Hindawi
Scientific Programming
Volume 2021, Article ID 6685978, 17 pages
https://doi.org/10.1155/2021/6685978

Research Article

Formal Modeling, Proving, and Model Checking of a Flood Warning, Monitoring, and Rescue System-of-Systems

Abdul Rehman,[1] **Nadeem Akhtar**,[2] **and Omar H. Alhazmi**[3]

[1]*Department of Computer Science and IT, Virtual University of Pakistan, Lahore, 54000, Pakistan*
[2]*Department of Computer Science and IT, The Islamia University of Bahawalpur, Punjab 63100, Pakistan*
[3]*Department of Computer Science, Taibah University, Medina 30001, Saudi Arabia*

Correspondence should be addressed to Nadeem Akhtar; nadeem.akhtar@iub.edu.pk

Received 24 December 2020; Revised 21 March 2021; Accepted 29 March 2021; Published 15 April 2021

Academic Editor: Tomàs Margalef

Floods after monsoon rains are frequent disasters that affect millions of lives in Pakistan. Human lives are lost, agriculture economies are destroyed, and livestock animals, houses, fruit farms, and crops are lost which are the major livelihoods of thousands of people in Punjab. Each year there are heavy rains in the monsoon season and, due to global warming, there is the rapid melting of snow in northern glaciers; these factors subsequently cause floods. There is also loss of life due to the spread of waterborne diseases and snake bites. Flood monitoring provides early detection of a flood and the calculation of its intensity, which results in reduced human life losses and economic losses. Most casualties are caused by the lack of timely real-time, authentic information about the high-risk areas, and flood intensity, speed, and direction. Therefore, the proposed approach is centered on formal modeling and verification of safety and liveness properties of flood monitoring perceivers. Each flood perceiver has several sensors. It requires the collection of information starting from the flood perceiver, observer, and environmental forecast. This information is processed to determine the flood intensity level. We have developed a CP-Nets' formal model and model-checked it. We have verified the safety and liveness properties of correctness by exhaustive verification of the system using model-based proof obligations (Event-B method using Rodin). Our objective in this research is to propose a correct, reliable, and efficient flood warning, monitoring, and rescue (WMR) SoS based on formal methods. We have used formal modeling and model-checking based on state-of-the-art hierarchical CP-Nets supported by exhaustive formal proof obligations of Event-B.

1. Introduction

River floods are a dangerous hydrological phenomenon that affects thousands of people each year around the globe. It is a complex task to anticipate the flood and prepare the evacuation plan. Sometimes flash floods develop instantly. As a result of these floods, human lives are lost, infrastructure is destroyed, crops and animals are lost, and people are left homeless to fight waterborne diseases.

In Pakistan, a flood in 2010 affected more than 20 million people. About two thousand people were reported dead. People were internally displaced in the country; properties, agricultural land, and crops were ruined [1]. River floods are a concern for a developing country such as Pakistan. Rivers, irrigation canals, and lakes are flooded after the rainy monsoon season every year. Sometimes, the water from these rivers and canals enter human inhabitant and destroy due to high flooding. They affect millions of people's lives and destroy houses, schools, bridges, livestock, animals, and crops. They cause waterborne diseases, health disorders, and loss of clean drinking water.

In Pakistan, with the rapid population growth, more and more people are living near river banks. Global warming has increased monsoon rains, and the rapid melting of glaciers in the Himalayas and Karakoram mountain ranges results in large quantities of water in rivers. Each year the reappearance of this disaster is making life tough for rural as well as urban inhabitants. Illegal construction by humans in river flood zones also aggravates the situation and causes huge losses in floods. For this purpose, the government and

private departments need planned urban construction, planned deforestation, and properly informed decisions to keep a check on illegal deforestation and illegal construction in flood zones.

The economic damages and destruction result in difficulty to recuperate from these losses. Flood damages can be avoided or substantially reduced by early flood detection, and flood perceivers are vital for early flood detection to stop human life losses and economic losses (Figures 1–3).

These flood perceivers are safety-critical systems and need to be correct as human lives are dependent on them. We have proposed a formal approach for flood monitoring, flood avoidance, and after-flood rescue services. This approach is cantered on formal modeling and verification for early flood detection and measurement of the flood intensity by using flood perceivers.

Pakistan is an agriculture-dependent country; there are five major rivers and hundreds of canals originating from these rivers. Without water, there is no agricultural economy. The role that water plays in agriculture is fundamental. Without an effective river and canal system, there is no agriculture. Historical facts and figures have shown that Pakistan is on the list of countries that face severe floods. Millions of people's lives are affected due to poor flood intimation, lack of prior information, and poor rescue services. Over the last 37 years, Pakistan faced 17 floods of different intensities after the megaflood in 1929 [2]. The floods in 2010 affected millions of people in Pakistan. An effective prevention strategy can minimize the destruction from floods. We have collected large amounts of flood data from flood perceiver sensors. We have performed advanced predictions and forecasts to calculate the probability of the flood using this flood-data.

A system-of-system (SoS) is the collaboration between existing systems to achieve some goals. It integrates independently useful systems into a system that delivers emergent behavior and new unique functions to users, emerging from the individual systems' combination. A flood-WMR SoS is a safety-critical system; applying formal methods and technology to this field is an emerging topic in industry and academia. In a flood-WMR SoS, failure can be potentially disastrous and correctness is of fundamental importance. Safety property is of particular importance in a safety-critical computer system [3] such as a flood-WMR SoS. In safety standards such as IEC61508 [4] and EN50128 [5], formal methods are vital for software requirements' specification and software design.

A correct and reliable flood-WMR SoS is vital, and if the system cannot warn people on time, then it is of no use. We have analyzed, designed, modeled, and verified a flood-WMR SoS to monitor rivers, canals, and dams before, during, and after floods and rescue and emergency services in a post-flood disaster situation. A model of a flood-WMR SoS is designed, model-checked, and verified by mathematical proofs. We have designed the system as system-of-systems, i.e., a composition of autonomous systems working together to attain a specific goal. We have used Coloured Petri-Nets (CP-Nets) [6–13] formalism to construct model and model-checking, as CP-Nets are exhaustive and human-understandable, have graphical syntax, and allow data flow in the form of tokens.

Metalanguage (ML) of CP-Nets, a functional language, is used to construct the model's data attributes and functional attributes. Afterward, the CP-Nets specifications are transformed into Event-B [14, 15] specifications for formal proofs. The correctness properties of safety and liveness are specified and formally verified. The models are simulated on the computer to provide that appropriate environment and check that it is working correctly according to the desired behavior. Without exhaustive models and generating proofs of safety and liveness, a correct, reliable system cannot be designed and developed. The safety and liveness properties are specified, model-checked using CP-Nets, and proof-checked using Event-B. We have used the Rodin [16, 17] platform for Event-B formal proofs.

The flood-WMR SoS must be correct to deliver accurate and precise alerts. Section 2 presents the objectives. Section 3 presents the motivation and problem statement. Section 4 describes the SoS. Section 5 presents the transformation from Event-B formal proofs to CP-Nets-based model checking. Section 6 presents the literature review. Section 7 presents the state of the art. Section 8 presents the material and methods. Section 9 presents the flood monitoring perceivers. Section 10 presents the flood-WMR SoS CP-Nets modeling and model-checking. Section 11 presents the flood-WMR SoS Event-B proofs. Section 12 presents the contributions. Section-13 puts forth the discussion and conclusion.

2. Objectives

The principal objective of a resilient flood-WMR SoS is to specify, design, and implement a correct model that ensures the properties of correctness (i.e., safety and liveness), reliability, and performance (i.e., efficiency). We have checked the correctness and reliability by CP-Nets formal modeling, CP-Nets model-checking (i.e., exhaustive state-space evaluation), and Event-B-based formal proofs. Formal timed CP-Nets ensure the performance attributes.

A correct and resilient model of SoS has the following pillars:

(1) *Formal Foundation*. Since safety-critical flood-WMR SoS has several mandatory requirements, a sound formal foundation is a prerequisite for the engineering approach.

(2) *Executable Language for Specification*. Designing and realizing large-scale critical SoS require suitable models at appropriate abstraction levels.

(3) *Run-Time Execution Platform*. Automatic adaptation and evolution of a critical system require a suitable run-time execution platform.

(4) *Open System*. As flood-WMR SoS is a large-scale critical system, it is the a long-lived system; therefore, it has to be prepared for openness. In an open environment, the context of the system can change at any time, availability of resources may change, services may evolve, services may disappear, or new services may become available.

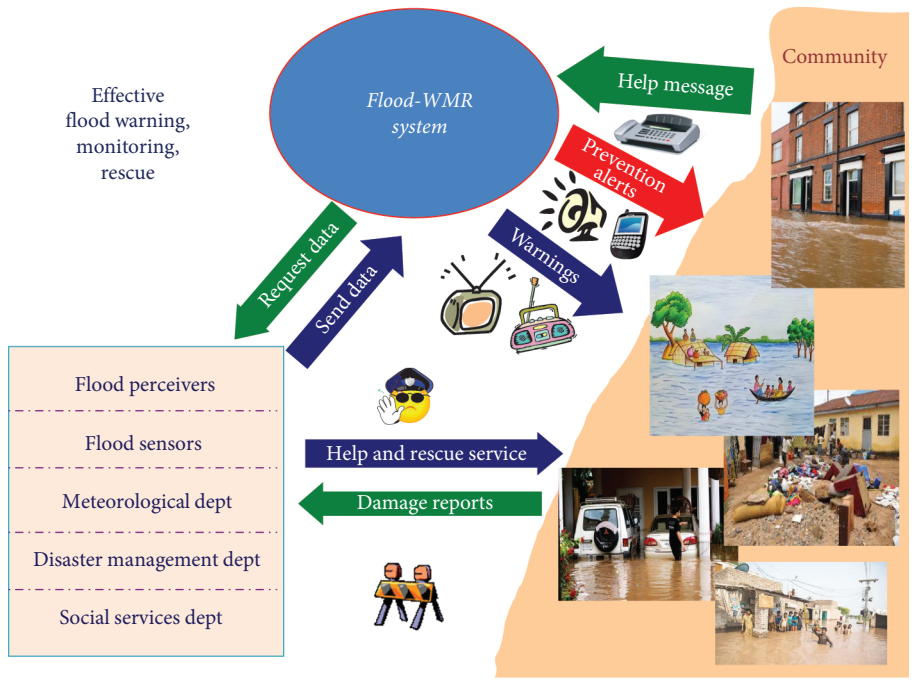

Figure 1: Stakeholders of the flood warning, monitoring, and rescue system-of-systems (flood-WMR SoS).

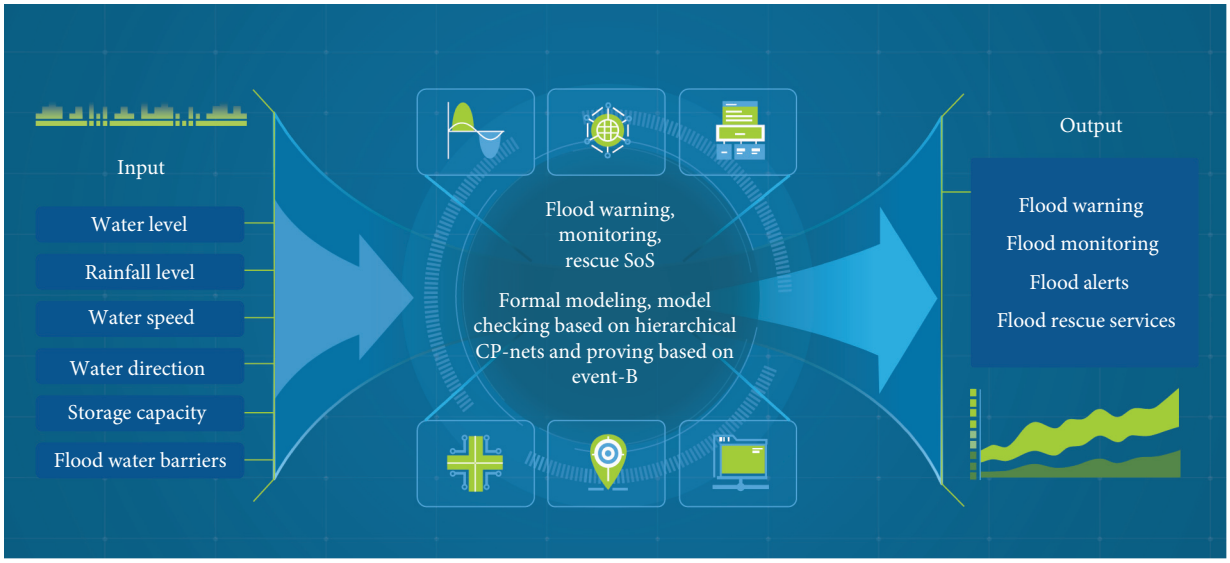

Figure 2: Flood warning monitoring and rescue system-of-systems (Flood-WMR SoS).

3. Motivation and Problem Statement

Flood-WMR SoS is a safety-critical software-intensive system. The correctness of flood-WMR SoS is important as errors in this system can cause human life loss and high economic loss. Therefore, human lives are dependent on the correctness of the flood-WMR SoS. This flood-WMR SoS has to ensure correctness properties of safety and liveness. As flood-WMR SoS has to operate in open environments in which it interacts and collaborates with other systems, concurrent processes are working in each system and there is communication between different processes. Ensuring correctness of the SoS through automated testing methods is not sufficient; therefore, formal modeling and verification provides exhaustive verification.

In our work, specification, modeling, verification, and validation of a safety-critical flood-WMR SoS depend on ensuring correctness at every stage. This goal is realized by using industrial strength formal methods and techniques of CP-Nets and Event-B.

The problem statement proposes a flood warning, monitoring, and rescue system built by multiple autonomous systems formally modeled and verified. Flood-

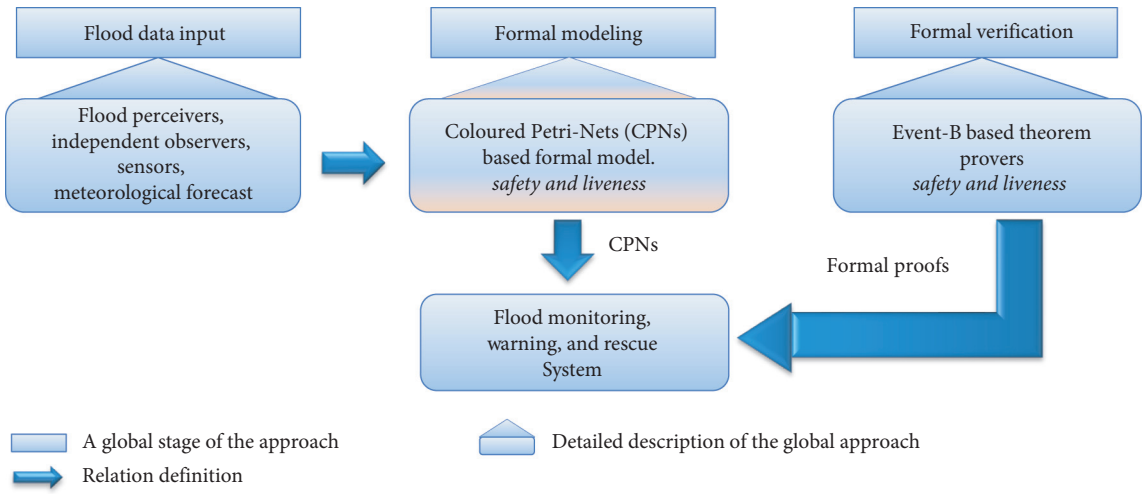

Figure 3: Proposed approach.

WMR SoS is correct and reliable, monitors floods, provides flood avoidance services, and facilitates rescue services.

4. System-of-Systems (SOS)

A system is composed of elements that work together according to some rules and regulations and produce results that cannot be obtained by an individual element. An individual element of a system can itself be complex and large and comprises subelements working together. Engineering SoS and guaranteeing their correctness, reliability, rigor, formality, and performance are complex.

The SoS was first applied to the American department of defense analysis and military systems' design [18]. "SoS principle objective is the collaboration among existing systems to reach goals. System-of-systems (SoS) integrates independently useful systems into a larger system, delivering new unique functions to users that emerge from the combination of the individual parts. Examples are intelligent goods transport robots, intelligent traffic systems, integrated surveillance systems, smart cities, and networked smart homes" [19, 20].

Maier [19], in 1998, identified three basic SoS types (virtual, collaborative, and directed), and five characteristics were identified as "operational independence, managerial independence, evolutionary development, emergent behavior, and geographic distribution" of SoS. Based on this characterization, Maier identifies a set of guiding design principles for SoS:

(1) *Stable Intermediate Forms*. The subsystems of SoS should be capable of working and fulfilling useful objectives, during operation and before full deployment

(2) *Policy Triage*. SoS design team should carefully make the policy to choose what to control, i.e., avoid under-control and over-control

(3) *Leverage at the Interfaces*. Its interfaces define the architecture of SoS

(4) *Ensuring Collaboration*. Mechanisms should be developed to create joint utility

The intrinsic characteristics of SoS are

(1) Operational independence of systems

(2) Managerial independence of systems

(3) Geographical distribution of systems

(4) Evolutionary development of SoS

(5) Emergent behavior of SoS

SoS creates emergent behavior, i.e., performs functions that cannot be performed by any of the constituent systems alone. Constituent systems fulfil their tasks and goals.

Boardman and Sauser [21] proposed "the nature of the composition to define the distinguishing characteristics of SoS as autonomy, diversity, connectivity, and emergence." Northrop et al. [22] put forth complexity and scale as the fundamental traits of ultralarge-scale systems, and the slogan "scale changes everything" is used". According to [23], "SoS is a combined arrangement of geographically distributed and managerially independent elements put together to provide a functionality that is not possible otherwise." DeLaurentis and Crossley [24] proposed that the flood monitoring systems, disaster management systems, health care systems, energy production systems, transport systems, and logistic systems can be designed and developed based on the SoS concept.

5. Transformation from CP-Nets-Based Model to Event-B Formal Proofs

The Event-B method and CP-Nets are two industry-recognized tools. Both formal methods are used extensively in many industrial projects in the domain of transportation and aeronautics. CP-Nets has a graphics language which has the advantage of easy understanding formalism. Therefore, many practical industry systems are specified and modeled by using high-level CP-Nets. We have performed a manual transformation from Event-B's context and machines into

the models of the CP-Nets. We have performed a technical review of the specification constructs of Event-B and CP-Nets, to understand the problems and limitations of these methods.

Event-B has a foundation on set theory, first-order predicate logic, formal proofs, proof obligations, variables, invariant, guards, and events. The Event-B method has been studied for urban railway systems and applied in some components such as automatic pilot system for the light driver-less shuttle for the airport and the subway-train platform screen door controlling system. The safety, liveness, and robustness of the Event-B developed systems have convinced people of its reliability because the final implementation code is generated from abstract. The Event-B machine is considered and proved to be safe. Event-B-proved models are accepted as strong safety proof [25, 26].

On the contrary, the metalanguage of CP-Nets is a functional language based on temporal logic. For large-scale and complex systems, the high-level CP-Nets could provide a formal, concise model. CP-Nets model checking is used here as an auxiliary method of verification. The translation approach consists of defining Event-B machines' transformation rules, guards, invariant, and events and representing these constructs in CP-Nets.

We have performed a transformation from CP-Nets models into Event-B models and integrated the behavioral specification and dynamic properties. Therefore, this paper describes the framework of a novel transformation process, based on a more in-depth state-space analysis of the functioning and the handling of CP-Nets with particular emphasis on their component tasks.

The CP-Nets assessment processes encounter the "state explosion" problem. Furthermore, there is a big gap between Event-B models and the final implementation because there are rarely any commercial tools to assess this process formally. In contrast, the Event-B method is well suited for formal assessment in the development of implementable codes. However, Event-B language demands a lot of pretraining for set theory and first-order logic. Besides, its notations and its commercial tools are not user friendly. So, it is only suitable to be used in vital safety-critical applications.

In the design phase, a model has been validated by industry experts, using dedicated industrial tools such as CP-Nets. The problem is how to prove it formally for the specifications and automatically generate the implementable codes. To be more precise, the problem to be solved is the translation from the CP-Net formalism into the abstract Event-B machine formalism. The motivation for such a transformation from the engineering perspective is that two formal methods are complementary. We have formal inductive proofs (proof obligations) and exhaustive state-space analysis (model checking).

It leads us to take advantage of both methods using the techniques of model-driven engineering. A solution may be a model transformation from CP-Nets models to the abstract Event-B machines and then the refinement of these Event-B abstract machines into concrete machines and code. Such a transformation can build a bridge between critical tasks, from a stable requirement analysis towards implementing an entire system.

The set of token colors in CP-nets are specified by ML programming language. These color sets are classified into two categories. The basic types which are derived from standard ML are simple color sets, and the others are compound color set using previously declared color sets. In this work, we have not considered the time-related colors (i.e., timed CP-Nets). The simple color sets and their corresponding data types in Event-B language are shown in Table 1. Similar data types for Integer, Boolean, and Enumerated are also there in the abstract machine notation.

The proposed approach to design, model, and verify flood-WMR SoS also leads to the definition of transformation rules from CP-Nets into Event-B specifications. Moreover, this new approach intends to provide an interactive software platform for automatic transformation from the CP-Nets model to Event-B machines.

The unit color sets can be represented with a set containing only one element, such as unit = {new unit}. The index color set is a concrete instance of a positive integer. It could be denoted by a set of finite positive integers index = {int-exp1..int-exp2}, where index is NAT. The string color sets is the set of all text strings, while in the abstract machine notation, the concrete type of "string" can only be used in the operation input parameters. In practical translation, we recommend using a finite enumerate set to indicate all the necessary strings. For the compound color sets, CP-nets use constructors to define more complex color sets.

The product color sets colset Name = product $name_1 * name_2 * ... * name_n$, where $n \geq 2$. In Event-B notation, there is the same Cartesian product '$*$' and definition will be Name: $= name_1 * name_2 * ... * name_n$. The record color set colset Name = record id_1: $name_1 * id_2$: $name_2 * ... * id_n$: namen. It is a fixed-length color set and has the same function as product. The only difference is that a product color is position-dependent, while in record color, each component has a unique label to be identified.

In B language, the record concept is the same, and the notations are as follows. The set of records is SET = struct(id_1: $name_1$,..., id_n: $name_n$) and an extensive record is REC = rec(id_1: $name_1$, ..., id_n: $name_n$), where n is an integer greater than or equal to 1. The access to a record field (quote operator) is idi'REC.

The List color set is a variable-length data type. The values are a sequence whose color must be the same type, colset name = list name0. As one of the most flexible structures in CP-Nets, the list structure has many predefined operations that can achieve functions such as inquiry, self-loop, and recursion. However, this concrete programming language is only accepted in the implementation phase of Event-B method. So, the mapping of the lists from CP-Net to Event-B machine is conditional, and the most suitable candidate is "Sequence expressions." Table 2 is a partial mapping of two data structures. The other list functions may only be realized inside the operations. Other predefined list functions are more programming like functions. It is not possible to write these functions in a compact form with set theory and first-order logic. In practice, depending on the

Table 1: Comparison of basic data types.

Color set	CPN specification	Event-B notation
Unit	Comprises a single element colset name = unit	No direct corresponding
Boolean	Set of true and false colset name = bool	BOOL
Integer	Integer numerals without a decimal point colset name = int	INTEGER
String	Sequences of printable characters colset name = string	STRING
Enumerated	Enumeration values colset name = with id0 | id1 | · · · | idn	SET = { E1, · · ·, En} only in operation input
Index	Sequences of values comprised of an identifier colset name = index id with exp1, .., exp2	No direct corresponding

Table 2: Mapping of List data types.

CPN specification	B Notation
ins new 1 x if x is not a number of list 1, then x is inserted at the end of 1, otherwise 1 is returned	ann <– ins new(ll,xx) = = if xx: ran(ll) then aa: = ll else aa: = ll < -xx end
nil or [] empty list	[]
hd lst head, the first element of the list	first(lst)
tl lst tail, the last element of the list	tail(lst)
Length lst length of list	size(lst)
elt:: List prepend element as head of list	elt - > lst
Rev lst reverse list	rev(lst)
Rev lst reverse list	lst(nth)
list.take(lst,n) returns first n elements of list	Lst/|\n
list.drop(lst,n) returns what is left after dropping first n elements of list	lst|/n
list.null lst returns true if element is empty	(lst = [])
mem.null lst returns true if element is in the list	elt:ran(lst)
Contains lst1 lst2 returns true if all elements in list 2 are elements in list 1 (ignoring the multiplicity of elements in list 2)	ran(lst1)<:ran(lst2)
union lst1 lst2 (or lst1^lst2 the concatenation of two list	lst1^lst2
Ins lst elt inserts element at the end of list	lst < -elt

context, list functions may appear in different forms in the operations. Sometimes, a function can only be developed in the implementation phase, where recursion and loop are allowed in Event-B machines. Here is an example of achieving the same function inside the operation.

Besides the declaration of color sets, the CPN tools [27] also allow constant declaration and a user-defined function declaration. A constant declaration binds a value to an identifier, which then works as a constant. The developer defines a constant in the Event-B constant part in context. The functions of an Event-B machine are used in operations. They can be accomplished by substitution, similar to a programming language, such as IF condition and case condition. Some simple parameterized functions may appear in the definition as reusable modules.

6. Literature Review

In [27], a web-based distribution platform that carries together important information on river floods is presented. The method contains various types of flood associated information such as water level, rainfall, water flow, and water direction collected through geo-sensor networks.

The platform acts as an information channel among authorities and experts to manage and coordinate teamwork and communication and as a web-based foundation of information for the community, sustaining their need of being timely informed about water level and flood situations. Ribeiro et al. [27] proposed a system consisting of three major components: processing unit, sensor network, and database of applications. The water level is monitored using wireless sensors that use mobile general package radio service communication to broadcast the flood water information towards a web-based server.

A system was developed in [28] called ArRoad, which analyzes and monitors rush-hour traffic and water-logged regions via sensors and image processing, which uses machine learning techniques to forecast traffic congestion and substitute redirecting traffic routes. Water sensor nodes are located to monitor and analyze the flooded locations, whereas real-time video-camera data detect the traffic capacity on the roads. Among natural disasters, floods are the primary cause of deaths around the world [2]. Pakistan has many flood-affected areas. Weather factors include heavy rainfall and increased snow-melting rate of northern glaciers due to global warming.

Alipio et al. [29] proposed a method for data collection of periodic flood updates from remote areas by people using the Internet and smartphones. This allows the population to become aware of river water changes: hydrological station or satellite image sensing. Cloud computing is used for better access and storage of flood information. Citizens (i.e., local population) are able to update their area flood events, get

flood information from other neighboring locations, and become aware of the floods.

Wan et al. [30] proposed an approach for the requirement specification, formal modeling, and verification of a flood monitoring system, which is a system which alerts the flood zone population through cell message services. This system uses a large number of sensors that provide information to a particular area via messages. The objective is to integrate the software with the system and construct the design, formal model, and implementation code for a flood monitoring system. Akhtar and Khan [31] proposed a flood monitoring system consisting of an ultrasonic wave system that automatically senses the level of water. This development is supported by advanced software programming and electronics, which automatically senses the water-level measurement and sends the water-level value to the control rooms to display the values.

Some studies propose a device running on electricity produced by solar panels that measure the real-time water level. The system supports the Global Positioning System (GPS) for mobile modem and microcontroller to send the early warning via messages to the flood zone people. The contact numbers of people living in flood zones are stored in a database, and the system warns about the flood water tidal level before and after the flood. The water level and flow precipitation level are necessary to prevent floods. Rahmtalla et al. [32] proposed a real-time flood monitoring and warning system to track storm paths from satellite images to make better decisions about flood warning and rescue services. The real-time water conditions can be monitored using a wireless sensors network that uses general packet radio service for public services to send data to the server application.

In literature, there is no work done in the formal specification, modeling, proving of safety and liveness properties of a flood warning, monitoring, and rescuing SoS. It is a safety-critical system and millions of human lives are dependent on this system; therefore, the focus is on formal, rigorous, and precise specification and the modeling and proving of safety and liveness properties of a flood monitoring SoS.

7. State of the Art

7.1. Formal Modeling and Verification. Formal modeling and verification are carried out using methods, techniques, and tools with a sound mathematical foundation. This mathematical foundation provides a rigorous, precise foundation in the requirement specification, modeling, verification, architecture definition, implementation, testing, maintenance, and evolution of SoS. In complicated systems such as SoS, multiple levels of formal methods are applied at different stages of the development lifecycle.

Formal verification is rigorous and ensures the correctness properties of the liveness and safety of the system. Formal methods are used in requirement specification and design to ensure correctness properties using state-transition machines, abstraction principle, refinement principle, proof obligations, theorem proving, and inductive assertions [33].

Formal verification is implemented by the CP-Nets model as well as the Event-B proofs constructed by using the platform Rodin. A formal CP-Nets model of the system is developed with precisely defined functional as well as nonfunctional properties.

A model provides a basis to improve and verify the system in a rigorous manner, and it is also important in safeguarding the system's correctness. It takes input and systemically checks whether the system holds the correctness properties of safety and liveness. The flood perceiver system model is constructed using CP-Nets and formal Event-B proofs developed in IDE Rodin.

7.2. Model Checking. Model checking [34–42] is one of the most practical formal methods for automatic, systematic, and exhaustive verification. "It is a computer-assisted method for the analysis of dynamical systems that can be modeled by state-transition systems" [36]. A mathematical model of the system is created and a comprehensive review of the model is performed. It includes checking all states and transitions in the model, i.e., exhaustive analysis of the model. It creates an abstract representation of a system with all the possible states and transitions.

The correctness of the model is verified through model checking techniques. The model takes input and systemically checks whether the system ensures the system properties. It is widely used in verification of critical engineering projects, i.e., verification of software systems of airplanes, spacecraft, trains, subway trains, nuclear reactors, and satellites. Its overall goal is to improve the quality of verification by specifying and checking correctness properties.

"Model checking is an automated technique that, given a finite-state model of a system and a formal property, systematically checks whether this property holds for (a given state in) that model. Model-based verification techniques are based on models describing the possible system behavior in a mathematically precise and unambiguous manner. The accurate modeling of systems leads to identifying incompleteness, ambiguities, and inconsistencies in informal system specifications. A system model is accompanied by algorithms that systematically explore all states of the system model. This provides the basis for verification techniques ranging from an exhaustive exploration (i.e., model checking) to experiments with a restrictive set of scenarios in the model (i.e., simulation), or in reality (i.e., testing)" [37].

Model checking uses temporal logic [43], linear-time temporal logic (LTL) [44–46], and computation-tree temporal logic (CTL) [42, 43, 47] for stating, checking, and verifying behavioral properties. The CP-Net and the Event-B language are both tool-supported formal method based on system state space. Since we have proved the static and dynamic properties of the translation by theorem analysis, we can use model checking as an auxiliary verification method. Model checking has a mathematical representation of a system, and its result consists of an exhaustive systematic exploration of the mathematical model [48, 49].

Because the well-known "state explosion" limitation exists, this approach can only be applied in a small system or used as an auxiliary method. The state-space of CP-net is calculated by the CPN Tools [48], and ProB calculates the state space of Event-B machine.

7.3. Correctness Properties. Safety and liveness properties are correctness properties. Correctness properties provide detailed system verification. Both safety and liveness properties complement each other. The safety property is an invariant which asserts that "something bad never happens and that an acceptable state of affairs is maintained". Calegari and Szasz [49] have defined safety property "$S = \{a1, a2, ..., an\}$ as a deterministic process that asserts that any trace including actions in the alphabet of S is accepted by S. ERROR conditions are like exceptions which state what is not required, as in complex systems, we specify safety properties by directly stating what is required."

According to Magee and Kramer [50], "a safety property is a property that can be specified by a safety formula of the form □p (i.e., temporal operator □ meaning always). This formula states that the property *p* holds throughout the computation." The liveness property asserts that "something good happens which describes the states of a system that SoS bring about given certain conditions" [51].

7.4. Coloured Petri-Nets (CP-NETS). A CP-Nets [6–13] is formal, discrete mathematics-based graphical language that constructs an executable system model. This model consists of places, transitions, and tokens flowing between places through transitions.

"A nonhierarchical CP-Nets is a nine-tuple CP-Nets = (P, T, A, Σ, V, C, G, E, I), where

(1) P is a finite set of places

(2) T is a finite set of transitions *T* such that $P \cap T = \varnothing$

(3) $A \subseteq P \times T \cup T \times P$ is a set of directed arcs

(4) Σ is a finite set of nonempty color sets

(5) V is a finite set of typed variables such that Type[v] $\in \Sigma$ for all variables $v \in V$

(6) C: P ⟶ Σ is a color set function that assigns a color set to each place

(7) G: $T \longrightarrow$ EXPRV is a guard function that assigns a guard to each transition *t* such that Type[G(t)] = Bool

(8) E: A ⟶ EXPRV is an arc expression function that assigns an arc expression to each arc a such that Type [E(a)] = C(p)MS, where *p* is the place connected to the arc a

(9) I: P ⟶ EXPR∅ is an initialization function that assigns an initialization expression to each place *p* such that Type[I(p)] = C(p)MS." [7]

These formal CP-Nets models can be compiled and executed. They are ideal to model the behavior of real-time safety-critical systems. A system can be modeled in the form of CP-Nets, and its correctness properties of safety and liveness can be exhaustively studied and verified. The CP-Net has a graphical and textual syntax, and it can exhaustively verify a system, in which concurrency, communications, and synchronizations play the key role.

7.5. Hierarchical Coloured Petri-Nets (CP-Nets). Hierarchical CP-Net has multiple layers (i.e., abstraction levels) of CP-Nets. It has a modular design as compared to simple CP-Net. Large complicated and complex systems are modeled by using hierarchical CP-Nets. "Hierarchical CP-Net is a four-tuple $CPN_H = (S, SM, PS, FS)$, where

(1) *S* is a finite set of modules. Every module is a coloured Petri-Net module $s = ((P^s, T^s, A^s, \Sigma^s, V^s, C^s, G^s, E^s, I^s), \mathbf{T^s_{sub}}, \mathbf{P^s_{port}}, PT^s)$. It is essential that $(P^{s1} \cup T^{s1}) \cap (P^{s2} \cup T^{s2}) = \varnothing$ for all *s1, s2* ∈ *S* such that s1 ≠ s2.

(2) SM: $T_{sub} \longrightarrow S$ is a submodule function that allocates a submodule to every substitution transition. The module hierarchy must be acyclic.

(3) PS is a port-socket relation function that allocates a port-socket relation $PS(t) \subseteq P_{sock}(t) \times \mathbf{P^{M(t)}_{port}}$ to each substitution transition *t*. It is essential that ST(p) = PT(p'), C(p) = C(p'), and $I(p)\langle\rangle = I(p')\langle\rangle$ for all (p, p') ∈ PS(t) and all $t \in T_{sub}$.

(4) $FS \subseteq 2^P$ is a set of nonempty fusion sets such that C (p) = C (p') and I (p) $\langle\rangle$ = I (p') $\langle\rangle$ for all p, p' ∈ fess and all fs ∈ FS" [7].

7.6. State Space Analysis and Behavioral Properties. The state space of CP-Nets is computed automatically to compute the behavioral properties of the Flood-WMR SoS. During model checking, temporal logics (i.e., LTL and CTL) are used for specification, checking, and verification of behavioral properties. These behavioral properties include the identification of the states in which the system terminates, minimum and a maximum number of tokens in a place, and identification of states in which the system always reaches (i.e., reach-ability analysis).

The CP-Nets toolkit named cpntools [8, 48] has a library for CTL model checking. In the case of the hierarchical CP-Nets of flood-WMR SoS, the state spaces can be generalized by replacing places with place instances and transitions with transition instances. The cpntools toolkit supports state spaces generation and analysis for hierarchical CP-Nets model of flood-WMR SoS. A directed graph is constructed, having a node for each reachable marking and an arc for each occurring binding element.

7.7. Event-B. Event-B [14, 15] is an extension of the B-Method for formal reasoning, modeling, and proving complex systems, including reactive and concurrent systems. It is a mathematical method for analysis and modeling the system. Event-B uses set theory as modeling notations. It performs the simulations by creating mathematical models that are exhaustively analyzed and proved.

The Event-B model is constructed in several abstraction layers consisting of context and machine. Static aspects of a

system are contained in context and dynamic aspects are specified by using the machine. Every context is made of carrier sets, constants, axioms, and theorems. Furthermore, context can be prolonged by other contexts and can be seen by more than one machine. The machine consists of variables, invariants, theorems, and events. A machine can be refined by a new machine. Variables are like constants that correspond to mathematical objects: binary relations, sets, numbers, and functions. A machine also contains several events that demonstrate its functionality. The event consists of three parts: (i) the event-name, (ii) guards, and (iii) actions. Guard is the critical condition for the event. The actions define how the state variables are accepted to evolve when carrying out the events.

Rodin [16, 17] is an Eclipse-based Integrated Development Environment (IDE) for the design and construction of Event-B [14, 15] proofs. The mathematical concepts of set theory, predicate logic, relations, and functions are the basis of Rodin. It integrates formal modeling as well as exhaustive proving using proof obligations. Instead of a compilation like higher-level programming language compilers, Rodin generates proof obligations, modeling trivial proof obligations, verifying liveness and safety properties. It analyzes, validates, and reasons for the flood monitoring perceivers' model. This platform is configurable and extensible; therefore, it can be integrated into different development projects and applications.

8. Materials and Methods

A correct model of Flood-WMR SoS depends on the following three major components:

(1) Flood data input: it obtains flood data from the flood perceivers sensor. This data include water level, water speed, wind direction, temperature, flood banks' status, dam status, and rainfall information.

(2) Formal modeling: it is semiformal modeling using UML Activity diagrams, and the formal CP-Nets models are built using these activity diagrams.

(3) Formal verification: the CP-Nets model checker models and model-checks the safety, liveness, deadlock freedom, and boundedness properties. The system is then specified and validated by writing axioms, preconditions, postconditions, invariant, guards, and Event-B events. The proofs are generated and checked by the Rodin theorem prover of Event-B.

The Event-B-based proof obligations have a purpose: to verify and validate the system using axioms, invariant, preconditions, postconditions, guards, and events. This Event-B model has both static as well as dynamic aspects of the system modeled and verified. Event-B generates proof obligations and refines and updates the system. In conjunction with the Event-B, CP-Nets performs model-checking. Some parts of the system are also specified and modeled by using CP-Nets modeling and model-checking. CP-Nets is a model-checker, and it performs exhaustive state-space analysis of the system. In this way, we have exhaustively and rigorously verified our system by using theorem prover-based method Event-B and model-checker CP-Nets.

8.1. Formal Modeling, Formal Proofs, and Model Checking. The formal foundation for specifying and verifying SoS architecture developed in the European ArchWare [52] project offers a sound foundation upon in which we can design and develop the formal-specification description. The flood-WMR SoS is based on SoS, consisting of multiple systems that work together. These systems consist of communication systems, monitoring systems, perceivers systems, central control systems, and information processing systems.

Our proposed approach has the phases of (i) flood perceivers, (ii) formal modeling, (iii) formal verification, and (iv) proof obligations. The model of a flood-WMR SoS has been specified, modeled, and verified to validate the approach. In the first phase of requirement specification, we have modeled the requirements in simple natural language. These specifications are then transformed into CP-Nets ML and are modeled and verified by CP-Nets. We have refined these specifications into proof obligations of Event-B proofs. In short, we have used the CP-Nets and Event-B methods for the formal specification, modeling, verification, and validation of the flood-WMR SoS.

9. Flood Monitoring Perceivers

Flood monitoring perceivers are used for early detection of floods as well as the analysis of flood intensity. Figure 4 shows the flood monitoring perceivers. Flood sensor data is of critical importance for the flood-WMR SoS. Based on this data, future predictions and decisions are made, and these decisions are critical for saving humans' lives and property. These flood-WMR perceivers are formally modeled and verified to ensure correct properties of safety and liveness.

After the feasibility study, the requirements are identified and specified. We have created UML activity diagrams [53–56] to understand the flood monitoring perceivers better, and then, in the subsequent stage, formal models and proofs are specified.

A CP-Nets-centered formal model is specified; then, the correctness properties of safety and liveness are ensured through CP-Nets model-checking and Event-B proof obligation models. The flood monitoring perceivers' requirements are elicited according to their specific functionality. The functional and nonfunctional requirements are specified for all the users of system. A functional diagram of flood perceivers is central in understanding the SoS. Flood monitoring and rescue based on flood perceivers is an iterative procedure that involves all the stakeholders, i.e., government flood monitoring agencies, disaster management departments, rescue services, hospitals, schools, mosques, and flood-area population.

The flood monitoring perceivers directly influence flood prevention as well as after-flood rescue services. Flood WMR

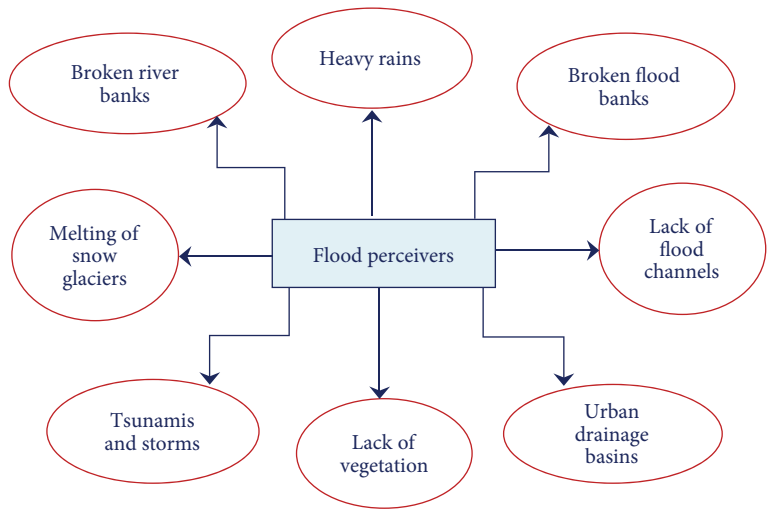

Figure 4: Flood monitoring perceivers.

```
FMS_Sensor_Data.cpn
  Step: 0
  Time: 0
  Options
  History
  Declarations
    Standard priorities
    Standard declarations
      colset UNIT = unit;
      colset BOOL = bool;
      colset INT = int;
      colset DATA = string;
      colset S_NO = int;
      colset S_NOxDATA = product S_NO*DATA;
      colset DATAPACK = record seq:S_NO*data:DATA;
      colset ACKPACK = S_NO;
      colset PACKET = union Data:DATAPACK + Ack:ACKPACK;
      colset RESULT = with success | failure | duplicate;
      val SensorInfo = 1`(1,"Sensor1_DATA") ++
        4`(2,"Sensor2_DATA") ++
        1`(3,"Sensor3_DATA") ++ 3`(4,"Sensor4_DATA") ++
        5`(5,"Sensor5_DATA") ++ 1`(6,"Sensor6_DATA");
      var n, k : S_NO;
      var d : DATA;
      var datapack : DATAPACK;
      var pack : PACKET;
      var res : RESULT;
  Monitors
  Sensor_Data
```

Figure 5: CP-Nets color sets data types of the flood.

SoS requirements are shown in Table 3. The primary goal is to build a correct, reliable, and efficient formal model.

10. Flood-WMR SoS: CP-Nets Modeling and Model Checking

The CP-Nets model for flood-WMR SoS is like a formal data flow model consisting of places, transitions, and labeled arcs between places and transitions. Two places are connected through a transition. It specifies the formal model centered on data flow. It performs model checking, i.e., exhaustive state-space evaluation of the system. CP-Net supports a specification language called CP-Nets ML, which is a functional metalanguage. It is formal, precise, well-defined, and unambiguous. CP-Nets ML has a concept of color sets as data types. New color sets can be easily declared and initialized. There are composite color sets that enable the construction of complex data structures. These composite color sets use (i) Cartesian product of two or more color sets, (ii) union, and (iii) record.

The primitive and composite data types (i.e., color sets) of the system's communication module are shown in Figure 5. This communication module ensures no loss of data packets during data transfer between the flood perceivers and the flood-WMR SoS. If a data packet is lost, its duplicate packet is generated, and after receiving a data packet, the SoS sends an acknowledgment to the perceiver. A complete CP-Nets model of the flood-WMR perceivers is constructed specifying all the sensors, actuators, and control units of the system. Composite data types are used, i.e., DATAPACK (which is a record) and PACKET (which is a union).

The flood-WMR perceivers of (i) heavy rain, (ii) overflowing rivers, (iii) damaged flood banks, (iv) urban drainage basins, (v) tsunamis and storms, (vi) damaged river banks, (vii) lack of forests and vegetation, and (viii) melting snow in glaciers. We have modeled these flood-WMR perceivers as CP-Nets models. These flood-WMR perceivers give information about floods and therefore are vital for saving millions of people's lives and properties. The people living in the affected areas get information about the flood intensity. In case of floods, they can move towards shelter houses and save their lives, material, livestock, and crops.

Figure 6 shows the CP-Nets model for communication between flood perceivers and the control unit of the SoS. The model is designed so that if some parts of the system fail (i.e., if some data packets are lost) the system ensures that there are duplicate packets and the information is transferred without any loss.

As a result, there is no loss of data packets in transport and communication. This model consists of places, transitions, and composite data types. The place limit in the model ensures that the system does not get congested and

Table 3: Flood-WMR SoS requirements.

Requirement	Description
Fun-1	Flood-WMR SoS estimates the floodwater and rainfall categorized as low, medium, and high
Fun-2	Flood-WMR SoS estimates the storage capacity of water in dams and rivers
Fun-3	When the storage capacity of water is min rainfall = high alert flood_warning immediately
Fun-4	When the storage capacity of water is max rainfall = high, medium alert flood_prevention immediately
Fun-5	No rainfall, sufficient storage of water flood-WMR alerts no_flood
Fun-6	During the disaster, guarantee the rescue services to the victims
Fun-7	Guarantee the communication of all stakeholders during flood

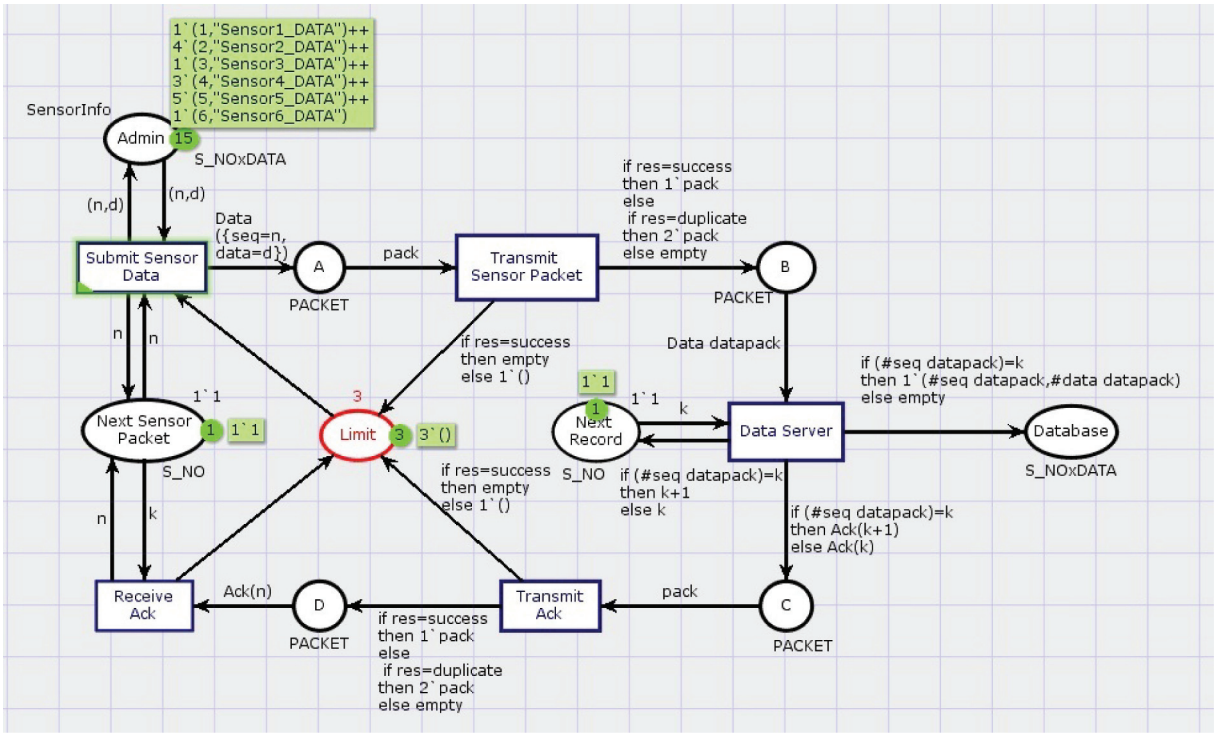

Figure 6: CP-Nets model for communication of Flood-WMR SoS perceivers.

deadlocked. Extra packets are stored in this place named limit, and it ensures that there are a limited number of packets passing through the system at a given time.

Figure 7 illustrates the state space of CPN tools describing the reachable states and their initial marking. Since here in this paper we cannot present all states, therefore only the firing of the transition gives access to the state-space analysis. A formal description and proof are to be created. However, this illustration might help the reader to have an improved understanding of the transformation process. The first number is the node number, and in this case, the node number is 1. The two numbers at the bottom of the node are the number of predecessors and successors calculated. In this case, five predecessors and eleven successors have been calculated.

11. Flood-WMR SoS: Event-B Proofs

The Event-B contexts define and specify the system's static aspects, i.e., carrier sets, constants, and axioms. In this SoS, the carrier sets determine the intensity value, direction, and speed of floodwater. The axioms are specified using carrier sets and constants. These static aspects of the system are used by the dynamic aspects, i.e., machines containing *variables, invariants*, and *events*. The static, as well as the dynamic aspects of the system, are presented below:

$$\begin{aligned} &\text{axm4: flood warning} \subseteq \text{FLOOD},\\ &\text{axm5: flood prevention} \subseteq \text{FLOOD},\\ &\text{axm6: no flood} \subseteq \text{FLOOD},\\ &\text{axm7: partition (VALUE, low, medium, high).} \end{aligned} \tag{1}$$

The Flood_Ctx defines the static aspects seen and used by machine_0, machine_1, machine_2, machine_3, and machine_4, as shown in Figure 8, as all of these machines deal with the flood warning, monitoring, and rescue services. Flood_Ctx uses four sets, twelve constants, and nine axioms:

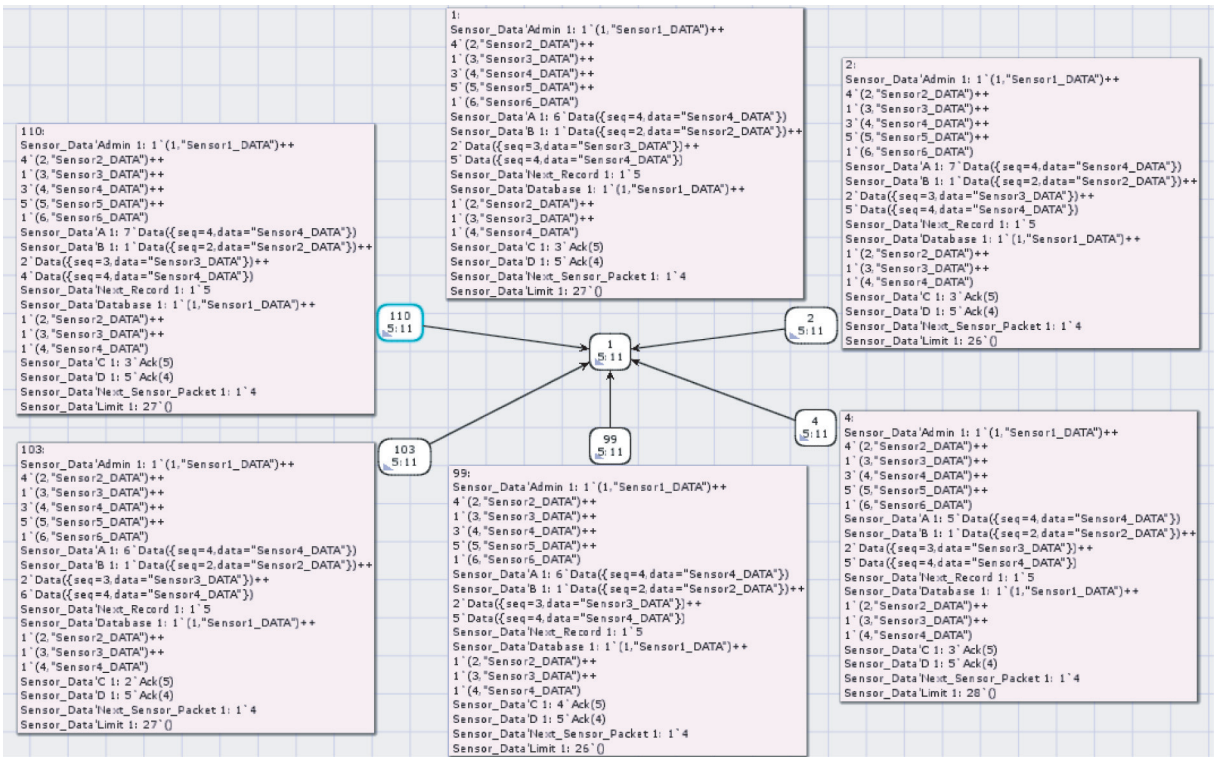

Figure 7: CP-Nets state space analysis for the initial marking.

axm8: partition (DIRECTION, towards Population, towards Crops, Towards Flood Lands)

axm9: partition (SPEED, slow_speed, medium_speed, high_speed).

(2)

The Rescue_Ctx contains the static aspects of the flood rescue activities. This context is seen by Machine_*3* and Machine_*4*, as these machines deal with the after-flood situation's rescue services. Rescue_Ctx uses two sets named as (TRANSMISSION, RESCUE_SERVICES) ten constants (InjuredPersons, electricityFailure, houseDestroyed, crops-Destroyed, Alarm, SMS, Mobile, Radio, TV, Internet) and five axioms demonstrated below:

axm1: injured Persons ⊆ RESCUE_SERVICES,

axm2: electricity Failure ⊆ RESCUE_SERVICES,

axm3: house Destroyed ⊆ RESCUE_SERVICES,

axm4: crops Destroyed ⊆ RESCUE_SERVICES,

axm5: partition (TRANSMISSION, Alarm, SMS, Mobile, Radio, TV, Internet).

(3)

Machine_*0* represents the first abstraction level that manages the flood warning; machine_*0* specifies and defines the system's dynamic properties (i.e., behavior). The requirement property of flood warning is verified to ensure correctness. The machine_1 have five variables named as Flood_Warning Water_Speed River_Capacity River_WaterLevel and Water_Direction Inv1, inv2, inv3, inv4, and inv5 which will ensure safety. If the states are unsafe, the invariants must be invalid, and when the invariant is true, the form should be valid or safe. Grd1 and grd are true when there is a flood alert or direction of water toward the population (a member of carrier set):

inv1: Flood_Warning ⊆ FLOOD,

inv2: River_Capacity ∈ low, medium, high,

inv3: Water_Speed ∈ slow_speed, medium_speed, high_speed,

inv4: River_WaterLevel ∈ low, medium, high,

inv5: Water_Direction ∈ towards population, towards crops.

(4)

EVENTS Flood_Warning STATUS ordinary ANY f_alert, f_level.

grd1: f_level = high,

grd2: f_alert ∈ FLOOD,

grd3: Water_Direction ∈ towards Population, towards Crops,

grd4: Water_Speed ∈ medium_speed, high_speed,

act1: Flood_Warning := Flood_Warning.

(5)

The Machine_*1* defines the event that manages flood prevention, water reserves' storage capacity, and water direction. Machine_*1* verifies the requirement property of Flood-WMR SoS. The invariant, grd, and act for machine_1 are shown below:

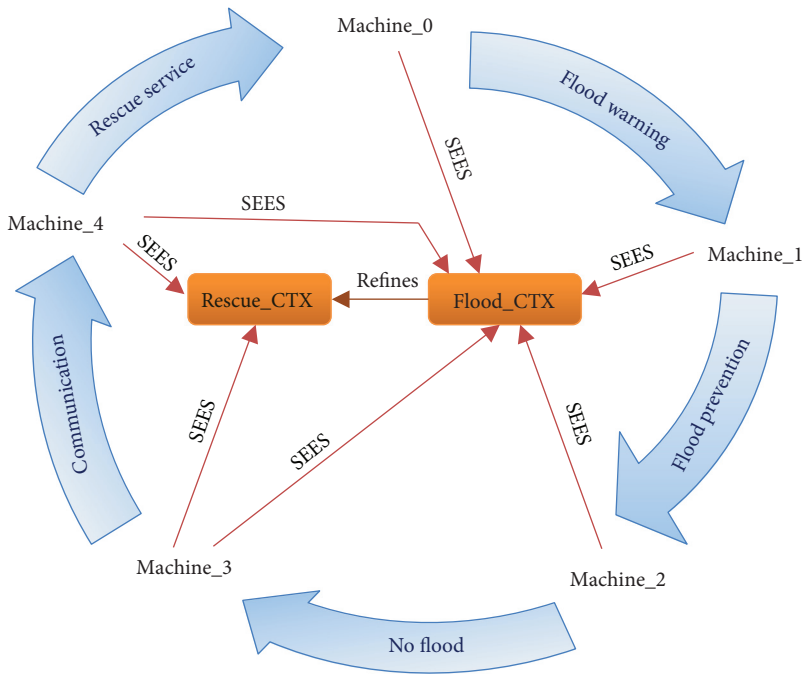

Figure 8: Flood-WMR SoS Event-B Model: the two contexts and five machine interactions.

inv3: Water_Direction ∈ towards Population,
towards Crops, towards FloodLands,
inv4: Water_Speed ∈ slow_speed, medium_speed,
high_speed. (6)

EVENTS Flood Prevention, STATUS ordinary, and ANY f_alert:

grd3: Water_Speed ∈ medium_speed, high_speed,
grd4: Water_Direction ∈ towards Population, towards Crops,
act1: Flood_Prevention ≔ Flood_Prevention. (7)

Machine_2 verifies the requirement property of No flood danger and alerts to the system and sees the only one context of rescue services. Event changes the variable (No_Flood Storage_Capacity Water_Direction Flood_Alert) values named as TRANSMISSION:

inv1: Storage_Capacity ∈ low, medium, high,
Inv2: No_Flood ⊆ FLOOD,
inv2: Flood_Alert ⊆ FLOOD,
inv3: Water_Direction ∈ towards Population,
towards Crops, towards Flood Lands. (8)

EVENTS NO_Flood and STATUS ordinary ANY FloodAlert:

grd1: Storage_Capacity = low,
grd2: FloodAlert ∈ FLOOD,
grd3: Water_Direction ∈ towards Population,
towards Crops,
act1: Flood_Alert ≔ Flood_Alert,
act2: No_Flood ≔ No_Flood. (9)

Machine_3 defines the event that manages communication and information transmission between flood perceivers and prevention services. Machine_3 sees the two contexts (Flood_Ctx and Rescue_Ctx). Communication Water_Direction Flood_Prevention Flood_ArrivalPrevention and Flood_ArrivalWarning variable are used in machine_2.inv1to inv8 are labeled as to describe the subsets values for a carrier set:

inv8: Water_Direction ∈ towards Population,
towards Crops, towards FloodLands,
inv9: Communication ∈ Alarm, SMS, Mobile, Radio,
TV, Internet. (10)

EVENTS Transmission, STATUS ordinary, ANY FloodAlert, and Message:

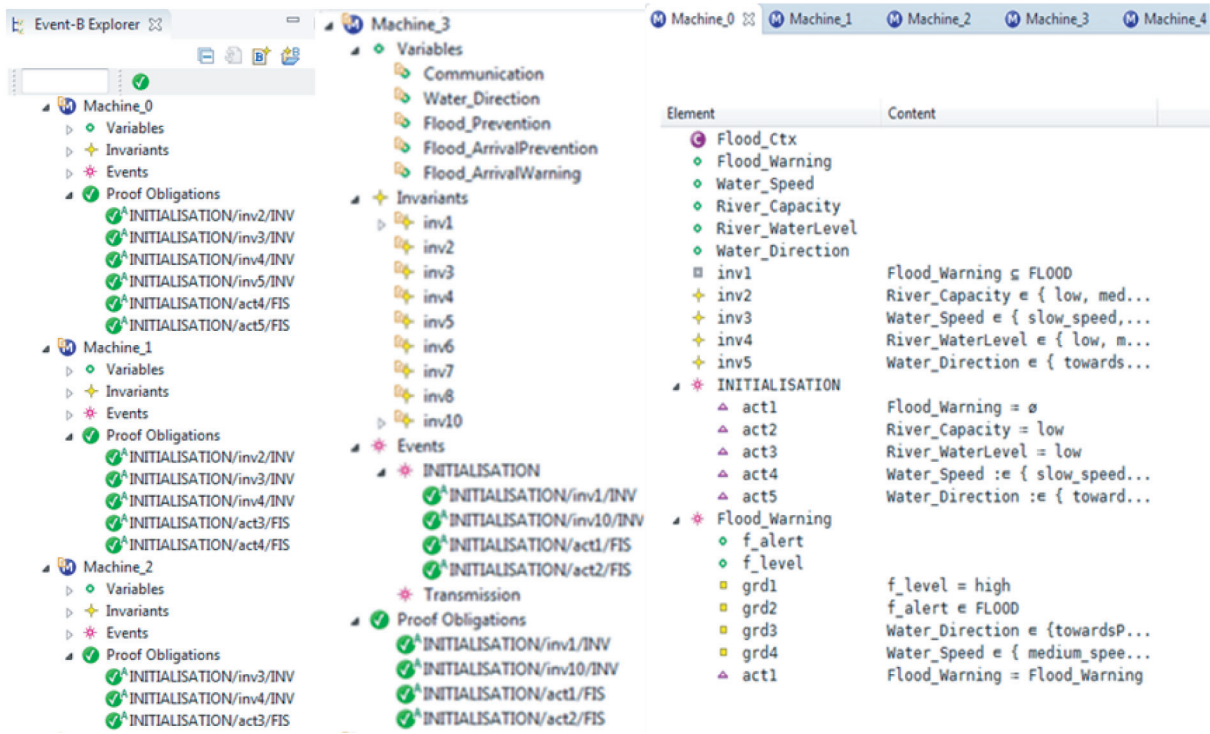

Figure 9: Dynamic behavior of machines.

grd5: Water_Direction ∈ towards Population, towards Crops,
grd6: Communication ∈ Alarm, SMS, Mobile, Radio,
TV, Internet,
act1: Flood_Prevention ≔ Flood_Prevention,
act2: Flood_ArrivalWarning ≔ Flood_ArrivalWarning,
act3: Flood_ArrivalPrevention ≔ Flood_ArrivalPrevention. (11)

Machine_4 manages all the rescue services. Events change these variable values. There are six invariants used in machine_4, and the first five invariants ensure those attributes as a subset of the rescue team when the ood has followed. The variables used in machine_4 are ProvideShelterHouses, ProvideFood, RescuePersons, ProvideElectricity, RescueTeamsSent, and Communication. All the labeled guards (named as grd1, grd2, grd3, grd4, and grd5) ensure the requirements' safety property. The act1 ensures the liveness property of correctness:

inv2: Provide Food ⊆ RESCUE_SERVICES,
inv3: Rescue Persons⊆ RESCUE_SERVICES,
inv4: Provide Electricity⊆ RESCUE_SERVICES,
inv5: Rescue Teams Sent⊆ RESCUE_SERVICES,
inv6: Communication ∈ Alarm, SMS, Mobile, Radio,
TV, Internet. (12)

EVENTS RescueServices STATUS ordinary ANY Rescue:

grd1: Rescue ∈ RESCUE_SERVICES,
grd2: Rescue Persons = injured Persons,
grd3: Provide Electricity = electricity Failure,
grd4: Provide Shelter Houses = house Destroyed,
grd5: Provide Food = crops Destroyed,
act1: Rescue Teams Sent: = Rescue Teams Sent. (13)

Rodin platform of Event-B generates proof obligations for machine_0, machine_1, machine_2, machine_3, and machine_4 represented in Figure 9. The proof obligations of the flood monitoring system are satisfied, and there are no conflicts. Refinements and updates fulfil all the conflicting proof obligations.

12. Contributions

The contributions in the analysis, modeling, and verification of a flood-WMR SoS are

(1) The flood-WMR SoS provides formal verification of correctness properties by using CP-Nets-based model checking and Event-B-based formal proofs.
(2) It is centered on uninterrupted communication between the flood monitoring system and stakeholders (i.e., affected citizens, rescue services, police, electronic media, and government authorities).

Communication with these responders ensures rapid propagation of information and early action after-flood alert. CP-Nets model of communications are constructed and model-checked.

(3) The flood monitoring perceivers are identified and modeled. These flood monitoring perceivers have sensors that provide real-time information about a number of perceivers to identify a flood situation before-hand.

(4) The flood-WMR SoS also emphasizes emergency and rescue services just after a flood. It proposes alternative infrastructures for uninterrupted communication if the disaster destroys the principal communication infrastructures.

13. Discussion and Conclusion

This work results in a formal, correct, exhaustive, and rigorous model of a safety-critical flood-WMR SoS. We have a solid background in mathematical analysis, design, specification, modeling, and verification of different distributed systems. Our previous expertise on ensuring correctness properties of safety and liveness for autonomous robotic multiagent system [57–59] and flood monitoring systems [30, 60–62] has helped us in the formal modeling and proving of this flood-WMR SoS.

Event-B-based proof obligations constants, variables, invariants, guards, and events are specified. Event-B verified the invariants and guards by using formal proofs. Events play the most vital role in proving the behavioral aspects of the system-of-systems. We have constructed multiple levels of abstractions for exhaustive-proofs, with each having a different refinement level, and safety and liveness properties are exhaustively analyzed and proved.

The CP-Nets model specifies a rigorous and precise model of the flood-WMR SoS, centered on the correct properties of safety and liveness. The proposed approach uses two industrial-strength formal methods for the requirement analysis, modeling, formal proving, refinement, abstraction, and model-checking of the flood-WMR SoS.

Induction-based exhaustive formal proof methods such as Event-B proofs and CP-Nets model checking perform the formal verification and validation. One of the main challenges of this work is to introduce verification and validation methodologies based on an amalgamation of formal methods and techniques. We have proposed transformations from one formal-method specification to another specification for exhaustive verification and state-space evaluation. This transformation allows Event-B context and machine conversion into CP-Nets, allowing set theory and first-order predicate-calculus-based formal proofs and temporal logic-based model checking.

A flood-WMR SoS shows the process of such formal verification and validation approach in the flood monitoring context. This contribution has several perspectives. One of them is a detailed formal-representation of the transformation based on the rules and a software platform developed for its automation. The design and development of an automatic conversion tool are required to facilitate the generation of Event-B context and machines, as it takes a long time to establish them manually, especially for large models.

This work also allows bridging the gap between formal proofs of Event-B and model checking based on CP-Nets. It allows rigorous specifications and verification. The objective is to design and develop an approach for the formal specification, verification, and validation of a correct flood-WMR SoS.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

References

[1] MacDonald, M., "Guidelines for Climate Compatible Construction & Disaster Risk Reduction in Rural Punjab", 2013. [Online]. Available: http://cdkn.org/wp-content/uploads/2012/09/Climate-Compatible-Construction-Guidelines_Final.pdf, [Accessed: Aug. 05, 2014].

[2] S. I. Bukhari and D. S. Rizvi, "Pakistan's flooding of july-august 2010:not only a natural disaster," *Academic Research Journals*, vol. 3, no. 6, pp. 160–167, 2015.

[3] N. R. Storrey, *Safety Critical Computer Systems*, Addison-Wesley Longman Publishing Co, Boston, MA, USA, 1996.

[4] IEC61508, "Functional Safety of Electrical/Electro-nic/programmable electronic safety related systems-part3: software requirements" (IEC, 2000).

[5] EN50128, "Railway Applications: Communications, Signaling and Processing Systems"-- Software for Railway Control and Protection Systems (CENELEC, 2001).

[6] W. M. van der Aalst and C. Stahl, *Modeling Business Processes: A Petri Net-Oriented Approach*, The MIT Press, Cambridge, MA, USA, 2011.

[7] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*, Springer Publishing Company, Berlin, Germany, 2009.

[8] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3–4, pp. 213–254, 2007.

[9] L. M. Kristensen, S. Christensen, and K. Jensen, "The practitioner's guide to coloured petri nets," *International Journal on Software Tools for Technology Transfer*, vol. 2, no. 2, pp. 98–132, 1998.

[10] K. Jensen, "A brief introduction to coloured petri nets," in *Proceedings of the International Workshop on Tools and Algorithms for the Construction and Analysis of Systems*, pp. 203–208, Springer, Luxembourg City, Luxembourg, 1997.

[11] K. Jensen, "Coloured petri nets. Basic concepts, analysis methods and practical use," *Practical Use*, Vol. 3, Springer, Berlin, Germany, 1997.

[12] K. Jensen, "Coloured petri nets. Basic concepts, analysis methods and practical use," *Analysis Methods*, Vol. 2, Springer, Berlin, Germany, 1994.

[13] K. Jensen, "Coloured petri nets. Basic concepts, analysis methods and practical use," *Basic Concepts*, Vol. 1, Springer, Berlin, Germany, 1992.

[14] J.-R. Abrial, *Modeling in Event-B System and Software Engineering*, Cambridge University Press, Cambridge, UK, 2010.

[15] J. R. Abrial, *The B-Book*, Cambridge University Press, New York, NY, USA, 1996.

[16] J.-R. Abrial, M. Butler, S. Hallerstede, T.-S. Hoang, F. Mehta, and L. Voisin, "Rodin: an open toolset for modelling and reasoning in Event-B," *International Journal on Software Tools for Technology Transfer*, vol. 12, no. 6, pp. 447–466, 2010.

[17] M. Jastram and M. Butler, *Rodin User's Handbook: Covers Rodin V.2.8*, Create Space Independent Publishing Platform, Scotts Valley, CA, USA, 2014.

[18] C. DiPetto, "Office of the deputy under secretary of defense for acquisition and technology, testimony before the house committee on armed services, subcommittee on readiness, march 13, 2008a," *As of December*, vol. 28, 2010.

[19] M. W. Maier, "Architecting principles for systems-of-systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998.

[20] D. Firesmith, *Profiling Systems Using the Defining Characteristics of Systems of Systems (SoS)*, Software Engineering Institute, Carnegie Mellon University Tech. Rep. TN-001, Pittsburgh, PA, USA, 2010.

[21] J. Boardman and B. Sauser, "System of systems-the meaning of of," in *International Conference on System of Systems Engineering*, IEEE, Budapest, Hungary, June 2006.

[22] L. Northrop, P. Feiler, R. P. Gabriel et al., *Ultra-Large-Scale Systems - the Software Challenge of the Future*, Carnegie Mellon, Pittsburgh, PA, USA, 2006.

[23] B. S. Blanchard and W. J. Fabrycky, *Systems engineering and analysis*, Vol. vol. 4, Prentice-Hall, Englewood Cliffs, NJ, 1990.

[24] D. A. DeLaurentis and W. A. Crossley, "A taxonomy-based perspective for systems of systems design methods,"vol. 1, pp. 86–91, in *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 86–91, IEEE, Waikoloa, Hawaii, USA, October 2005.

[25] J.-L. Boulanger, *Formal Methods: Industrial Use from Model to the Code*, Wiley-ISTE, Hoboken, NJ, USA, 2012.

[26] Wiley, *Formal Methods Applied to Industrial Complex Systems: Implementation of the B Method*, Wiley-IEEE Press, Hoboken, NJ, USA, 1st edition, 2014.

[27] A. Ribeiro, A. Cardoso, A. S. Marques, and N. E. Simões, "Web-based platform for river flood monitoring," in *Proceedings of the 2017 4th Experiment@International Conference (exp.at'17)*, pp. 131-132, IEEE, Faro, Portugal, June 2017.

[28] K. P. Menon and L. Kala, "A review on flood monitoring: design, implementation and computational modules," in *Proceedings of the International Journal of Innovative Research in Computer*, 2017.

[29] M. I. Alipio, J. R. R. Bayanay, A. O. Casantusan, and A. A. Dequeros, "Vehicle traffic and flood monitoring with reroute system using Bayesian networks analysis," in *Proceedings of the 2017 IEEE 6th Global Conference on Consumer Electronics*, Las Vegas, USA, January 2017.

[30] Z. Wan, Y. Hong, S. Khan et al., "A cloud-based global flood disaster community cyber-infrastructure: development and demonstration," *Environmental Modelling & Software*, vol. 58, pp. 86–94, 2014.

[31] N. Akhtar and S. Khan, "Formal architecture and verification of a smart flood monitoring system-of-systems," *The International Arab Journal of Information Technology (IAJIT)*, vol. 16, no. 2, 2018.

[32] A. Rahmtalla, A. Mohamed, and W. G. Wei, "Real time wireless flood monitoring system using ultrasonic waves," *International Journal of Science and Research*, vol. 3, no. 8, pp. 2012–2015, 2014.

[33] J. Sunkpho and C. Ootamakorn, "Real-time flood monitoring and warning system," *Songklanakarin Journal of Science and Technology*, vol. 33, no. 2, pp. 227–235, 2011.

[34] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods and experience," *ACM Computing Surveys*, vol. 16, no. 4, Article ID 19, 2009.

[35] E. M. Clarke Jr, O. Grumberg, D. Kroening, D. Peled, and H. Veith, *Model Checking*, MIT press, Cambridge, MA, USA, 2nd edition, 2018.

[36] E. M. Clarke, T. A. Henzinger, H. Veith, and R. Bloem, *Handbook of Model Checking*, Springer International Publishing AG, part of Springer Nature, Berlin, Germany, 1st edition, 2018.

[37] E. M. Clarke, T. A. Henzinger, and H. Veith, "Introduction to model checking," *In Handbook of Model Checking*pp. 1–26, Springer International Publishing AG, part of Springer Nature, Berlin, Germany, 1st edition, 2018.

[38] C. Baier and J.-P. Katoen, *Principles of Model Checking (Representation and Mind Series)*, The MIT Press, Cambridge, MA, USA, 2008.

[39] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*, MIT Press, Cambridge, MA, USA, 1999.

[40] J. Katoen, *Concepts, Algorithms, and Tools for Model Checking*, University of Erlangen–Nürnberg, Erlangen, Germany, 1999.

[41] E. M. Clarke, E. A. Emerson, S. Jha, and A. P. Sistla, "Symmetry reductions in model checking,," in *Lecture Notes in Computer Science*, A. J. Hu and M. Y. Vardi, Eds., vol. 1427pp. 147–158, 1998.

[42] E. M. Clarke, R. Enders, T. Filkorn, and S. Jha, "Exploiting symmetries in temporal logic model checking," *Formal Methods in System Design*, Kluwer Academic Publishers, vol. 9, no. 1–2, , pp. 77–104, 1996.

[43] E. M. Clarke, E. A. Emerson, and A. P. Sistla, "Automatic verification of finite-state concurrent systems using temporal logic specifications," *ACM Transactions on Programming Languages and Systems*, vol. 8, no. 2, pp. 244–263, 1986.

[44] E. A. Emerson, "Temporal and modal logic," in *In Handbook of Theoretical Computer Science*, J. van Leeuwen, Ed., pp. 995–1072, MIT Press, Cambridge, MA, USA, 1990, http://dl.acm.org/citation.cfm?id=114891.114907.

[45] A. Valmari, "A stubborn attack on state explosion," *Formal Methods in System Design*, vol. 1, no. 4, pp. 297–322, 1992.

[46] M. Vardi and P. Wolper, "An automata-theoretic approach to automatic program verification (preliminary report)," in *Proceedings of the 1st IEEE Symposium on Logic in Computer Science*, pp. 332–344, IEEE, New York, NY, USA, July 1986.

[47] O. Kupferman, M. Y. Vardi, and P. Wolper, "An automata-theoretic approach to branching-time model checking," *The Journal of the ACM*, vol. 47, no. 2, pp. 312–360.

[48] R. Gerth, R. Kuiper, D. Peled, and W. Penczek, "A partial order approach to branching time logic model checking," in *Proceedings of the Third Israel Symposium on the Theory of Computing and Systems*, pp. 130–139, Tel Aviv, Israel, January 1995.

[49] D. Calegari and N. Szasz, "Verification of model transformations: a survey of the state-of-the-art," *Electronic Notes in Theoretical Computer Science*, vol. 292, pp. 5–25, 2013.

[50] J. Magee and J. Kramer, *Concurrency: State Models and Java Programs*, John Wiley & Sons, 2nd edition, 2006.

[51] M. Zohar and P. Amir, *Temporal Verification of Reactive Systems: Safety*, Springer-Verlag New York, Inc., New York, NY, USA, 1995.
[52] D. Giannakopoulou, J. Magee, and J. Kramer, *"Fairness and Priority in Progress Property Analysis," Technical Report*, Department of Computing, Imperial College of Science, Technology and Medicine, Queens Gate, London, UK, 1999.
[53] M. Westergaard and H. Verbeek, "Cpn tools." [Online]. Available: http://cpntools.org/,[Dec.01,2019].
[54] F. Oquendo, B. Warboys, R. Morrison et al., *"Archware: Architecting Evolvable Software," in European Workshop on Software Architecture*, pp. 257–271, Springer, Berlin, Germany, 2004.
[55] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*, Pearson Higher Education, Midrand, South Africa, 1999.
[56] OMG "Unified Modeling Language Superstructure Specification, version 2.1.1." Document Formal/2007-02-05, Object Management Group, February 2007. http://www.omg.org/cgi-bin/doc?formal/2007-02-05.
[57] S. Harald and J. H. Hausmann, "Semantics of uml 2.0 activities with data-flow," in *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, Washington, DC, September 2004.
[58] N. Akhtar, Y. L. Guyadec, and F. Oquendo, "Formal specification and verification of multi-agent robotics software systems: a case study," in *Proceedings of the International Conference on Agents and Artificial Intelligence (ICAART 09)*, INSTICC Press, Porto, Portugal, January 2009.
[59] N. Akhtar, "Contribution to the formal specification and verification of a multi-agent robotic system," PhD Thesis, Ecole Doctorale, Laboratory VALORIA, Valoria, Cantabria, 2010.
[60] N. Akhtar, Y. L. Guyadec, and F. Oquendo, "Formal requirement and architecture specifications of a multi-agent robotic system," *Journal of Computing*, vol. 04, no. 04, pp. 75–80, 2012.
[61] N. Akhtar, A. Rehman, M. Hussnain et al., "Hierarchical coloured petri-net based multi-agent system for flood monitoring, prediction, and rescue (fmpr)," *IEEE Access*, vol. 7, no. 1, p. 180, 2019.
[62] N. Akhtar, A. Rehman, and D. M. Khan, "Formal verification of safety and liveness properties using coloured petri-nets: a flood monitoring, warning, and rescue system," *Journal of Information Communication Technologies and Robotics Applications (JICTRA)*, ISSN, vol. 09, no. 01, pp. 80–88, 2018.