enter next phase

Figure 3: Software Performance Engineering Process (Source: Smith, 2007)

III. Merits and Demerits of Formal Methods for Performance Evaluation

a) Merits

It is effectual to write a specification formally rather than writing an informal specification and then translating it. To detect inconsistency and incompleteness, it is efficient to analyze the formal specification as early as possible (Mona, Amit & Meenu, 2010). Given below are some of the merits of formal methods in software performance evaluation:

i. *Measure of correctness:* The use of formal methods provides a measure of the correctness of a system, as opposed to the current process quality measures.
ii. *Early defect detection:* Formal Methods can be applied to the earliest design artifacts, thereby leading to earlier detection and elimination of design defects.
iii. *Guarantees of correctness:* Formal analysis tools such as model checkers consider all possible execution paths through the system. If there is any possibility of a fault/error, a model checker will find it. In a multithreaded system where concurrency is an issue, formal analysis can explore all possible interleaving and event orderings. This level of coverage is impossible to achieve through testing.
iv. *Error Prone:* Formal description forces the writer to ask all sorts of questions that would otherwise be postponed until coding. This helps to reduce the errors.

b) Demerits

Formal methods are generally viewed with suspicion by the professional engineering community (Bowen, 93). Given below are some of the demerits of formal methods in software performance evaluation:

i. *Expansive*

Formal Methods are expense. This is because of the rigor involved, formal methods are always going to be more expensive than traditional approaches to engineering. Also, the tool development cost is high.

ii. *Limits of Computational Models*

While not a universal problem, most formal methods introduce some form of computational model, usually hamstringing the operations allowed in order to make the notation elegant and the system provable. Unfortunately, these design limitations are usually considered intolerable from a developer's perspective.

iii. *Usability*

Traditionally, formal methods have been judged on the richness of their descriptive model. That is, 'good' formal methods have described a wide variety of systems, and 'bad' formal methods have been limited in their descriptive capacities.

iv. *Adaptability*

SPE activities are not easy to adapt and economical for future environments. So it needs to evolve in order to make SPE adaptable.

IV. Conclusion

Formal Methods (FM) is a very active research area with a wide variety of methods and mathematical models. There is not available any one method that fulfills all the related needs of building a formal specification. Just like the No Free Lunch theorem is highly essential in the field of machine learning because good number of correctly classified instances in predicting valid disease outcomes using supervised machine learning techniques is not just a function of accuracy (Akinsola, Adeagbo, Awoseyi, Ayomikun, 2019).Performance evaluation of software using formal methods can be carried out using hybridization of machine learning and Multi Criteria Decision Making (MCDM) techniques. MCDM methods can be used to find the optimal classification and regression models in relation to supervised machine learning algorithms (Akinsola, Kuyoro, Awodele & Kasali, 2019).

Researchers and practitioners are continuously working in this area and there by gaining the benefits of using formal methods. Furthermore, formal methods are only part of the solution to the problem related to requirement analysis and success depends crucially on integrating them into a larger process. Formal method axioms are being used in structural evaluation of a software model especially data mining model. Survey of Markovian Behavioral Equivalences supports a merely qualitative analysis, in the sense that it only allows one to establish whether two models pass an arbitrary test in the same way.

Generalized Stochastic Petri nets (GSPNs) can be conveniently used for the analysis of complex models of Discrete Event Dynamic Systems (DEDS) and for their performance and reliability evaluation. Classical Process algebra (CPA) can be used to develop models which may be used to calculate performance measures as well as deduce functional properties of the system.

Markovian Bisimilarity ~MB, Markovian Testing equivalence ~MT, and Markovian Trace equivalence ~MTr with respect to a number of criteria such as exact aggregation, congruence property , sound and complete axiomatization, logical characteristics and verification complexity can be used to model by taking advantage of symmetries within the model. Stochastic model checking can be used to cover both the theory and practical aspects for two important types of probabilistic models such as discrete- and continuous-time Markov chains.

Software Performance Engineering (SPE) should become better integrated into capacity planning. There has been a tremendous amount of research in the

SPE field since it was first proposed as a discipline in 1981. The emphasis will change from finding and correcting design flaws to verification and validation that the system performs as expected. The verification and validation can be implemented using predictive analytics with proper application of the best fit machine learning algorithms. Supervised predictive machine learning, ML algorithms require precise accuracy and minimum errors in addition to putting several factors into consideration (Osisanwo, Akinsola, Awodele, Hinmikaiye, Olakanmi & Akinjobi, 2017)

Software Application Gap Analysis

Software assessment must be determined in a manner whether business requirements are being met, if not, what steps should be taken to ensure they are met successfully. The following must be considered for critical performance evaluation.

1. The natural language axioms deals with first-order predicate logic notation, therefore, it cannot be used for implementing more complex software performance evaluation.
2. The axioms are too generic and might not be robust enough to cope with evaluating certain software classes effectively. Therefore, domain specific axioms should be developed
3. The verification and validation components of the software performance evaluation process should include more analyzer to make it efficient and highly scalable with focus on machine learning.
4. In Markovian Behavioral Equivalences none of the proposals seems to induce an exact aggregation at the Continuous-Time Markov Chains (CTMC) level
5. Markov chains focuses on numerical analysis of modelling but cannot handle novel approaches concerning the special structures in performance evaluation, thus cannot handle complex models.
6. There is need for the development of the solid theoretical framework of model construction and analysis for Generalized Stochastic Petri nets (GSPNs).
7. Determination of the execution probability and the average duration of the computations in the presence of passive transitions is highly is a challenge. Also, the set of logical operators necessary to characterize Markovian behavioral equivalences decreases as the discriminating power of the equivalences decreases.
8. PRISM model checker for stochastic model checking may prove too simplistic for some modelling applications.
9. There is need to extend the quantitative methods to model emerging hardware-software developments, to extend hardware-software measurement technology to support SPE, and to develop interdisciplinary techniques to address the more general definition of performance.

References Références Referencias

1. Akinsola, Jide E. T., Adeagbo, Moruf A., Awoseyi, Ayomikun A. Breast Cancer Predictive Analytics Using Supervised Machine Learning Techniques. International Journal of Advanced Trends in Computer Science and Engineering, 8 (6), 3095-3104, ISSN 2278-3091, November – December 2019. Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse70862019.pdf DOI: https:// doi.org/10.30534/ijatcse/2019/70862019.
2. Akinsola, Jide E. T.; Awodele, Oludele; Idowu, Sunday A.; Kuyoro, Shade O. SQL Injection Attacks Predictive Analytics Using Supervised Machine Learning Techniques, International Journal of Computer Applications Technology and Research, Volume 9–Issue 04, 139-149, 2020, ISSN:-2319–8656. April, 2020. Available at: https://ijcat.com/archieve/volume9/issue4/ijcatr09041004.pdf doi: https://10.7753/IJCATR0904.1004.
3. Almeida, J.B., Frade, M. J., Pinto, J. S., & Melo de Sousa, S., (2011). Rigorous Software Development, A Practical Introduction to Program Verification. Series: Undergraduate Topics in Computer Science. Springer Verlag, 1st Edition, 2011, XIII, 307 p. 52 illus. Soft cover, ISBN 978-0-85729-017-5.
4. Anwaar,A., Junaid,Q., Raihan,R., Arjuna,S.,Andrej,Z. & JonC. (2016). Big data for development: applications and techniques. Big Data Analytics, 1/2,1-24. ISSN: 2058-6345.doi:10.1186/s41044-016-0002-4. Available from: http://i.stanford.edu/~ullman/mmds/ book.pdf.
5. Balbo, G. (2007). Introduction to Generalized Stochastic Petri Nets. Universit`a di Torino, Dipartimento di Informatica Corso Svizzera, 185, 10149 Torino, Italy. SFM 2007, LNCS 4486, pp. 83–131, 2007. © Springer-Verlag Berlin Heidelberg 2007.
6. Balsamo, S & Andrea Marin, A. (2007). Queuing Networks. Dipartimento di Informatica Universit`a Ca' Foscari di Venezia Via Torino 155, 30172 Venezia Mestre, Italy. SFM 2007, LNCS 4486, pp. 34–82, 2007. © Springer-Verlag Berlin Heidelberg 2007.
7. Bernardo, M. & Hillston, J. (2007). Formal Methods for Performance Evaluation. 7th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2007 Bertinoro, Italy, May 28-June 2, 2007. Advanced Lectures. ISSN 0302-9743 © Springer-Verlag Berlin Heidelberg 2007.
8. Bernardo, M. (2007). A Survey of Markovian Behavioral Equivalences. Universit`a di Urbino "Carlo Bo" – Italy Istituto di Scienze e Tecnologie dell'Informazione. SFM 2007, LNCS 4486, pp. 180–219, 2007. © Springer-Verlag Berlin Heidelberg 2007.

9. Bowen & Stavridou (1993). "Safety Critical Systems, Formal Methods and Standards.
10. Bowen, J. P., & Hinchey, M. G. (1995). "Seven More Myths of Formal Methods." IEEE Software 12.4 (1995): pp. 34-41.
11. Clark, A., Gilmore, S., Hillston, J. & Tribastone, M. (2007). Stochastic Process Algebras. LFCS, School of Informatics, University of Edinburgh. SFM 2007, LNCS 4486, pp. 132–179, 2007. © Springer-Verlag Berlin Heidelberg 2007
12. F. Y. Osisanwo, J. E. T. Akinsola, O. Awodele, J. O. Hinmikaiye, O. Olakanmi and J. Akinjobi. Supervised Machine Learning Algorithms: Classification and Comparison. International Journal of Computer Trends and Technology (IJCTT) – Volume 48 Number 3, 2017, https://doi: 10.14445/22312803/IJCTT-V48P126
13. Giammarco, K., Xie, G. & Whitcomb, C. A. (2012). "A Formal Method for Assessing Interoperability using Architecture Model Elements and Relationships." (2012)
14. Jean-Yves & Boudec, L. (2010). "Performance Evaluation of Computer and Communication Systems", EPFL Press, Lausanne, Switzerland, 2010.
15. J. E. T. Akinsola, S. O. Kuyoro, O. Awodele & F. A. Kasali. Performance Evaluation of Supervised Machine Learning Algorithms Using Multi-Criteria Decision Making Techniques. International Conference on Information Technology in Education and Development (ITED) Proceedings, 17 – 34, 2019.
16. Kling, R. (1994). "Systems Safety, Normal Accidents and Social Vulnerability".
17. Kwiatkowska, M., Norman, G. & Parker, D. (2007). Stochastic Model Checking, School of Computer Science, University of Birmingham Edgbaston, Birmingham B15 2TT, United Kingdom SFM 2007, LNCS 4486, pp. 220–270, 2007. © Springer-Verlag Berlin Heidelberg 2007.
18. Lamsweerde, A. V. (2000). "Formal Specification: A Roadmap". Proceedings of the Conference on the Future of Software Engineering. ACM, 2000. pp. 147-159.
19. Leskovec, J., Rajaraman, A. &Ullman, D. J. (2014). Mining of Massive Datasets.
20. Melo de Sousa, S. (2011). Rigorous Software Development: An introduction. (LIACC/DIUBI). RELEASE (UBI), LIACC (Porto), CCTC (Minho) Computer Science Department, University of Beira Interior, Portugal.
21. Michael Collins, M. (1998). Formal Methods. Carnegie Mellon University, 18-849b Dependable Embedded Systems, Spring 1998. Available at: https://users.ece.cmu.edu/~koopman/des_s99/formal_methods/.
22. Miller & Srivas, (1995). Formal Verification of the AAMP5 Microprocessor.
23. Mona, B., Amit, M. & Meenu, D. (2010). Formal Methods: Benefits, Challenges and Future Direction, Journal of Global Research in Computer Science © JGRCS 2010, Volume 4, No. 5, May 2013. Pp. 21-25
24. Rechtin, E. (1992). "The Art of Systems Architecting." IEEE Spectrum 29.10 (1992): pp. 66-69.
25. Rodano, M. & Giammarcob, K. (2013). A Formal Method for Evaluation of a Modeled System Architecture. Procedia Computer Science, Volume 20 (2013). Complex Adaptive Systems, Publication 3 Conference Pp. 210 – 215. Published by Elsevier B.V.
26. Rushby, J. (1995). Formal Methods and their Role in the Certification of Critical Systems - SRI - 1995.
27. Smith, C. U. (2007). Introduction to Software Performance Engineering: Origins and Outstanding Problems. Performance Engineering Services, Santa Fe, NM 87504. SFM 2007, LNCS 4486, pp. 395–428, 2007. © Springer-Verlag Berlin Heidelberg 2007.
28. Stewart, W. J. (2007). Performance Modelling and Markov Chains. Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA. SFM 2007, LNCS 4486, pp. 1–33, © Springer-Verlag Berlin Heidelberg 2007.
29. Wikipedia (2017). Available at: https://en.wikipedia.org/wiki/Formal_methods.