**SPCRD GLOBAL
PUBLISHING**
Sustainable Solutions

RAMSS Review of Applied Management
and Social Sciences

Journal homepage: <http://ramss.spcrd.org>

ISSN (Print): 2708-2024

ISSN (Online): 2708-3640

Formal Specification and Design of E-Learning IMS

^a Amnah Firdous, ^b Najia Saher, ^c Afsah Imtiaz Elahi, ^d Uzma Waheed

^a Lecturer, Computer Science Department, The Government Sadiq College Women University Bahawalpur, Pakistan

Email: amnah@gscwu.edu.pk

^b Assistant Professor in Department of Computer Science and IT, The Islamia University Bahawalpur, Pakistan

Email: najiasaher@gmail.com

^c Assistant Professor in Department of Information Technology, The Islamia University Bahawalpur, Pakistan

Email: afsahimtiazelahe@gmail.com

^d Lecturer, Computer Science Department, The Government Sadiq College Women University Bahawalpur, Pakistan

Email: uzma.waheed@gscwu.edu.pk

ARTICLE DETAILS	ABSTRACT
History: Accepted 25 March 2021 Available Online March 2021	<p><i>The purpose of this research study is to look at, how the utilization of planning techniques / strategies having the most important impacts on any project /organization's success and imaginative problem-solving. In particular Project Planning Techniques, three most important arranging basic strategies including, Gantt graphs, case-based arranging and basic way investigation, were studied & analyzed. As this study is an empirical in nature which purposes to highlight the project standards used to examine the accomplishment of projects and the project planning techniques being practically applicable. It is quantitative research study and questionnaires were asked to access the variables and associate with the existing literature. Project Success criteria were graded which were found to be dissimilar in this field as compare to others. It also presented the knowledge and usage of project planning techniques for categorizing the room of improvement as for as the Project Success is concerned, where some professional trainings are required for the project team members. The role of Effective Project Governance is also very vital for project success as well. The main focus and targeted area is the construction industry of Pakistan.</i></p>
Keywords: Correctness, Colored Petri Nets (CPNs), E-Learning, Information Management System (IMS), Modeling, Verification	
JEL Classification: L86, M15C63, C69	
DOI: 10.47067/ramss.v4i1.125	

© 2021 The authors. Published by SPCRD Global Publishing. This is an open-access article under the Creative Commons Attribution-NonCommercial 4.0

Corresponding author's email address: najiasaher@gmail.com

1. Introduction

Formal specification and verification is based on mathematical proven methods, techniques, languages, and tools. The mathematical concepts (i.e. set theory, predicate logic, temporal logic,

calculus etc.) define the methods, techniques and tools which in turn formally describe, model, and verify problems. Formal methods are rigorous, practical and precise way of software verification. It is important to find precise and exhaustive way to define and model a system, in order to study and verify it's safety and liveness properties. In a software engineering process formal techniques give greater assurances in the application of development methods.

Our goal is to model and verify an Information Management System. Here verification means to ensure the correctness of the system. The prime focus is on correctness properties constituted by a combination of safety and liveness.

To address this problem statement the research questions are given below.

Q. 1. How an e-Learning IMS be formally specified?

Q. 2. How the correctness properties of safety and liveness are ensured?

Q. 3. How CPNs are used to state the behavioral architecture of an e-Learning IMS?

Q. 4. How to ensure that the e-learning IMS model verifies each property specified in the requirement and preliminary design specification.

2. State of the Art

To address this problem statement the research questions are given below.

2.1 Formal methods

A formal specification is precise, unambiguous, structured, and consistent. Formal methods have a mathematical foundation (i.e. discrete mathematics, set theory, propositional calculus, predicate calculus etc.) and are extensively used for the rigorous specification, design, and verification of software systems. They are intended to systematize and introduce rigor in each phase of software development. They provide the basis for consistency and provides standard systematic methods to record assumptions and results. By providing precise description methods, a formal method helps the understanding required to unite and verify artifacts of different phases of software development.

- Formal Specification is precise, unambiguous, structured, and consistent. They are defined by using formal methods. Primarily there are two types of formal languages; Model-Oriented and Properties-Oriented (Algebraic Logic, Temporal Logic etc.).
- Formal Proofs are achieved by using languages, tools based on First-order Predicate Logic, and Calculus.
- Model Checking make use of the formal specification and formal proof to make sure that the system contains all possible properties to be able to handle all possible scenarios that could happen for a given specification.
- Abstraction use mathematical models to create prototype of the entire system for simulation. These formal models define and refine the properties and characteristics of the system.

Why consider formal specification, modeling, and verification?

The development of a formal specification, modeling, and verification provides insights and understanding of the software requirements and software design:

- Clarify customer requirements as formal specifications are precise, well-defined and unambiguous.

- Detect and remove incompleteness, ambiguity, and inconsistency.
- Facilitate property-preserving transformation from requirements to design; and from design to implementation
- Provide a sound foundation for a correct software design
- Provide traceability so that system-level requirements are traceable to sub-systems or components.
- Formal specifications can be applied with different degree of formalism at various points throughout the development process.

2.2 Coloured Petri Nets

CPNs (Kristensen, Christensen, & Jensen, 1998; Kurt Jensen, Lars Michael Kristensen, 2007) is a discrete-event modeling language that combines the Petri-nets and functional programming language CPN-ML which is founded on Standard ML (“Standard ML of New Jersey,” 2020; Ullman, 1998).

CPN has a graphical syntax to construct working models of systems especially concurrent systems and to specify, analyze and verify the correctness properties of these systems. This formal method, with its modeling language, graphical syntax, and exhaustive state-space evaluation is ideal to model complex systems in which concurrency and communication are key characteristics.

There are a number of domains in which CPNs are applied on industrial projects with great results, as in modeling of communication protocols (Billington, Gallasch, & Han, 2004), simulation of data networks (Billington, Diaz, & Rozenberg, 1999), implementation of distributed algorithms, and verification of embedded systems (Adamski, Karatkevich, & Wegrzyn, 2005; Hummel & Fengler, 2005), business process and workflow modeling (Haoxiang, 2019; Lo, Chen, Cheng, & Kung, 2011), and manufacturing systems (Desrochers & Al-Jaar, 1995; Quaglini, 2003).

CPN are ideal to develop and verify concurrent systems. As these systems are complex and its a difficult task to ensure correctness of concurrent processes present in these systems. CPN builds an executable model of these systems. By first building an executable model and then simulating it results into an exhaustive, detailed insight into the analysis, design, and operation of the system. Its results into a less complex and more perfect design (Zhang et al., 2013). Developing an executable model leads to a complete working specification, allowing a systematic investigation of properties which results into a significant decrease of errors.

In the early stage of analysis and design, CPN model representing high level of abstraction is specified. The constructed model is then step-wise refined to build a more refined, detailed, and precise description of the system. CPN models are executable and they can be simulated. This simulation exhaustively investigates, debugs, and validate the system design. Each single step of simulation has its behavior, and all those steps define the collective behavior of the system.

2.3 Formal verification

One of the most important aims of a high-quality software development process is to ensure correct behavior. Verification comprises, analytical methods, such as logical analysis and tracing, intelligent investigation, empirical methods, and experimental strategies i.e. testing and simulation.

Software verification's objective is to guarantee that product completely fulfills all the major essentials. The ultimate goal of verification of procedures, is to set up certainty that the product system is "*fit for reason*". This means that the system must be good enough for its future use. Formal modeling use methods, procedures, and techniques to model complex systems. By building a numerically thorough exhaustive model of an unpredictable system, it is conceivable to check the software properties in a more in-depth than observational testing.

Two well-established and popular approaches for formal verification, used in research as well as large industrial projects are *model checking* and *theorem proving*.

23.1 Model Checking

Model checking focuses on constructing a precise, well-defined, un-ambiguous model of a system, and ensuring that non-functional properties (i.e. constraints and quality attributes) holds in that model. Model checking takes into account time-based logic (i.e. temporal logic). In model checking, the models M are move frameworks and the properties ϕ are recipes in transient logic. To check that a framework fulfills a property, we must do three things (A, 2019):

- Specify the model of the system by a formal notation (i.e. language) of a model checker. This specification notation is the input of the model M ;
- Specify the liveness, progress, safety, and deadlock-freedom properties using the specification notation of the model checker. This results into a temporal logic formula ϕ ;
- These contributions M and ϕ are input to the model checker.

The model checker takes the M and ϕ as input, creates an exhaustive model for state space evaluation and yields the response "yes" if M satisfies ϕ and "no" otherwise; in the latter case, the model checker creates the trace of system actions that results into this failure (i.e. error state).

Model checking systems take as data two elements: a system, whose accuracy we demand to confirm, and an announcement from the pre-requisites that we wish to hold in the system. Most existing model checking systems let the states about the system's performance to be composed in temporal logic. It has stages of:

- **Modeling:** A framework model.
- **Specification:** Regular language detail property in formal basis.
- **Verification:** Algorithm for checking whether a model satisfies a property.

2.3.2 Theorem Proving

The second most important approach for formal verification is theorem proving, which uses *axioms*, *pre-condition*, *post-condition*, *invariants*, and *proof-obligations* to prove properties of the system. Most of the industrial strength Theorem-Provers are founded on set theory, relations and function, propositional calculus, and first-order predicate calculus (Yuan & Herbert, 2011).

2.4 Safety and Liveness

The correctness properties of safety and liveness are specified in formal notation. These properties are then checked exhaustively by model checker by creating an exhaustive state space (i.e. all possible traces). If the specified property is proven to be not correct by the model checker, than it

constructs the counter example to demonstrate the wrong behavior. The counter example gives the exact trace of the fault (Ongenaes et al., 2013). This trace to the faulty behavior gives insight into the actual reasons that leads to the failure, as well as important steps that can be taken to fix the problem. Model checking proves and ensures the correctness properties, and if it is provided with the required input it terminates with a yes/no answer.

Fig. 1. Model Checking.

Safety and liveness properties are vital and can be expanded and formally confirmed by utilizing formal strategies and methods. A safety (i.e. well-being property) stipulates that "terrible things" don't happen amid execution of a system and a liveness property stipulates that "great things" do happen (in the long run). The characteristics of a system to check whether it is workable and something "bad" will never happen. Its objective is to identify protection requirements. Formal verification also focuses on the safety properties (Desrochers & Al-Jaar, 1995). Liveness identifies a set of properties of parallel systems that involve a system to make progress in different situations.

TABLE 1: SAFETY AND LIVENESS (Aziz, Ejaz, & Alam, 2013)

	Safety	Liveness
1.	Something "bad" will never occur.	Something "good" will eventually occur (i.e. but it is not known when).
2.	The program will never create incorrect result (i.e. partial correctness).	The program will produce an outcome (i.e. termination).
3.	These properties contradiction (i.e. conflict) always has a finite witness (i.e. "if something bad happens on an infinite run, then it happens already on some finite prefix").	These properties contradiction (i.e. conflict) never has a finite (i.e. restricted) witness (i.e. "no matter what happens along a finite run, something good could still happen later").
4.	These properties can be checked on controlled performances.	These properties can be checked on infinite execution (i.e. infinite run), they cannot be checked on finite executions (i.e. finite run).

3. Material and Methods

3.1 E-Learning IMS Architecture

Online courses are based around the e-learning IMS. E-learning IMS provides the student a formally designed system where student can view course information like course overview, search courses, chat, web links, download of lectures, scheme of study, course contents etc. Student admin and instructor takes their ID and Passwords, submit their material and participate in discussion boards with tutor or students from E-Learning IMS. A formal CPN based model of this e-learning IMS is proposed.

3.1.1 User View point:

Step-1: Depict the E-Learning IMS in a CPNs form Login.

Step-2: Validate Acceptance check or unacceptable inputs from the database (if unregistered user then asks user to register herself/himself).

Step-3: Describe the user functions or the services (select CS related courses) provided by the E-Learning IMS.

Step-4: Find the required course material text/audio/video.

Step-5: Get the required course material.

Step-6: Logout.

3.1.2 Admin View point:

Step-1: Depict the E-Learning IMS in a CPN form login

Step-2: Validate Acceptance check or unacceptable inputs.

Step-3: Describe the Admin Privileges to add user, update user, delete user, update course material, delete course material, validate or invalidate course material.

Step-4: Logout.

3.1.3 Instructor View point:

Step-1: Depict the E-Learning IMS in a CPNs form Login.

Step-2: Add material in any course and in any format i.e. text, audio and video.

Step-3: Delete material in any course and in any format i.e. text, audio, and video.

Step-4: Logout.

CPNs are especially appropriate for demonstrating and investigating extensive and complex systems.

3.2 CPN based formal verification

CPN based formal verification answers functional checks that confirms (or refutes) accuracy of the e-learning IMS specifications. Formal confirmation of any of the e-learning IMS property is like an exhaustive investigation of the system configuration related to that property (Yuan & Herbert, 2014).

The e-learning IMS system is specified as a directed bipartite graph, the nodes represent the transitions (i.e. events, specified by rectangles) and places (i.e. conditions, specified by circles) (Gonzalez et al., 2016; Mierlo et al., 2018). The pre- and/or post conditions are specified as places joined respectively before and after a transition by directed arcs (i.e. arrows). This graphical representation (i.e. bipartite graph) specifies the step-by-step procedures that incorporate decisions. Data is specified in the form of tokens, these token moves in the graphical model and therefore data can be exhaustively examined and changed when a move fires (Mochalov, Bratchenko, Nikulin, & Yakovlev, 2021).

4. Case Study: Formal Modeling of E-learning Information Management System

CPNs based e-Learning IMS delivers analysis techniques which are used to prove the correctness of e-learning IMS workflow procedures. Verification based on formal modeling of an E-Learning IMS is proposed.

Fig. 3. The graphical formalism elements of CPNs (i.e. place, transitions, and tokens).

4.1 Introduction

The formal CPN models the e-Learning IMS. This CPN model ensures correctness properties and enforces *completeness* and *consistency* of the system.

4.2 Functional Architecture

The CPN model of the e-Learning IMS is organized, and well-structured and provide formal modeling. CPN artifacts of places, transitions, and tokens are used for interactive simulations. A place is represented in ellipse symbol and it can hold a number of tokens. Each token can hold a data values of a number of data types like String, Integers, Unit, Boolean. Values can also be combined to form new values with the help of arithmetic operators. The action are triggered with the help of Transitions. Data flow through Arcs in the form of tokens (Shams & Zamanifar, 2014). The Arcs get input in the form of tokens, and they process these tokens as output into transition.

Fig. 4. User/Admin/Instructor login of E-learning IMS

Fig. 4 shows the basic entry point of the system, it lets the user to login in his particular mode and use the system according to its privileges (i.e. the user gets the services to find computer science related course material in text, audio, and video files).

Fig. 5. User login, enable all Transitions

Fig. 5 shows all transitions are enabled, and all tokens are passed, as a result allowing the three types of users to avail e-learning IMS services.

Fig. 6. New user Registration.

Fig. 6 defines the new login process of E-Learning IMS. There are two major processes Login (User ID, Password) and New-user registration. Each of these processes is represented by a token. First, the user provides the "User Id and Password"; if it is available in database then it is validated and then user is allowed to use the interface. Otherwise it will consider as invalid user.

In the second process the new user submits the form and goes into database. Then next time this new user uses the user id and password to login and getting the privilege of interface. Each token has its variable and value, and it shows the movement of data.

Here below another model that describes the effect of all new users' registration transitions by providing the value of 'x' for Users Id and value of 'y' for password.

Fig. 7. All newly registered users

Fig. 8. Newly registered users as well as invalid users.

Fig. 8 represents the complete picture of this login process, here there are already registered new users with new transitions i.e. “Ibrahim” got the Interface, and in INVALID process two users are invalid due to wrong ID or Password. The performance of the system is evaluated with the help of a number of simulations, these simulations provide performance measure of the proposed system.

Fig. 9. Subject selection, Add material and Add subject

The CPN model gives a proper view of how the tokens pass and how the actions are triggered. Here, consider how CPNs can be used to model the subject's selection, adding course material and adding a new subject into the E-Learning IMS.

Fig. 10. All subjects' course material successfully added including "Java"

Admin Level:

Fig. 11. Admin Roles

Fig. 12. Update course material by Admin

Fig. 13. Delete user by Admin

Fig. 14. All enabled transitions of Admin side tokens

4.3 Instructors Level

Course Instructor has a major role in E-Learning IMS. An instructor Instructors can be many with same privileges to add material and they can also delete material from subject's database in case they found the irrelevant or inappropriate materials.

Fig. 15. Add and delete material

Fig.15 shows the instructors' role in E-Learning IMS after login. Instructors can be many with same privileges to add material and they can also delete material from subject's database in case they

found the irrelevant or inappropriate materials. In this CPN model only one variable is used to pass string.

Fig. 16. Simulation for Logout Colored Petri Nets model

In the above fig. 16, color set variables are initialized to accomplish the logout process for User, Admin and Instructor.

5. Results and Discussion

Formal modeling and verification of E-learning IMS is specified. Our system provides online support for multiple subjects and their relevant audio, video lectures. The subject notes, slides, and books are also provided.

Question 1. How an e-Learning IMS be formally specified?

Animated performance of the finite state mechanism is adopted by using CPNs. The finite state automaton is colored to show the current state of the automaton. CPNs are specifying time base automaton and showing the data flow. The formal and data flow characteristics of CPNs are based on state space evaluation.

Question 2. How the correctness properties of safety and liveness are ensured?

The properties of CPNs are functional and shows the correctness of the system that is related to the behavioral and functional modeling. The main behavior is ensuring liveness and safety that is presented in CPN i.e. login, subjects selection, adding subjects, validating subjects, admin role, instructor role, interface for learner, logout etc.

Question 3. How CPNs are used to state the behavioral architecture of an e-Learning IMS?

CPN block diagrams of the E-Learning IMS show the architectural as well as behavioral design of the system. It elaborates all elements in a top-down structure. It presents a clear picture of the system functions and also depicts the interactive behavior. Both the static structure as well as dynamic behavior of a system is projected using CPNs.

Question 4. How to ensure that the e-learning IMS model verifies each property specified in the requirement and preliminary design specification.

The correctness properties are specified formally. The state space analysis checks and verifies the desired properties.

6. Future Work

Exhaustive investigation and formal validation in the form of formal CPNs of E-Learning IMS is proposed. System correctness and verification are utmost reality and systems are formally modeled to give better understanding of current and upcoming requirements. Our proposed formal model provides a complete understanding for modeling, analysis, and formal verification.

The following recommendations have been made:

- CPNs provide exhaustive formal verification of the e-learning IMS.
- Time-based automata (i.e. temporal logic) is used for every single transition to evaluate systems performance and its effectiveness.

References

- A, P. P. (2019). Enhanced Edge Model for Big Data in the Internet of Things Based Applications. *Journal of Trends in Computer Science and Smart Technology*, 1(1), 63-73. <https://doi.org/10.36548/jtcsst.2019.1.006>
- Adamski, M., Karatkevich, A., & Wegrzyn, M. (2005). Design of embedded control systems. Design of Embedded Control Systems. Springer US. <https://doi.org/10.1007/o-387-28327-7>
- Aziz, S., Ejaz, A., & Alam, S. E. (2013). Mortality pattern in a trust hospital: A hospital based study in Karachi. Retrieved April 23, 2021, from https://www.researchgate.net/publication/286805160_Mortality_pattern_in_a_trust_hospital_A_hospital_based_study_in_Karachi
- Billington, J., Diaz, M., & Rozenberg, G. (Eds.). (1999). Application of Petri Nets to Communication Networks (Vol. 1605). Berlin, Heidelberg: Springer Berlin Heidelberg. <https://doi.org/10.1007/BFb0097770>
- Billington, J., Gallasch, G. E., & Han, B. (2004). A coloured Petri net approach to protocol verification. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 3098, 210-290. https://doi.org/10.1007/978-3-540-27755-2_6
- Desrochers, A. A., & Al-Jaar, R. Y. (1995). Applications of Petri Nets in Manufacturing Systems; Modeling, Control, and Performance Analysis. *IEEE Control Systems*. <https://doi.org/10.1109/MCS.1995.476396>
- Gonzalez, A., Gronsund, P., Mahmood, K., Helvik, B., Heegaard, P., & Nencioni, G. (2016). Service Availability in the NFV Virtualized Evolved Packet Core (pp. 1-6). Institute of Electrical and Electronics Engineers (IEEE). <https://doi.org/10.1109/glocom.2015.7417254>
- Haolang, D. W. (2019). Trust Management of Communication Architectures of Internet of Things. *Journal of Trends in Computer Science and Smart Technology*, 2019(2), 121-130. <https://doi.org/10.36548/jtcsst.2019.2.005>
- Hummel, T., & Fengler, W. (2005). Design of embedded control systems using hybrid petri nets. In *Design of Embedded Control Systems* (pp. 139-149). Springer US. https://doi.org/10.1007/o-387-28327-7_12
- Kristensen, L. M., Christensen, S., & Jensen, K. (1998). The practitioner's guide to coloured Petri nets. *International Journal on Software Tools for Technology Transfer*, 2(2), 98-132. <https://doi.org/10.1007/s100090050021>
- Kurt Jensen, Lars Michael Kristensen, L. W. (2007). Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use. Volume 1 | Kurt Jensen | Springer. Retrieved April 23, 2021, from <https://www.springer.com/gp/book/9783540609438>
- Lo, C. C., Chen, C. H., Cheng, D. Y., & Kung, H. Y. (2011). Ubiquitous healthcare service system with

- context-awareness capability: Design and implementation. *Expert Systems with Applications*, 38(4), 4416–4436. <https://doi.org/10.1016/j.eswa.2010.09.111>
- Mierlo, S. Van, Tendeloo, Y. Van, Dávid, I., Meyers, B., Gebremichael, A., & Vangheluwe, H. (2018). A multi-paradigm approach for modelling service interactions in model-driven engineering processes. Retrieved from <https://research.vu.nl/en/publications/a-multi-paradigm-approach-for-modelling-service-interactions-in-m>
- Mochalov, V. P., Bratchenko, N. Y., Nikulin, V. I., & Yakovlev, S. V. (2021). Methods for functional parallel processes implementation for network functions virtualization based on coloured petri nets. In *AIP Conference Proceedings* (Vol. 2318, p. 130004). American Institute of Physics Inc. <https://doi.org/10.1063/5.0035806>
- Ongenaes, F., Claeys, M., Dupont, T., Kerckhove, W., Verhoeve, P., Dhaene, T., & De Turck, F. (2013). A probabilistic ontology-based platform for self-learning context-aware healthcare applications. *Expert Systems with Applications*, 40(18), 7629–7646. <https://doi.org/10.1016/j.eswa.2013.07.038>
- Quaglini, S. (2003). Workflow Management—Models, Methods and Systems. *Artificial Intelligence in Medicine*, 27(3), 393–396. [https://doi.org/10.1016/s0933-3657\(03\)00011-3](https://doi.org/10.1016/s0933-3657(03)00011-3)
- Shams, H., & Zamanifar, K. (2014). MVCC: An architectural pattern for developing context-aware frameworks. In *Procedia Computer Science* (Vol. 34, pp. 344–351). Elsevier B.V. <https://doi.org/10.1016/j.procs.2014.07.035>
- Standard ML of New Jersey. (2020). Retrieved April 23, 2021, from <https://www.smlnj.org/>
- Ullman, J. (1998). *Elements of ML Programming*, ML97 Edition: Retrieved April 23, 2021, from <https://www.amazon.com/Elements-ML-Programming-ML97-2nd/dp/0137903871>
- Yuan, B., & Herbert, J. (2011). Non-intrusive Movement Detection in CARA Pervasive Healthcare Application, 360–362.
- Yuan, B., & Herbert, J. (2014). Context-aware hybrid reasoning framework for pervasive healthcare. *Personal and Ubiquitous Computing*, 18(4), 865–881. <https://doi.org/10.1007/s00779-013-0696-5>
- Zhang, D., Huang, H., Lai, C. F., Liang, X., Zou, Q., & Guo, M. (2013). Survey on context-awareness in ubiquitous media. *Multimedia Tools and Applications*, 67(1), 179–211. <https://doi.org/10.1007/s11042-011-0940-9>.