OPEN ACCESS

Kashf Journal of Multidisciplinary Research
Vol: 02 - Issue 3 (2025)
P-ISSN: 3007-1992 E-ISSN: 3007-200X
https://kjmr.com.pk

A Systematic Approach to Formal Verification and Validation of Embedded Systems: Enhancing Reliability and Safety

Aafia Latif
Department of CS & IT, Govt Graduate College Burewala, Pakistan
Sadia Latif*
Department of Computer Science , Bahauddin Zakaria University Multan, Pakistan
Alina Shaikh
Department of Computer Science, NCBA&E Multan, Pakistan
Rana Muhammad Nadeem
Department of CS & IT, Govt Graduate College Burewala, Pakistan
Abdul Manan Razzaq
Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan

****Corresponding author: Sadia Latif (sadialatifbzu@gmail.com)***

Article Info

Abstract

This article addresses the problem of model-based early design verification of systems engineering applications expressed using System Modelling Language (SysML). This thesis describes the formal specification and verification approach for the early design verification of real time embedded systems. The main objective is to assess the design from its functional requirements to ensure the conformance of system requirements at early design phase. My main contribution is a novel approach to model and verify the hybrid systems using SysML Block Diagram and hybrid automata. A formal verification technique "Model checking" has been used for the formal verification of SysML Block Diagram of real time embedded systems. The PRISM models Probabilistic Timed Automata (PTA) and Continuous Time Markov Chain (CTMC) are taken for the formalization and mapping of SysML Block Diagram. The user requirements are expressed as temporal logics Probabilistic Computational Tree Logic (PCTL) and Continuous Stochastic Logic (CSL) for the properties verification against the PTA and CTMC model. Moreover, it define hybrid automata based formal specification to extend the behavior of SysML Block Diagram. The upgraded SysML Block Diagram has more ability to capture the discrete and continuous behavior of hybrid systems accurately. The SysML block diagram is verified with PRISM in order to show the correctness of system functionality. This thesis presents the effectiveness and validity of proposed approach with the help of four case studies. The discrete and continuous time constraints are considered in case studies along with the system functionality. The discrete and continuous time constraint helps the system and software engineer to verify the real time aspect of system along with its system functionality.

Keywords:
SysML (System Modeling Language) , Distributed Environment, Temporal Logic , Continuous Stochastic Logic (CSL) , Probabilistic Computational Tree Logic (PCTL), Extended Computation Tree logic (ECTL) , Unified Modeling Language (UML)

Introduction

This research project consists of three parts. .

Embedded systems are very large and complex, and such systems need to be checked and verified to reduce risk during the development phase. Graphic modeling of such real-time systems requires greater accuracy and reliability. Products, features. Use mathematical thinking to check the accuracy of your software. In other words, in reviews, i.e. H. software testing, this process is used to recognize errors. Can be used to verify consistency of technical methods during design. This can lead to financial issues, product failures, human errors, and other issues, leading to dangerous consequences. Important times are related to actual times and are important when the system is unable to work on time. Such current technologies must be identified and implemented at the initial design stage to prevent risks that may arise at later stages. The cost of correcting errors after the development process is approximately 500 more than the cost of correcting errors during the initial design phase. Graphic modeling and early reviews of graphic models are necessary to reduce the risks that may occur at later stages of development. Design, specifications and analysis of real-time am-bed systems is a challenging task for software and systems engineers. Therefore, these issues need to be identified and verified at the early stages of system design. The purpose of this paper is to model real-time diagrams and perform validation using SYSML block diagrams to verify the accuracy of the system according to the requirements. Recognizing the graphics models of these systems is difficult due to time addiction. The SYSML block diagram represents all the components and explains the interactions between them. These graphic patterns should be recorded during the early construction stage. Otherwise, it could lead to product failures and loss of sales. It helps you make more accurate and accurate decisions about your users. The main purpose of this paper is to demonstrate the properties and analysis of SYSML block diagrams with the help of real-time visualization. The goal is to prove the accuracy of system operations in both discrete and continuous time. The review process [3] includes model testing, interrupt analysis, white box testing, black box testing, and more. We chose this as one of the corresponding verification methods for the system for the system. inspection. A clear code is provided to explain the rights of users of the system. This specification includes formal processes and written models for real modeling and machine learning. Specific guidelines provide guidelines for developing reliable and correct applications.

2. Materials and Methods

For graphic modeling of embedded systems in real time, SYSML was chosen as the graphic modeling language for modeling such systems. Although SYSML offers a variety of diagrams, it takes over the SYSML block diagram for graphic modeling of embedded systems, as it provides clearer and faster system functionality. Internal block diagram programs provide an internal view of a system block or white box view, as they are instantiated from a block definition diagram that represents only the main system block. Compound blocks are displayed as block portions of the internal block diagram and provide a view of the internal system components. The internal structure of system components is easy to inspect in internal block diagrams and therefore provides the basis for checking the functionality of internal and external components.

Presenting formal specifications using hybrid machines, overcome the limitations of SYSML models and improved SYSML, and formal specifications I created a block diagram from The updated SYSML block diagram is used to model hybrid system components. To confirm system modifications, model testing technology was selected as one of the formal verification techniques for checking the graphic models of actual time packaging systems. A methodology for automatic review of SYSML models of embedded systems in real time has been proposed. The SYSML block diagram is commented on the flow port to display the interactions between different blocks. The SYSML model is translated into the PRISM model (PTA and CTMC), and the user requirements are translated into time logic (PCTL and CSL). The examiner of the PRISM model occupies both the PRISM model and the property, showing results indicating that the property is true or incorrect. The PRISM model tester shows counters where properties are not true

and helps to modify the SYSML model. Figure 3.1 illustrates the proposed approach to explain how a graphic model of a system embedded in real time is validated using the model testing tool PRISM.

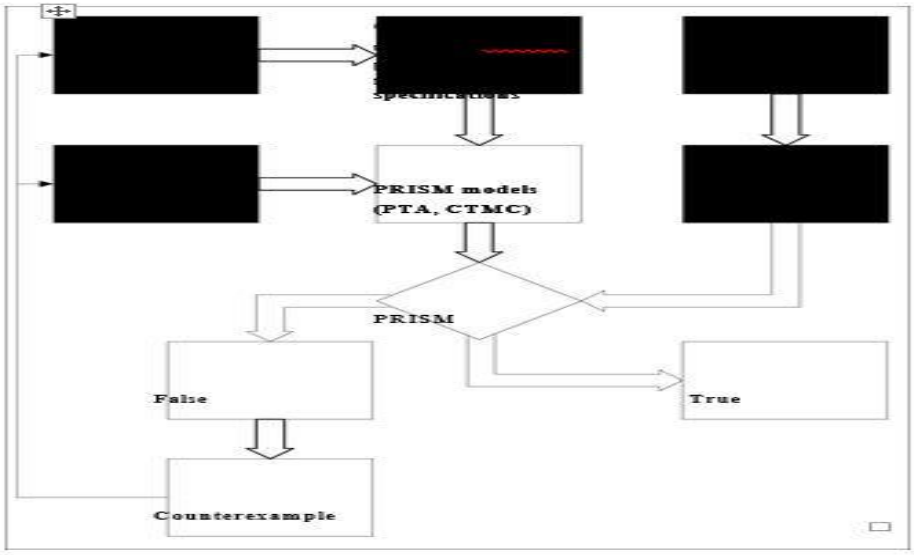

Figure 3.1 Proposed Methodology

3. PRISM

3.1 Why PRISM

There are different model testing tools to verify graphic models. Uppal is a model testing tool to verify real systems that can verify only single time-based requirements. Divine is a parallel model tester that does not support time-controlled machines and is not applied for time-controlling. Prism is a parallel model tester that supports both single and continuous time real-time systems. You can use prisms to view both time control and unlimited requirements. I used the PRISM model tester and tested the functionality of the system through the SYSML-NERTERNAL block diagram and the enhanced SYSML block diagram. Prisma is a concurrent model for model testing and allows parallel review of components. Thus, PRISM model checkers are appropriate for case studies of real-time embedded systems since they can verify models within less time compared to other model auditors. In addition, prisms are also appropriate for verifying stochastic time coefficients. It can also give counters if the properties of the model are not satisfied. This assists you in identifying the absence of models.

3.2 Comparison between PRISM and other Model Checkers

The model checking tool Prisma has many features when compared to other model review tools. The PRISM model tester supports four types of models. Markov Chain (CTMC), Stochastic Machine (PA), Stochastic Time-Control Automatic (PTA), Markov Decision Process (MDP). Discrete-Time Markov Chains (DTMCS) are models based on discrete-time transition systems and are probability distributions that are used randomly for synchronization. The Continuous Time Markov Chain (DTMC) is designed to support models with infinite small time steps continuous periods. Markov Decision Process (MDP) is an extension of the DTMC model that supports non-deterministic features and is suitable for parallel processes. Stochastic Automata (PAS) is a model that supports time-controlled machines, while Stochastic Automatic (PTA) is the best model for determining the probability of a state without supporting time-controlled machines. PRISMA supports a wide range of property specifications, such as PCTL, CSL, LTL, PLTL, PCTL*, and more, compared to other model auditors. Features such as deadlock recognition, counterexamples, witnesses, diagrams, and graphic user interfaces (GUIs). PRISM uses reviews in a variety of application domains, including distributed algorithms, communications, multimedia protocols, security protocols, and biological systems. In recent years, PRISM has been used to check UML/SYSML-based models, and using this model testing tool, only UML/SYSML models such as activity diagrams and

state diagrams have been validated. We chose Prism as a model checking tool for wide support for features. A comparison between the characteristics of different model testers is shown in Table 3.1.

3.2 Comparison between PRISM and other Model Checkers

Table 3.1 Model Checking Tools-Comparison Chart

Name	Type/	Modeling	Properties	Suitability for discrete		Concurrent	GUI	Latest	Target Domain	Available
	Support	Language	specification	and continuous		support		Release		mapping
			language	systems						technique for
				Discrete	Continuous					SysML/UML
				systems	systems					
PRISM	Probabi list	PEPA,		Discrete	Continuous	Yes	Yes	PRSM	Distributed	Activity
	ic, real	PRISM	PCTL, CSL,					version 4.2.1,	algorithms,	Diagram,
	time,	language,	LTL, PLTL,					4 December,	communication	State chart
	hybrid	Plain						2014	and multimedia	diagram,
	approach,								protocols,	Sequence
	linear,								security	diagram
	nonlinear,								protocols,	
	DTMC,C								biological	
	TMC,								systems	
	MDP,									
	PA,PTA									
Hy tech	Linear	C++, linear	CTL	Discrete	Continuous	Yes	No	Hy tech	embedded	SysML
	hybrid	Hybrid						version 1.04,	systems	

	model	Automata						October, 1996		
Space Ex	Non-	ECML		Discrete	Continuous	No	Yes	Space Ex	reachability	No
	Linear,	(ETRI CPS						v0.9.8b, 11-	algorithm	
	piecewise	Modeling	None					07-2013		
	constant,	Language)								
	affine,									
PHAVer	Linear,	Linear		Discrete	Continuous	No	Yes	PHAVer	safety and	No
	piecewise	Hybrid	None					0.38, 12 Feb,	reachability	
	affine	Automata						2007	analysis	
	dynamics									
UPPAAL	Real-time	Timed	TCTL subset	Discrete	No	No	Yes	Uppal 4.0.13	Algorithmic	State machine
		automata,						(Sep 27,	verification	Diagram
		C subset						2010)		

Table 3.2 (cont'd)

DIVINE	**Plain,**	**DVE input**	**LTL**	**No**	**No**	**Yes**	**Yes**	**DIVINE**	**Algorithmic**	**Sequence**
	parallel	language,						3.2.1,	verification	Diagram,
	verification	C/C++,						November		Activity
	n	Timed						2014		Diagram
		Automata								
CIVL	concurrent	C++	Languages/A	No	No	Yes	No	CIVL V0.14	safety properties	No
			PIs to CIVL-					31-10-2014	of CIVL-C	
			C						programs	

Charon	Hybrid	Charon	MEDL	Discrete	Continuous	Yes	Yes	January 23,	Embedded	No
	system	Language						2003	systems	
KRONOS	Real-time	Kronos	TCTL	Discrete	No	No	No	Sep 5, 2002	Embedded	State machine
		Language							systems	diagram
d/dt	hybrid	d/dt	CTL	Discrete	Continuous	No	No		reachability	No
	systems	Language							analysis	
NuSMV	Plain	SMV	CTL, LTL	No	No	No	No	Version 2.5.4,	synchronous	Sequence
								October 28,	finite-state and	diagram,
								2011	infinite-state	
									systems	
Cadence	Plain	Cadence	CTL, LTL	No	No	No	No	version 2.5,	computer	functional block
SMV		SMV,						July 30, 1998	hardware designs	diagram
		Verilog								
PAT	Probabilist	CSP	LTL	Yes	No	Yes	Yes	Version 3.5.1,	real-time system	state machine
	ic, Plain,							August 13,		diagram
	Real-time							2013		
SPIN	Plain	Promela	LTL	No	No	Yes	Yes	Version 6,	Multi-threaded	No
								December,	software	
								2010	applications	
TAPAAL	Real Time	Petri nets	TCTL	Yes	Yes	No	Yes	Version	Petri nets,	No
								3.1.3, Dec 2,	Timed-Arc	

								2014		

4. Preliminaries

Functional block diagram is a block diagram that shows the relationship between systems and functions in software engineering. A block diagram is a representation of an obstructing system function, and indicates that it is linked to other blocks and displays their relationship. figure. In SYSML, a block is viewed as a fundamental unit that represents the structure of a system. Blocks may represent software and hardware components. SYSML employs a block definition diagram to define blocks from a functional view and indicate the structural relationships of blocks. The internal block diagram is analogous to the traditional block diagram and describes the internal structure of the block. The internal block diagram [16] is derived from the UML composite structure diagram [15] with some extensions and rules that are specified by SYSML. The internal block diagram consists of parts, ports, and connections that illustrate the activity of the block's internal structure. The building blocks from the block definition diagram are part of the internal block diagram. A connection is utilized to connect two blocks, and the parts are linked using a connection and communicated using a port. Two types of ports in the block exist. Standard port and river connection. Communication between blocks is done with the assistance of ports. A block contains two ports, that is, H. Standard Port and Fluid Sport, which is utilized for transferring data from a block to another. A flow port is a point of communication through which help or output of elements like energy and data move [16]. There is River Sports for data transfer between blocks. There are two kinds of flow connections: H. Nuclear Fluss Sport and Non -Tatomar River Sports. Atomic flow ports are categorized after the blocks of communication elements and non-a commercial flow ports, by flow specifications for element communication. Nuclear flow ports were utilized for communication among blocks. Standard ports call surfaces for communication and operation between blocks through interfaces.

4.1Timed Automata

A timed automata (TA) models the real-time behavior of embedded systems. We employ timed automata which relate the functionality of internal system components of internal block diagram and real-time attributes of embedded system. Timed automata [43] represents a finite set of real value clocks that can grow with time and which can be reset.

Timed automata [43] is a tuple TA = (L, Li, E, C, clock, guard, inv, lab) where,

- L is a finite set of location
- Li∈L is an initial location
- E is a set of edges
- C is a finite set of clocks
- Clock: a function assigned to each edge
- Guard: a function labels each edge with a clock constraint
- Inv: function is assigned to each location an invariant
- lab: L→2AP is a labeling function assign to each location, a set of atomic proposition

4.2 Continuous Time Markov Chain

Continuous time Markov chain is used for the modeling of real time systems that have small time steps and it is used for the performance evaluation, reliability and dependability analysis.

A continuous-time Markov chain [44] is a tuple C = (S, AP, L, α, P, E) where,

- S is a finite set of states
- AP is a finite set of atomic propositions
- L : S → 2 AP is the labeling function
- α ∈ Distr (S) is the initial distribution
- P : S × S → [0, 1] is a stochastic matrix
- E : S → R≥0 is the exit rate function

4.3 Hybrid Automata

Hybrid automation is a finite state machine with discrete jump transition and continuous variables. Discrete action is an event whose value change at different point of time while transition to a finite set of control location. The continuous behavior shows the continuous variables whose value change with the passing of time within state. Hybrid automation is ideal for those systems that are composed of both discrete and continuous behavior.

A hybrid automaton [45] is a tuple HA = (L, ∑, lab edge, X, Init, Invariant, Flow, Jump_func) where,

- L represents the finite set of control location and it shows the control mode
- ∑ represents the finite set of event names
- lab edge is a finite set of labeled edges and it represents the discrete changes of control mode. The changes are labeled with event names taken
- from label ∑
- X represent the finite set of real valued variable
- Init is a predicate that indicates the value start from initial location
- Invariant is a predicate that sets the value constraint for location
- Flow is a flow predicate, which represents the continuous evolution when control of hybrid system is in location l
- Jump_func is a function assigned to each edge a predicate that provide discrete events when move from one location to another

4.4 Probabilistic Computational Tree Logic

For the incorporation of user requirements as properties, PRISM supports two specification languages PCTL and CSL for property specification. To verify a Probabilistic timed automata, we write properties in PCTL to express its related specifications. PCTL is an extension of CTL with probabilistic operator [23]. The „P" operator in PCTL [46] has a probability bound (p∈ [0, 1]) and a relational operator (∼∈{<, ≤, ≥, >}).

PRISM verifies probabilistic properties with temporal logic PCTL and CSL but it can also provide supports for verifying non-probabilistic properties by using operators from computation tree logic (CTL). CTL uses the A and E operators instead of the probabilistic operator „P" to show whether all paths or some path satisfy a particular path formula.

4.5 Continuous Stochastic Logic

Continuous Stochastic Logic (CSL) called branching time logic and it is similar to temporal logic Computation Tree Logic (CTL). PCTL operators have been adopted by CSL [47] and PCTL has operators like probability bound (p∈ [0, 1]) and a relational operator (∼∈ {<, ≤, ≥, >}). For the incorporation of user requirements to model checker property specification language, PRISM supports two property specification languages i.e. Probabilistic Computation Tree Logic (PCTL) and Continuous Stochastic Logic (CSL). PCTL is used for the MDP, DTMC models and CSL is used with CTMC model.

5. Implementation and Experimental Results:

In this section, a case study of Ticket Vending machine is given and then applied that shows the correctness of proposed approach in discrete time constraint. First, a case study is given and Internal Block diagram is employed for graphical modeling of case study. The graphical model is translated into the input language of PRISM model checker. Last, the user needs are given, converted to temporal logic PCTL and subsequently proved against the PRISM model. In the last, experimental outcomes are also included of this case study.

It's an example of ticket vending machine. The ticket vending machine contains following components; magnetic strip card reader, keypad, display panel, controller, and ticket generator. When the user inserts the card in the magnetic strip card reader, then the

In Figure 5.1, the data is flowing among the blocks through flow ports and it shows the direction of data in two ways, either in one direction (←) (→) or in both directions (↔). The left arrow represents that the data is incoming while the right arrow describes that the data is out-going. In Figure 5.1, the block magnetic strip card reader receives input from the user who interacts with the card reader block through <<flow port>> in: card inserted and then send this data through <<flow port>> out: read data to controller block. The dotted arrow as shown in Figure 5.1 shows the transition of state among blocks. The block is receiving data through <<flow ports>> if the arrow head is towards the block and the block is sending data if the arrow tail is towards the block.

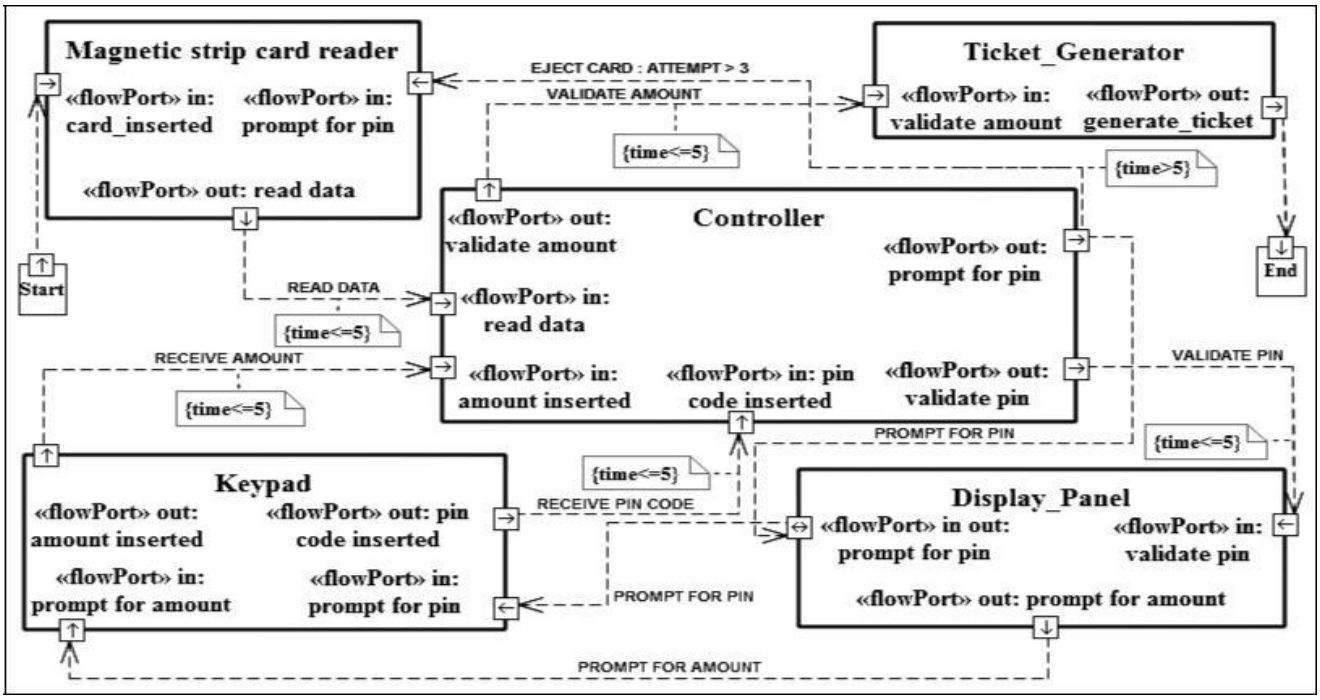

Figure 5.1 Internal Block Diagram- Ticket Vending Machine

5.2.1 Mapping of Internal Block Diagram with PRISM Notations for Verification of Model

The SYSML model of ticket sales machine has been checked against a model checker prism. The following is an explanation of SYSML internal block diagrams allocation in the stochastic time controller model in Prisma. The PRISM model tester relies on a module where every module is a system component [46]. The module has variables to show the state transition when all actions are invoked. The command begins with a square bracket[] and the guard (condition), and, if the action is indicated, depict interactions between various modules. Module. Magnetic strip card reader, keyboard, controller, display panel, ticket generator. PRISM model mapping - SSML - The fragment code for the INTERNAL block diagram is given in Figure 5.2.

```
module magnetic_strip_CardReader
card_reader: [0..3] init 0;
x:clock;
[]card_reader = 0 -> (card_reader' = 1) & (x'= 0);
[prompt_for_pin_1]card_reader = 1 & x<=5-> (card_reader' = 2);
[]card_reader = 2 & x>=6 -> (card_reader' = 3);
[]card_reader = 3 -> (card_reader' = 0);
[eject_attempt]card_reader = 2 -> (card_reader' = 3);
[eject_timeunit_pin]card_reader = 2  -> (card_reader' = 3);
[eject_timeunit_amount]card_reader = 2  -> (card_reader' = 3);
endmodule
```

Figure 5.2 Fragment code of PRISM- Ticket Vending Machine

For instance, if a system is in State S; then its transition is represented by the symbol S'. The transition between states in the case study is shown as "[ ] card reader = 0
→ (card reader' = 1) & (x'=0);". In above guard (condition), 0 represents "ready" state and 1 represents "card inserted" state. The card_ reader is a variable that changes its state from "ready" to "card inserted "and "x" represents the clock that reset to zero. The data type "clock" is defined as "x: clock;" where "x" is a variable represents a clock.
In PRISM, the blocks interact with each other using synchronizing mode. In figure 5.2, The label "[eject attempt]" between modules "magnetic strip card reader" and "controller" indicates the synchronization between both modules and a transition arrow label shows it with Eject card: attempt>3 as shown in Figure 5.1. The synchronization shows that the module "controller" interacts with module "magnetic strip card reader" to eject card if the number of attempts to insert pin code is greater than 3.
In figure 5.2, the label [eject_timeunit_pin] between modules "magnetic strip card reader" and "controller" indicates that the "controller" module will eject the card, if the user does not enter the pin code within five units of time. The label [eject_timeunit_amount] between modules "magnetic strip card reader" and "controller" indicates that the "controller" module will eject the card; if the amount is not entered within five units of time

5.2.2 Properties

We have identified the properties that are checked against the translated model to establish internal block diagram of ticket vending machine correctness. We employed PCTL as temporal logic and checked the properties. PRISM employed CTL operators like A and E to satisfy a specific path formula. The operator "A" means that the property will always be true for all paths and operator "E" means that the property will eventually be true for all paths. We apply "E" operator with properties that fulfill a specific path formula. The "E" operator means that there should be a minimum of one path exists that leads to the desired state and only those paths will lead to the desired state that fulfill the given condition. The "F" operator is a special instance of Until "U" operator and it means that the property is true only when it fulfills the given condition. The temporal operator "F" holds true at some point on the path. The properties are given in Table 5.1 and its PCTL code is mentioned in Table 5.2.

5.2.3 Experimental Results and Analysis

In sections 5.2, 5.2.1 and 5.2.2, we conduct the experiment with case study and describe the mapping of internal block diagram to PRISM. We employ PRISM to illustrate the correctness of our proposed methodology. We check functional requirements given in Table 5.1 that establishes the correctness of ticket vending machine. For checking, the five properties are converted to PCTL format and then checked with PRISM. Table 5.2 demonstrates the PCTL formula of five properties and Figure 5.3 demonstrates experimental results in PRISM.

Table 5.1 Properties-Ticket Vending Machine

Property No.	**Description**
P1	Magnetic strip card reader may take maximum 5 units of time to read the card information
P2	Validation of pin may not take more than 5 units of time
P3	Validation of amount may not take more than 5 units of time

	Magnetic strip card reader should eject the	car d	i f	i t	takes
P4					
	more than 5 units of time to insert pin code				
	Magnetic strip card reader should eject the	car d	i f	i t	takes
P5					
	more than 5 units of time to insert amount				

Table 5.2 Properties Verified on PRISM-Ticket Vending Machine

Property		
	PCTL properties	**Status**
No.		
P1	E [F card reader=2&x<=5]	Satisfied
P2	E [F controller=3&y<=5]	Satisfied
P3	E [F controller=6&y<=5]	Satisfied
P4	E [F (controller=1&y>=6) & card reader=3]	Satisfied
P5	E [F (controller&=4y>=6) &card reader=3]	Satisfied

The five properties in Table 5.1 are classified as timed properties and considering the discrete time aspect. Table 5.2 shows PCTL code for five properties that are verified with PRISM and it shows the status of properties as well. All properties are satisfied as shown in Table 5.2. The PRISM tool allows us to verify discrete time based properties.

The properties mentioned in Table 5.1 are the functional requirements that reflect the normal behavior of ticket vending machine. The non-functional requirements such as performance, reliability and efficiency are also met along with functional requirements with the assistance of discrete time constraint. The card reader reads the card information within five units of time that reflects the performance and effectiveness of the system. The correctness of the system depends on the aspect that the system will not exceed five time units in reading the card's information. And system effectiveness proves that the system will carry out its task as described in user requirements.

The case study given and proven is one of real-time embedded systems. The proposed method assists in the verification of the real-time embedded systems. The method can display correct results if we implement it to the real-time embedded systems. We took time factor into consideration in our case study and proved the system correctness together with discrete time constraint at an early design phase. The time constraint is a very important aspect in real-time embedded systems and therefore it is required for system engineer to verify, either the real-time embedded systems would finish the system functions within

the given time or not. The approach gives the system engineer a means to verify the system correctness as well as a discrete time limit at early design stage so that redesigning the entire system in the future is not costly.

The methodology provides the correct and reliable results. It facilitates the system engineer to model and verify the real-time embedded system in less time as the internal block diagram helps to represent the system features clearly and quickly. The internal block diagram was not used as a design model in the past for the verification of early design of real-time systems. We involve the discrete time constraint in internal block diagram for the timed based modeling of case study and then verified the timed based properties that show the correctness and effectiveness of the system. PRISM also generates a counter example which shows the path of the state where condition is not true.

The counter example is more useful to find errors in the early design phase and encourages us to improve the SysML model that ultimately saves the time, cost and efforts of system engineer that can take place later during the development of real-time embedded system.

Figure 5.3 Automatic Verification of Properties in PRISM- Ticket Vending Machine

5.3 Continuous Time Case Study and its Graphical Modeling

In this section, a case study of Liquid fertilizer mixing plant is described and then implemented that proves the validity of proposed approach in continuous time constraint. First, a case study is described and Internal Block diagram is employed for graphical modeling of case study. The graphical model is translated into the input language of PRISM model checker. Lastly, the user requirements are given, converted to temporal logic CSL and then model-checked against the PRISM model. In the latter, the experimental findings are also mentioned of this case study.

The plant of liquid fertilizer mixing consists of three sensors, four regulators, a mixing unit and a boiling unit. There are three containers with different liquids and each container has a regulator to control the flow of liquid from the container. Liquid flows to mixing unit from first, second and finally from third container. All regulators switch-on and switch-off one by one. Three sensors are attached in order to sense the level of the liquid. The sensor one, in turn, switches-on regulator one until the liquid from container one reaches a desired level. The sensor two then lets regulator two switch-on until the liquid from container two reach a specific level. Finally, sensor three enables regulator three to switch-on until the liquid in container three reach a desired level.

Any regulator does not turn-on if any other regulator is turned-on. Then, mixing unit blends all liquids for up to 70.5 units of time but at least 50.5 units of time. Then, the regulator four turns-on to pump liquid from mixing unit to boiling unit. Lastly, boiling unit boils liquid for up to 110.5 units of time but at least 80.5 units of time.

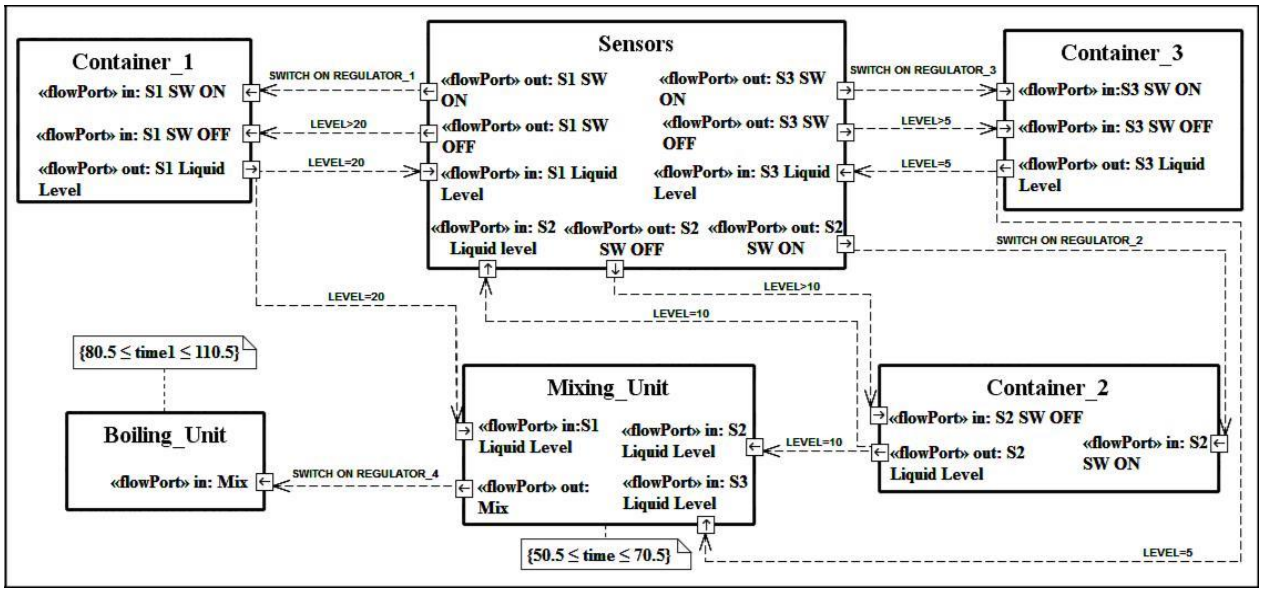

Figure 5.4 Internal Block Diagram-Liquid Fertilizer Mixing Plant

We represent the liquid fertilizer case study with internal block diagram of SysML as depicted in Figure 5.4 which gives six blocks for sensor, container_1, container_2, container_3, mixing unit and boiling unit. Interaction between blocks is represented with the assistance of flow-ports. The two blocks mixing unit and boiling units are labeled with a time period ti= [x, y], where „x" are the earliest time for action completed and „y" are the maximum time for action completed. The action mixing _unit takes at least 50.5 and at most 70.5 units of time to end and action boiling unit takes at least 80.5 and at most 110.5 units of time to end.

In Figure 5.4, data sending and receiving is done by flow ports and its direction of arrow towards block (←) shows that block is receiving the data and direction of arrow away from block (→) shows that block is sending the data. The dashed arrow between blocks indicates the transition of state from one to another state as shown in Figure 5.4.

5.3.1 Mapping SysML Internal Block Diagram to CTMC

SysML internal block diagram of mixing plant is checked with the help of „PRISM".

The SysML model is converted into CTMC model and continuous time constraint is employed to exhibit the real time behavior of the embedded system. PRISM language is module based and thus we converted every block in Figure 5.4 into module and converted six blocks into six modules. The module has variables, guard and commands. The command marked with action depicts the communication between modules. The flow-ports are utilized for interaction between blocks in internal block diagram and we utilized command label with action to indicate the interaction between modules in CTMC model. The section of module "boiling unit's translated code is demonstrated in Figure 5.5. In Figure 5.5, there is a variable "regulator4" in module boiling unit which indicates the two states switch-on or switch-off. The switch-off indicates state „0" and switch-on indicates

„1". The "regulator4" transit its state from 0 to 1 when mixing of all liquids have completed. The command label with action [flow to boil] represents the synchronization between modules mixing unit and boiling unit and liquid mixture will start flowing from mixing unit to boiling unit when mixing of liquid will be completed. The variable

"mix=1" represents that it contain the mixture of all three liquids. The clock variable is declared with variable "time1" which represents the time constraint of the action completed in specified time. The time constraints 80.5 and 110.5 represents the continuous time because in continuous time we can define state at any value of time. The continuous time "time1>=80.5" represents that boiling unit take at least 80.5 units of time to boil the liquid and "time1<=110.5" represents that the boiling unit will take maximum

110.5 units of time to complete the action. The last line of code in Figure 5.5 indicates that the regulator1, regulator2 and regulator3 will be switched-off when the regulator of boiling unit is switched-on.

```
module boiling_unit
regulator4:[0..1]; // 0 switch off, 1 switch on
time1:[80..111];
[flowtoboil] regulator4=0  & mix=1&time1>=80.5 & time1<=110.5  -> (regulator4'=1);
[] regulator4=0 & regulator1=0& regulator2=0 & regulator3=0  -> (regulator4'=1);
endmodule
```

Figure 5.5 PRISM code-Liquid Fertilizer Mixing Plant

Table 5.3 Properties- Liquid Fertilizer Mixing Plant

Property No.	**Description**
P1	On detection of specific liquid level through three sensors, the regulators should be turned off automatically
P2	When mixing unit is turned on, then all regulators and boiling unit should be turned off
P3	When boiling unit is turned on, then all regulators and mixing unit should be turned off
P4	At the same time, no two regulators should be turned on
P5	The mixing unit should not be turned on for more than 70.5 units of time
P6	The boiling unit should not be turned on for more than 110.5 units of time

Table 5.3 Properties- Liquid Fertilizer Mixing Plant

For the measurement of some liquid flow from the container1, container2 and container3, we have adjusted threshold limit on regulator of containers. We adjusted threshold=20 for the container1 that regulator1 should be switched-off automatically when the liquid level comes to the threshold of 20. We adjusted threshold=10 for container2 and threshold=5 for container3, when liquid level comes to these limits then regulator2 and regulator3 should switch-off.

Property No.	**CSL properties verified in PRISM**	**Status**
P1	E [F s=1 &regulator1=0|s=2 & regulator2=0|s=3&regulator3=0]	True
P2	E [F regulator1=0 & regulator2=0 & regulator3=0&regulator4=0&mix=1]	True
P3	A [F regulator1=0 &regulator2=0 &regulator3=0&regulator4=1]	True
P4	A [F regulator1=1 & (!(regulator2=1 &regulator3=1&regulator4=1))|regulator2=1&(!(regular tor1=1&regulator3=1&regulator4=1))|regulator3=1&(!(regulator2=1&regulator1=1&regulator4=1))|regulator 4 =1&(!(regulator2=1&regulator3=1&regulator1=1))]	True
P5	A [F mix=1&time<=70.5]	True
P6	A [F regulator4=1&time1<=110.5]	True

Table 5.4 Properties Verified on PRISM- Liquid Fertilizer Mixing Plant

5.3.2 Properties

The six properties are proposed to verify the SysML model of real-time embedded systems. The properties are stated in Table 5.3. We have used operators „A" and „E" to verify properties as shown in Table 5.4. The „A" operator states that property will always hold for all path and „E" operator states that the property will eventually hold for all paths.

In Table 5.4, we have used the „F" operator and it is a special case of operator Until „U" that tells that the property will satisfy only when the condition written along with the „F" operator is true. A witness is also shown with property E [F condition] when the property result is true [47]. A counterexample is shown with property A [G condition] when the property result is false [47].

5.3.3 Experimental Results and Analysis

The CTMC model is verified for both timed and untimed properties given in Table 5.3. In addition, Table 5.4 demonstrates continuous stochastic logic (CSL) code for the aforementioned properties of Table 5.3. The verified status indicates all properties are true. The mixing plant is demonstrating the right behavior as described in requirements. The confirmation of two timed properties proves that the mixing and boiling units strictly obeying the strict time constraint and finished the operation in given time units. Time unit is an important feature in real time embedded systems and if system does not obey the strict time constraint then it will produce wrong results and system will divert from executing its actual functionality.

The outcome indicates that the system is strictly adhere to the continuous timed feature as well as untimed properties and verify the system correctness at early design stage with the assistance of model checking tool. PRISM has the capability to support parallel composition of module, therefore it requires less time in verification compared to other model checkers. The proposed methodology assists the systems engineer in minimizing the risks that can be encountered during development stage by checking the systems at earlier design stage. Such risks would especially be related to real-time embedded systems and can lead to a number of problems such as redeveloping the entire system, product failure, loss of money or life. The re-development of the entire system costs much more than the verification of graphical models of real-time embedded systems at early design stage.

5.4 Hybrid System Case Study of Temperature Control System

Hybrid systems are real time digital and embedded control systems [45], hence we are taking the case study of embedded control systems that is "Temperature control system". In this subsection, the case study of Temperature control system is given to depict discrete and continuous behavior of the system. Initially, we formulate the formal specification of the case study and afterwards create the upgraded SysML block diagram from such specifications. The SysML model is translated into the input language of model checker „PRISM" and user requirements are converted to CSL properties for verification. At last, the experimental results of this case study are discussed.

Conclusion

With the growth in technology and ever growing need of software intensive applications, the real time and embedded system development require some serious attention for providing error free and high quality application. The early design verification and modeling is actually a tough job particularly for industrial and large scale application development. The methodology proposed in this paper allows formal modeling and verification of the system early design. We had four case studies and checked it. We simulate the system behavior of two case studies that comprised discrete and continuous time using SysML internal block diagram individually and checked it. We then have two case studies of hybrid systems, specify the system functionality formally, and represent the system components by enhanced SysML block diagram. Formal specification specifies how the components communicate with each other using communication channels and it also specifies the passage from one state to another state. Formal specification assists in supplying the semantics for every component supporting rich support to map to model checker language. Graphical model is mapped into PRISM model and user specifications are mapped into property specification language. The PRISM displays the verification results compared to the developed PRISM model. PRISM indicates the results as „True" when the property is satisfied and indicates the results as

„False“ when the property is not satisfied. Counterexample is generated too if the property does not hold and it helps us to rectify mistakes in our system model.

The graphical model of discrete and continuous timed based real time embedded systems i.e. SysML block diagram is validated using model checking tool. These SysML based graphical models enable us to verify the system correctness at the initial design stage. Additionally, we have taken into account two case studies of hybrid systems, given the formal specifications using the aid of hybrid automata, create SysML block diagram and then validated. Since hybrid automata are well-suited to the modeling of hybrid systems hence we have analyzed the two examples of hybrid systems that show discrete and continuous behavior. The initial design verification lowers the efforts and time of the system engineers and it assists to create a secure application. The suggested approach delivers substantial and precise outcomes and motivates the system and software engineers to apply our approach to formal modeling and verification of real time and embedded systems. The system and software engineers can apply our suggested approach to early design graphical modeling, specification and verification of real time embedded systems.

References

Myers, G. J., Sandler, C., & Badgett, T. (2011). The art of software testing. John Wiley & Sons.

Norman, G., & Parker, D. (2014). Quantitative Verification: Formal Guarantees for Timeliness, Reliability and Performance. London Mathematical Society and Smith Institute.

Jéron, T., Veanes, M., & Wolf, B. (January 2013). Symbolic Methods in Testing. Saarbrücken/Wadern, Germany: Dagstuhl Publishing.

Baier, C., & Katoen, J.-P. (2008). Principles of Model Checking (26202649 ed.). Cambridge: MIT press.

Clarke, E. M., Grumberg, O., & Peled, D. (1999). Model Checking. MIT press.

Samat, P. A., & Zin, A. M. (2012). Common Modeling Language for Model Checkers. Journal of Computer Science , 8 (1): 99-106.

Mazzini, S., Puri, S., Mari, F., Melatti, I., & Tronci, E. (2009). Formal Verification at System Level. DAta Systems In Aerospace (DASIA), Org. EuroSpace, Canadian Space Agency, CNES, ESA, EUMETSAT. Instanbul, Turkey.

Clarke, E., Grumberg, O., Jha, S., Lu, Y., & Veith, H. (2000). Counterexample-Guided Abstraction Refinement. In Computer Aided Verification (pp. 154-169). Springer Berlin Heidelberg.

Clarke, E., Fehnker, A., Jha, S. K., & Veith, H. (2005). Temporal Logic Model Checking. In Handbook of Networked and Embedded Control Systems (pp. 539-558). Birkhäuser Boston.

Ciesinski, F., & Größer, M. (2004). On Probabilistic Computation Tree Logic. In Validation of Stochastic Systems (pp. 147-188). Springer Berlin Heidelberg.

M. Waqas, Z. Khan, S. U. Ahmed and A. Raza, "MIL-Mixer: A Robust Bag Encoding Strategy for Multiple Instance Learning (MIL) using MLP-Mixer," 2023 18th International Conference on Emerging Technologies (ICET), Peshawar, Pakistan, 2023, pp. 22-26.

Gao, Y., Xu, M., Zhan, N., & Zhang, L. (2013). Model checking conditional CSL for continuous-time Markov chains. Information Processing Letters , Volume 113, Issues 1–2, 44– 50.

HUSSAIN, S., Raza, A., MEERAN, M. T., IJAZ, H. M., & JAMALI, S. (2020). Domain Ontology Based Similarity and Analysis in Higher Education. IEEEP New Horizons Journal, 102(1), 11-16.

Axelsson, R., Hague, M., Kreutzer, S., Lange, M., & Latte, M. (2010). Extended Computation Tree Logic. In Logic for Programming, Artificial Intelligence, and Reasoning (pp. 67-81). Springer Berlin Heidelberg.

Raza, A., Salahuddin, & Inzamam Shahzad. (2024). Residual Learning Model-Based Classification of COVID-19 Using Chest Radiographs. Spectrum of Engineering Sciences, 2(3), 367–396

Mahmood, F., Abbas, K., Raza, A., Khan,M.A., & Khan, P.W. (2019). Three Dimensional Agricultural Land Modeling using Unmanned Aerial System (UAS). International Journal of Advanced Computer Science and Applications (IJACSA) [p-ISSN : 2158-107X, e-ISSN : 2156-5570], 10(1).

Khan, U. S., & Khan, S. U. R. (2024). Boost diagnostic performance in retinal disease classification utilizing deep ensemble classifiers based on OCT. Multimedia Tools and Applications, 1-21.

Khan, M. A., Khan, S. U. R., Haider, S. Z. Q., Khan, S. A., & Bilal, O. (2024). Evolving knowledge representation learning with the dynamic asymmetric embedding model. Evolving Systems, 1-16.

Raza, A., & Meeran, M. T. (2019). Routine of encryption in cognitive radio network. Mehran University Research Journal of Engineering & Technology, 38(3), 609-618.

Al-Khasawneh, M. A., Raza, A., Khan, S. U. R., & Khan, Z. (2024). Stock Market Trend Prediction Using Deep Learning Approach. Computational Economics, 1-32.

Khan, U. S., Ishfaque, M., Khan, S. U. R., Xu, F., Chen, L., & Lei, Y. (2024). Comparative analysis of twelve transfer learning models for the prediction and crack detection in concrete dams, based on borehole images. Frontiers of Structural and Civil Engineering, 1-17.

Khan, S. U. R., & Asif, S. (2024). Oral cancer detection using feature-level fusion and novel self-attention mechanisms. Biomedical Signal Processing and Control, 95, 106437.

Raza, A.; Meeran, M.T.; Bilhaj, U. Enhancing Breast Cancer Detection through Thermal Imaging and Customized 2D CNN Classifiers. VFAST Trans. Softw. Eng. 2023, 11, 80–92.

Dai, Q., Ishfaque, M., Khan, S. U. R., Luo, Y. L., Lei, Y., Zhang, B., & Zhou, W. (2024). Image classification for sub-surface crack identification in concrete dam based on borehole CCTV images using deep dense hybrid model. Stochastic Environmental Research and Risk Assessment, 1-18.

Khan, S.U.R.; Asif, S.; Bilal, O.; Ali, S. Deep hybrid model for Mpox disease diagnosis from skin lesion images. Int. J. Imaging Syst. Technol. 2024, 34, e23044.

Khan, S.U.R.; Zhao, M.; Asif, S.; Chen, X.; Zhu, Y. GLNET: Global–local CNN's-based informed model for detection of breast cancer categories from histopathological slides. J. Supercomput. 2023, 80, 7316–7348.

Khan, S.U.R.; Zhao, M.; Asif, S.; Chen, X. Hybrid-NET: A fusion of DenseNet169 and advanced machine learning classifiers for enhanced brain tumor diagnosis. Int. J. Imaging Syst. Technol. 2024, 34, e22975.

Khan, S.U.R.; Raza, A.;Waqas, M.; Zia, M.A.R. Efficient and Accurate Image Classification Via Spatial Pyramid Matching and SURF Sparse Coding. Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol. 2023, 7, 10–23.

Farooq, M.U.; Beg, M.O. Bigdata analysis of stack overflow for energy consumption of android framework. In Proceedings of the 2019 International Conference on Innovative Computing (ICIC), Lahore, Pakistan, 1–2 November 2019; pp. 1–9.

Shahzad, I., Khan, S. U. R., Waseem, A., Abideen, Z. U., & Liu, J. (2024). Enhancing ASD classification through hybrid attention-based learning of facial features. Signal, Image and Video Processing, 1-14.

Khan, S. R., Raza, A., Shahzad, I., & Ijaz, H. M. (2024). Deep transfer CNNs models performance evaluation using unbalanced histopathological breast cancer dataset. Lahore Garrison University Research Journal of Computer Science and Information Technology, 8(1).

Bilal, Omair, Asif Raza, and Ghazanfar Ali. "A Contemporary Secure Microservices Discovery Architecture with Service Tags for Smart City Infrastructures." VFAST Transactions on Software Engineering 12, no. 1 (2024): 79-92.

Bilal, O., Asif, S., Zhao, M., Khan, S. U. R., & Li, Y. (2025). An amalgamation of deep neural networks optimized with Salp swarm algorithm for cervical cancer detection. Computers and Electrical Engineering, 123, 110106.

Khan, S. U. R., Asif, S., Zhao, M., Zou, W., Li, Y., & Li, X. (2025). Optimized deep learning model for comprehensive medical image analysis across multiple modalities. Neurocomputing, 619, 129182.

Khan, S. U. R., Asif, S., Zhao, M., Zou, W., & Li, Y. (2025). Optimize brain tumor multiclass classification with manta ray foraging and improved residual block techniques. Multimedia Systems, 31(1), 1-27.

M. Wajid, M. K. Abid, A. Asif Raza, M. Haroon, and A. Q. Mudasar, “Flood Prediction System Using IOT & Artificial Neural Network”, VFAST trans. Softw. Eng., vol. 12, no. 1, pp. 210–224, Mar. 2024.

Waqas, M., Ahmed, S. U., Tahir, M. A., Wu, J., & Qureshi, R. (2024). Exploring Multiple Instance Learning (MIL): A brief survey. Expert Systems with Applications, 123893.

Ouchani, S., Mohamed, O. A., & Debbabi, M. (2013, September). A probabilistic verification framework of SysML activity diagrams. Intelligent Software Methodologies, Tools and Techniques (SoMeT), 2013 IEEE 12th International Conference on (pp. 165 - 170). IEEE.

Waqas, M., Tahir, M. A., Al-Maadeed, S., Bouridane, A., & Wu, J. (2024). Simultaneous instance pooling and bag representation selection approach for multiple-instance learning (MIL) using vision transformer. Neural Computing and Applications, 36(12), 6659-6680..

Khan, Z., Hossain, M. Z., Mayumu, N., Yasmin, F., & Aziz, Y. (2024, November). Boosting the Prediction of Brain Tumor Using Two Stage BiGait Architecture. In 2024 International Conference on Digital Image Computing: Techniques and Applications (DICTA) (pp. 411-418). IEEE.

Khan, S. U. R., Raza, A., Shahzad, I., & Ali, G. (2024). Enhancing concrete and pavement crack prediction through hierarchical feature integration with VGG16 and triple classifier ensemble. In 2024 Horizons of Information Technology and Engineering (HITE)(pp. 1-6). IEEE https://doi. org/10.1109/HITE63532.

Khan, S.U.R., Zhao, M. & Li, Y. Detection of MRI brain tumor using residual skip block based modified Mobile Net model. Cluster Comput 28, 248 (2025). https://doi.org/10.1007/s10586-024-04940-3

Waqas, M., Tahir, M. A., & Qureshi, R. (2023). Deep Gaussian mixture model based instance relevance estimation for multiple instance learning applications. Applied intelligence, 53(9), 10310-10325.

Raza, A., Soomro, M. H., Shahzad, I., & Batool, S. (2024). Abstractive Text Summarization for Urdu Language. Journal of Computing & Biomedical Informatics, 7(02).

Ouchani, S., Mohamed, O. A., & Debbabi, M. (2014). A property-based abstraction framework for SysML activity diagrams. Knowledge-Based Systems , 56, 328–343.

Waqas, M., Tahir, M. A., & Khan, S. A. (2023). Robust bag classification approach for multi-instance learning via subspace fuzzy clustering. Expert Systems with Applications, 214, 119113..

Ouchani, S., Mohamed, O. A., & Debbabi, M. (2013, June). A Security Risk Assessment Framework for SysML Activity Diagrams. Software Security and Reliability (SERE), 2013 IEEE 7th International Conference on (pp. 227 - 236). IEEE.

Ashraf, M., Jalil, A., Salahuddin & Jamil, F. (2024). DESIGN AND IMPLEMENTATION OF ERROR ISOLATION IN TECHNO METER. Kashf Journal of Multidisciplinary Research, 1(12), 49-66.

Meeran, M. T., Raza, A., & Din, M. (2018). Advancement in GSM Network to Access Cloud Services. Pakistan Journal of Engineering, Technology & Science [ISSN: 2224-2333], 7(1).

Ouchani, S., Mohamed, O. A., & Debbabi, M. (2012). Efficient Probabilistic Abstraction for SysML Activity Diagrams. In Software Engineering and Formal Methods (pp. 263-277). Springer Berlin Heidelberg.

Soomro, M. H., Salahuddin, Irtaza, G., Ali, G., & Batool, S. (2024). USE IMAGE PROCESSING MODEL TO FRUIT QUALITY DETECTION. Kashf Journal of Multidisciplinary Research, 1(11), 85-106..

Waqas, M., & Khan, M. A. (2018). JSOPT: A framework for optimization of JavaScript on web browsers. Mehran University Research Journal of Engineering & Technology, 37(1), 95-104.

Ouchani, S., Jarraya, Y., & Mohamed, O. A. (2011, July). Model-based systems security quantification. Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on (pp. 142 - 149). IEEE.

Salahuddin, Hussain, M., & hamza Shafique, P. (2024). PERFORMANCE ANALYSIS OF MATCHED FILTER-BASED SECONDARY USER DETECTION IN COGNITIVE RADIO NETWORKS. Kashf Journal of Multidisciplinary Research, 1(10), 15-26.

Jarraya, Y., Soeanu, A., Debbabi, M., & Hassaine, F. (2007, March). Automatic Verification and Performance Analysis of Time-Constrained SysML Activity Diagrams. Engineering of Computer-Based Systems, 2007. ECBS '07. 14th Annual IEEE International Conference and Workshops on the (pp. 515 - 522). IEEE.

Jarraya, Y., Debbabi, M., & Bentahar, J. (2009, April). On the Meaning of SysML Activity Diagrams. Engineering of Computer Based Systems, 2009. ECBS 2009. 16th Annual IEEE International Conference and Workshop on the (pp. 95 - 105). IEEE.

Syed Shahid Abbas, Salahuddin, Abdul Manan Razzaq, Mubashar Hussain, Meiraj Aslam, Prince Hamza Shafique, & Muhammad Asif Nadeem. (2024). Optimized AI-Driven Intrusion Detection in WSNs: A Semi-Supervised Learning Paradigm. Journal of Computing & Biomedical Informatics.

Jarraya, Y., & Debbabi, M. (2012, July). Formal Specification and Probabilistic Verification of SysML Activity Diagrams. Theoretical Aspects of Software Engineering (TASE), 2012 Sixth International Symposium on (pp. 17 - 24). IEEE.

Debbabi, M., Hassaïne, F., Jarraya, Y., Soeanu, A., & Alawneh, L. (2010). Probabilistic Model Checking of SysML Activity Diagrams. In Verification and Validation in Systems Engineering (pp. 153-166). Springer Berlin Heidelberg.

Salahuddin, Abdul Manan Razzaq, Syed Shahid Abbas, Mohsin Ikhlaq, Prince Hamza Shafique, & Inzimam Shahzad. (2024). Development of OWL Structure for Recommending Database Management Systems (DBMS). Journal of Computing & Biomedical Informatics, 7(02).

Guimaraes, F. P., Célestin, P., Batista, D. M., Rodrigues, G. N., & de Melo, A. C. (2013). A Framework for Adaptive Fault-Tolerant Execution of Workflows in the Grid: Empirical and Theoretical Analysis. Journal of Grid Computing , 12(1), 127-151.