ANALYSIS, DESIGN, ARCHITECTURE SPECIFICATION, AND FORMAL VERIFICATION OF A SMART FLOOD MONITORING SYSTEM-OF-SYSTEMS

Saima Khan[1], Muhammad Anwar[2] and Nadeem Akhtar[3]
[1]Department of Computer Science, Virtual University Of Pakistan, Pakistan
[2]Instructor of Computer Sciences, Virtual University of Pakistan (VU), Pakistan
[3]Assistant Professor, Department of Computer Science & IT, The Islamia University of Bahawalpur, Pakistan
Corresponding Author: nadeem.akhtar@iub.edu.pk

Abstract: F*loods are the most common hazards in pakistan. In a flood situation, forecast of necessary information and an effective evacuation plan are vital. Smart flood monitoring system-of-systems (sos) is a flood monitoring and rescue system. It collects information from weather forecast, flood onlookers and observers. This information is processed and then made available as alerts to the clients. The system also maintains continuous communication with the authorities for disaster management, social services, and emergency responders. Smart flood monitoring sos synchronizes the support offered by emergency responders with the community needs. This paper presents the architecture specification and formal verification of the proposed smart flood monitoring sos. The formal model of this sos is specified as well as model-checked to ensure the correctness properties of safety and liveness.*

Keywords: Flood Monitoring; System-of-Systems (SoS); Behavioural Modeling; Formal verification; Model Checking; Correctness properties; Safety; Liveness

1. INTRODUCTION

Natural disasters affect millions of people every year around the globe. These disasters (i.e. floods, earthquakes, windstorms, hurricanes, fires) cause loss of precious human lives, animal lives, crops, infrastructure and properties. Resulting in drastic impact on the economy, especially for a developing country like Pakistan.

In Pakistan, floods are the most common hydrological disasters showing time and again the destructive power of moving water. Over the last few decades, floods are the single most common cause of destruction. Sometimes we are able to anticipate the flood and prepare the evacuation plan, i.e. a flood situation arising as a result of rains takes a little longer for the flood to develop. But there are also flash floods that develop instantly.

1. In our part of the world, the major factors that cause floods are:
2. Massive unorganized illegal constructions of buildings and roads on riversides hinder the natural flow of river water during low floods. It also reduces the area for absorption of excessive rain water.
3. Blocking of flood drain canals. These canals require periodic maintenance.
4. Cutting down of forests contributes in landslides as well as floods.
5. Breaking of levee or damages to dams. These levee and dams require continuous periodic maintenance.
6. Each year the seasonal weather system in Pakistan brings heavy monsoon rains.

When a flood threat becomes a reality, infrastructure is destroyed, and people are left homeless to fight with water borne diseases. The local economy suffers. A flood in 2010 affected more than 20 million people. Almost 2000 people were reported dead. People were dislocated; crops and properties were ruined [1].

Flood resistant structures and redirection are the ways to reduce losses. Massive infrastructure of dams and levees also helps. Advanced computer forecasts and predictions based on weather and flood related data help to inform with accuracy about the possibility and intensity of the flood. Smart Flood Monitoring SoS observes and monitors floods and broadcasts the flood warnings to emergency responders and potential victims. These flood warnings are sent to the affected areas via SMS, television broadcast, radio broadcast, email and a web portal. The emergency responders are kept informed using the direct dedicated high speed links. It keeps the flood affected people informed about the availability of food, shelter, new road plans, power services, gas services, health services, and rescue services. They also keep the people connected with one another during the disaster. It is important to have an efficient and robust communication infrastructure running uninterrupted throughout the calamity. There is a backup plan to keep the communication up during the disaster even if the communication wires and towers go down because of floods.

We have proposed a Smart Flood Monitoring SoS for the monitoring of floods as well as rescue and emergency services in a post-flood disaster situation. Our SoS includes booster stations equipped with booster balloons to ensure uninterrupted communication. These booster balloons keep the TV, radio, telephone, internet and cell phone services active throughout the course of the disaster. The use of these technologies keeps the people connected with each other; and also allows the statistical analysis of the information in hand to make correct decisions.

2. PROBLEM STATEMENT

In Pakistan floods are frequent and cause a constant threat to human lives, economy and infrastructure. Thus, a flood monitoring, warning and rescue system is of critical importance.

1) The proposed Smart Flood Monitoring SoS emphasizes 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 early flood alert.
2) The proposed Smart Flood Monitoring SoS also emphasizes on emergency and rescue services just after a flood. It proposes alternative infrastructures for uninterrupted communication if the principal communication infrastructures are destroyed by the disaster.
3) The correctness of the system is verified by formal Model-Checking techniques.

3. STATE OF THE ART

3.1 System-of-Systems (SoS)

SoS facilitates development of large and complex systems. It is an integration of autonomous systems that are geographically distributed and support continuous evolution. These are the systems that are functionally and managerially independent. These systems on integration, share their resources and services to serve a larger, complex and unique functionality that is not possible to achieve otherwise.

Blanchard and Fabrycky in [2] describe SoS as a combined arrangement of managerially independent and geographically distributed elements (i.e. already fulfilling some purposes) put together to work and provide a functionality that is not possible otherwise.

3.2 Colored Petri-Nets (CPN)

Coloured Petri Nets are high level extension of Petri-Nets that help to validate concurrent and distributed systems involving synchronizations. It is a formal, mathematical and graphical language used to develop executable model of the system representing places and transitions. Jensen in [8] presented the concepts and the applications of CPNs in system modelling. CPN Tools [9] allow construction of CPN models with timing constraints. This tool also allows simulating system behaviour and verifying the system by using state space analysis [10]. The state space analysis performed by CPN Tools facilitates to analyze some standard properties including bounded-ness, home, liveness and fairness [11].

Berthomieu and Diaz in [3], Merlin and Farber in [4] have acknowledged Petri-Nets very suitable to model time-critical systems. Petri-Nets cater the time-critical requirements including associating the timing constraints with the places or transitions. Denaro and Pezze in [5]; He and Murata in [6] have presented a good review of various new developments and applications of Petri-Nets in software engineering. Xudong in [7] has presented a review showing the development and applications of Petri-nets in many disciplines. This review also highlights the Petri-Net support for general software engineering paradigms.

3.3 Model Checking

Model Checking [12-16] is a method for automatic and algorithmic verification of finite state concurrent systems. It takes as input a finite state model of a system and a logical property, it then systematically checks whether this property holds for a given initial state in that model. Model checking is performed as an exhaustive state space search that is guaranteed to terminate since the model is finite. It uses temporal logic to specify correct system behaviour.

Model checking addresses finite systems but can be scaled up to a more complex system as a System-of-System. Here, by complex we mean a system with a large number of independent interacting components, with non-linear aggregate activity, with concurrency between components and constant evolution. Model checking basic idea is to use algorithms executed by software tools to verify the correctness of the system. The user inputs a description of a model of the system, the possible behaviour, and a description of the requirements specification i.e. the desirable behaviour, and leaves the verification up to the machine. If an error is recognized the tool provides a counter-example showing under which circumstances the error can be generated. This allows the user to locate the error and to repair the model specification before continuing. If no errors are found, the user can refine its model description e.g. by taking more design decisions into account so that the model becomes more concrete and can restart the verification process.

3.4 Correctness: Safety and liveness properties

Safety property is an invariant which asserts that "something bad does never happen", that is an acceptable state of the system is maintained. For example, a property which assures that a power reactor temperature would never exceed 100 degree Centigrade etc. Magee and Kramer in [21] have defined safety property S = {a1, a2 … an} as a deterministic process that states that a trace consisting of the actions in the alphabet of S, is accepted by S. ERROR conditions are like exceptions which present the states that are not required. In complex systems safety properties are specified by directly specifying what is required.

Liveness property states the "something good happens" that shows and specifies the states of system that can be brought about by an agent under certain given conditions [21]. The work by [17][18][19][20] have used LTS for the verification of correctness properties in multi-agent based robotic systems.

3.5 Labelled Transition System (LTS)

LTS is a collection of techniques for the automated formal verification of finite-state concurrent systems. It consists of interacting finite state machines along with their properties; it performs compositional analysis to exhaustively search for violations of the required properties. Each component of a specification is described as LTS, which has all the possible states a component can reach and all possible transitions it can perform.

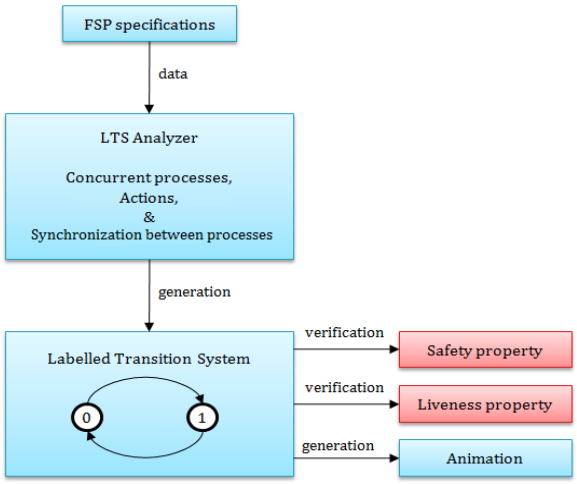

Figure 1: **LTS Analyzer takes FSP as input**

FSP is a process algebra notation having finite state processes used for the concise description of component behaviour particularly for concurrent systems. It is an implementation of formal methods that provides construct to formalize specifications of software components and architecture. Each component consists of processes; each process has a finite number of states and is composed of one or more actions. There exists concurrency between elementary calculator activities for which there is a need to manage the interactions, communication and synchronization between processes. Magee and Kramer [21] have proposed an analysis tool LTS Analyzer for FSP notation.

.

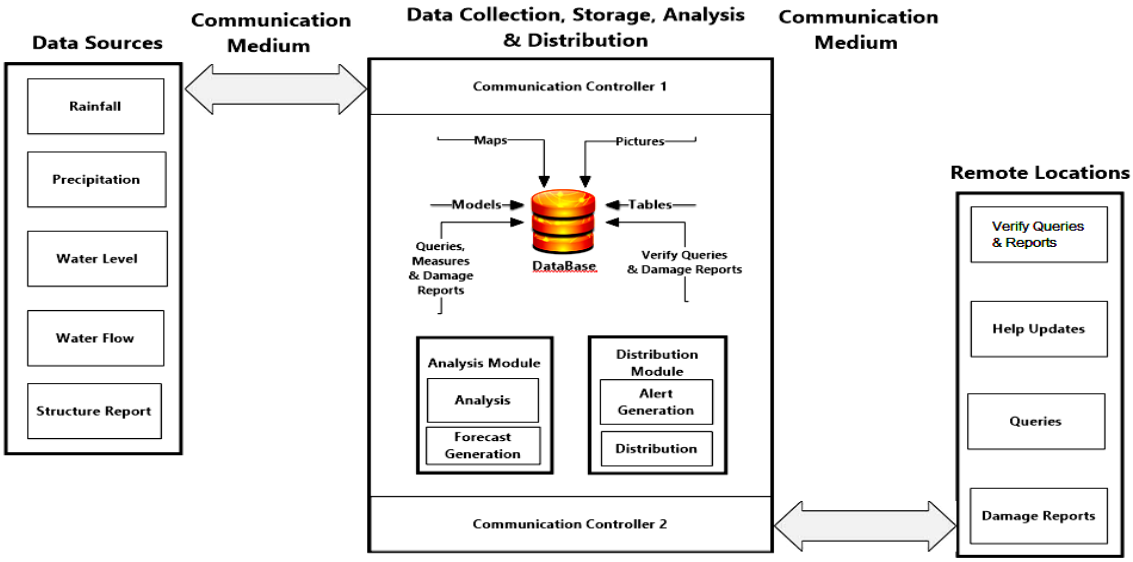

Figure 2: High-Level Architecture Specifications of Smart Flood Monitoring SoS

4. PROPOSED SYSTEM - SMART FLOOD MONITORING SOS

4.1 High-level System Architecture

High-level system architecture presents the system elements at a higher level of abstraction. It gives a well-defined picture of the SoS parts and how they fit together. The high-level architecture of the Smart Flood Monitoring SoS is shown in Figure 2

Data Sources collect the elements that are vital for flood monitoring. These measures include rainfall, precipitation, water level, water flow and structure report. On getting a request from Communication Controller 1, these measures are then sent to Data Storage through a communication medium. This weather and flood related data can be in various forms. It may include maps, pictures, models and tables. Here the stored data is further analyzed to produce results and forecasts. If the forecast is found to be for a flood warning, alerts are generated and distributed to the remote location through communication medium. These remote locations include Emergency Responders, Community and Independent Observers. Data collection, storage, analysis and distribution centres also receive and store data from remote locations in the form of queries help measures, damage reports and verification results. This data is used for evaluating the performance of the constituent components and finally the overall performance of system. Communication Controller 2 helps to maintain the communication with the Remote Locations.

4.2 Structural Architecture

This phase takes a transition from high-level architecture to a detailed structural architecture (i.e. components and interfaces) of the SoS.

The interactions and dependencies of components and interactions through interfaces with one another are also presented. Component diagram represents the structural architecture of the SoS. All components are put together and the UML component diagram is shown in Figure 3. External interfaces to interact with the autonomous components are highlighted in red for clarity.

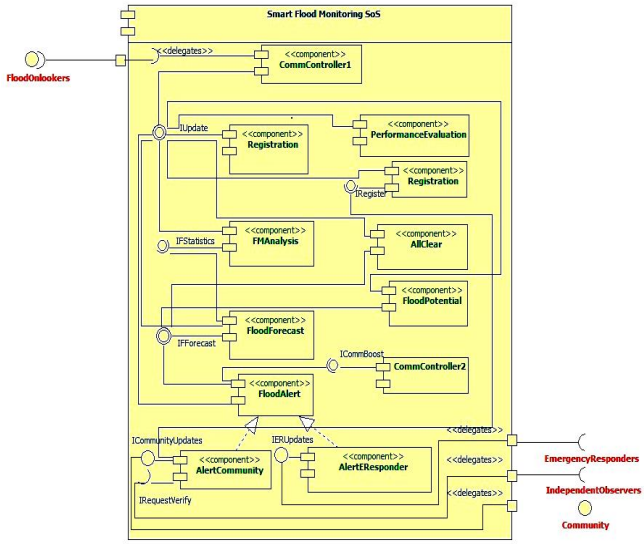

Figure 3: UML Component Diagram for Smart Flood Monitoring SoS

4.3 Formal Behaviour Modelling of Smart Flood Monitoring SoS

Behavioural model of Smart Flood Monitoring SoS highlights the system functionality and actions. It is critical in situations when a system is to be observed for its response to user requests, its interaction with other systems etc. Further, behavioural modelling helps to assess, verify SoS correctness properties and validate the system. It also serves to bridge the gap and makes a smooth transition from analysis to implementation.

The proposed Smart Flood Monitoring SoS is a distributed system. The objective is to design and develop a formal correct system. Formal modelling aids to check the system behaviour and ensure the correctness of the system.

Smart Flood Monitoring SoS is divided into four components: *Data Collection*, *Flood Analysis*, *Alert Community*, *Alert Emergency Responders*. CPN models are constructed for each component. The behavioural modelling and analysis is done by adopting the following

steps: constructing CPN models, running and simulating the models in CPN tool, entering the state space tool to analyze the SoS correctness properties (i.e. safety property, liveness property), and if some error is found, analyzing to correct it.

State space tool is applied to analyze the models for bounded-ness, home, liveness properties. These properties answer a number of the most important inquiries made as a result of the state space analysis.

Bounded-ness Property: How many and which tokens a place may hold?

Home Property: Does there exist a single home marking? i.e. the marking that can be reached from any reachable marking.

Liveness Property: Are all transitions live and can be enabled again?

The following section presents and discussed the CPN model for one of the module of Smart Flood Monitoring SoS.

The CPN model for **Flood Monitoring Analysis** is constructed as shown in figure-4. To perform analysis, the flood data is requested from database using RQ variable of the color set STRING. After retrieving the required data represented here by the variable FD of the color set FData, analysis is performed. Analysis results are represented by flood statistics FS from the color set FStats. FS contains the analysis results that may be one of these three: All Clear, Flood Potential, Flood Warning. If the analysis result into All Clear results, then the system is in Normal Operation state; in case of Flood Potential, Flood Prevention state is reached; and if the analysis indicates a Flood Warning, then system further generates a Forecast about the flood. Any action that is taken in Normal Operation, Flood Prevention or Flood Warning is also recorded in the Database.

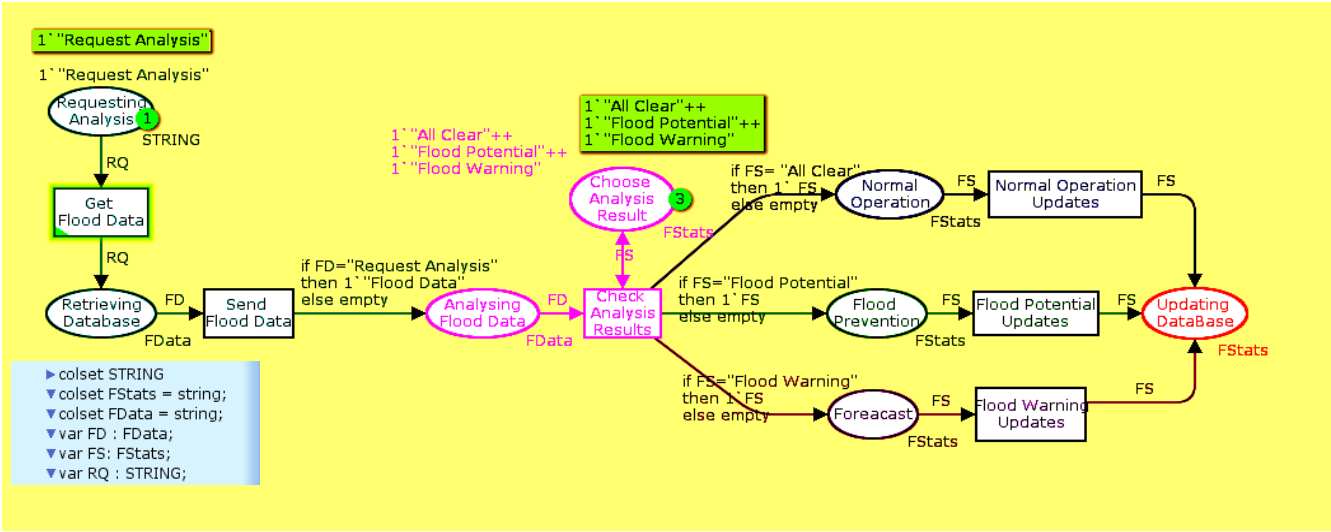

Figure 4: CPN model for Flood Monitoring Analysis at start with tokens

To make a choice for possible analysis results, transition is bound manually as shown in Figure 5. Binding is chosen before firing the transition.

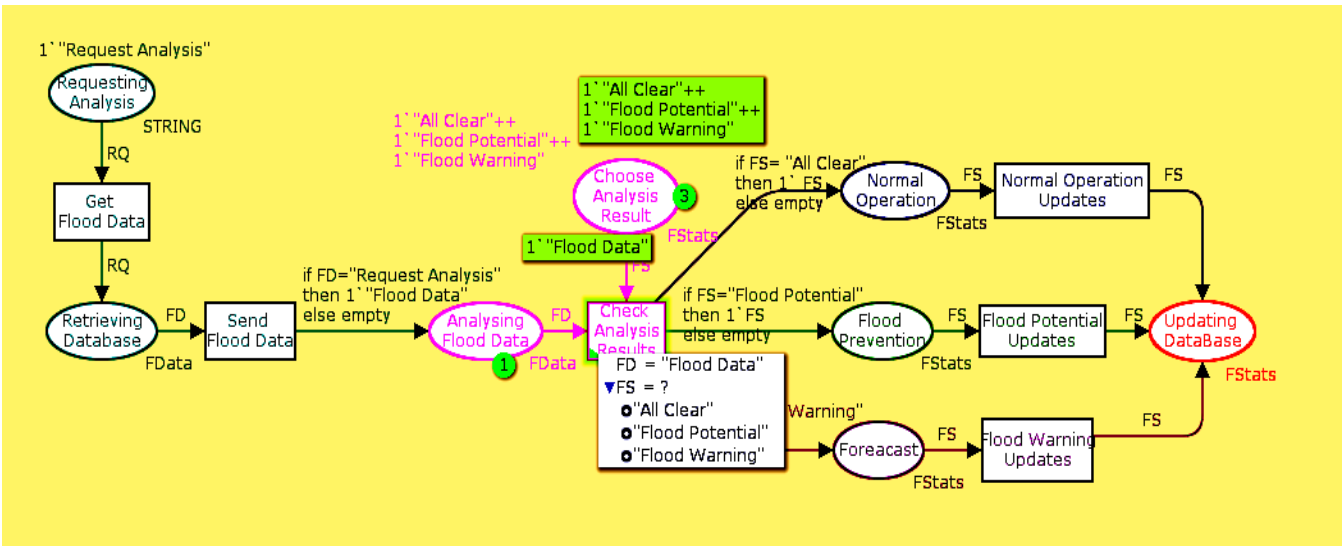

Figure 5: CPN model for Flood Monitoring Analysis during Simulation (Binding the Transitions Manually)

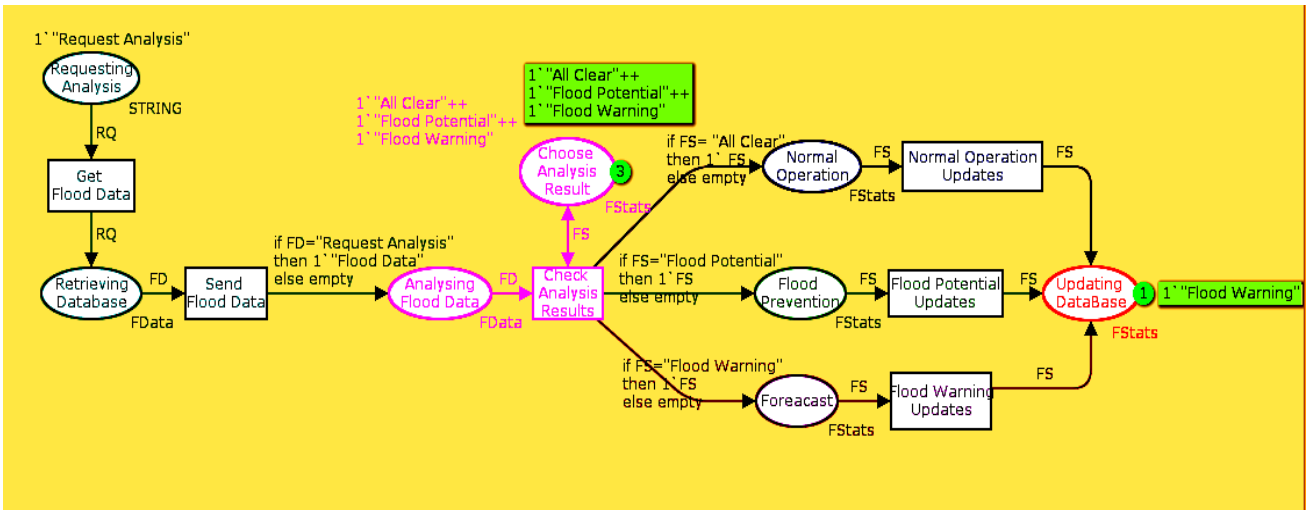

Figure 6: CPN model for Flood Monitoring Analysis after Simulation Completion

Figure 6 shows the final simulation results. It shows the color sets and variables used in the CPN model for Flood Monitoring Analysis. Color set STRING is used to model the request for flood related data. Color set FData and FStats represent the flood data before and after the analysis respectively. FD and FS are the variables of the color sets FData and FStats. While RQ is a variable from color set STRING. The state space analysis of the CPN for Flood Monitoring Analysis is done by using CPN Tools and generating state space report.

4.4 Safety property verification of Smart Flood Monitoring SoS

The CPN model presented above verify some behavioural properties of the system. These properties are boundedness, home, liveness and fairness. It is important to verify the correctness property of safety. Liveness properties are checked and presented in state space analysis report of the CPN. The safety properties of the SoS are checked by LTS based model checking. Model checking verifies the proposed system behaviour for safety property. Properties are specified in FSP syntax consisting of sequence of states, set of action labels and transition relations, FSP in turn generates a Labelled Transition System (LTS). LTSA [21] analyzes and verifies these properties. Table-1 shows the safety property Flood Monitoring Analysis of Smart Flood Monitoring SoS.

Table 1: Safety Property

Safety Property	Flood warning state is a mutually exclusive state and system cannot be in Normal Operation or Flood Potential with Flood Warning at the same time.

FSP	`property NORMAL_OPERATION = ( processData -> ANALYSIS_RESULT ),`
	`// If analysis result does not indicate any flood situation, then normal`
	`// operation continues. And in the case of a flood situation, system is`
	`// in alert state.`
	`ANALYSIS_RESULT = ( flood_warning -> FLOOD_ALERT`
	`| no_flood_warning -> NORMAL_OPERATION ),`
	`FLOOD_ALERT = ( alertGeneration -> NORMAL_OPERATION ).`
	`// After analysis result system performs corresponding activities, system`
	`// repeats the Normal Operation to track the updates`

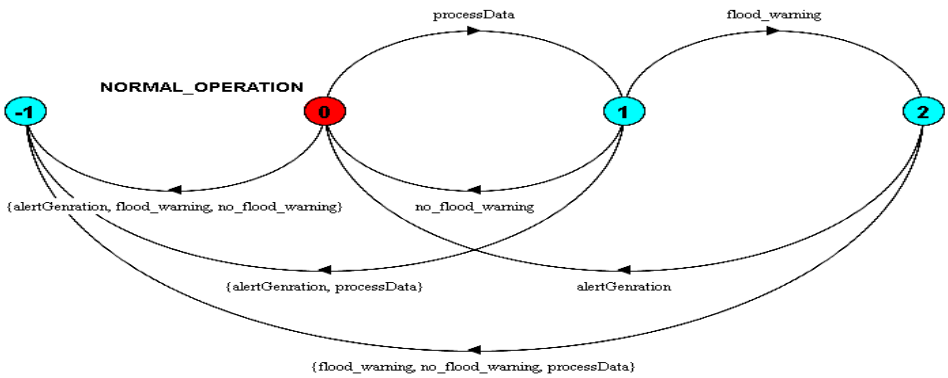

Figure 7: LTS for Flood monitoring analysis

5. CONCLUSION AND FUTURE WORK

A smart flood monitoring SoS has its importance in not only to evacuate the victims but also to reduce the damages by providing rescue and help services. Our proposed system will save the human lives, infrastructure and economy.

The Smart Flood Monitoring SoS is a system that has been designed and modelled using formal methods to ensure correctness properties in each step. The emphasis is on keeping the communication up which ensures that rescue services remain available during the disaster. Our proposed model addresses the SoS challenges involved in system development, system working, and system evolution. Architecture as well as design has been proposed covering both static as well as dynamic aspects of the system architecture. Formal verification and model checking is adopted to verify the correctness properties of the specified system. The elicited requirements are evolved towards the high-level design in the form of workflow diagrams. This design is further refined to develop architecture of the Smart Flood Monitoring SoS.

Formal verification of the SoS ensures correctness properties. CPN models with the timing constraints are constructed, and model checking is done by specifying the safety properties as FSP and generating LTS (Labelled Transition System). The long term objective, after the system gets implemented, is achieving the operational excellence in community services. Thus, reducing damages caused by floods.

REFERENCES

[1] M. MacDonald, "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] B. Blanchard and W. Fabrycky, "Systems Engineering and Analysis", 3rd Edition. Prentice Hall, (1998).

[3] B. Berthomieu and M. Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri nets", *IEEE Trans. Software Engineering*, **17**(3): 259-273(1991).

[4] P. M. Merlin and D. J. Farber, "Recoverability of Communication Protocols", *IEEE Trans. Communications*, **24**(4): 1036-1043(1976).

[5] G. Denaro and M. Pezze, "Petri nets and Software Engineering", *Lecture Notes in Computer Science (LNCS)*, **3098**: 439-466(2004).

[6] X. He and T. Murata, *High-level Petri nets Extensions, Analysis, and Applications*, Electrical Engineering Handbook, Ed. Wai-Kai Chen, Elsevier Academic Press, pp. 459-476(2005).

[7] H. Xudong, "A Comprehensive Survey of Petri Net Modeling In Software Engineering", *Int. Journal of Software Engineering and Knowledge Engineering*, **23**(5): 589-625(2013).

[8] K. Jensen, "Colored Petri Nets Basic Concepts, Analysis Methods and Practical Use", vol. **1** (1992).

[9] CPN Tools website, 2012. [Online]. Available: http://www.cpntools.org. [Accessed: June 27, 2015].

[10] K. Jensen, L. M. Kristensen, J. L. Wells, "Coloured Petri Nets and CPN Tools for Modeling and Validation of Concurrent Systems", *Int. Journal on Software Tools for Technology Transfer (STTT)*, **9**(3): 213-254(2007).

[11] L. Wells, *Performance Analysis Using Coloured Petri Nets*, PhD [Dissertation]. Dept. of Comp. Sc.: Univ. of Aarhus, 2002.

[12] J. P. Quielle, J. Sifakis, "Specification and verification of concurrent systems in CESAR". Proceedings of the 5th International Symposium on Programming, pp. 337-350 (1982).

[13] E. M. Clarke, O. Grumberg, D. Peled, "Model Checking". MIT press, (1999).

[14] E. M. Clarke, E. A. Emerson, A. P. Sistla, "Automatic verification of finite state concurrent systems using temporal logic specifications". ACM transactions Prog. Lang. Syst., **8**(2): 244-263(1986).

[15] E. M. Clarke, O. Grumberg, D. E. Long, "Model checking and abstraction". ACM Transactions Prog. Lang. Syst., **16**(5):1512-1542(1994).

[16] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, "Counter example-guided abstraction refinement for symbolic Model Checking". Journal ACM, **50**(5): 752-794(2003).

[17] N. Akhtar, "Requirements, Formal Verification and Model transformations of an Agent-based System: A Case Study", *Computer Engineering and Intelligent Systems (IISTE)*, **5**(3): 1-16 (March 2014).

[18] N. Akhtar and M. M. S. Missen, "Contribution to the Formal Specification and Verification of a Multi-Agent Robotic System", *European Journal of Scientific Research (EJSR)*, **117**(1):35-55(January 2014).

[19] N. Akhtar and M. M. S. Missen, "Practical Application of a Light-weight Formal Implementation for Specifying a Multi-Agent Robotic System", *Int. Journal of Computer Science Issues (IJCSI)*, **11**(1): 247-255(January 2014).

[20] N. Akhtar, Yann Le Guyadec, and Flavio Oquendo, "Formal Specification and Verification of Multi-Agent Robotics Software Systems: A Case Study", *In Proc. of Int. Conf. on Agents and Artificial Intelligence (ICAART'09)*, pp. 475-482 (January 2009).

[21] J. Magee and J. Kramer, "Concurrency - State Models and Java Programs", *John Wiley and sons*, 2nd edition, pp. 1-223 (2006).