Formal Architecture and Verification of a Smart Flood Monitoring System-of-Systems

Nadeem Akhtar[1] and Saima Khan[2]
[1]Department of Computer Science and IT, the Islamia University of Bahawalpur, Pakistan
[2]Department of Computer Science Faculty of Computer Science and Information Technology, Virtual University of Pakistan, Pakistan

Abstract: *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 System-of-System 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 to ensure the correctness properties of safety and liveness.*

Keywords: *Flood monitoring; system-of-systems; behavioral modeling; formal verification; correctness; safety property.*

Received September 15, 2015; accepted June 1, 2016

1. Introduction

Natural disasters affect millions of people every year around the globe. 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. Most of the times we are able to anticipate the flood and prepare the evacuation plan. But sometimes there are flash floods that develop instantly. When a flood threat becomes a reality, infrastructure is destroyed, and people are left homeless to fight with water borne diseases. A flood in 2010 affected more than 20 million people. Almost 2000 people were reported dead. People were dislocated; crops and properties were ruined [16].

Advanced forecasts and predictions based on flood data help to inform with accuracy about the possibility and intensity of the flood. Smart Flood Monitoring System-of-Systems (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. It also keeps 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.

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 existing research work, there is no application of System-of-Systems in flood monitoring. The problem statement is to propose a System-of-System based Smart Flood Monitoring system that is formally designed and verified. A System-of-System that is correct and reliable; monitor floods, provides flood avoidance services; and facilitates rescue services

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 [5] 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 [14] presented the concepts and the applications of CPNs in system modeling. CPN Tools [11] allow construction of CPN models with timing constraints. This tool also allows simulating system behavior and verifying the system by using state space analysis [15]. The state space analysis performed by CPN Tools facilitates to analyze some standard properties including bounded-ness, home, liveness and fairness [20].

Berthomieu and Diaz [6], Merlin and Farber [18] 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 [12]; He and Murata [13] have presented a good review of various new developments and applications of Petri-Nets in software engineering. Xudong [21] 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 [7, 8, 9, 10, 19] 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 behavior. Model checking basic idea is to use algorithms executed by software tools to verify the correctness of the system.

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 [17] 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 [17]. The work by [1, 2, 3, 4] have used LTS for the verification of correctness properties in multi-agent based robotic systems.

3.5. Labelled Transition System (LTS)

Magee and Kramer [17] have proposed an analysis tool LTS Analyzer for FSP notation. 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. FSP is a process algebra notation having finite state processes used for the concise description of component behavior particularly for concurrent systems. 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.

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 as shown in Figure 1.

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 centers 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.

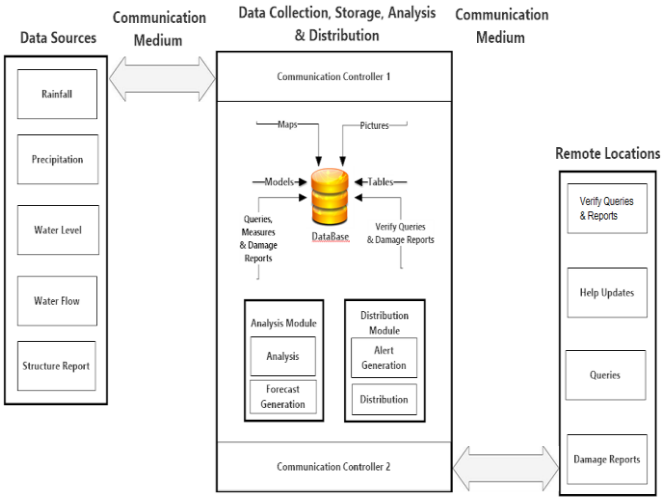

Figure1. High-level architecture of smart flood monitoring SoS.

4.2. Structural Architecture

The high-level architecture is refined into a detailed structural architecture (i.e., components and interfaces). The interactions and dependencies of components and interactions through interfaces with one another are specified. Component diagram represents the structural architecture of the SoS. All components are put as shown in Figure 3. External interfaces to interact with the autonomous components are highlighted in red.

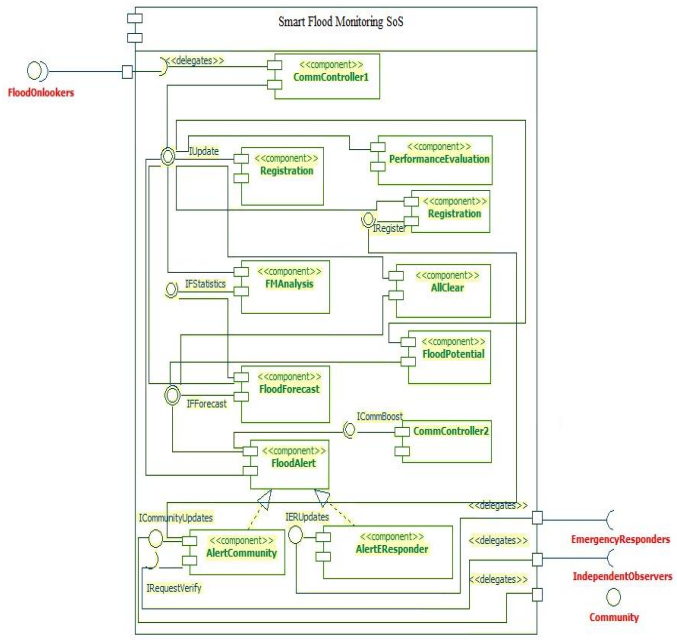

Figure 2. Structural architecture of smart flood monitoring SoS.

4.3. Formal Behavior Modeling of Smart Flood Monitoring SoS

Behavioral 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, behavioral modeling 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 objective is to design and develop a formal correct system. Formal modeling aids to check the system behavior 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 behavioral modeling 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).

State space analysis is performed to analyze the models for bounded-ness, home, liveness properties. These properties answer a number of 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 CPN model for Flood Monitoring Analysis is shown in Figure 3. 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 recorded in the Database.

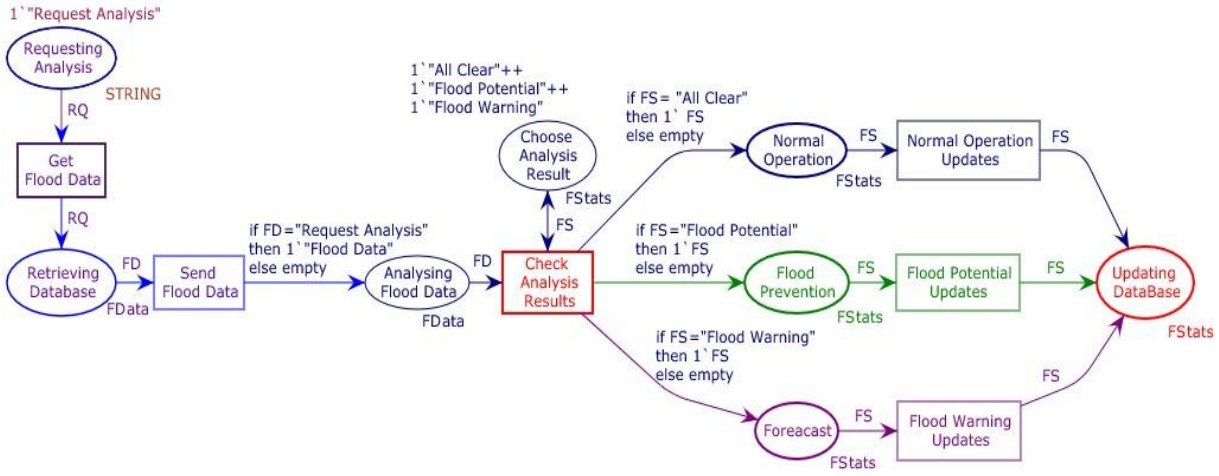

Figure 3. CPN model for flood monitoring analysis at start with tokens.

To make a choice for possible analysis results, transition is bound manually as shown in Figure 4.

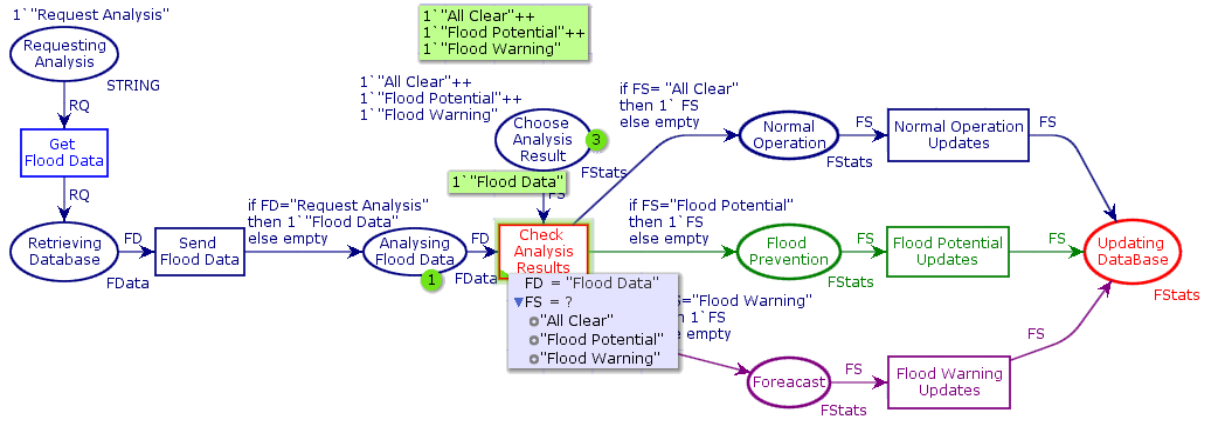

Figure 4. Manual binding of transitions during simulation.

Figure 5 shows the final simulation results along with 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 CPN state space report is generated for state space analysis.

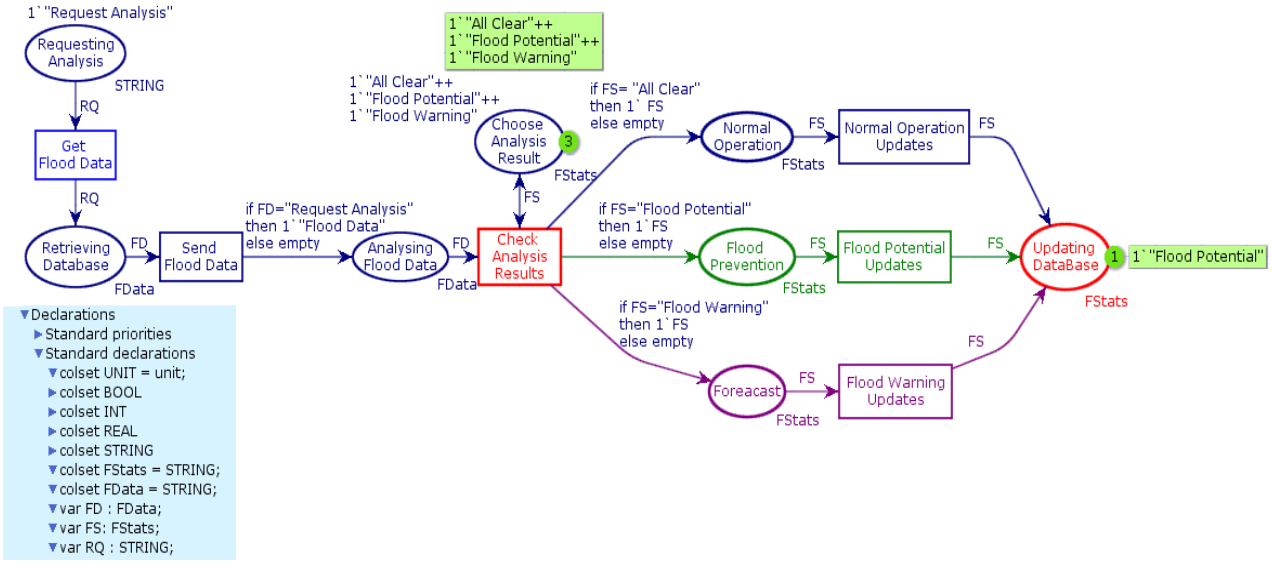

Figure 5. CPN model for flood monitoring analysis after simulation completion.

4.4. Safety Property Verification of Smart Flood Monitoring SoS

The CPN model presented above specify behavioral properties of the system. These properties are boundedness, home, liveness and fairness. It is important to verify the correctness property of safety. The safety properties of the SoS are checked by LTS based model checking. Properties are specified in the form of LTS consisting of sequence of states, set of action labels and transition relations. Table-1 shows one of the safety properties.

Table 1. Safety property of smart flood monitoring SoS.

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).
	// Whatever is the analysis result (Normal or a Warning), after performing
	// corresponding activities, system returns to Normal Operation to track the updates.

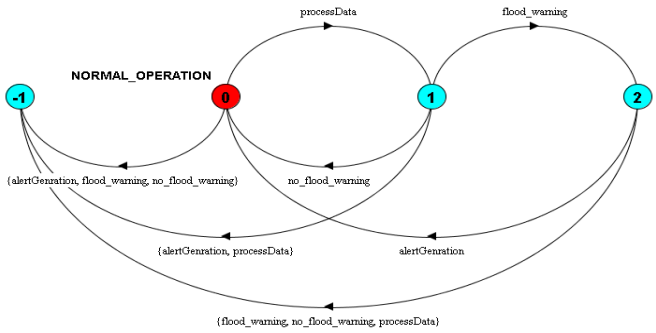

Figure 6. LTS for flood monitoring analysis.

5. Contribution

Our contributions in the analysis, design and verification of a smart flood monitoring, warning and rescue system are:

1. The proposed Smart Flood Monitoring SoS 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.
2. The proposed 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. Our approach provides formal verification of correctness properties by using LTS model-checking.

6. Conclusions and Future Work

Smart Flood Monitoring SoS has its importance in flood forecast as well as to reduce the damages by providing rescue and help services. Our proposed system will save human lives, infrastructure and economy.

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 Labelled Transition System (LTS).

Our Smart Flood Monitoring SoS has been designed using formal methods to ensure correctness properties in every phase. The emphasis is on keeping the communication up which ensures that rescue services remain available during the disaster. The proposed architecture covers both the structural as well as behavioral aspects of the system. The elicited requirements are evolved towards the high-level architecture in the form of workflow diagrams. This architecture is further refined in to detailed architecture. Formal verification is adopted to verify the correctness properties of the specified system.

References

[1] Akhtar N., "Requirements, Formal Verification and Model Transformations of an Agent-based System: A Case Study," *Computer Engineering and Intelligent Systems*, vol. 5, no. 3, pp. 1-16, 2014.

[2] Akhtar N. and Missen M., "Contribution to the Formal Specification and Verification of a Multi-Agent Robotic System," *European Journal of Scientific Research*, vol. 117, no. 1, pp. 35-55, 2014.

[3] Akhtar N. and Missen M., "Practical Application of a Light-weight Formal Implementation for Specifying a Multi-AgentRobotic System," *International Journal of Computer Science Issues*, vol. 11, no. 1, pp. 247-255, 2014.

[4] Akhtar N., Guyadec Y., and Oquendo F., "Formal Specification and Verification of Multi-Agent Robotics Software Systems: A Case Study," *in Proceedings of 1st International Conference on Agents and Artificial Intelligence*, Porto, pp. 475-482, 2009.

[5] Blanchard B. and Fabrycky W., *Systems Engineering and Analysis*, Prentice Hall, 1998.

[6] Berthomieu B. and Diaz M., "Modeling and Verification of Time Dependent Systems Using Time Petri Nets," *IEEE Transactions on*

Software Engineering, vol. 17, no. 3, pp. 259-273, 1991.

[7] Clarke E., Grumberg O., and Peled D., *Model Checking*, MIT press, 1999.

[8] Clarke E., Emerson E., and Sistla A., "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.

[9] Clarke E., Grumberg O., and Long D., "Model Checking and Abstraction," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 5, pp. 1512-1542, 1994.

[10] Clarke E., Grumberg O., Jha S., Lu Y., and Veith H., "Counter Example-Guided Abstraction Refinement for Symbolic Model Checking," *Journal ACM*, vol. 50, no. 5, pp. 752-794, 2003.

[11] CPN Tools website, 2012. [Online]. Available: http://www.cpntools.org. Last Visited, 2015.

[12] Denaro G. and Pezze M., "Petri Nets and Software Engineering," *Lecture Notes in Computer Science*, Berlin, pp. 439-466, 2004.

[13] He X. and Murata T., *High-level Petri nets Extensions, Analysis, and Applications*, Elsevier Academic Press, 2005.

[14] Jensen K., *Colored Petri Nets Basic Concepts, Analysis Methods and Practical Use*, Springer-Verlag, 1997.

[15] Jensen K., Kristensen L., and Wells J., "Coloured Petri Nets and CPN Tools for Modeling and Validation of Concurrent Systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3, pp. 213-254, 2007.

[16] MacDonald M., *Guidelines for Climate Compatible Construction and Disaster Risk Reduction in Rural Punjab*, 2013. [Online]. Available: http://cdkn.org/wp-content/uploads/2012/09/Climate-Compatible-Construction-Guidelines_Final.pdf, Last Visited, 2014.

[17] Magee J. and Kramer J., *Concurrency-State Models and Java Programs*, John Wiley and Sons, 2006.

[18] Merlin P. and Farber D., "Recoverability of Communication Protocols," *IEEE Transactions on Communications*, vol. 24, no. 4, pp. 1036-1043, 1976.

[19] Quielle J. and Sifakis J., "Specification and Verification of Concurrent Systems in CESAR," *in Proceedings of the 5th International Symposium on Programming*, Berlin, pp. 337-350, 1982.

[20] Wells L., Performance Analysis Using Coloured Petri Nets, PhD, University of Aarhus, 2002.

[21] Xudong H., "A Comprehensive Survey of Petri Net Modeling in Software Engineering," *International Journal of Software Engineering and Knowledge Engineering*, vol. 23, no. 5, pp. 589-625, 2013.

Nadeem Akhtar is working as Assistant Professor at the Department of Computer Science & IT, The Islamia University of Bahawalpur (IUB), PAKISTAN. He has a PhD from the Laboratory VALORIA of Computer Science, University of South Brittany (UBS), FRANCE with the highest honour "Tres Honorable". He has aMasters with specialization in Information System Archite Institut Universitaire Professionnalisé Vannes, FRANCE. He is the recipient of a number of awards, scholarships and research grants i.e. Study in France 2004 French Embassy scholarship for Master studies, HEC in France, Teaching assistant for ENSIBS=million, research award in year 2014 from The Directorate of Research and Development, The Islamia University of Bahawalpur. His multi-agent systems, system-of-systems and self.

Saima Khan is pursuing her MS Her research work is focused on presenting and formally verifying a SoS based model for Smart Flood Monitoring. She studied MCS from Quaid working as an Assistant Professo Women Rahim Yar Khan, Pakistan for the last 4 years. She has 12 years of teaching experience and one year experience of software development and quality assurance. She has received many certificates and medals for throughout an outstanding academic career. Above all is the President Award given to her by the President of Islamic Republic of Pakistan for her outstanding academics. Aside from teaching, she is determined for education reforms and character building of the students. Her interests include: Software Engineering, Databases, Project Management, Analysis and Design Tools.