A Model Driven Framework for Modeling and Verification of Underwater Wireless Sensor Networks

Author

Ayesha Tariq

Regn Number

FALL 2017-MS-17(CSE) 00000203649

MS-17 (CSE)

Thesis Supervisor:

Dr. Farooque Azam

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

August, 2021

A Model Driven Framework for Modeling and Verification of Underwater Wireless Sensor Networks

Author

Ayesha Tariq

FALL 2017-MS-17(CSE) 00000203649

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Software Engineering

Thesis Supervisor:

Dr. Farooque Azam

Thesis Supervisor's Signature: ____

DEPARTMENT OF COMPUTE ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

August, 2021

DECLARATION

I certify that this research work titled “A Model Driven Framework for Modeling and Verification of Underwater Wireless Sensor Networks “is my own work under the supervision of Dr. Farooque Azam. This work has not been presented elsewhere for assessment. The material that has been used from other sources; it has been properly acknowledged / referred.

Signature of Student

Ayesha Tariq

FALL 2017-MS-17(CSE) 00000203649

LANGUAGE CORRECTNESS CERTIFICATE

This thesis is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the University for MS thesis work.

Signature of Student

Ayesha Tariq

FALL 2017-MS-17(CSE) 00000203649

Signature of Supervisor

COPYRIGHT STATEMENT

ACKNOWLDGEMENTS

I am highly thankful to ALLAH Almighty for his blessings and constant help throughout my thesis. Indeed, this would not have been possible without his bountiful and gracious help in each and every step. I am thankful to HIM for putting me on a path where people helped me graciously to achieve my goal. Indeed, none is worthy of praise but ALLAH Almighty.

I am profusely grateful to my beloved mother for her constant love, support, prayers and sacrifices. I would like to pay my special thanks to my beloved mother, for encouraging me to avail best opportunities in my life. I am also thankful to my siblings for all the support and prayers throughout my time of research.

I would also like to express my gratitude to my supervisor **Dr. Farooque Azam** and my cosupervisor **Dr. Wasi Haider Butt** for their constant motivation, patience, enthusiasm, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my MS study.

I would also like to pay special thanks to my Guidance Committee Members **Dr. Arsalan Shaukat and Dr. Urooj Fatima**. Their recommendations are very valued for improvement of the work. I would like to pay special thanks to **Muhammad Waseem Anwar** for his tremendous support and cooperation. Each time I got stuck in something, he came up with the solution. Without his help, I wouldn’t have been able to complete my thesis. I appreciate his patience and guidance throughout the whole thesis.

I would also like to thank my family, my siblings, my class fellows and my seniors for their support and cooperation. Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Thanks for all your encouragement!

Dedicated to my beloved mother whose tremendous motivation and support helped me to accomplish this achievement

ABSTRACT

The domain of underwater wireless sensor networks (UWSNs) has recently captured the significant attention of researchers from both industry and academia due to its substantial enhanced capabilities in ocean surveillance, marine tracking, and device deployment for detecting underwater targets. However, wide-scale implementation of UWSNs is a long way off for a number of reasons. One of the most critical reasons is that ocean-centric applications are both expensive and time-consuming to deploy offshore and to test in the field. Therefore, verification of network requirements prior to actual deployment requires a quick and reusable solution. Although there are tools (simulators) for the analysis of UWSNs infrastructure, protocol and algorithms, these operate on lower abstraction level with higher complexities where knowledge of low-level code is essential. On the other hand, Model-Driven Engineering (MDE) is system development approach to provide higher level of abstraction. Although it has been utilized in the domain of UWSNs for system level design, the early analysis and verification of system design is usually performed at lower abstraction level in isolation. As a result, a major gap between design and verification has emerged significantly. To fill this gap, a user-friendly and higher abstraction layer is required that allows design and verification aspects to be modelled at the same level as system design.

In order to bridge this gap, this thesis proposes a Model-Driven Framework for Underwater Wireless Sensor Networks (MFUWSN). Particularly, it permits the modeling of system design and verification aspects of UWSNs at higher abstraction level altogether. To achieve this, a UML Profile for Underwater Wireless Sensor Networks (UPUWSN) is developed to model design and verification aspects of UWSNs. A transformation engine named “UWSN Transformation Engine (UWSNTE)" has been developed as part of the research to automatically transform high-level MFUWSN source models into low-level C/C++ code. As a result, the AquaSim NS3 simulator can be used to do early analysis and verification of UWSNs design in the early stages of development. The applicability of framework is established through two benchmark case studies i.e., Monitoring Offshore Oil & Gas Reservoirs and Smart Cites Underwater. The results indicate that the proposed framework significantly simplifies the system design and verification of UWSNs as both design and verification aspects are managed altogether at higher abstraction level.

Keywords: Wireless Sensor Networks, Underwater Simulator, Network Verification, UWSN, Unified Modeling Language (UML), UML Profile, Model-to-Text Transformation (M2T), Model-Driven Engineering (MDE)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

CHAPTER 1: INTRODUCTION

This chapter provides the detailed introduction of the research which is categorized in different sections. The background study is documented in **Section 1.1**, problem statement in **Section 1.2**, proposed methodology in **Section 1.3**, research contribution in **Section 1.4** and thesis organization in **Section 1.5.**

1.1. Background Study

The objective of this section is to introduce the background study of multiple concepts that has been used in this research. These concepts include:

- Underwater Wireless Sensor Network
- Underwater Wireless Sensor Network Architecture
- Underwater Wireless Sensor Network Challenges
- Model-Driven Engineering

1.1.1. Underwater Wireless Sensor Network

Wireless sensor networks (WSNs) have a lot of potential for monitoring aquatic environments since they can sense, collect, and transmit data wirelessly in real time to users. It has indirectly resulted in the establishment of UWSNs, a new paradigm in wireless sensor technology (UWSNs) [50]. UWSNs (Underwater Wireless Sensor Networks) are a hybrid of wireless technology and compact micromechanical sensor equipment with intelligent sensing, processing, and smart communication capabilities. Underwater wireless sensor networks (UWSNs) enable a wide range of underwater exploration applications for scientific, environmental, and military purposes [1]. Sensor nodes, controllers, memory, and power supplies are all included in UWSNs. Sensor nodes are spatially dispersed underwater in UWSNs to monitor variables including pressure, conductivity, turbidity, water quality, and temperature. The data collected can then be used in a variety of underwater applications such as coastal monitoring, pollution monitoring, military protection, water-based disaster prevention, and underwater resource exploration, among others [2]. Sensor nodes, which can be stationary or mobile, communicate with one another via wireless communication units [3]. Transceivers are utilized in UWSNs to transmit collected data to the sinks at the surface (Sonobuoys). The Sinks use two modes of communication: acoustic and radio, to collect data from sensor nodes via acoustic communication and then communicate it to tracking stations via radio communication. Underwater monitoring is difficult due to the characteristics of acoustic communication, which include a long propagation delay [4], a high packet loss probability, high energy consumption, low dependability, and limited link capacity and bandwidth [5].

However, wide-scale deployment of UWSNs, is a long way off for a variety of reasons. One of the most prominent reasons is that offshore deployment and field-level testing of ocean-centric applications are both costly and time-consuming. These networks are becoming complex which makes their development and verification more difficult than before.

1.1.2. Underwater Wireless Sensor Network Architecture

As depicted in Figure 1.1, UWSNs are made up of multiple components: an onshore sink, a surface buoy, an underwater sink node, and underwater sensor nodes. Satellites, vessels, and autonomous underwater vehicles (AUVs) can also be utilized to extend the range of detection and communication. Pressure, sound, temperature, and other physical or environmental factors are monitored by underwater sensor nodes, which collectively relay data to the underwater sink node. The data is transmitted through wired or wireless link to a surface buoy, and then received via radio communication at an onshore sink or surface sink. The architectures of UWSNs can be categorised in a variety of ways. Static, semi-mobile, and mobile architectures are all classified differently. Another common way to categorize UWSNs is to divide them into one-dimensional, two-dimensional, three-dimensional, and four-dimensional categories. UWSNs can be single-hop, multi-hop, or hybrid in nature (single-hop individual sensors, multi-hop clusters). Short-term, time-critical applications and long-term, non-time-critical applications are two types of architectures [51].

In **one-dimensional architecture** underwater sensor nodes are spread autonomously. Each sensor node is a self-contained network that can collect, process, and transmit data to the base node. The nodes can be deployed for a prolonged period of time in this case to reflect data back to the base station. In this architecture, sensor nodes communicate by radio frequency (RF), optical fiber transmission, or acoustic communication. Star topology is the most commonly used topology in this architecture.

Sensor nodes are organized together as “clusters” in **two-dimensional architecture**. The network of sensor nodes has a cluster head, just like any other cluster network (anchor node). Data is collected by each node in the cluster and sent back to the cluster head. A horizontal communication link connects all nodes in the cluster to the cluster head, while the cluster head delivers data back to the base station through a vertical communication link. Depending on the deployment scenario, communication between nodes is accomplished using RF, optical fiber transmission, or acoustic signal. The most common topologies are star, ring, and mesh.

Sensors are clustered in a **three-dimensional architecture** to collect data at various depths below sea level. The sensor nodes' connection progresses to the next level of 2D architecture. Inter-cluster communication among sensor nodes deployed at varying depths for data acquisition; intra-cluster communication among sensor nodes and cluster head (anchor node); and anchor–buoyant communication between cluster head and remote base node are the three types of communication in three-dimensional architecture. Depending on the type of data capture and deployment scenario, communication is done via RF, optical fiber, or acoustic signal in this architecture. The most common topologies are star, mesh, and ring.

The integration of static UWSN architecture, three-dimensional architecture, and mobile UWSN is known as **four-dimensional architecture**. Underwater autonomous vehicles (UAVs) are used in this scenario to collect data from underwater sensor nodes and return it to the base station. Submarines, underwater vehicles, robotics, and even miniature underwater operational ships are all examples of UAVs. The distance between the UAV and the base station nodes determines the type of communication used, which might be RF or acoustic [52].

1.1.3. Underwater Wireless Sensor Network Challenges

Propagation delay, poor sound speed, poor channel quality, inadequate bandwidth, packet loss, multi path delay, and other issues and challenges faced by UWSNs. Only a few of the most important challenges are listed below. *Bandwidth restriction:* The bandwidth is strictly limited. Underwater networks present a significant problem in this regard. *Devices cost:* When compared to other wireless networks, the cost of devices used in underwater sensor networks is relatively expensive. Additional protection techniques are necessary to protect hardware devices, which increases network costs [53]. *Propagation delay:* When compared to terrestrial sensor networks, the underwater sensor community has a lower propagation latency.

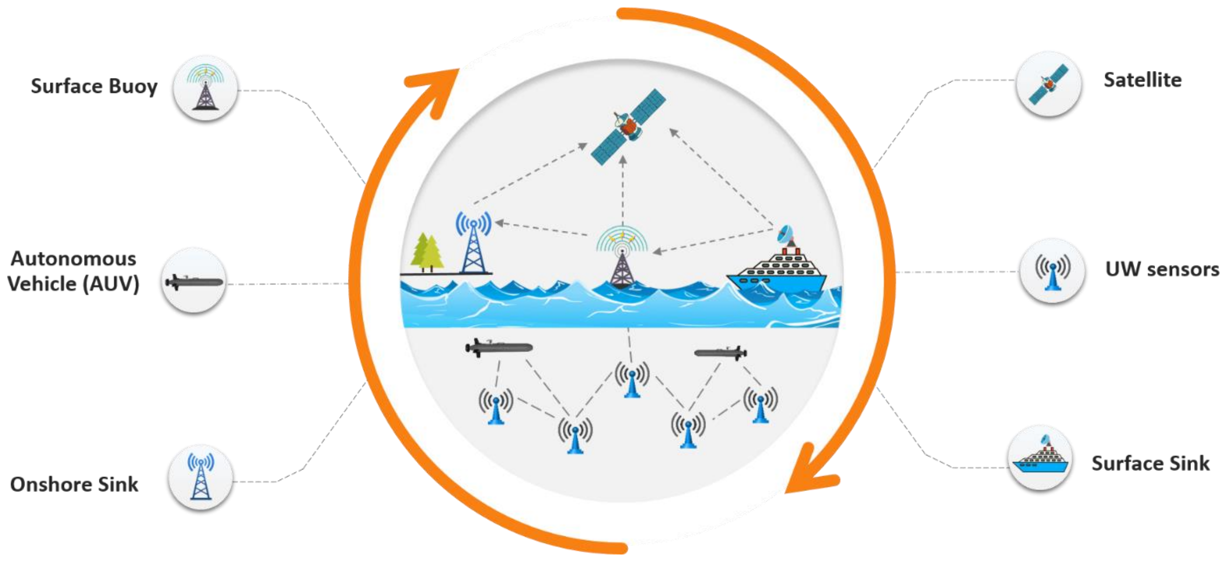


Figure 1.1: UWSNs Architecture

In the UWSN, designing the best propagation model is a crucial concern. *Bit error rate:* UWSN has a high bit error rate, which results in high prices and occasional connectivity outages. In comparison to TWSNs, the error rate in UMSN is extremely high. *Energy usage:* UWSN has a high energy consumption, however battery electricity is limited. Batteries, in general, cannot be revived. It is impossible to use solar energy in an underwater environment. The problem of developing a dependable and efficient energy consumption model has yet to be resolved. *Battery backup:* One of the most difficult aspects of using WSN nodes for underwater communication is the battery backup. The underwater node's battery backup can be limited, and once the price drops, the battery could be scrapped. *Lack of dependable communication:* UWSN's communication system does not provide for dependable communication [54]. As a result, different concerns such as packet loss, propagation delay, and so on exist. *Sensor node localization:* The deployment of sensor nodes is a critical difficulty in the UWSN. UWSN necessitates localization with high precision and scalability. The necessity for high precision localization remains a concern due to propagation latency, limited bandwidth, and node mobility. Low data rates: UWSN cannot use radio frequency waves. Acoustic communication is important in UWSN; however, it takes a long time to transport data between nodes due to its narrow range and low frequency. *Deployment:* Offshore deployment and field-level trials for ocean-centric applications are both costly and time-consuming [55]. Above mentioned issues can be reduced by taking actions against Hardware and Software components of UWSN. At hardware level, design energy efficient sensor nodes, employ renewable energy resources and reduce attenuation using acoustic medium. To address deployment challenges, provide completely configurable and adaptable simulations/models that allow researchers to set or customize hardware or software parameters for efficiently and accurately evaluating different novel protocols and algorithms.

Underwater wireless sensor networks lab-level experimentation and testing consists of three primary phases 1) *Design* 2) *Verification* 3) *Simulation*. *Design* is the first step in the UWSN experimental process. The development of structural and behavioral characteristics of a network is referred to as design. oTcl, Tcl, C, C++, C#, Embedded C, Proto C and other low-level technologies are commonly utilized for structural and behavioral representations. Once design phase is completed, *verification* phase begins, which includes selection of experimentation platforms. Their selection evaluates whether the chosen platform is easily adjustable, adaptable, time synchronized, efficient, dependable, and channel accurate, allowing researchers to configure hardware or software parameters for testing various techniques. Aqua-Sim, Aqua-Lab, UANT (Underwater Acoustic Networking plaTform), UPPER (Underwater Platform to Promote Experimental Research) etc. are common platforms for building and deploying a simulation tool. Simulation is carried out once the design and experimentation platforms have been implemented successfully. *Simulation* is the process of using simulator i.e. software tool for the testing new protocol, their communication capabilities and improvements of the network infrastructure for Underwater sensor Networks. Monte Carlo simulation, Discrete-Event Simulations, and Trace-Driven Simulation are the three types of simulation. Most common simulators to simulate the real-world underwater networks are Network Simulator version-2 and 3 (NS-2/NS-3), TinyOS simulator (TOSSIM), UWSim, J-Sim etc. [56].

Low level implementation technologies cause implementation challenges in UWSNs lab-level experimentation and testing at lower abstraction levels, i.e., low productivity, time consuming, costly, time to market difficulties, and increase the difficulty level. Verification of UWSNs is challenging due to this complexity. Model-Driven Engineering (MDE) is the answer to the problems that low-level implementation technologies cause.

1.1.4. Model-Driven Engineering

Model Driven Engineering is a software development approach that deals with software complexity by generating and manipulating the problem's conceptual models and meta-models. It is an abstract representation of the knowledge and processes required for the development and execution of software. MDE enables users to create complicated programs by abstracting pre-built components into simpler abstractions. Rather than telling, these visual building blocks demonstrate the business requirement and solutions to technical issues. MDE is the most fundamental principle of low-code development: it's the link that connects IT and business domain specialists, allowing them to cooperate and turn ideas into useful applications. MDE encapsulates complexity and automates human-process intervention, making it easier to understand. Instead of being interpreted into code, the model in model-driven engineering projects is executable at runtime. MDE is able to avoid frequent operational and quality difficulties with code-centric projects as a result of this [57].

Model Driven Engineering creates conceptual and meta-models of a specific problem in a specific domain using a variety of modeling languages. Many software engineers and researchers have utilized UML as one of the most powerful languages in this field. UML provides a standard approach to present a system's design [58]. The system is represented using two types of UML diagrams: structural and behavioral. The behavior of the system, or the actions of a component of a system, is depicted in a behavioral diagram. The structural diagram, on the other hand, is focused with a system's overall structure, or the components and their interdependencies. A UML profile diagram can be used to customize and expand UML conceptual models. UML Profile allows to customize the UML language to meet the needs of your customers or application domain. Model Transformation, which accomplishes automatic model transformation, is another key part of MDE. It is accomplished the transformation of one or more source models into one or more target models. The transformation process is guided by mapping rules that specify how each source component should be translated into the target model. Applying transformation rules to a given source model, as well as meta-models from both the source and destination models, can build target domain applications automatically. This transformation can be from a platform independent model to a platform specific model (Model-to-Model) or from a platform-specific model to code (Model-to-Text). The resulting output is validated after transformation.

1.2. Problem Statement

Researchers and industrial institutions are becoming more interested in the usage of underwater wireless sensor networks as technology progresses (UWSNs). Underwater wireless sensor network is a new wireless technology in which small sensors with limited energy, memory, and bandwidth are deployed in deep sea water and used for a variety of monitoring operations such as tactical surveillance, environmental monitoring, and data collection. Exploration of undersea resources, oceanographic data collecting, flood or disaster avoidance, tactical surveillance systems, and autonomous underwater vehicles all these use underwater wireless sensor networks. In UWSNs offshore deployment and field-level experimentation of ocean-centric applications in UWSNs are both costly and time-consuming. That's why the real time verification of network design is difficult due to the constrained imposed by the open acoustic channel, harsh underwater environment, and their own peculiarities. The importance of verification motivates us to carry it out as early as feasible before the deployment of network. Early design verification will save a lot of time and resources in the network design. MDE enables us to do design verification in the early stages of development.

There is a need for a strategy to reduce the complexity of UWSN development, provide a higher-level abstraction with fully automated end-to-end implementation, controlled development during the early stages, and an open-source generic and real-time solution that can verify network requirements prior to actual deployment. In regard to the design and verification problem, this research proposes Unified Modeling Language Profile, which provides a high level of abstraction and automatically generates target code, as a solution that simplifies design and low-level implementation complexity.

1.3. Proposed Methodology

The entire research is carried out in a systematic manner. Step-by-step research flow is depicted in Figure 1.2. We identified the problem in the first step. The optimum solution for the problem identified in the first phase was then provided. We conducted a thorough and in-depth literature review that assisted us in determining the best solution to the situation. We examined the research that had been done on our proposed solution, examined it, and compared it. A comparison of underwater wireless sensor networks protocols/techniques was also carried out.

Furthermore, the proposed solution includes an automated model-based approach for supporting the modeling of UWSNs at platform independent level. It also includes higher abstraction level automated end-to-end implementation of core concepts of the UWSNs using UML diagrams i.e., Class diagram, State Machine diagram, and modeling and standardizing the meta-level concepts using UML profile diagram. It also includes a tool for transforming models to text. The transformation engine is based on mapping rules, which transforms the UML model to code in implementation phase. The proposed methodology has been validated for two case studies of different sizes.

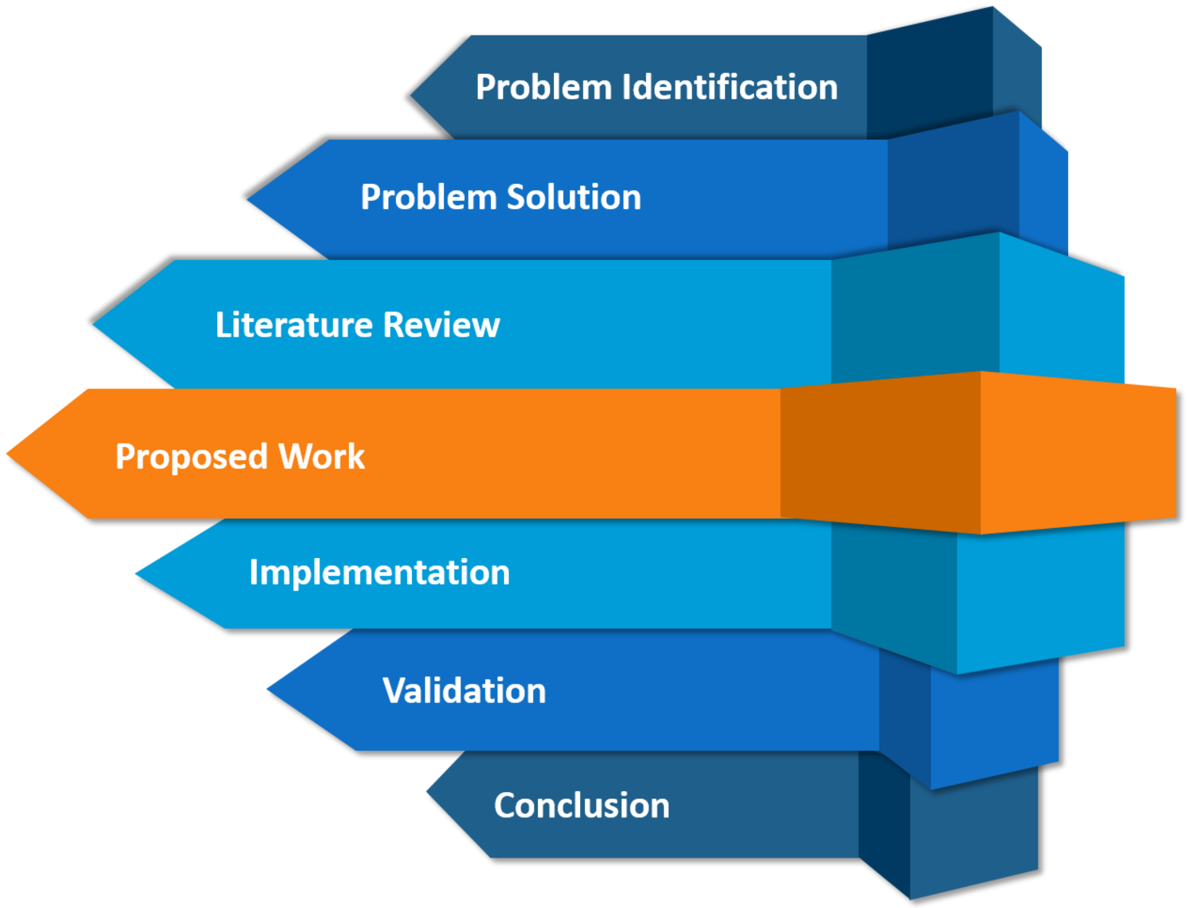


Figure 1.2: Flow of Research

1.4. Research Contribution

Comprehensive automated code generation for underwater wireless sensor network requirements at a platform-independent level is proposed in this work. The detailed set of contributions of the proposed approach are as follows:

- ➢ We have presented a model-driven framework to design network requirements for underwater wireless sensor networks for early design verification. It helps to provide a higher-level abstraction of the system and reduces the design complexity at early development stages.

- UML Profile for UWSNs has been developed to design/model the underwater framework.
- We have provided a transformation engine to generate underwater wireless sensor network implementation by transforming higher-level models to low-level implementation code in C++. The transformation engine is developed using Java and Acceleo transformation language.
- We have provided validation of our proposed work using two benchmark case studies from various domains and sizes to ensure that our suggested approach works in all scenarios.

1.5. Thesis Organization

The organization of the thesis can be viewed in Figure 1.3. **Chapter 1: INTRODUCTION** presents the introduction which consists of a background study of the concepts used in this research, the problem statement, proposed methodology, research contribution, and thesis organization. **Chapter 2: LITERATURE REVIEW** contains the literature review which provides a description of work done in the domain of UWSNs. In the Literature review, we also highlight the research gaps that we encountered. **Chapter 3: PROPOSED METHODOLOGY** describes the details of the proposed methodology used for an identified problem. It presents MFUWSN (Model-Driven Framework for Underwater Wireless Sensor Networks) for designing underwater applications. **Chapter 4: IMPLEMENTATION** covers the detailed implementation regarding the transformation rules for the proposed UML profile and transformation engine along with its architecture. **Chapter 5: VALIDATION** provides the validation performed for our proposed methodology using two important case studies i.e., Monitoring Offshore Oil & Gas Reservoirs and Smart Cites Underwater case studies. The two case studies chosen for validation have various domains and sizes to ensure that our suggested approach works in all scenarios. In the validation, modeling of these case studies is performed, then these models are transformed to C++ code, and code is simulated on a simulation tool i.e., AquaSim NS3 **Chapter 6: DISCUSSION AND LIMITATION** provides a brief discussion of the work done as well as the research's limitations. **Chapter 7: CONCLUSION AND FUTURE WORK** concludes the research and recommends future work for the research.

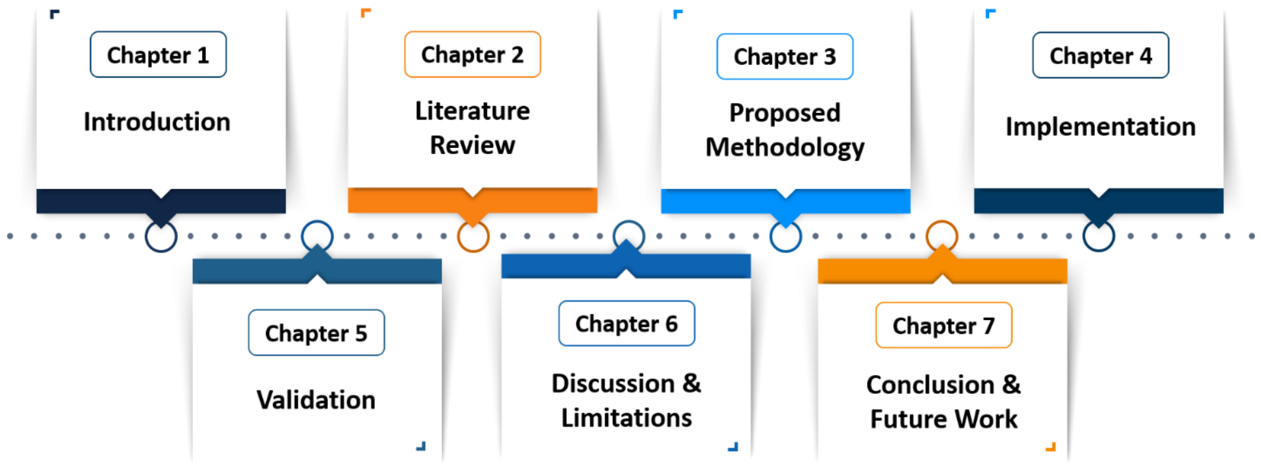


Figure 1.3: Thesis Outline

Chapter 2

Literature Review

CHAPTER 2: LITERATURE REVIEW

This chapter discusses studies in the field of underwater wireless sensor networks. The background study of UWSNs in academia and industry are discussed in **section 2.1.** Moreover, we carried out a systematic literature review to examine the most recent advancements in UWSNs, shown in **section 2.2.** In this section we identified the existing tools, shown in Table VIII, and routing protocols, shown in Table VI, and optimization techniques in Table VII, and communication technologies in Table VII and internal architecture types. This section conclude that no work has been presented for the automatically generating design and verification of network infrastructure, protocol and their communication capabilities. And **section 2.3** highlights the research gaps that forms the foundation of our research.

2.1. UWSN Background

UWSNs (Underwater Wireless Sensor Networks) is a new technology for monitoring aquatic assets that is being used in a variety of applications such as underwater data collection, ocean sampling networks, anonymous vehicles, disaster avoidance, and submarine detection. Researchers from academia and industry have recently been paying close attention to UWSNs. As a result, a number of studies have been conducted in order to develop UWSN approaches, tools, protocols, and architectures. UWSNs (Underwater Wireless Sensor Networks) are a hybrid of wireless technology and compact micromechanical sensor equipment with intelligent sensing, processing, and smart communication capabilities. Underwater wireless sensor networks (UWSNs) enable a wide range of underwater exploration applications for scientific, environmental, and military purposes [1]. Sensor nodes are spatially dispersed underwater in UWSNs to monitor variables including pressure, conductivity, turbidity, water quality, and temperature. The data collected can then be used in a variety of underwater applications such as coastal monitoring, pollution monitoring, military protection, water-based disaster mitigation, and undersea resource exploration, among others [2]. Sensor nodes, which can be stationary or movable, communicate with one another via wireless communication devices [3]. Transceivers are utilized in UWSNs to transmit collected data to the sinks at the surface (Sonobuoys). The Sinks use two modes of communication: acoustic and radio, to collect data from sensor nodes via acoustic communication and then communicate it to tracking stations via radio communication.

Underwater monitoring is difficult due to the characteristics of acoustic communication, which include a long propagation delay [4], a high packet loss probability, high energy consumption, low dependability, and limited link capacity and bandwidth [5].

There are various studies [47] that focus at the evolution of Wireless Sensor Networks as a whole, rather than only UWSNs. Furthermore, some recent studies provide an overview of various areas of underwater wireless sensor networks, such as routing problems, protocol operations, assessment, and supporting technologies. For example, an investigation of existing routing protocols, forwarding mechanisms, dynamic structure, route management, and discovery is performed in a study [35]. Another study [36] uses analytical and simulation methods to investigate routing algorithms based on node mobility. The features of underwater wireless sensor networks and possible data gathering approaches are discussed in [37], and the authors of [38] explore the features of underwater wireless sensor networks and prospective data gathering techniques. Although several elements of UWSNs, such as protocols, tools, and communication forms, have been extensively explored independently, to the best of our knowledge, there is no research accessible that analyzes and summarizes recent UWSNs advances. The goal of this study is to highlight the most recent research efforts and technological advancements aimed at improving the performance of UWSNs and permitting their widespread use, while also serving as a starting point for practitioners and researchers interested in working in this field. We ran a Systematic Literature Review to find this prominence of stuff (SLR) in section 2.2.

2.2. Systematic Literature Review

2.2.1. Review Protocol

The review protocol demonstrates the overview of our study, research questions, criteria of selection and rejection, the method of search process, assessment of quality, extraction of data and the mechanism used for data synthesis. The details of these elements are given in following sub-sections.

2.2.2. Research Questions

The following are the research questions that will be explored in this section: **RQ1:** What major studies involving UWSNs have been published between 2012 and 2020? **RQ2:** At what level the

Model Driven in UWSN is used? **RQ3:** Which tools / simulators are recurrently used during 2012-2020 to aid UWSNs verification? **RQ4:** What are the foremost routing protocols, architecture types and communication technologies that have been utilized for UWSNs during 2012-2020 researches? **RQ5:** What are the future perspectives in the area of UWSNs?

To answer the above questions a Systematic Literature Review (SLR) is performed to comprehensively analyze the latest developments in UWSNs. Particularly, 34 research studies published during 2012-2020 have been selected from 4 scientific libraries IEEE, SPRINGER, ACM AND ELSEVIER. Finally, a comparative analysis of routing protocols is done on the basis of important evaluation metrics which are used for the validation of UWSNs. It has been concluded that there exist adequate approaches, protocols and tools for the monitoring of UWSNs. However, the design verification capabilities of existing approaches are insufficient to meet the growing demands of UWSNs. In this context, the findings of this article provide solid platform to enhance the current UWSNs tools and techniques for large and complex networks. It is also clear from the analysis that no article has been found which proposes higher abstraction layer for verification of underwater wireless sensor systems.

2.2.3. Inclusion and Exclusion Criteria

For the selection of researches, four parameters are defined which result in high quality outcomes. These parameters are:

1. Selected researches must be related to underwater wireless sensor networks.
2. Selected researches must be published during 2012-2020.
3. Research study is selected only if it belongs to any of the following notable scientific databases: IEEE, SPRINGER, ACM, and ELSEVIER.
4. Selected researches should be result oriented and provide genuine solution to some UWSNs problems.

On the other hand, the research studies do not follow all the aforementioned selection parameters will be discarded.

2.2.4. Search Process

Four scientific databases (i.e., IEEE, Springer, ACM and Elsevier) are selected to get the authentic and positive studies from impact factor journals and conference proceedings. Different search terms are used for search process like Underwater Wireless Sensor Networks, UWSN protocols, Underwater Wireless Acoustic Networks, UWSN Architecture and UWSN Model Driven Engineering as listed in Table 1. The first column of Table 1 represents serial number (Sr. #) and tables are sorted through it for systematic representation. The second column contains different search terms and corresponding numeric results against each scientific database are given in third column of Table 2.1.

Table 2.1: Search terms with results

Sr.#	**Search Term**	**Search Results (2012-2020)**			
		IEEE	Springer	ACM	Elsevier
1	Underwater Wireless Sensor Networks	1291	1202	955	1089
2	UWSN Model Driven Engineering	246	362	187	638
3	UWSN protocols	548	687	578	247
4	Underwater Wireless Acoustic Networks	1179	614	502	457
5	UWSN Architecture	487	599	688	333

To acquire relevant results, different filters are applied like year filter (2012-2020), AND operator etc. This facilitates us to get significant number of studies to perform further evaluation as shown in Fig. 2. Initially, 12,889 researches are selected from four databases. Subsequently, 10,200 researches are rejected after reading title and 2002 researches are excluded by reading their abstracts. Subsequently, 585 researches are rejected after reading different sections. We comprehensively examine the remaining 102 researches to ensure the fourth selection parameter. During this, we found that few studies are not fully compliant with selection criteria. For example, the results are not properly presented in study [48]. Furthermore, authors in study [49] do not carried out performance analysis. Consequently, we exclude such 68 studies after detailed analysis and finally select 34 researches which fully satisfy selection and rejection criteria as shown in Figure 2.1.

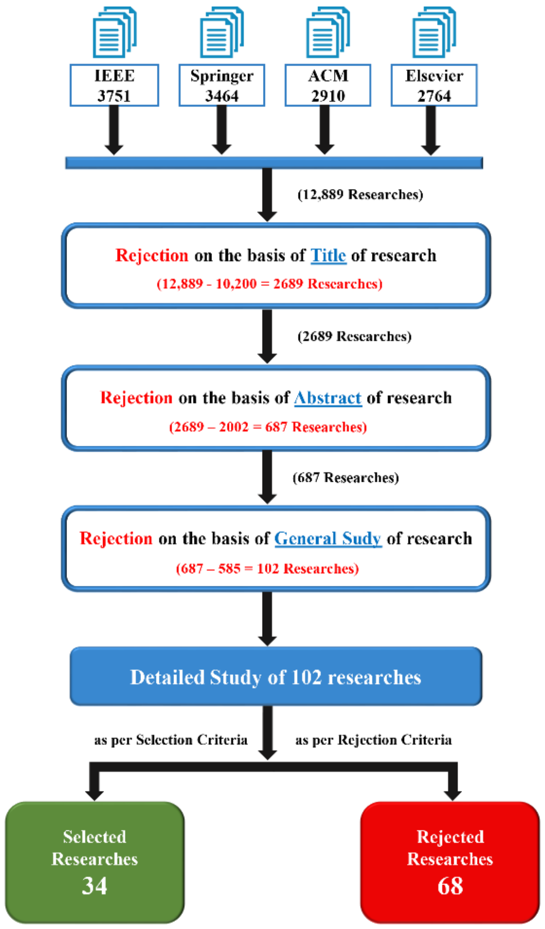

IEEE
3751
Springer
3464
ACM
2910
Elsevier
2764
(12,889 Researches)
Rejection on the basis of Title of research
(12,889 - 10,200 = 2689 Researches)
(2689 Researches)
Rejection on the basis of Abstract of research
(2689 – 2002 = 687 Researches)
(687 Researches)
Rejection on the basis of General Sudy of research
(687 – 585 = 102 Researches)
Detailed Study of 102 researches
as per Selection Criteria
as per Rejection Criteria
Selected Researches 34
Rejected Researches 68

Figure 2.1: Search Process

2.2.5. Quality Assessment

To ensure the trustworthy results of this SLR, we attempt our best to select high quality researches. To achieve this, we only selected studies from four prominent and reliable databases (i.e., IEEE, Springer, ACM and Elsevier). The distribution of studies with respect to four databases is shown in Table 2.2. We thoroughly searched all four scientific databases with different keywords as shown in Table 2.1.

Table 1.2: Number of selected studies with respect to scientific databases

Sr. #	Scientific Database	Type	References	Total References
1	IEEE	Journal	[4][6][8][10][11][12][13][14][15][18][19][20][21][22][23][24][30][34]	18
		Conference	[7][16][17][31]	4
2	Springer	Journal	[25][26][27][28][29][33]	6
		Conference	Nil	0
3	ACM	Journal	Nil	0
		Conference	[32]	1
4	Elsevier	Journal	[1][2][3][5][9]	5
		Conference	Nil	0

To ensure reliable outcomes of this SLR, we select only those researches where appropriate validation of proposed approach is performed as stated in parameter 4 of selection and rejection criteria (Section 2.1). For example, we discard study [49] as its validation is improper. Moreover, we do not select any review paper. Furthermore, we try to select newest studies as much as possible. Figure 2.2 presents year wise distribution of selected researches.

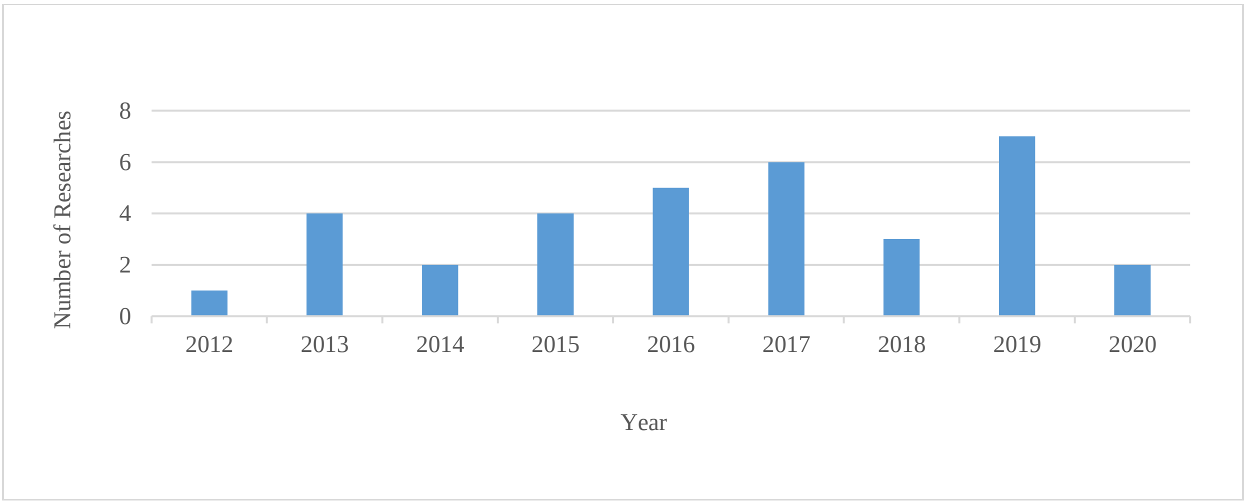


Figure 2.2: Yearly Distribution of Selected Researches

2.2.6. Data Extraction

Different data extraction parameters are defined to acquire the appropriate data as illustrated in Table 2.3. We examine the extracted information to recognize the tools, architecture types, communication technologies, proposed protocols with routing techniques. Finally, the evaluation of routing protocols is performed. Basic bibliographic information from each selected research is also extracted for the completion of analysis.

Table 2.2: Data Extraction

Sr. #	**Description**	**Details**
1	Bibliographic Details	Authors of study, Title, Publisher Publication year and Research Type (i.e., Journal / Conference)
	Data extraction	
2	Outline	Basic objective of selected study like purpose of proposed approach and related tool / protocols in the context of underwater wireless sensor networks
3	Results	Fallouts of selected researches
4	Assumptions	Qualitative or Quantitative method used
5	Validation	Evaluation of results e.g., case study etc.
	Data Synthesis	
6	Categorization	Relevance of selected study with pre-defined categorization (Table 2.4 and Table 2.5)
7	Tools	Tools used in Underwater wireless sensor networks (Table 2.6)
8	Architecture Types	Architecture types used in UWSNs (Table 2.7)
9	Protocol and Optimization Techniques	Protocols with associated optimization technique used in each selected study (Table 2.8)
10	Communication Technologies	Communication types used in UWSN (Table 2.9)
11	Performance Evaluation of each proposed Protocol	Each protocol is evaluated on the basis of performance metrics (Table 2.10)

2.2.7. Results

In this section, an in-depth analysis of selected studies and the results achieved are presented. The references of corresponding studies are provided against each category for further exploration. We categorized the 34 selected researchers into four groups on the basis of their symmetry with the appropriate domain as given in Table 2.4. Description of these categories are as follows:

1. The *Model Driven* approach enables the reusability and design verification simplicity [44]. In this regard, the Model Driven category involves those selected studies where model driven approaches are applied for UWSNs.

2. *Cloud computing* is highly relevant concept with wireless networks [42]. Therefore, the studies that involves the architecture of cloud computing are placed in Cloud Computing category.

3. The *Mobile UWSNs* category involves those selected studies where ad hoc networks are applied in UWSNs.

4. The *IOT category* involves those selected studies where different approaches are applied and belongs to Internet of things. This is in fact a general category of UWSNs.

Table 2.3: Categories of researches on the basis of different Domains

Sr. #	Category	References
1	Model Driven	[30][31]
2	Cloud Computing	[30]
3	Mobile UWSNs	[4][7][18][19][33]
4	IOT	[1] [2] [3] [5] [6] [8] [9] [10] [11] [12] [13] [14] [15] [17] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [32][34]

Although Table 2.4 clearly provides the domains where UWSNs are applied, there is a need to categorize selected researches in terms of quality objective targeted for validation of UWSN in each research. Hence, Table 2.5 provides a classification of selected researches in terms of metrics. Description of these categories are as follows:

1. The *Energy Consumption* category involves researches in which efficiency measures are proposed to reduce the energy consumption according to the structure and need of UWSNs. Energy consumption is highly significant characteristic [39] of wireless networks.

2. The *Localization* category includes researches focuses on enhancing the accuracy and precision of nodes placement in UWSNs.

3. *Execution time* is important quality measure in wireless networks [43]. Therefore, the Execution Time category involves those selected studies where time to deliver data packets is targeted when the number of nodes exceeds.

4. The *System Utilization* category involves selected studies those optimize the usage of networks by applying different slotting strategy in UWSNs.

5. The *Throughput* category consider studies those focus on improving packet delivery ratio in UWSNs.

Table 2.5: Categories of researches on the basis of Targeted Objective

Sr. #	**Category**	**References**
1	Energy Consumption	[2][4][7][8][9][10][14][16][17][21][23][25][28][33][34]
2	Localization	[1][3][6][13][15][18][26][27][31]
3	Execution Time	[11]
4	System Utilization	[19][20][30]
5	Throughput	[5][12][22][24][29][32]

In UWSNs, testing of any technique/algorithm is expensive as well as difficult in terms of deployment in deep oceanic water. Therefore, testing prior to actual deployment setup of UWSNs is an essential step. Number of tools/ simulators has proven to be widely useful for the analysis of underwater wireless networks [11]. We identify 11 tools/simulators from 34 selected researches as given in Table 2.6. Identified tools/simulators are categorized on the basis their availability in market. İt can be open source and free to use for general public or it can be commercial. These tools/simulators plays a substantial role to emulate the data gathering and monitoring process in

UWSNs. Each tool/simulator has numerous benefits as well as limitations. It can be seen from Table 2.6 that Network Simulator (NS) is most frequently utilized tool for UWSNs. Particularly, NS2 is the older version of Network Simulator which is now replaced with new version NS3. However, few selected studies reported the use of NS2 while other reported the use of NS3. Therefore, in Table 2.6, we combine all studies under Network Simulator (both NS2 and NS3 versions). We analyze that Network Simulator with AquaSim are the most general purpose used simulators/tools in selected researchers for UWSNs because these complies the Object-Oriented style, endure 3D modeling and can emulate acoustic signal attenuation, packet delivery conflicts and delays in propagations.

Table 2.6: UWSNs Tools / Simulators

Sr. #	**Tools / Simulators**	**Availability**	**References**
1	Network Simulator (NS2 and NS3 Versions)	Open Source	[1] [2] [5][7] [9][11] [12][18][19][21][22][25][26] [27][28][29] [31][32][33][34]
2	AquaSim	Open Source	[3][5][7][18][22][25][26][34]
3	Matlab	Commercial	[3][10][17]
4	QualNet	Commercial	[6][8][11]
5	J-Sim	Open Source	[11]
6	SUNSET SDCS	Open Source	[23]
7	GME (Generic Modeling Environment)	Open Source	[30]
8	Eclipse Modeling Framework (EMF)	Open Source	[31]
9	DigitizeIt	Open Source	[11]
10	OPNET	Commercial	[16][11]
11	GloMoSim	Commercial	[11]

Additionally, all of the identified tools/simulators are not particularly developed to work for UWSNs. Many of aforementioned tools/simulators are inherently developed for non-wireless networks. However, there are the extended versions for UWSNs such as OPNET and NS2. The NS2 is not supported anymore but NS3 with Aquasim is specifically designed for underwater network simulations and has all the NS2 features embedded. NS3 is an emerging underwater acoustic network tool and many researches have recently utilized it. Beside this, Model driven approach is easy to implement for UWSNs through simulators like NS3. Particularly, NS3 is platform specific simulator while model driven approach deals with platform independent models. In this regard, platform independent model can be automatically transformed to platform specific models [46] like NS3 for the simulation of UWSNs environment. As a result, the combination of model driven with UWSNs tools can certainly reduce complexity of network modeling and verification. However, the possibilities for such combination still need to be further investigated. Furthermore, all identified tools does not provide modeling, transformation and verification facilities for UWSN domain. Though, most of the identified tools are an enterprise solution and not freely available. In the category of open-source free solutions, MDE can be considered a worthy option.

From 34 selected researches, we identify four types of UWSNs communication architectures i.e. One dimensional, two dimensional, three dimensional and four dimensional. It is important to note that such networks can be classified in many ways, but we consider the most common architecture types which are the basis for creating the UWSN applications. The classification of selected studies on the basis of architecture types is given Table 2.7. The identified architecture types in UWSNS allows a remote facility to analyze, monitor and report any defect, risk or vulnerability. These architecture types are designed for reliable communication [12].

Table 2.4: UWSNs Communication Architectures

Sr. #	Architecture Type	References
1	One-dimensional (1D)	[3][8]
2	Two-dimensional (2D)	[1][13] [25][28]
3	Three-dimensional (3D)	[2][4][6][9][16][17][20][21][24][26][27][29][33]
4	Four-dimensional (4D)	[1][6][7]
5	Other	[5] [8]

Routing protocols in UWSNs optimally picks the paths of routing in regard to energy consumption and then contributes in prolonging the network lifetime. These protocols can be classified into different categorize according to the application scenario or the basic measures considered while routing. Typical routing protocols are based on localization, cross-layer, opportunistic routing, geo-based, clustered, hop-by-hop and reinforcement learning. To achieve required outcomes, several optimization techniques can be applied on routing protocols as per UWSNs requirements. In selected studies, we identified 21 important routing protocols and their corresponding optimization technique as shown in Table 2.8. Finding and maintaining the routes for complex underwater conditions with energy constraint and unexpected topology changes is a challenging task. Recently, many routing optimization techniques have been presented or already proposed techniques has been enhanced for UWSNs. For example, in order to handle node mobility problem, dynamic addressing based technique has been applied to all network nodes [1] where each node has been assigned route address without the need of configuration information. While in each hop, the packet is routed to a locally optimal next-hop node to the destination by using greedy opportunistic forwarding strategy [5][6][7][8][20][29]. This technique is used by many studies with adjustments (e.g., sequence number, hop count and depth information) according to the goals and requirements of the application.

In another study [2], to handle the problem of non-uniform sensor node distribution, an optimization technique of adaptive packet transmission range has been adopted in the routing decision process. Similarly, dynamic multi access reservation slotting schemes [19] are proposed to change the number of access slots accordingly, and to explore the optimal slotting strategy to optimize system usage. Deep learning is now emerging trend to manage routing issues [40]. In this regard, the authors in [22] merge neural network with Q-learning to make globally optimal routing decision. In another study, self-learning based dynamic firefly mating optimization intelligence technique [34] is used to locate highly stable and secure routing paths for packets in UWSNs across communication voids and shadow zones.

In few selected studies [9] [27] [33], the optimization technique is not explicitly mentioned rather such studies identify few parameters (e.g. node energy, hop count, propagation delay, channel quality, routing pipe radius, depth information) which should be consider while proposing a protocol. In summary, it can be analyzed that any single optimization technique is not solely suitable for different scenarios because each technique has some definite strengths and disadvantages. The findings of this study are highly useful for new researchers as latest routing protocols and their associated optimization techniques are systematically summarized as given in Table 2.8.

Table 2.8: Proposed Routing Protocols with Corresponding Optimization Techniques

Reference	Protocol	Optimization Technique
[1]	Hop-by-Hop Dynamic Addressing Based (H2-DAB)	Hop-by-Hop Dynamic Addressing Algorithm
[2]	Adaptive hop-by-hop Vector-Based Forwarding (AHH-VBF)	Adaptive Forwarding/Packet Transmission Range Technique
[5]	Geographic Routing Protocol	Anycast Greedy Geographic Forwarding Strategy
[6]	Void-Aware Pressure Routing (VAPR)	Greedy opportunistic direction forwarding approach
[7]	Geographic and Opportunistic Routing Protocol with Depth Adjustment (GEDAR)	Greedy Opportunistic Forwarding Strategy with Depth Adjustment
[8]	HydroCast	Voids Recovery with Greedy Opportunistic Forwarding
[9]	Agent Based Approach (ABA)	Route establishment based on node energy, hop count and propagation delay
[19]	Dynamic Reservation Access Protocols (scheme I) (scheme II)	System-state-aware dynamic multi-access reservation slotting approach
[20]	Enhanced CARP (E-CARP)	Location-free and Greedy hop-by-hop Routing Strategy
[21]	Energy-Aware and Void-Avoidable Routing Protocol (EAVARP)	Opportunistic Directional Forwarding Strategy based on void hole avoidance
[22]	Deep Q-Network-Based Energy- and Latency-Aware (DQELR)	Deep Q-Network Algorithm with Off-Policy and On-Policy Methods
[23]	Channel-Aware Reinforcement Learning-Based Multi-Path Adaptive (CARMA)	Distributed Reinforcement Learning Framework
[24]	Proactive Routing Approach with Energy efficient Path	Dijkstra's algorithm with vertical layering approach and cluster formation
[25]	Energy-Efficient Multipath Grid-based Geographic Routing (EMGGR)	Gateway Election Algorithm with Packet Forwarding mechanisms
[26]	Energy-Efficient Localization-Based (EEL)Geographic Routing	NADV (Normalized Advancement) and TOA (Time of Arrival) localization
[27]	Improved VBF (Vector Routing Forwarding)	Geographic routing strategy with routing pipe radius
[28]	Multi-Layered Routing Protocol (MRP)	Localization-free routing mechanism
[29]	Grid Division Polar Tracing (GDPT)	Greedy Algorithm using Cubic Grid Model and Polar Tracing
[32]	On-Surface Wireless-Assisted Opportunistic (SurOpp)	Opportunistic routing in buoy nodes

[33]	Energy Efficient Data Gathering (EEDG)	Depth information and Nodes Forwarding Approach
[34]	Dynamic Firefly Mating Optimization Inspired Routing Protocol (FFRP)	Self-Learning based Dynamic Firefly Optimization Intelligence Algorithm

Table 2.9 gives a glimpse of different specifications of three communication technologies (i.e. Acoustic, Optical and Electromagnetic) which are frequently applied in UWSNs. Particularly, each communication technology is evaluated against different attributes like communication range and data rate which refers to transmission speed of each communication technology. Similarly, latency denotes the delay in UWSN communication and transmission power refers to the energy utilize during communication. The cost represents the deployment and operational cost of each technology. The directional may refers to omni directional in which signal is transmitted and received from any of the possible directions or uni directional in which signal is transmitted or get in only one direction. Finally, the propagation path loss (attenuation) and energy [8] [12] [15] represents corresponding concepts of each communication technologies.

Electromagnetic (RF) communication technology is typically best for land wireless networks, but it can also be successfully use in underwater wireless sensor networks. It accomplishes high data rate in Gbps but has a limitation with respect to short communication range. On the other hand, optical communication technology has a high bandwidth such as gigabits per seconds but has a short communication range and highly effected by turbidity and salinity. It works best in clear shallow underwater. Lastly, acoustic technology is the typical communication type of underwater wireless sensor networks because of its long communication range. Acoustic technology propagates substantially in deep water, but its propagation delay is high and its speeds mainly depends on salinity and temperature of water.

We analyze that Acoustic is the main technology used in UWSNs presently. Particularly, nine selected studies ([5][6][8][12][13][14][15][17][18]) utilize Acoustic communication technology alone. Moreover, one study [10] utilize the combination of Acoustic and Optical technologies. Furthermore, nine studies ([1][2][3][7][9][16][26][32][33]) utilize the combination of both Acoustic and RF technologies. Finally, it is also analyzed that only three studies ([4][11][34]) utilize RF technology alone. It is important to note that rest of the selected studies do not provide any information regarding the use of communication technology, therefore, such studies are not considered in the analysis.

Coarse underwater environment possesses huge difficulties and challenges when designing deep water communication systems in temporally and spatially fluctuating propagation media. Therefore, each communication technology advantages and disadvantages should be keenly considered while selecting for a particular network. From the analysis, it is also observed that each communication media has its own advantages and disadvantages, therefore, the application of each technology is subject to the given requirements. However, the major characteristics while selecting the underwater communication technologies are bandwidth, channel attenuation, life time of network and routing protocol.

Table 2.9: Comparison of Different Underwater Communication Technologies

Technologies	**Comm. Range**	**Data Rate**	**Latency**	**Energy**	**Transmission Power**	**Cost**	**Propagation Path loss**	**Directional**
Acoustic	<20 km	<10 Kbps	High	100 bits/Joules	≈ 10 W	High	High	Omni
Optical	100 m-200 m	<10 Gbps	Low	30,000 bits/Joules	≈ 1 W	High	Turbid Dependent	Uni
Electromagnetic (RF)	< 100 m	<0.1 Gbps	Moderate	N/A	≈ 100 W	Low	Moderate	Omni

2.2.8. Comparison

In this section, performance analysis and evaluation of identified UWSNs routing protocols is carried out as given in Table 2.10. Particularly, six evaluation parameters are defined to perform genuine investigation of routing protocols. The details of evaluation parameters are as follows:

1. *Energy efficiency* is the usage of energy during successful information transmission towards the destination. Since UWSNs have limited energy because of larger size nodes, this parameter becomes the most significant one. Therefore, to provide reliable data transmission and higher network lifetime, it is necessary to use energy efficient routing protocol.

2. *Delivery Ratio* is proportion of data packets which are sank in time of communication by total number of data packets originated at source. The parameter is really important to measure the overall effectiveness of protocol.

3. *End-to-End Delay* is a time consumed when data packets transmitted from origin to destination. This ratio is mostly at a greater altitude due to the use of acoustic communication in underwater networks.

4. *Cost efficiency* includes different types of costs related to overall network e.g. deployment cost, nodes cost, underwater sensor devices cost etc. The actual deployment of applications for verification in underwater networks is really costly. Therefore, underwater routing protocols has to be cautiously considered and designed in terms of performance evaluations, simulations etc. for initial verification without actual deployment.

5. *Node Mobility* is also an essential performance metric for the efficient data forwarding in UWSNs. As the deployment of deep-water network is sparse, the mobility and time synchronization between nodes is very challenging.

6. *Integration* with underwater MAC protocols explore the integration possibilities of proposed routing protocol with MAC protocol. Such integration is commonly required for complex underwater networks.

Table 2.10: Qualitative Analysis of Proposed Routing Protocols through Performance Metrics

Reference	**Protocol**	**Energy Efficiency**	**Delivery Ratio**	**End-to-End Delay**	**Cost Efficiency**	**Node Mobility**	**Integration with underwater MAC protocol**
[1]	H2-DAB	Medium	Medium	Low	Low	High	IEEE 802.11
[2]	AHH-VBF	High	Medium	Low	Low	NA	Aloha
[5]	Geographic routing protocol	Medium	High	Medium	Medium	NA	CSMA
[6]	VAPR	NA	High	Low	NA	High	CSMA
[7]	GEDAR	High	High	Medium	High	Medium	CSMA
[8]	HydroCast	High	High	Low	Medium	Medium	CSMA
[9]	ABA	High	Medium	Medium	High	NA	
[19]	Dynamic reservation access protocols (scheme I) (scheme II)	High	NA	Low	Medium	Medium	Aloha
[20]	E-CARP	High	Medium	NA	Medium	High	NA
[21]	EAVARP	Medium	Medium	Low	NA	High	NA
[22]	DQELR	High	Medium	Medium	NA	High	NA
[23]	CARMA	High	High	Low	NA	NA	NA

[24]	PA-EPS-Case I	Medium	High	Medium	Low	Medium	NA
[25]	EMGGR	High	High	Low	NA	Medium	NA
[26]	EEL	High	High	Medium	Medium	Low	NA
[27]	Improved VBF Algorithm	High	High	NA	NA	High	NA
[28]	MRP	High	High	Medium	NA	Low	802.11- DYNAV
[29]	GDPT	High	High	Low	Medium	Medium	NA
[32]	SurOpp	High	High	Low	High	Medium	CSMA, IEEE 802.11n
[33]	EEDG	High	Medium	Low	NA	NA	NA
[34]	FFRP	High	High	Low	NA	NA	CSMA

The comparative analysis of underwater routing protocols is summarized in Table 2.10. The aforementioned parameters are evaluated though different attributes like High, Medium, Low and NA (Not Available) to analyze the performance of each identified protocol. For example, consider H2-DAB protocol [1] in Table 2.10, its energy efficiency and delivery ratio are Medium while its cost, efficiency and end to end delay is Low. Furthermore, Node mobility of this protocol is High. Similarly, if we consider Geographic routing protocol in Table 2.10, its delivery ratio is High while energy efficiency, end to end delay and cost efficiency is Medium. Its node mobility is Not Available (NA) which means authors do not discuss this metric at all. Other routing protocols, as given in Table 2.10, can analyzed in similar way.

From detailed analysis of Table 2.10, it is concluded that most of the studies do not take node mobility into consideration and relative cost efficiency is low which eventually lead to high communication expenditures, high energy utilization and unreliable ways of communication in underwater environment. The analysis and performance metrics, as given in Table 2.10, facilitate practitioner and researchers in the selection of right protocol for a specific underwater application under particular circumstances. For example, in case of underwater pollution observation and monitoring demands, the time efficiency is not usually required. Therefore, energy efficient routing protocols are highly suitable for such applications. On the other hand, in case of real time applications such as catastrophic avoidance, less attenuate protocols having small path coverage with minimum propagation delay are preferred.

We can find clear evidence in literature that researchers have proposed simulation tools for validating and debugging various underwater acoustic protocols. These requires writing simulation manually at lower-level abstraction and this inevitably requires knowledge of how such codes are written. Hence complexity of underwater wireless sensor networks makes the verification process difficult. MDE is an effective approach that provides promising features to perform design verification in early phases using higher abstraction layers.

2.3. Research Gap

In this section, research gaps and proposed solution is discussed. After analyzing the above literature, it has been observed that as the technology and economy advances, UWSNs has becoming the main source of interest for researchers and industrial practitioners. However, only a small amount of work has been done with Model Driven Engineering. No design validation and transformation model has been defined for generating UWSN's architecture. Industries and human activities are continuously demolishing the underwater environment thus underwater circumstances monitoring becomes a crucial issue for the researchers. Deployment of UWSNs is difficult and costly activity due to the involvement of several underwater constraints. These constraints includes limited hardware resources (computing power, repositories), expensive proprietary tools, communication technologies are not reliable, prolong and irregular attenuation, limited lifetime of a network, fixed bandwidths, noise, physical susceptibility, network attacks and errors while transmitting data. Due to the unavoidable constraints imposed by underwater wireless sensor networks, fully fledged solutions and applications are hard to implement in this area. Researchers propose several simulators and approaches for conducting simulation or experimentation in order to minimize costs and increase speed research activities and findings. Designing efficient and dependable simulation and experimentation systems, on the other hand, has proven to be more difficult than anticipated. Detailed and reliable verification of UWSNs prior to actual deployment is very critical. Simulators causing the individuals to write low level code for simulation of design and properties at lower abstraction level. Due to large development and debugging efforts, writing low level code at a lower abstraction level takes a long time. Furthermore, it results in higher costs less user friendly and a longer development time. Moreover, coding is less reusable and more prone to errors. As a result, we'll need to use modeling approach to get around the low-level coding implementation issues and to overcome design and verification gap.

To address the identified gaps after a comprehensive literature review of current applications and developments in the domain of Underwater Wireless Sensor Networks (UWSNs), a novel Model-Driven Engineering based solution has been proposed to overcome design and verification gap. Our proposed solution provides the higher level of abstraction with complete, open source, fully automated controlled, platform-independent and real time solution to deal with changes and complexity of the network verification requirements.

The following are some of the characteristics of the proposed approach:

- Early analyze and do verification of network design before deployment
- Network constraints are verified in preliminary development period
- Saves time of debugging and low-level coding
- Reduces time complexity and provide automatic network model
- Is an open source system for Researchers to test and enhance?
- Includes abstraction, separation of concerns, reusable components and automation
- Provides the design correctness at early stages according to its global specification
- Increase the network's performance and QoS
- Less Error Prone

Chapter 3

Proposed Methodology

CHAPTER 3: PROPOSED METHODOLOGY

As discussed, earlier UWSN is a new wireless sensor technology that is being used to provide the most promising mechanism and ways for detecting the aquatic environment. It has a variety of important applications in the harsh underwater environment. It performs admirably in a variety of settings, including commercial, military, emergency monitoring, data collection, and environmental monitoring. However, verification of network design prior to actual deployment is a time consuming, tedious task and require expert skills and knowledge. To provide a comprehensive, well-coordinated and interoperable platform for underwater network systems, a Model-Driven Framework for Underwater Wireless Sensor Networks (MFUWSN) is proposed. This method enables developers to generate a network design from a requirement's global specification. To verify that its design adheres to its specifications, the derived behaviors being evaluated and verified before being deployed in the early stages of development. Figure 3.1 depicts a graphic representation of the suggested framework. Multiple ideas such as UML Profile and Model to Text (M2T) transformation concepts have been combined into the suggested solution described below in order to give a complete UWSN behavior requirement.

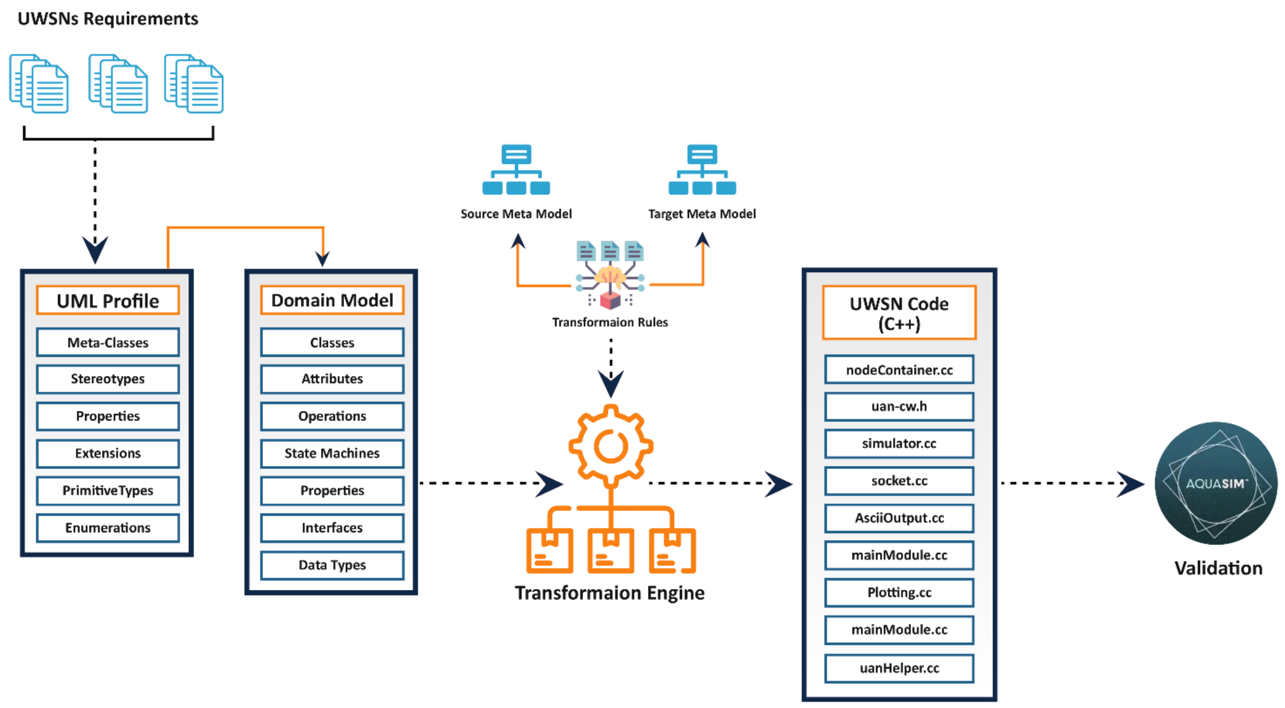
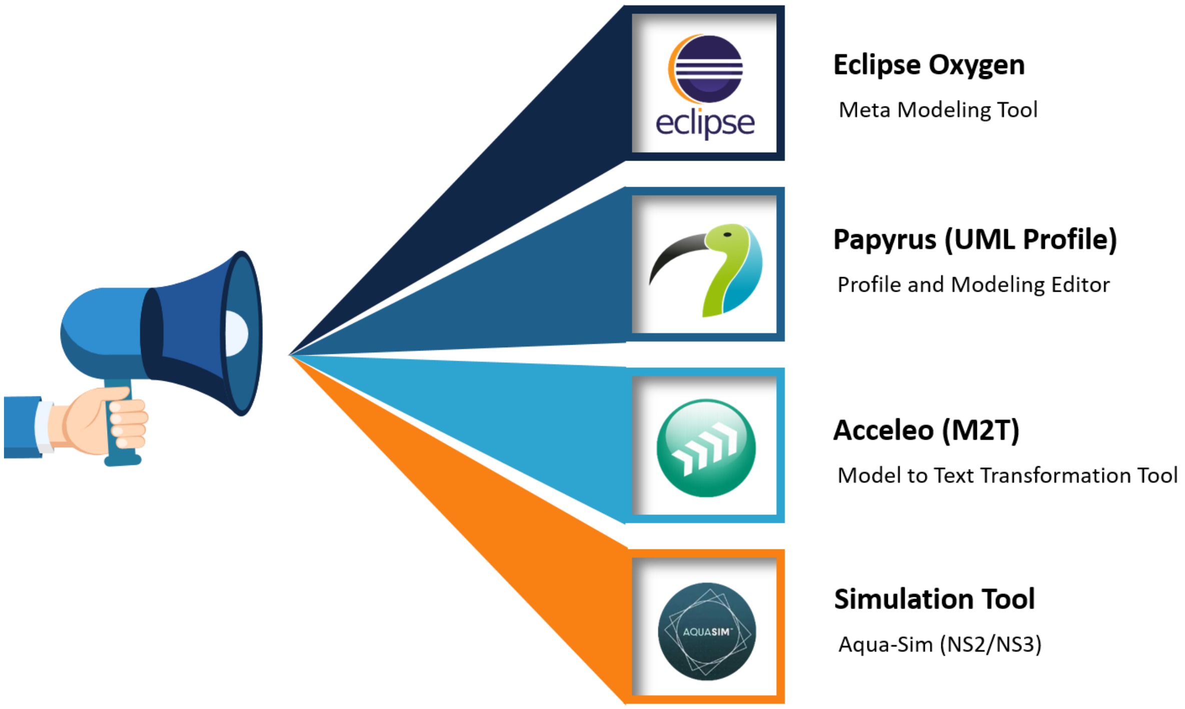


Figure 3.1: Model-Driven Framework for Underwater Wireless Sensor Networks (MFUWSN)

The purpose of this chapter is to give detail of concepts used in the proposed solution. Model-driven engineering is used in the recommended approach, which involves the use of the UML Profile Diagram to expand and apply meta-level principles in the modeling of data acquisition system requirements. Then, using the Model to Text transformation technique, the domain model is used as an input model to build entire NS3 C++ code for underwater wireless sensor network simulation. All of the generated files are then required to deploy in an actual environment to be completely operational i.e., we used AquaSim NS3 simulation tool to simulate the generated artifacts. The required tools and techniques for this purpose can be seen in Section 3.1 and Figure 3.2. The proposed model-driven framework, with its principles of abstraction, separation of concerns, reuse, and automation, can be considered the right tool for reducing the complexity of underwater wireless sensor network development.

3.1. Tools and Techniques Architecture

The use of effective tools can contribute in the accelerating of the software development process. The architecture of the tools and techniques used in the creation of MFUWSN is depicted in Figure 3.2. Metamodeling environment, Modeling editor, Modeling Transformation Tools, and Simulation tool are the four layers it comprises of. Meta-modeling environment [59] contains the Eclipse Oxygen which is an open-source environment having variety of plugin available in market for software development and modeling purpose. Modeling editor layer consists of Eclipse modeling plugin i.e. Papyrus [60]. It is a model-based IDE which is used to realize the UML profiling mechanism. It is easily extensible as it is based on the principles of UML Profiles. Our proposed profile, UUWSN, is developed in papyrus editor. Modeling transformation layer consists of transformation tool Acceleo. Acceleo [61] is an open-source code generator that allows people to use model driven approach for developing applications. It can also be used to convert models to text (M2T) and is available as an Eclipse plugin. UWSNTE, our proposed transformation engine, is developed in Acceleo. A tool for testing and deploying the converted or generated code is included in the fourth layer. AquaSim NS3 [62] is a discrete event driven network simulator that supports a wide range of protocols and features for modeling underwater wireless sensor networks. We have used AquaSim NS3 for validating the generated code from transformation engine "UWSNTE".



3.2. PUWSN (UML Profile for Underwater Wireless Sensor Networks)

In the domain of software engineering, the Unified Modeling Language (UML) is a general-purpose, developmental modeling language that is designed to provide a standard way to depict a system's architecture. It is defined within the framework of OMG's four-layer metamodeling architecture (Object Management Group). The language for specifying the *meta model layer* is defined by the topmost *meta-meta model layer*. The meta model layer, in turn, defines legal specifications in a modeling language; the UML meta model, for example, defines legal UML specifications. Models of individual software systems are defined in the *model layer*. The *user objects layer*, on the other hand, is used to create specific instances of a model [63].

The profile technique can be used to extend UML and turn it into a domain-specific language. UML Profile is a framework for extending and customizing UML by adding additional concepts, attributes, and semantics to make it more appropriate for a specific domain. Profile uses the notation of a package with a keyword <<profile>> written before name of package. Stereotypes, constraints, and tagged values constitute the majority of a UML Profile. In our proposed framework, we have used UML profiling method for specializing the general constructs of existing modeling language and refining refine them according to our domain i.e., underwater wireless sensor network. The complete overview of proposed UML profile is shown in Figure 3.3.

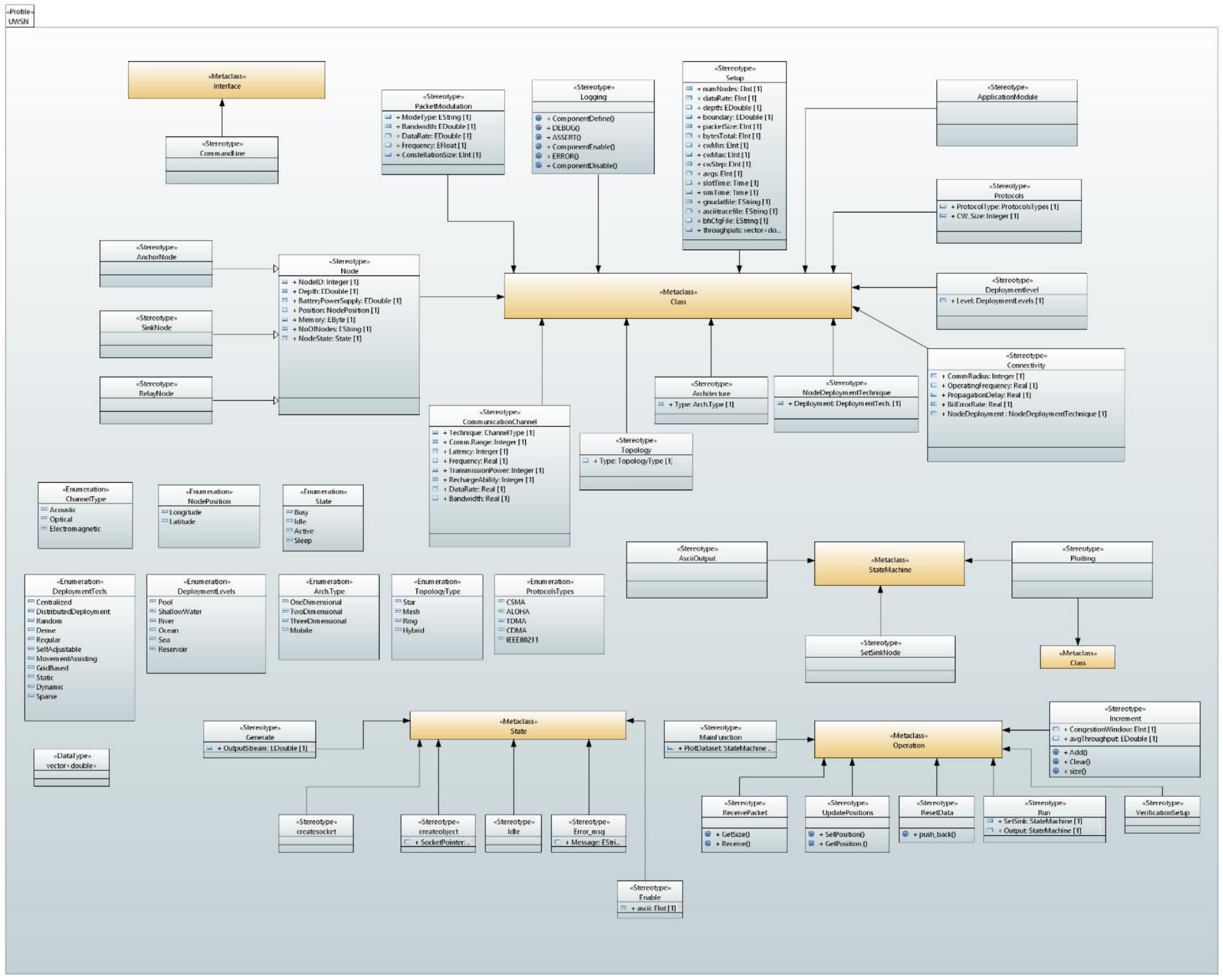


Figure 3.3: PUWSN (UML Profile for Underwater Wireless Sensor Networks)

The Papyrus modeling editor tool has been used to model PUWSN. It is a Java-based eclipse plug-in. Multiple stereotypes are included in the proposed profile to facilitate in the introduction of underwater wireless sensor network associated domain concepts into UML modeling. PUWSN has been divided into five sections to make it easier to understand. Sections are defined logically on the basis of UWSN architecture.

- Structural Concepts
- Verification Related Concepts
- Behavior Level Stereotypes

➢ Operation Level Concept

3.2.1. Structure Level Concepts

Figure 3.4 represents the stereotypes introduced to model the structure concepts of underwater network. This section of profile contains 9 stereotypes i.e. *Node*, *CommunicationChannel*, *Topology*, *Architecture*, *NodeDeploymentTechnique*, *Connectivity*, *Deploymentlevel*, *Protocols and PacketModulation.* These are derived from the named class meta-class in UML.

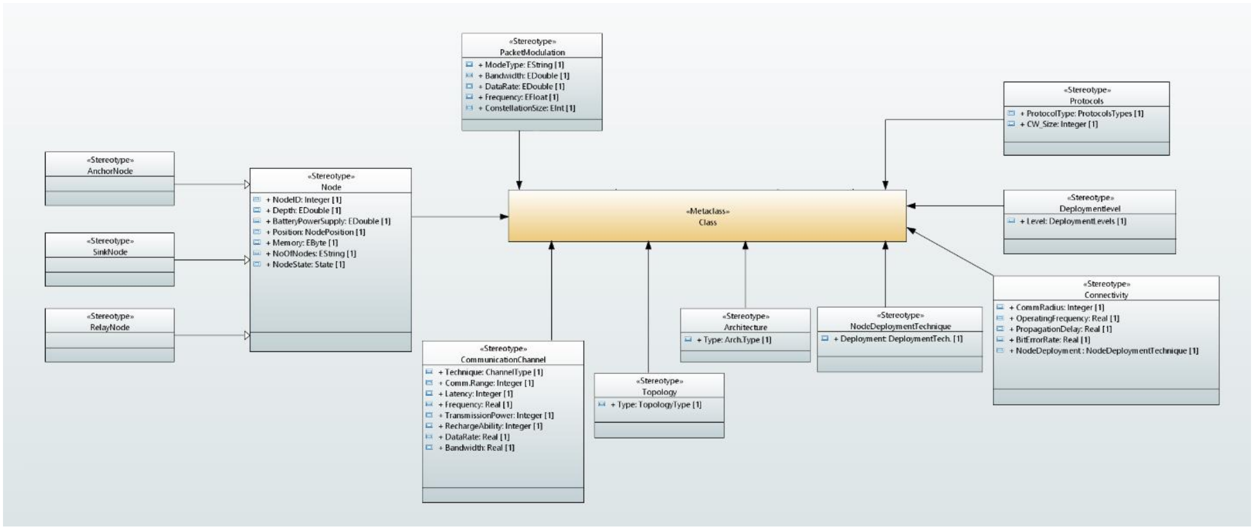


Figure 3.4: Structural Concepts

<<Node>>

Node stereotype represents "Node" concept of UWSN. It is a UML meta-class named class that has been extended. The node is in charge of communicating and exchanging data with other nodes in the network as well as the base station. In UWSNs, three types of nodes are commonly utilized i.e. AnchorNode, SinkNode and RelayNode. *AnchorNode* is the sensing node which already know their location through some method like GPS. The *Sink node* is in responsible for collecting data from all sensor nodes and transmitting it to the monitoring center. A sensor node that receives data from a source node or another relay node and passes it to a sink node or another relay node is specified by the stereotype *RelayNode*. Every Node has assigned a *NodeID* which represents the unique id of each node. The tag value *Depth* defines the depth of each node. *BatteryPowerSupply* tag value defines the assigned battery power for each node in a network. The tag value *Position* refers to the estimated position of each sensor node within a network with respect to their Longitude and Latitude. The data type of Position is modeled using an Enumeration. A battery is installed in each UWSN sensor node. It is modeled using the property named *Memory* having value of type Ebyte. There are specific number of nodes in underwater setup which is define using the tag value *NoOfNodes*. While each node has a state which is modeled using the property named *NodeState* having value of type *State*. The data type of NodeState is modeled using an Enumeration. The enumeration class *State* defines several enumeration literals i.e., Active, Idle, Sleep and Busy.

<< *CommunicationChannel* >>

CommunicationChannel stereotype represents "Communication Channel" concept of UWSNs. It is a UML meta-class named class that has been extended. This stereotype encapsulates the transmission medium notion of UWSNs, which are made up of sensors and autonomous underwater vehicles (AUVs) that interact, coordinate, and share data in order to perform sensing and monitoring activities.Each Communication Channel has some properties which are considered for smooth data transmission. The tag value *Technique* defines type of channel used for communication and having value of type ChannelType. The data type of *Technique* is modeled using an enumeration class ChannelType defines several enumeration literals i.e., acoustic, optical, and RF. Each channel has a communication range which is modeled using the property named *Comm.Range*. The tag value *Latency* defines the time it takes for some data to get to its destination across the network. The tag value *Frequency* defines the assigned frequency for communication. Power for transmission is defined by a property named *TransmissionPower*. While the tag value *RechargeAbility* defines the ability of channel to recharge. Transmission speed of channel is defined by property *DataRate* having value of type Real. Maximum rate of data transfer across a network is defined by property *Bandwidth* having value of type Real.

<< *Topology*>>

Topology stereotype represents the manner in which the nodes, devices and links of an underwater network are arranged to relate to each other. It is a UML meta-class named class that has been extended. The types of topologies are defined by a tag value *Type*, which has a value of type *TopologyType*. The data type of *TopologyType* is modeled using an Enumeration. This enumeration class defines several enumeration literals i.e., star, mesh, ring and hybrid.

<< *Architecture*>>

Architecture stereotype represents "Communication Architectures" concept of UWSNs. It is a UML meta-class named class that has been extended. This stereotype refers to an underwater network in which a collection of sensor nodes (cluster) is installed. The attribute *Type* defines the architecture type in underwater network, which has a value of type *Arch.Type*. Four types of architecture are present which are defined in the enumeration Arch.Type which is discussed in Section 3.2.4.

<< NodeDeploymentTechnique>>

NodeDeploymentTechnique stereotype represents "Node Deployment Technique" concept of UWSN. It is a UML meta-class named class that has been extended. Underwater node sensor deployment is a very crucial process as it decides the achievable sensing, network connectivity and communication coverage. The type of available technique is defined by a tag value *Deployment*, which has a value of type *DeploymentTech.* The data type of *DeploymentTech* is modeled using an Enumeration. This enumeration class defines several enumeration literals.

<< Connectivity>>

The number of nodes that can communicate with the base station divided by the total number of nodes is known as the *connectivity* stereotype. It is a UML meta-class named class that has been extended. Connectivity is a metric that defines the ease with which the nodes can connect to the surface station. It ensures that the detected event is conveyed to base station. Connectivity depends on various attributes like node deployment, transmission power, operating frequency, propagation delay, communication radius, the internodal distances and bit error rate. The node deployment techniques are defined by a tag value *NodeDeployment*, which has a value of type *NodeDeploymentTechnique.*

<< Deploymentlevel>>

Deploymentlevel stereotype represents "nodes deployment at different water levels" concept of UWSN. It is a UML meta-class named class that has been extended. Deploymentlevel is an important concept in UWSNs which defines possible deployment levels involves adjusting the depths of nodes in water. The attribute *Level* defines different deployment levels for sensing nodes in underwater network, which has a value of type *DeploymentLevels*. The data type of

DeploymentLevels is modeled using an Enumeration. The enumeration class DeploymentLevels defines several enumeration literals.

<< *Protocols*>>

Protocols stereotype represents “Routing Protocols” concept of UWSN. It is a UML meta-class named class that has been extended. This stereotype is capable of efficiently transferring data from the source node to the network's destination node. Different *ProtocolType* optimally picks the paths of routing in regard to energy consumption and hence contributes to the network's longevity. The data type of *ProtocolsTypes* is modeled using an Enumeration which defines several enumeration literals. While the attribute *CW_Size* defines congestion window size that limits the amount of data that can send into the network before receiving an ACK.

<< *PacketModulation*>>

PacketModulation stereotype is extended from UML meta-class named Class. This stereotype captures abstraction of packet modulation information. The attribute *ModeType* defines the modulation techniques for optimized transmission. The tag value *Bandwidth* defines the bandwidth of transmitted signal. *Frequency* captures center frequency of transmitted signal. Number of constellation points in modulation is defined by property *ConstellationSize.*

3.2.2. Verification Related Concepts

This section of PUWSN profile represents *Verification Related Concepts* of UWSNs framework as shown in Figure 3.5. There are three main stereotypes defined in this section *ApplicationModule, Setup* and *Logging*.

<< *Setup*>>

Setup stereotype is extended from UML meta-class named Class. Setup stereotype in UWSN framework is known to be the “verification component”. It captures design level objects of a network to verify the correctness. Verification setup has predefined size of *boundary* area in meters. There are number of nodes (*numNodes)* which will perform the specific monitoring tasks and are placed at estimated transmitting *depth*. These nodes have transmission speed which is define using the tag value *dataRate*. Each packet receives or send by sensing nodes has *packet size* and the total bytes received during setup is defined using the tag value *bytesTotal*. Implementing a backoff rule improves the performance of MAC protocols. Each node utilizes the parameters

CW (contention window size), *cwStep*. *cwMax* , *cwMin* and slot duration named *slotTime* for backoff purposes. The *avgs* attribute specifies the number of topologies to test for each cw point, as well as the duration per trial (*simTime*) and the *throughputs* for each run. Further the GNU Plot output of setup environment is defined by a tag value *gnudatfile*. A tag value named *asciitracefile* has been provided to generate standard tracing for popular tracing sources and to modify which objects generate the tracing. Generic preference file i.e., CFG that stores settings and configuration information is defined by a tag value *bhCfgFile*.

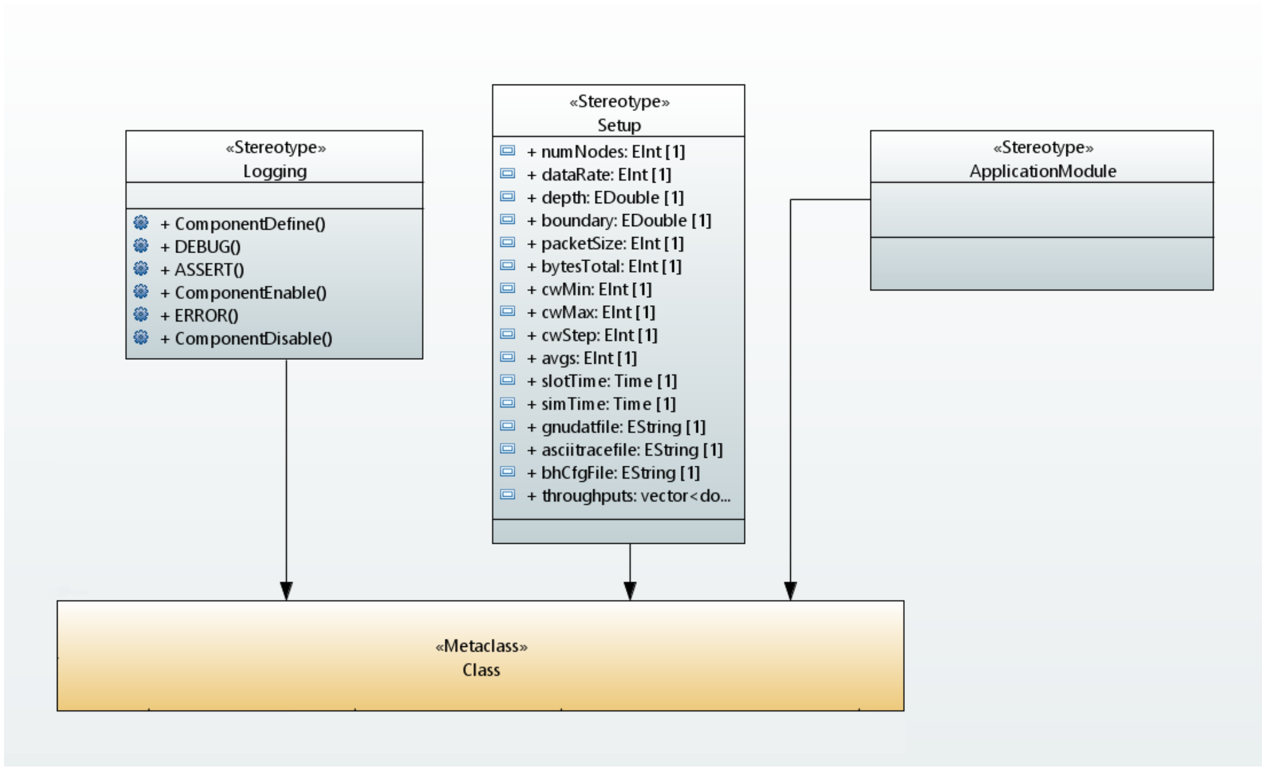


Figure 3.5: Verification Related Concepts

<< *ApplicationModule*>>

ApplicationModule stereotype is extended from UML meta-class named class. This stereotype provides a streamlined environment to model and run the functions of underwater network efficiently. It's an all-in-one package for an underwater network application that can run on its own. This module encapsulates the application binaries, as well as the software dependencies and hardware requirements, into a single, self-contained package.

<< *Logging*>>

Logging stereotype represents arbitrary messages at a specific log level in UWSN. It is a UML meta-class named class that has been extended. Mainly, this stereotype is defined to allow developers to send information out on screen and can be used to monitor or debug the progress of underwater experiments. The *ComponentDefine* attribute specifies the name of a log component.It used at the top of every file in which we want to use the NS_LOG macro. Behavioral aspects of log component are modeled using six functions i.e. *ComponentDefine()*, *DEBUG()*, *ASSERT()*, *ComponentEnable(),ERROR() and ComponentEnable().*

3.2.3. Behavior Level Stereotypes

UML State machine meta-model is extended to propose stereotypes, in order to add diverse behavioral elements of underwater wireless networks in PUWSN. As illustrated in Figure 3.6, three stereotypes are extended from the meta-class "State Machine" and six stereotypes are extended from the meta-class "state."

<<*SetSinkNode*>>

SetSinkNode is applied on UML state machine. This stereotype is defined to represent “setting sink position” function of metaclass operation *Run*. The *Run* operation receives unique identifier of a sink node from an interface and creating socket instances. This can be used as an interface in a node to generate PacketSockets that can be used to connect to network devices. An app can connect to a network device using a PacketSocket. The data buffers are provided by the application, the socket converts them to raw packets, and the network device adds the protocol specific headers and trailers. Actual logic of *Run* function is modeled using the state machine diagram.

<< *AsciiOutput* >>

AsciiOutput is applied on UML state machine. This stereotype is defined to represent “enable ascii output” function of metaclass operation *Run*. Actual logic of *Run* function can be modeled using the state machine diagram. This feature surrounds the low-level tracing system in order to assist us with the finer points of configuring some easily understandable packet traces. We will see output in ASCII files if we enable this functionality. It is based on the ideas of distinct tracking sources and sinks, as well as a standardized technique for connecting sources and sinks.

<< *Plotting*>>

It is applied on UML state machine and is a UML meta-class named class that has been extended. This stereotype is defined to represent “Plotting with GNUplot” function of metaclass operation *MainFunction*. From a set of datasets, this generates gnuplot ready plotting commands. The actual logic of plotting is implemented by using state machine diagram.

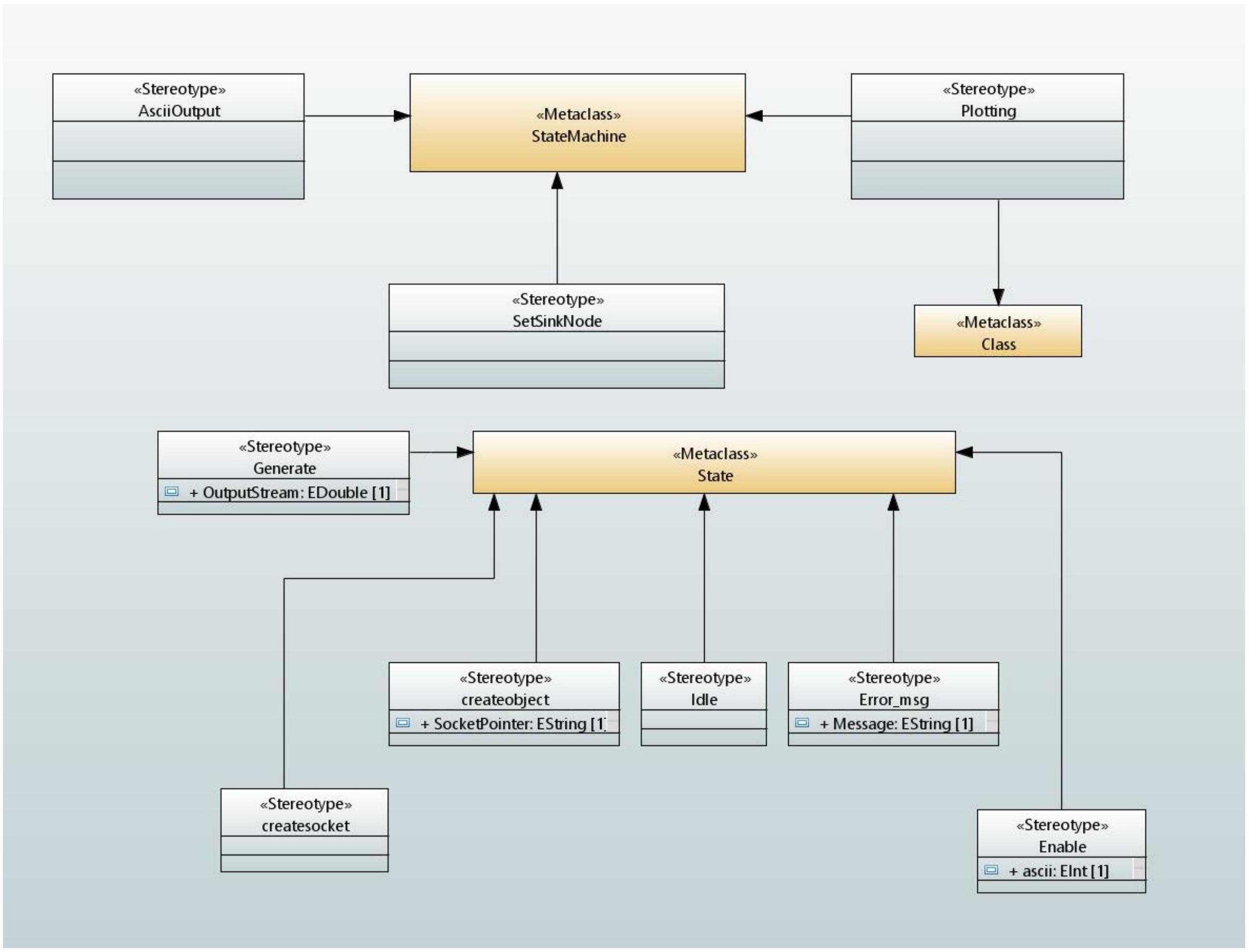


Figure 3.6: Behavior Level Stereotypes

<<Idle>>

Idle stereotype is extended from meta-class named state. It represents the system's conceptual state in accordance with the specifications. Adding conceptual state in PUWSN not only assists in modeling of conceptual state of system but also helps in transformation.

<< *Error_msg* >>

Error_msg stereotype is extended from meta-class named state. In run and main function, the errors of conceptual states are represented by message. To implement that behavior this, stereotype i.e., error message is proposed in PUWSN. In order to determine and display the specific error, attribute *Message* is defined. It has value of type *Estrings*.

<< *Enable*>>

Enable stereotype is extended from meta-class named state. In run function, the ASCII file of conceptual states are enabled on each device to test the output behavior of system. To implement that behavior this, stereotype i.e. enable is proposed in PUWSN.

<< *Generate*>>

Generate stereotype is extended from meta-class named state. In main function, the 2D points plot of conceptual state are generated from a set of datasets. It creates a single output stream that contains both gnuplot commands and data values. To implement that behavior this, stereotype i.e., enable is proposed in PUWSN.

<< *createsocket*>>

createsocket stereotype is extended from meta-class named state. A SocketFactory supplied by TypeId performs the generation of conceptual state sockets on a particular node in the run function. To implement that behavior this stereotype is proposed in PUWSN.

<< *createobject*>>

createobject stereotype is extended from meta-class named state. Every conceptual state of system involves creating and manipulating objects. Therefore, this stereotype is extended from state to represent the objects being used in the conceptual state of system.

3.2.4. Operation Level Concepts

This section of PUWSN profile represents *operations* of UWSNs framework as shown in Figure 3.7. There are seven main stereotypes defined in this section *ReceivePacket*, *UpdatePositions*, *ResetData*, *Run*, *VerificationSetup*, *MainFunction* and Increment.

<< *ReceivePacket*>>

ReceivePacket stereotype represents “Receive Packet” concept of UWSN. It is a UML meta-class named Operation that has been extended. Main task of this stereotype is to keep information of all available received packets from a socket. Behavioral aspects of this operation are modeled using two functions i.e. *GetSize()* which returns the size of the received packet in bytes and *Receive()* which read a single packet from the socket.

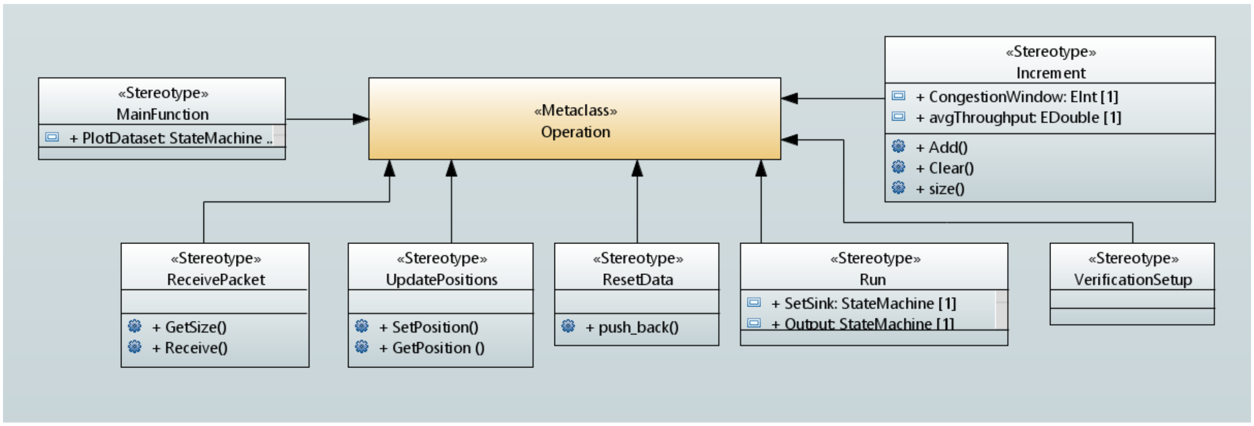

«Stereotype» MainFunction
+ PlotDataset: StateMachine ..
«Metaclass» Operation
«Stereotype» Increment
+ CongestionWindow: EInt [1]
+ avgThroughput: EDouble [1]
+ Add()
+ Clear()
+ size()
«Stereotype» ReceivePacket
+ GetSize()
+ Receive()
«Stereotype» UpdatePositions
+ SetPosition()
+ GetPosition ()
«Stereotype» ResetData
+ push_back()
«Stereotype» Run
+ SetSink: StateMachine [1]
+ Output: StateMachine [1]
«Stereotype» VerificationSetup

Figure 3.7: Operation Level Concepts

<< *ResetData*>>

ResetData stereotype represents “Reset Data” concept of UWSN. It is a UML meta-class named Operation that has been extended. *ResetData* saves the throughput of verification setup from a single run. This stereotype push the calculated throughput into a vector from the back and increases size of vector by one using a function *push_back()*.

<< *Run*>>

Run stereotype represents “Run an experiment” concept of UWSN. It is a UML meta-class named Operation that has been extended. Run is a function in UWSN framework that is known to be the “verification component”. It sets the parameter to configure nodes in the model then returns data set of values and measured throughput. In order to model the behavior of Run operation, two attributes named SetSink and Output is specified. To invoke the UML State Machine diagram, the SetSink attribute is defined. It refers to the state machine diagram, which is used to simulate the behavior of the SetSink run operation. While the Output attribute is used to invoke a UML State Machine diagram that models the behavior of ascii output operations.

<< *MainFunction*>>

MainFunction stereotype represents “Main functionality” concept of UWSN. It is a UML meta-class named Operation that has been extended. MainFunction is responsible for controlling program execution of design via its interface. Main behavior of this operation, for defining the actual logic of running the verification on CommandLine interface. In order to model the behavior of MainFunction operation, attribute named PlotDataset is specified. The UML State Machine diagram is called using the PlotDataset attribute. It refers to the state machine diagram, which is used to depict the behavior of a 2D plotting operation.

<< Increment>>

Increment stereotype represents "Increment value" concept of UWSN. It is a UML meta-class named Operation that has been extended. The computation of average throughput for a group of runs is represented by *Increment*, followed by increment contention window size. The attribute *CongestionWindow* defines the CW value for completed runs. It has value of type *EInt*. While the attribute *avgThroughput* defines the calculated throughput for each run. Behavioral aspects of this operation are modeled using three functions i.e. *Add()* which appends the values of *CongestionWindow* and *avgThroughput* to a container for the verification data. *Clear()* deletes the previous values of throughput and sets size to zero. For each cw point, *size()* returns the throughput size and compares it to the number of topologies to test.

3.2.5. Data types and Enumerations

The **Figure 3.8** shows all the enumerations and data type used for modeling of entire UWSN concepts discussed above. Enumeration ChannelType represents three types of communication technologies which are frequently used in UWSNs i.e., acoustic, optical and electromagnetic. Enumeration DeploymentTech. defines different techniques applied during deployment of nodes i.e., centralized, random, dense, regular, self-adjustable, GridBased, DistributedDeployment etc. Enumeration DeploymentLevels deals with types of deployment areas i.e., pool, shallow water, river, ocean, sea and reservoir. Enumeration Arch.Type defines four types of UWSNs communication architectures i.e., one dimensional, two dimensional, three dimensional and mobile. Enumeration State represents four types of nodes states which are scheduled to avoid energy wasting in UWSNs i.e., Busy, Idle, Active and Sleep. Enumeration ProtocolTypes defines different types of protocols which optimally picks the paths of routing i.e., CSMA, ALOHA, TDMA, CDMA etc. And primitive type defines the data types of the attributes used in modeling of above-mentioned concepts. While an enumeration Topology type defines different topologies in which the nodes, devices and links of an underwater network are arranged to relate to each other. i.e., star, mesh, ring and hybrid.

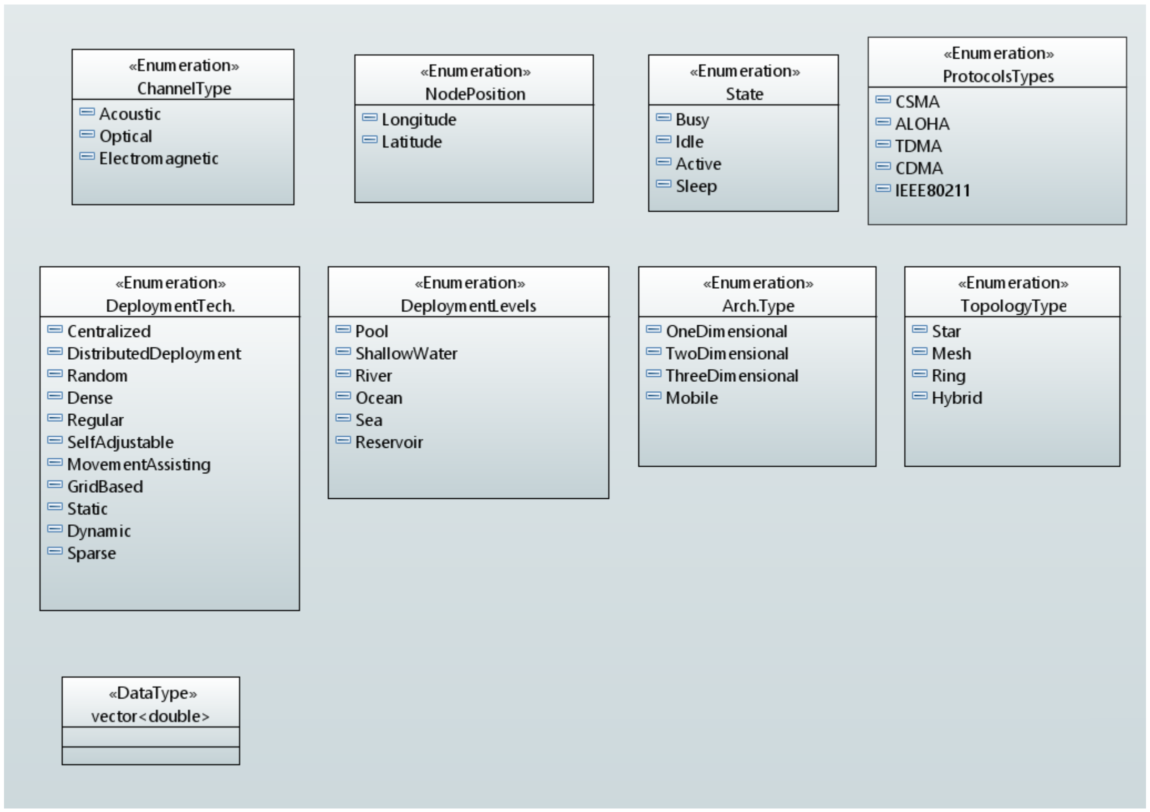


Figure 3.8 Data types and Enumerations

Chapter 4

Implementation

CHAPTER 4: IMPLEMENTATION

This chapter primarily focuses on the applied implementation strategy. It provides detail about the **UWSNTE (**Model-Driven UWSN Transformation Engine) –our proposed transformation engine that generates design verification c++ code from UML model for underwater wireless sensor applications. This chapter provides details about the architecture of transformation engine in **Section 4.1** and the transformation rules that we applied are discussed in **Section 4.2.**

4.1. Architecture of Transformation Engine

Figure 4.1 depicts a visual representation of our architecture. UWSNTE accepts input models and uses transformation rules to convert them to c++ code. The User Interface (UI) and Transformer are both required components of this transformation engine. Acceleo, an open-source Eclipse plugin for Model-to-Text (M2T) transformation, is used to carry out the transformation. JAVA and the Acceleo transformation language are the languages utilized for transformation.

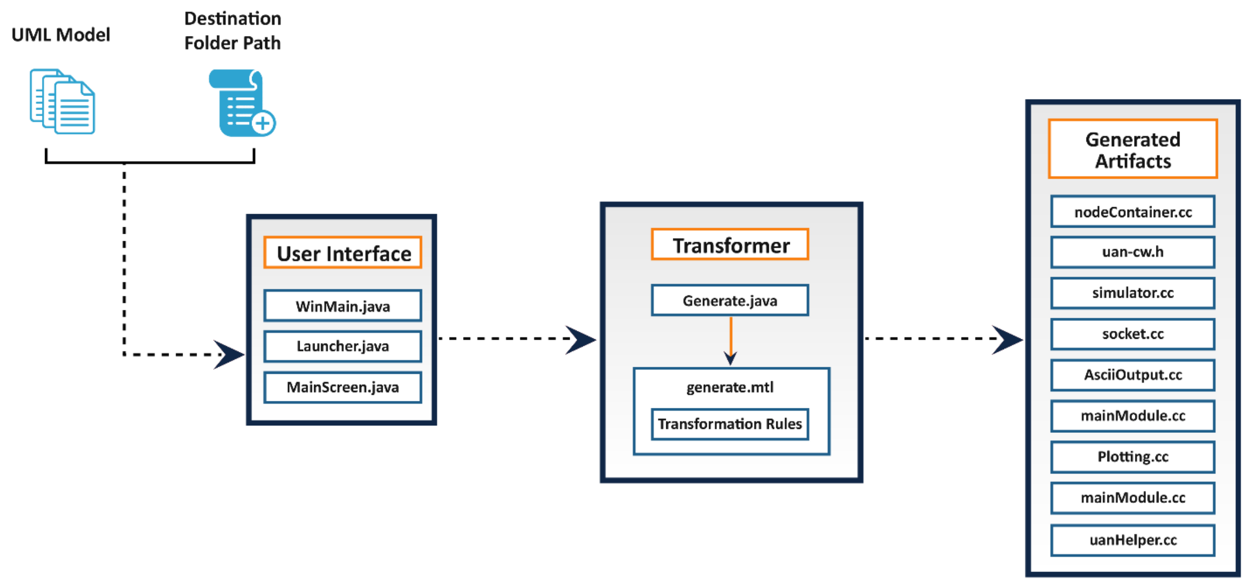


Figure 4.1: Architecture of Transformation Engine

4.1.1. User Interface

Figure 4.2 represents the main interface of our tool UWSNTE. It has three components: a system modeler, a transformation engine, and a help document. *System Modeler* allows users to utilize the

Papyrus tool in order to model UWSN domain applications. Second, the *Transformation Engine* option enables the user to convert input models into deployable code.Lastly, *Help Document* option of interface opens a “help document”.

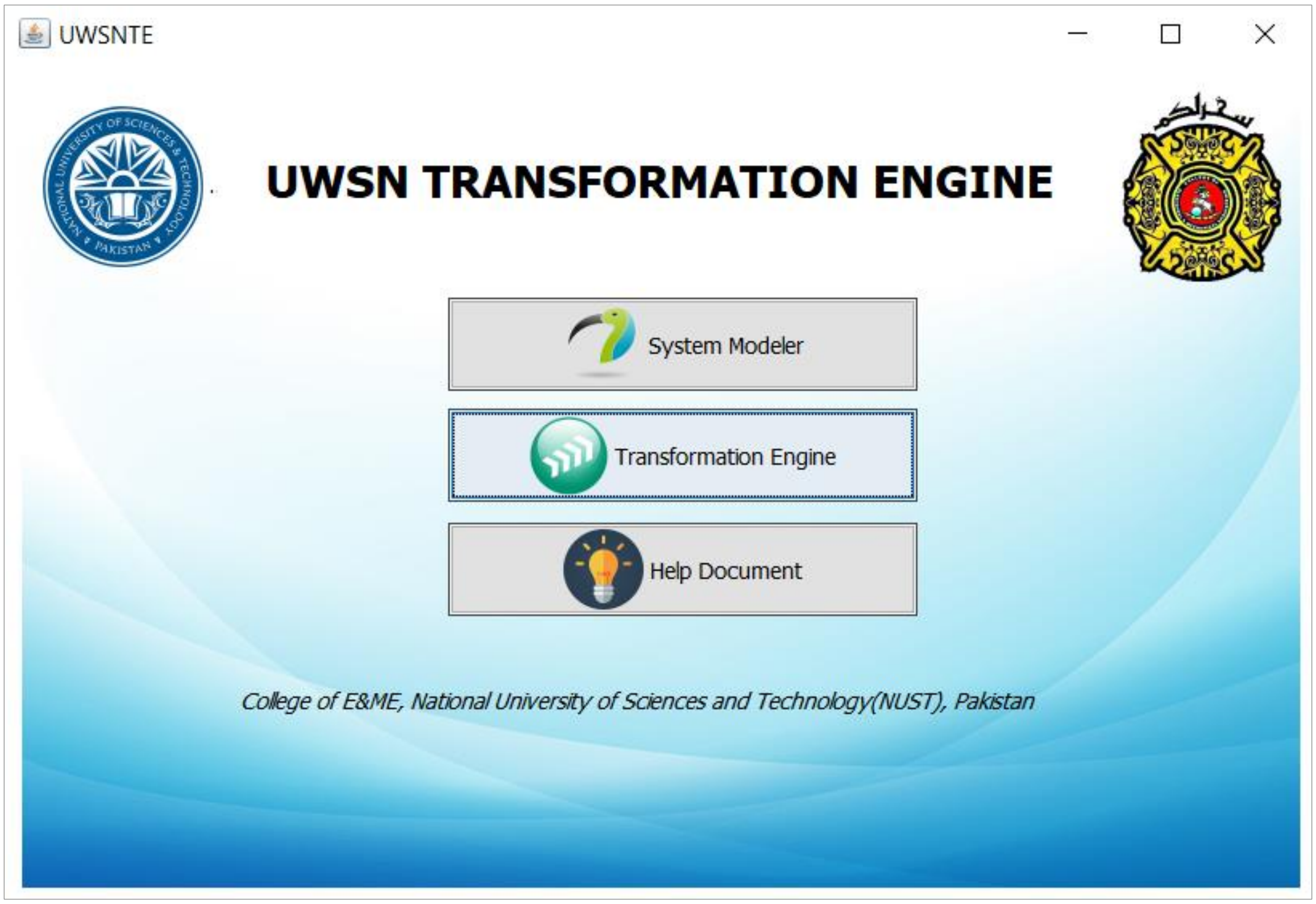

UWSNTE
UWSN TRANSFORMATION ENGINE
System Modeler
Transformation Engine
Help Document
College of E&ME, National University of Sciences and Technology(NUST), Pakistan

Figure 4.2: Main Screen of UWSNTE

The interface in Figure 4.3 provides an option for user to input the model of system to transform. This interface is based on Input Model, Destination Folder, Generate, Transformation Status, Reset and Open Folder options. The user can browse and pick available .uml files in his system using the Input Model. By using browse button in front of Destination Folder user can define the destination location for the generated code. The Generate button generates code files automatically based on the input model. Reset button allows user to reset input fields’ values. Transformation Status displays the current state of the tool, indicating whether files were successfully created or if an issue occurred. The open folder command takes you to the folder where the output files were created. Open generated files, if checked, opens all the generated files when the transformation is completed.

UWSNTE's user interface is comprised of three primary Java classes: Launcher, MainScreen, and WinMain. The transformation engine's main executor is MainScreen. It includes a list of accessible functions as well as a graphical user interface (GUI) with buttons and input fields. The java-based controller classes that implement these features are Launch and WinMain.

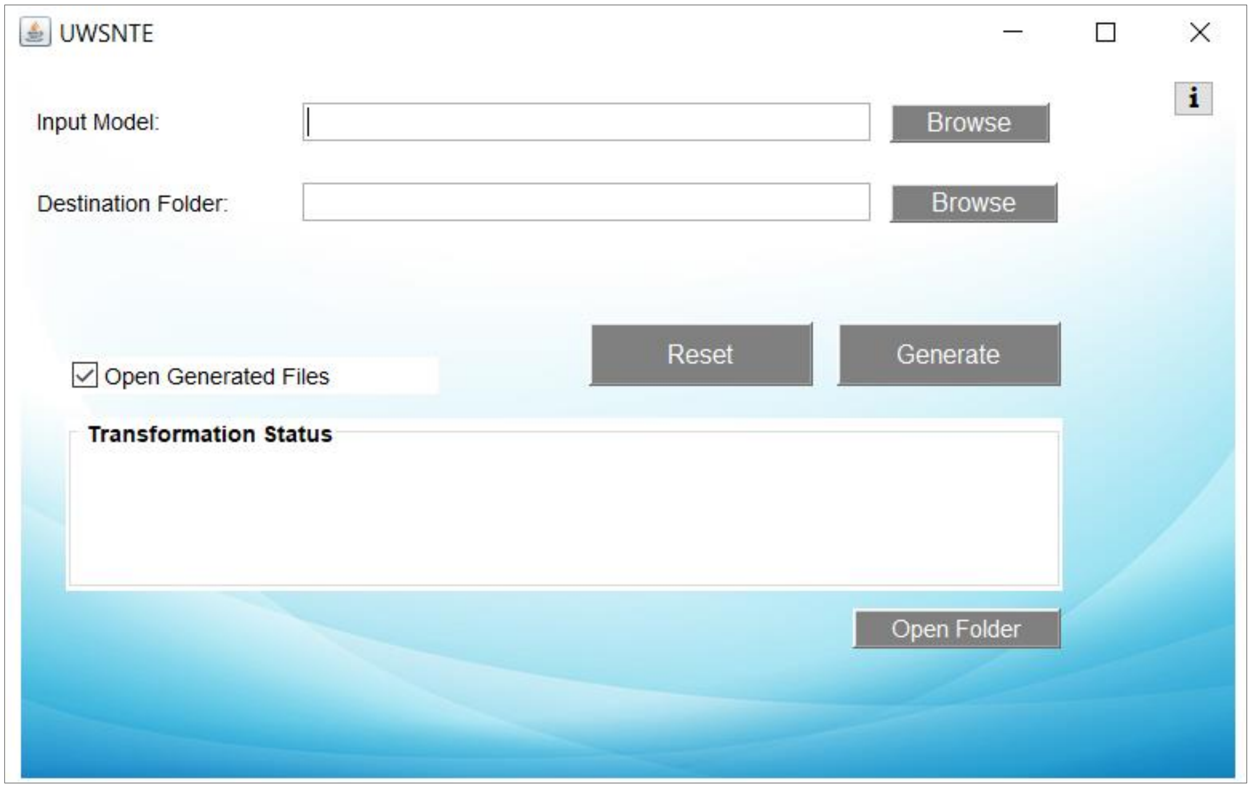


Figure 4.3: Input Interface of UWSNTE

4.1.2. Transformer

The generate.mtl file consumes the user's UML model and uses Acceleo code to transform it into the appropriate artifacts as shown in Figure 4.1. To implement transformation rules, Transformer relies on two primary components: Generate (Generate.java) and Template (generate.mtl) files. The Template file is the major component, and it is made up of several sub templates. It gets the input models and sends them to the appropriate sub templates. To generate the result, each sub template applies its transformation rules to each UML model element. The generated code in the target folder, specified by the user, consists of code written in *C++* language. These generated codes are then imported into AquaSim NS3 to simulate and verify the design.

4.2. Transformation Rules

In this section transformation rules are discussed which are used to map the model and transform it into desired system code as shown in Figure 4.1. The process of modifying or converting a model into text or another model is known as model transformation and the predefined formal rules which are used to transform the concept of one model to another model or text are usually denoted as transformation rules. Transformation rules are the building blocks essential for converting UML models into desired UWSN code [64]. The main aim of transformation is focused to develop the set of formal rules to ensure model continuity and to reduction of information loss and effort as much as possible.

4.2.1. Transformation Rules for Code Generation

Transformation of a model to text or code is usually denoted as M2T transformation and the focus is to develop the textual or code artifacts from the model. The engine which is used to carry out M2T process is known as Transformer. Mapping rules used to transform the Model into Code is provided and discussed in this section. Transformation rules used to transform the model artifacts into respective code artifacts are defined in Table 4.1.We have used UML state machine diagram for behavioral representation of UWSN components and Class diagram for structural representation.

Table 4.1: Transformation Rules for Model to Text (M2T)

Sr#	Model Artifacts	Code Artifacts	Mapping
1	**Package**		
	Package Name	Folder	Package—Name ➔ Folder Name
2	**Model Class**		
	Class Name	Class	Association—Member ends ➔ Owning Class/ Owned Class.
	Visibility		Model Class—Visibility ➔ Access specifiers class
	Attribute		Model Class—Attribute ➔ Owned Attributes

	Attribute Visibility		Model Class—Attribute Visibility ➔ Owned Attribute Access Specifiers
	Applied Stereotype		Model Class—Applied Stereotype—Property ➔ Owned Attributes
3	**Association**		
	Member Ends	Class Instance	Association—Member ends ➔ Owning Class/ Owned Class
	Member Ends Visibility	public/private/ protected	Association—Member ends visibility ➔ Owned Instance Access specifier.
	Member Ends Multiplicity	List<Type> Instance Name OR Single Instance	Association—Member ends Multiplicity/ Cardinality ➔ List<Type> or Single Instance.
4	**Operation**		
	Operation Name	Method	Operation—Name ➔ Method Name.
	Owned Parameter	Method Definition	Operation—Owned Parameter ➔ Method Parameter (in, out), Return Type of Method (return).
	Opaque Behavior		Opaque Behavior—Description ➔ Body of Method.
5	**Enumeration**		
	Enumeration Name	enum Enumeration	Enumeration—Name ➔ enum Name
	Owned Literal	Enumeration Literal	Enumeration—Owned Literal ➔ Enum Values.
	Generalization/Specialization		
	Generalization/ Specialization		Generalization/ specialization ➔ Generalization / Specialization
6	**State Machine**		
	State Machine	Function	State Machine ➔Class Function

	States		State ➔ Function Bodies
	Choice (Pseudo state)		Choice ➔ if/else

The concept of mapping of a source model to a desired target textual artifact is termed as Model-to-Text (M2T) transformation. *Model Artifacts* for modeling underwater wireless sensor systems are founded on components of the Unified Modelling Language (UML). *Code Artifacts* are c++ code components that are used to translate UML model elements into text during transformation. *Mapping* connects UML model elements to their code counterparts. In the Unified Modeling Language (UML), a *Package* is an element that is used to group other elements and determine the hierarchical arrangement of elements. The name of the package is transferred to the name of the folder containing code files during transformation. In UML, *Model Classes* represent the system's static structure. The Model Class's name is mapped to the class's name in the code artifact. The access specifier of the code class is transferred to the *visibility* of the Model Class. Property and Visibility of the *Applied Stereotype* are mapped to attributes and access specifiers of the class's attributes. In UML, *associations* form semantic relationships, and their Member ends are mapped to owing and owned class instances. The instance access specifier and multiplicity of containing instances, i.e. 1 as single instance and 0...* as List instance>, are mapped to the visibility and multiplicity or Cardinality of association. *Operations* define the behavioral features of Model Classes and mapped to methods of classes. The name of the method is mapped to the name of the operation, and the name of *owned parameters* is mapped to the name of the parameters. The direction of owned parameters is mapped to the kind of parameter, such as in, out, inout, and return. *Opaque Behavior* is a UML semantic that is implementation specific. The body of the method or the method definition is mapped to the description of opaque behavior In UML, an *Enumeration* is a user-defined set of named elements. *Enumeration Literal* is translated to user defined Enum values in code, while Enumeration Name is mapped to name of Enum in code. *State Machine* provides logical mapping of the functions to desired output UWSN elements and mapped to function of classes. It is a behavior model that groups all possible system occurrences, called *states*. States are described as a condition in which a UML object exists and changes when an event is triggered, and function bodies are mapped to it. While a *Choice (pseudo state)* is a control element that influences the sequence of events in a state machine, and it is mapped to if/else choices in code.

4.2.2. Transformation Rules for Behavior Level Stereotypes

This section describes the transformation of behavior level stereotypes into low level UWSN code. Behavior level stereotypes help in transformation of three important functions in UWSN framework. These functions are 1) SetSinkNode 2) AsciiOutput 3) Plotting. For the actual implementation of the body of aforementioned functions, these stereotypes are used. Table 4.2 provides logical mapping of the stereotypes to desired output UWSN elements.

Table 4.2: Behavior Level Stereotypes Transformation

Sr#	Stereotype	Transformation Rule
1	<< SetSinkNode>>	SetSinkNode ➔ Body of setting sink position method
2	<< AsciiOutput >>	AsciiOutput ➔ Body of enable Ascii output method
3	<< Plotting>>	Plotting ➔ Body of plotting method
4	<< idle>>	<< idle>> ➔ represents initialization of behavior
5	<< Enable>>	<< Enable >> ➔ represents enabling certain behavior in that specific state on which this stereotype is applied
6	<< Generate>>	<< Generate>> ➔ represents generating certain behavior in that specific state on which this stereotype is applied
7	<< Error_msg>>	<< Error_msg>> ➔ represents messages being used in that state
8	<< createsocket>>	<< createsocket>> ➔ represents creating socket for certain behavior in that specific state on which this stereotype is applied
9	<< createobject>>	<<createobject>> ➔ represents creating objects for certain behavior in that specific state on which this stereotype is applied

4.2.3. Transformation Rules for Class Level and Operation Level Stereotypes

In this section we have discussed the transformation of the stereotypes designed specifically for class level and operation level stereotypes. Mapping rules are mentioned in Table 4.3.

There are total 9 classes in UWSN which are defined as structural elements of UWSN. Node, CommunicationChannel, Topology, Architecture, NodeDeploymentTechnique, Connectivity, Deploymentlevel, Protocols and PacketModulation stereotypes help in generation of structure classes. As all previously mentioned classes are structure elements therefore transformation rules defined in Section 4.2.2 Serial 1-9 are applicable on them.

Table 4.3: Class Level and Operation Level Stereotypes Transformation

Sr #	Stereotype	Transformation Rule
1	<< Node>>	▪ Maps node attributes in underwater network system ▪ Class must extend ➔ UWSN_Node class ▪ NodeID ➔ Unique identifier of each node ▪ Depth ➔ depth of each node ▪ BatteryPowerSupply ➔ assigned battery power for each node ▪ Position ➔ estimated position of each sensor node ▪ Memory ➔ assigned memory ▪ NoOfNodes ➔ specific number of nodes ▪ NodeState ➔ current state of each node
2	<<CommunicationChannel>>	▪ Maps transmission medium concept of UWSN ▪ Class must extend ➔ UWSN_Communication-Channel class ▪ Technique ➔ communication channel type ▪ Comm.Range ➔ communication range ▪ Latency ➔ time it takes for some data to get to its destination ▪ Frequency ➔ assigned frequency ▪ TransmissionPower ➔ power for transmission ▪ RechargeAbility ➔ ability of channel to recharge ▪ DataRate ➔ Transmission speed of channel ▪ Bandwidth➔ maximum data transfer rate
3	<< Topology>>	▪ Class must extend ➔ UWSN_Topology class ▪ Type ➔ types of topologies

4	<< Architecture>>	▪ Class must extend ➔ UWSN_Architecture class ▪ Type ➔ architecture types
5	<<NodeDeploymentTechnique>>	▪ Class must extend ➔ UWSN_Node-DeploymentTechnique class ▪ Type ➔ types of available deployment technique
6	<<Connectivity>>	▪ Class must extend ➔ UWSN_Connectivity class ▪ CommRadius, NodeDeployment, BitErrorRate, OperatingFrequency, PropagationDelay ➔ Class members
7	<< Deploymentlevel >>	▪ Class must extend ➔ UWSN_Deploymentlevel class ▪ Level ➔ deployment levels for sensing nodes
8	<< Protocols>>	▪ Class must extend ➔ UWSN_Protocols class ▪ ProtocolType, CW_Size ➔ Class members
9	<< PacketModulation >>	▪ Class must extend ➔ UWSN_PacketModulation class ▪ ModeType, Bandwidth, Frequency, DataRate, ConstellationSize ➔ Class members
10	<< Setup>>	▪ Maps verification settings in underwater network system ▪ Class must extend ➔ UWSN_Setup class ▪ numNodes, depth, packetSize, dataRate, bytesTotal, cwStep, cwMax, cwMin, slotTime, simTime, throughputs, gnudatfile, asciitracefile, bhCfgFile ➔ Class members
11	<< Logging>>	▪ Arbitrary messages at a specific log level in network system ▪ Class must extend ➔ UWSN_Logging class ▪ ComponentDefine() ➔ Create a log component with a unique name ▪ DEBUG() ➔ output a message of level log DEBUG ▪ ASSERT() ➔ Enable asserts at compile time ▪ ERROR() ➔ Report a fatal error with a message and terminate ▪ ComponentEnable() ➔ Allow the logging output for that log component to be enabled ▪ ComponentEnable() ➔ Disable the logging output associated with that log component

12	<< CommandLine >>	▪ CommandLine ➔ parse command-line arguments
13	<< Run>>	▪ Section 4.2.2 Serial 1-9 ➔ all rules apply here ▪ Class must extend ➔ UWSN_ Run class ▪ SetSink ➔ State Machine ➔ set sink node method body (Section 4.2.1 Serial 6, Section 4.2.2 1-9 all rules apply) ▪ Output ➔ State Machine ➔ ascii output method body (Section 4.2.1 Serial 6, Section 4.2.2 1-9 all rules apply)
14	<< Increment>>	▪ Class must extend ➔ UWSN_ Increment class ▪ CongestionWindow, avgThroughput ➔ Class members ▪ Add() ➔ add function ▪ Clear() ➔ clear function ▪ size() ➔ size function
15	<< ReceivePacket>>	▪ Class must extend ➔ UWSN_ ReceivePacket class ▪ GetSize() ➔ coding block for returning size in bytes of the received packet ▪ Receive() ➔ read packets coding block
16	<< MainFunction>>	▪ Section 4.2.2 Serial 1-9 ➔ all rules apply here ▪ Class must extend ➔ UWSN_ MainFunction class ▪ PlotDataset ➔ State Machine ➔ Plot 2D dataset method body (Section 4.2.1 Serial 6, Section 4.2.2 1-9 all rules apply)
17	<< UpdatePositions>>	▪ Class must extend ➔ UWSN_ UpdatePositions class ▪ SetPosition() ➔ Set Position function ▪ GetPosition () ➔ Get Position function
18	<< ResetData>>	▪ Class must extend ➔ UWSN_ ResetData class ▪ Push_back() ➔ push the calculated throughput into a vector

In MainFunction and Run operations, state machine diagrams are defined for defining the actual body of set sink node, ascii output, and plot dataset methods. Transformation of state machines and states is defined in Section 4.2.1. Serial 6 and Section 4.2.2 1-9. In Node class, three types of nodes are created depending on the value of properties i.e. position, node state, depth etc.

CommunicationChannel stereotype is transformed into applied channel in underwater system. Technique property in CommunicationChannel stereotype represents the avaiable communication channels of UWSN, specifically defined for underwater network systems. Latency, frequency, range, dataRate, bandwidth is transformedlLatency, frequency, range, dataRate, bandwidth of channel. CommunicationChannel is defined as enumeration.

Update positions is mapped into an operation class in NS3. It is logically mapped in update positions class as new random locations assigned to a group of nodes within the bounding area. GetPosition function of update positions stereotype is transformed into get position method, in which they get the current position of nodes. While SetPosition function of update positions stereotype is transformed into set position method, in which they set the position of these nodes.

CommandLine interface stereotype is applied on an interface which is transformed into NS3 interface. This is an interface which acts as a parse command-line argument. With input streams (operator>>), argument variable types can be set directly; more complex argument parsing can be achieved by giving a Callback that follows transformation rules.

Chapter 5

Validation

CHAPTER 5: VALIDATION

Two thorough case studies are used to demonstrate the applicability and validity of our proposed framework in this chapter. Figure 5.1 represents the flow towards validation process. It also shows the overview of our research. Firstly, UWSN framework concepts are identified and utilized to model PUWSN profile. This profile modeling is followed by the transformation of domain model (Class Diagram) and behavioral model (State Machine Diagram) to the deployable c++ code using our proposed transformation engine UWSNTE. Finally, the proposed framework MFUWSN is validated and verified with two case studies i.e., Monitoring Offshore Oil & Gas Reservoirs and Smart Cites Underwater case studies. These case studies are discussed and documented in descriptive form. Monitoring Offshore Oil & Gas Reservoirs case study is discussed and validated in **Section 5.1** and Smart Cites Underwater case study is presented in **Section 5.2** correspondingly. **Section 5.3** gives details of Quality evaluation of MFUWSN, and **Section 5.4** discusses transformation loses.

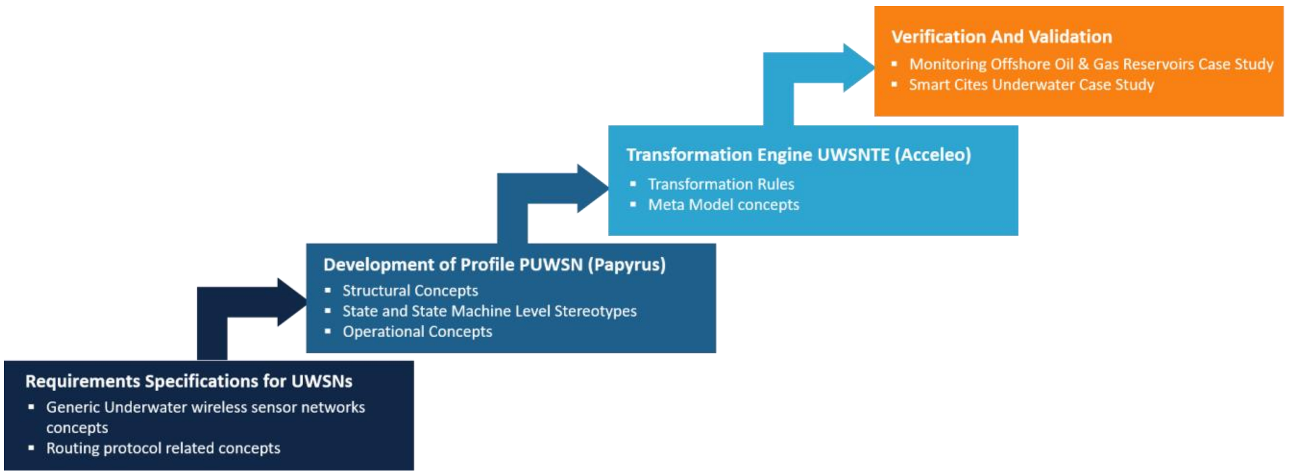

Verification And Validation
Monitoring Offshore Oil & Gas Reservoirs Case Study
Smart Cites Underwater Case Study
Transformation Engine UWSNTE (Acceleo)
Transformation Rules
Meta Model concepts
Development of Profile PUWSN (Papyrus)
Structural Concepts
State and State Machine Level Stereotypes
Operational Concepts
Requirements Specifications for UWSNs
Generic Underwater wireless sensor networks concepts
Routing protocol related concepts

Figure 5.1: Flow towards Validation

5.1. Monitoring Offshore Oil & Gas Reservoirs

This case study is explained and validated by dividing it in four sections. **Section 5.1.1** describes the Requirement Specification for the Monitoring Offshore Oil & Gas Reservoirs system. **Section 5.1.2** contains the UML modeling of domain concepts with applied profile to present the system architecture of the required system in Eclipse plugin Papyrus. **Section 5.1.3** shows the transformation results in the form of generated code. And then verification of the case study in **Section 5.1.4**.

5.1.1. Requirements Specifications

Underwater resources are currently a major focus of new offshore exploration activities, as well as a major growth area for offshore production in the coming years. The benign and shallow water resources still dominate offshore oil production. As the industry matures and technology improves, there is a greater demand for automated data collection, recording, and transmission prior to deep water deployment. Underwater wireless sensor networks may be the greatest option for quick environmental monitoring and boosting the spatial and temporal resolution of oceanographic observations. The use of UWSN necessitates the use of good localization algorithms in order to obtain the appropriate geographical metadata for oceanographic measurements. The most significant feature of UWSN may be its capacity to monitor in near-real time via acoustic communication lines, making it significantly more trustworthy than anchoring.

With our proposed method user can achieve a model that can generate network infrastructure, protocols and their communication capabilities in offshore oil & gas monitoring activities. Acoustic conditions and node locations have a significant impact on the maximum attainable data rate, according to capacity computations and link performance. It allows users to model and derive a system design from a global requirement specification, and then produce c++ code automatically. User can deploy code on his system and generate offshore oil & gas monitoring application to monitor temperature, gas fraction, sand rate, flow rate, pressure, fluid fraction and chemical properties before real deployment of such network. This section contains the details of network specifications for underwater systems deployed in offshore oil & gas reservoirs.

Deployment Settings

The primary requirement of every Underwater Wireless Sensor Networks is its deployment setup that allows the researcher/ practitioner to setup the network according to their specific requirements. These deployment necessities should be configured for boundary area size with total number of transmitting and sink nodes with their data rate, depth, generated packet sizes, bytes received and number of topologies. It also includes time per trial, slot time duration with contention window size.

UWSN Configuration Helper

UWSN monitoring system requires configuration of underwater *channel*, *PHY* and *MAC* models. Three propagation models are needed to model the underwater acoustic *channel*: the ideal channel model, the Thorp propagation model, and the Bellhop propagation model. While the *PHY* model's main role is to handle packet capture, error detection, and successful packet forwarding to the MAC layer. The signal to noise ratio (SINR) and packet error rate (PER) are calculated using two models in the PHY. The PER and SINR models work together to determine packet reception success. A Transducer function connects the PHY model to the channel. The Transducer is in capable of ensuring track of all arriving and departing packets throughout the event. The "Mode" of the transmission determines how the PHY, PER, and SINR models respond to packets (UanTxMode). Finally, a MAC protocol that uses a slotted contention window (CW) comparable to the IEEE 802.11 DCF is modeled. The contention window for nodes is always measured in slot times (configured via attribute). If the channel is felt to be busy, nodes backoff and choose a slot to transmit in at random (uniform distribution). The durations of the slot times can also be set using an attribute.

Receiving Data/Packets

The routing algorithm for Offshore Oil & Gas Reservoirs should keep track of all available and received packets from a socket. This socket is an abstraction that enables monitoring applications to connect to other Internet hosts and, among other things, exchange reliable byte streams and erroneous datagrams. The TypeId is supplied by the interface and assists in the establishment of sockets on a certain node. These sockets perform different operations like notify Oil & Gas When new data/packets are available to be read, reservoirs monitor the application, build Callbacks when packets are received, and allocate a local endpoint for the socket.

Processing Interface

The UWSNs requires an interface that will define, parse, and execute network requirements input then prints the values of each variable. The Attribute and GlobalValue inputs are processed by this interface. A shorthand parameter name can be used to set default values for certain properties. It requires an attribute named TypeID which records a lot of meta-information about each instance.

5.1.2. Modeling

Figure 5.2 depicts the complete domain model of the UWSN application for offshore oil and gas reservoirs. It represents all structural elements of UWSN. There are two main concepts in this model i.e., Deployment settings for underwater network and channel, PHY and MAC configurations to develop monitoring system for oil & gas. The developed model consists of several classes, as well as the stereotypes, attributes, and methods that are required for the system's proper execution.

In this model, <<Setup>> stereotype is applied to *VerificationSetup* which define network layout and properties of nodes. The concepts of configurations are based on twelve classes; *UanHelper* class set the physical layer, MAC layer, Transducer and protocol for the required system network. <<CommunocationChannel >> stereotype is mapped to *Channel* which provide the channel information with three propagation models to handle complicated ocean acoustic models. <<Protocol >> stereotype is mapped to *PacketSocketAddress* which set the protocol then provide an address for a packet socket, set destination address and get the device this address is bound to. <<Node >> stereotype is mapped to *NodeContainer* which hold the multiple node pointers to keep track of nodes. <<ApplicationModule>> stereotype is mapped to *ApplicationContainer* which provides configuration of parameters and tracing. This ApplicationConatiner will create an application, install it in a node, then add that application to a Container for use by the caller for each of the Nodes in the NodeContainer. *NetDeviceContainer* define the installation of a NetDevices on nodes. The NetDeviceContainer will create a net device for each of the Nodes in the NodeContainer, assign it a MAC address and a queue, and install it on the node. The helper additionally puts each of the devices to a Container so that the caller can use them later. <<InterfaceCommandLine >> stereotype is mapped to *CommandLine* which is used to parse command-line arguments. Argument variable types with input streamers (operator>>) can be set directly; more complex argument parsing can be accomplished by providing a Callback. *Packet*, *Socket* and *PacketSocketHelper* classes provide details of packets and keep track of all available packets from a socket. Classes *Gnuplot* and *Plot2DDataset* allows to create gnuplot(2D points plot.)-ready plotting instructions from a collection of datasets. Packet modulation information is provided through the UanTxMode, Modeslist, and UanTxModeFactory classes. In the Global database of UanTxMode objects, this includes a lightweight globally unique id for the mode. Nodes' current position and velocity are tracked by the *MobilityHelper* and *MobilityModel* classes.

It also supports in the assignment of node placements and mobility models. *MainModule* class is responsible for controlling program execution of design via its interface. Main behavior of this class, for defining the actual logic of running the verification on CommandLine interface. The *OnOffHelper* class assists in the generation of traffic to a single destination on a set of nodes using an OnOff pattern. The "On" and "Off" states alternate after Application::StartApplication is called. The onTime and offTime random variables are used to determine the duration of each of these stages. There is no traffic generated in the "Off" state. cbr traffic is generated when the switch is turned on. The initial packet transfer occurs after a delay equal to (packet size/bit rate) when an application is started. *OnOffHelper* class also used to set the underlying application attributes. Each attribute value has a specific datatype.

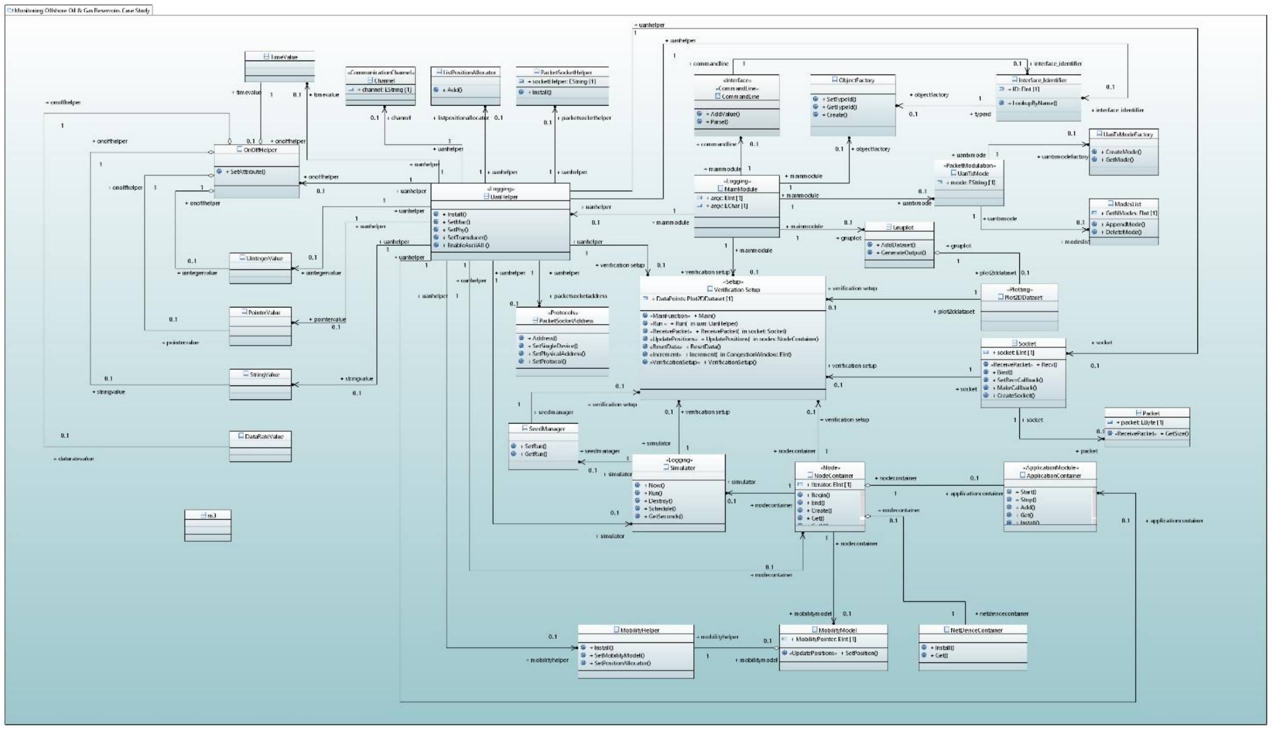


Figure 5.2: Domain Model of UWSN for Offshore Oil & Gas Reservoirs (Structural Elements)

To this point we have modeled structural elements of UWSN for Offshore Oil & Gas Reservoirs. UML state machine diagram is used for representing behavioral elements of UWSNs. Figure 5.3 provides the model i.e. state machine diagram, which is developed for representing actual behavior of set sink node method. Similarly, Figure 5.4 and Figure 5.5 provide behavioral representation of

Asccii output function and behavioral representation of plotting 2D data points function respectively.

As shown in Figure 5.3, behavior of set sink node method is presented. Initially, idle state represents initialization of event. From idle state, after transition either create object or socket is driven on design under test. The CheckID pseudo state of the UML state machine diagram is used to depict this decision. The value of "ID" is used to determine the next state selection from the idle state. The number "1" indicates that the operation is to create a socket. On the other hand, if ID has value “0” it implies creating object operation. This condition is modeled as guards on the transition. After, driving creating operation, system will go back to idle state. Similarly, behaviors of Ascii output and Plotting methods are modeled as shown in Figure 5.4 and 5.5 respectively. In models PUWSN profile is applied. Therefore, user need to enter the values of properties defined in stereotypes as shown in Figure 5.6.

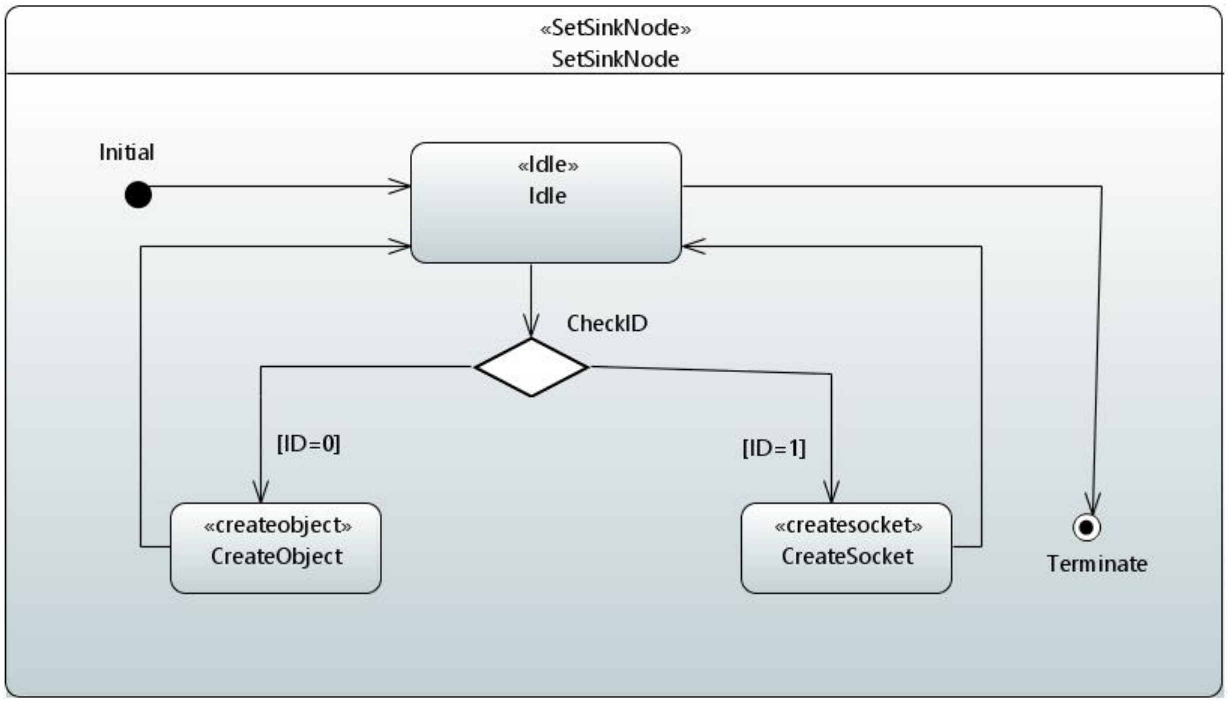


Figure 5.3: Behavioral Model of Set Sink Node Function for Offshore Oil & Gas Reservoirs

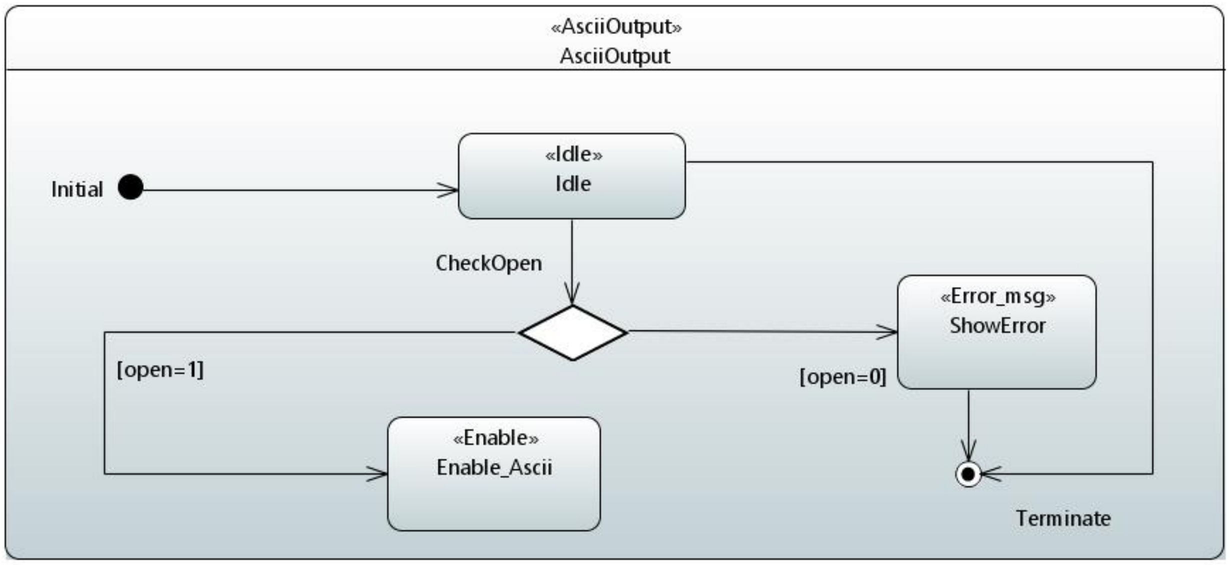


Figure 5.4: Behavioral Model of Ascii Output Function for Offshore Oil & Gas Reservoirs

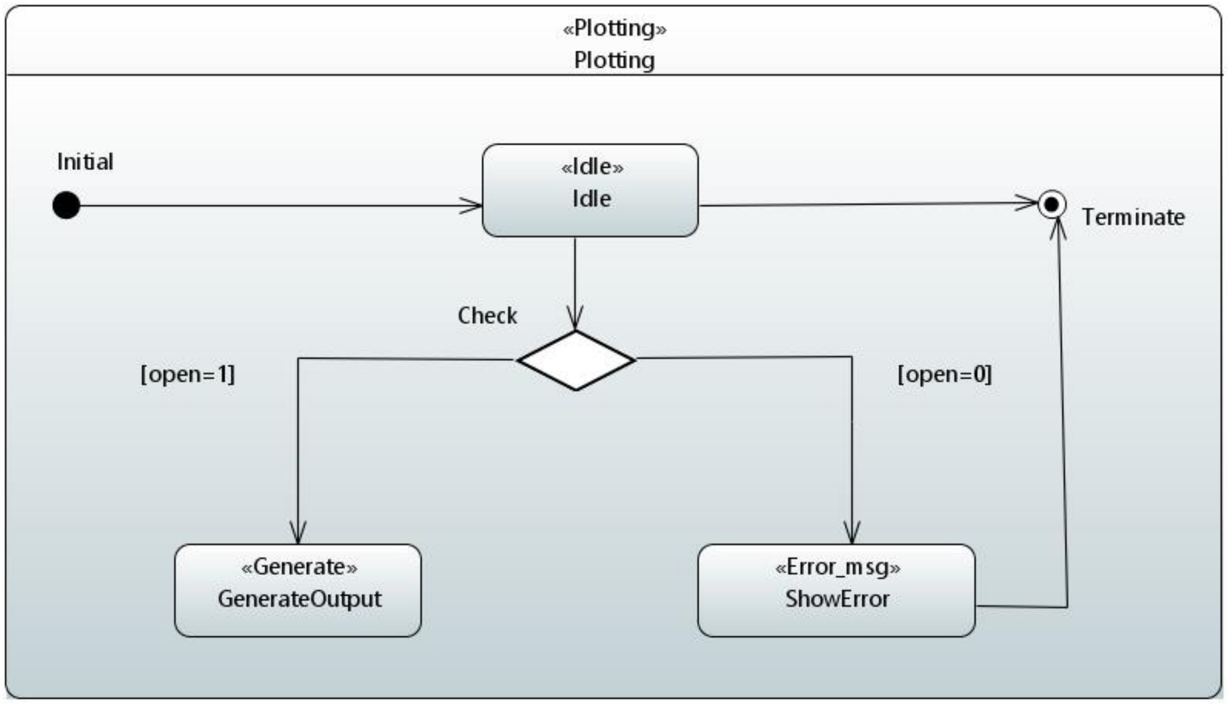
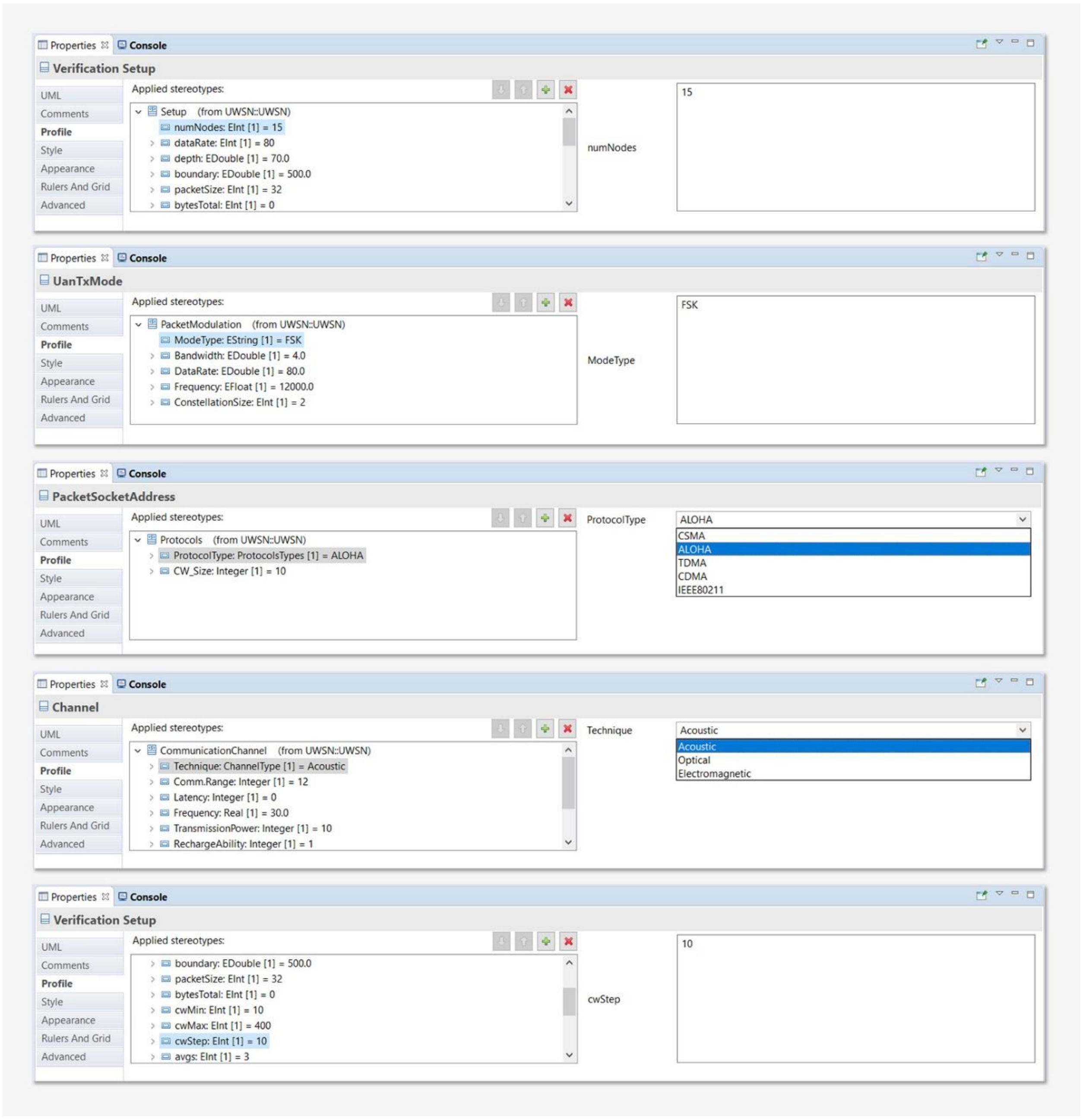


Figure 5.5: Behavioral Model of 2D Plotting Function for Offshore Oil & Gas Reservoirs

It is important to note that UML profiles require assignment of tagged values as defined in PUWSN. In Verification setup it is important to define the verification settings of network. Verification settings is represented by 16 tags i.e., numNodes, boundary, depth, dataRate, packetsize, bytesTotal etc. User defines the values of these tags as shown in Figure 5.6. Moreover, user need to define the type of modulation, ModeType, bandwidth, DataRate, frequency and ConstellationSize for the packets based on the “Mode” of the transmission. Similarly, values are assigned to all tagged values defined in PUWSN, as demonstrated in Figure 5.6.



Verification Setup

Setup (from UWSN::UWSN)
numNodes: EInt [1] = 15
dataRate: EInt [1] = 80
depth: EDouble [1] = 70.0
boundary: EDouble [1] = 500.0
packetSize: EInt [1] = 32
bytesTotal: EInt [1] = 0

numNodes: 15

UanTxMode

PacketModulation (from UWSN::UWSN)
ModeType: EString [1] = FSK
Bandwidth: EDouble [1] = 4.0
DataRate: EDouble [1] = 80.0
Frequency: EFloat [1] = 12000.0
ConstellationSize: EInt [1] = 2

ModeType: FSK

PacketSocketAddress

Protocols (from UWSN::UWSN)
ProtocolType: ProtocolsTypes [1] = ALOHA
CW_Size: Integer [1] = 10

ProtocolType: ALOHA
CSMA
ALOHA
TDMA
CDMA
IEEE80211

Channel

CommunicationChannel (from UWSN::UWSN)
Technique: ChannelType [1] = Acoustic
Comm.Range: Integer [1] = 12
Latency: Integer [1] = 0
Frequency: Real [1] = 30.0
TransmissionPower: Integer [1] = 10
RechargeAbility: Integer [1] = 1

Technique: Acoustic
Acoustic
Optical
Electromagnetic

Verification Setup

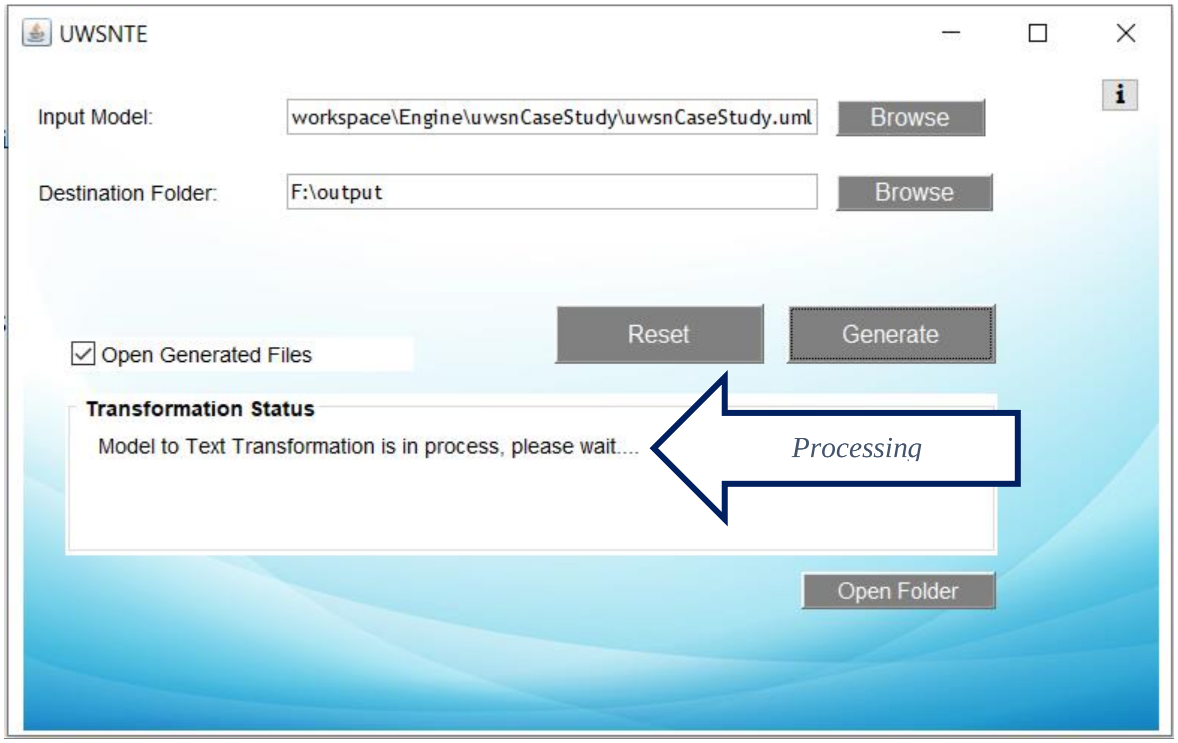
boundary: EDouble [1] = 500.0
packetSize: EInt [1] = 32
bytesTotal: EInt [1] = 0
cwMin: EInt [1] = 10
cwMax: EInt [1] = 400
cwStep: EInt [1] = 10
avgs: EInt [1] = 3

cwStep: 10

Figure 5.6: Assigning Tagged Values in Offshore Oil & Gas Reservoirs Design Model

5.1.3. Code Generation

This section highlights the code generation process from our proposed transformation engine "UWSNTE". It takes the domain model (.uml file) of Offshore Oil & Gas Reservoirs as an input and converts it to .cc file i.e., *C++*, shown in Figure 5.7, by applying transformation rules. First of all, Oil & Gas Reservoirs model and destination folder are provided in interface. Transformation status is visible on the screen as shown in Figure 5.7 and 5.8. Once transformation process is completed, generated files can be seen in destination folder as shown in Figure 5.9.



Input Model: workspace\Engine\uwsnCaseStudy\uwsnCaseStudy.uml

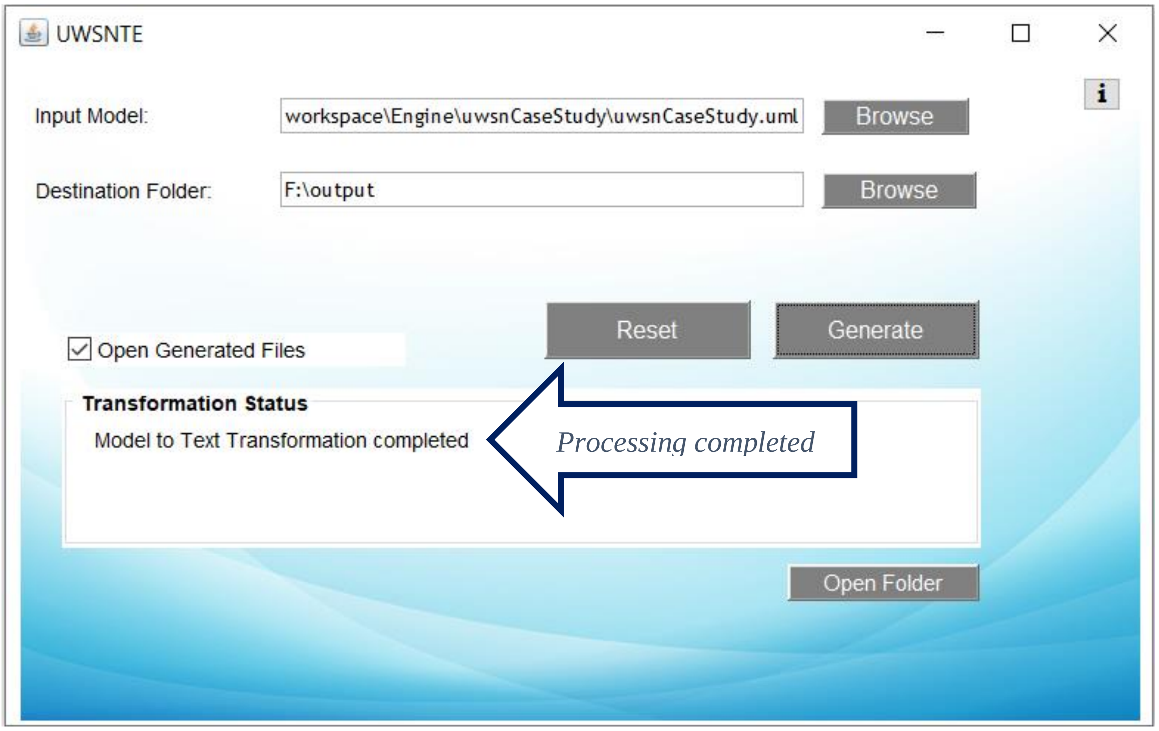
Destination Folder: F:\output

☑ Open Generated Files

Transformation Status

Model to Text Transformation is in process, please wait....

Figure 5.7: Model Transformation of Offshore Oil & Gas Reservoirs



Input Model: workspace\Engine\uwsnCaseStudy\uwsnCaseStudy.uml

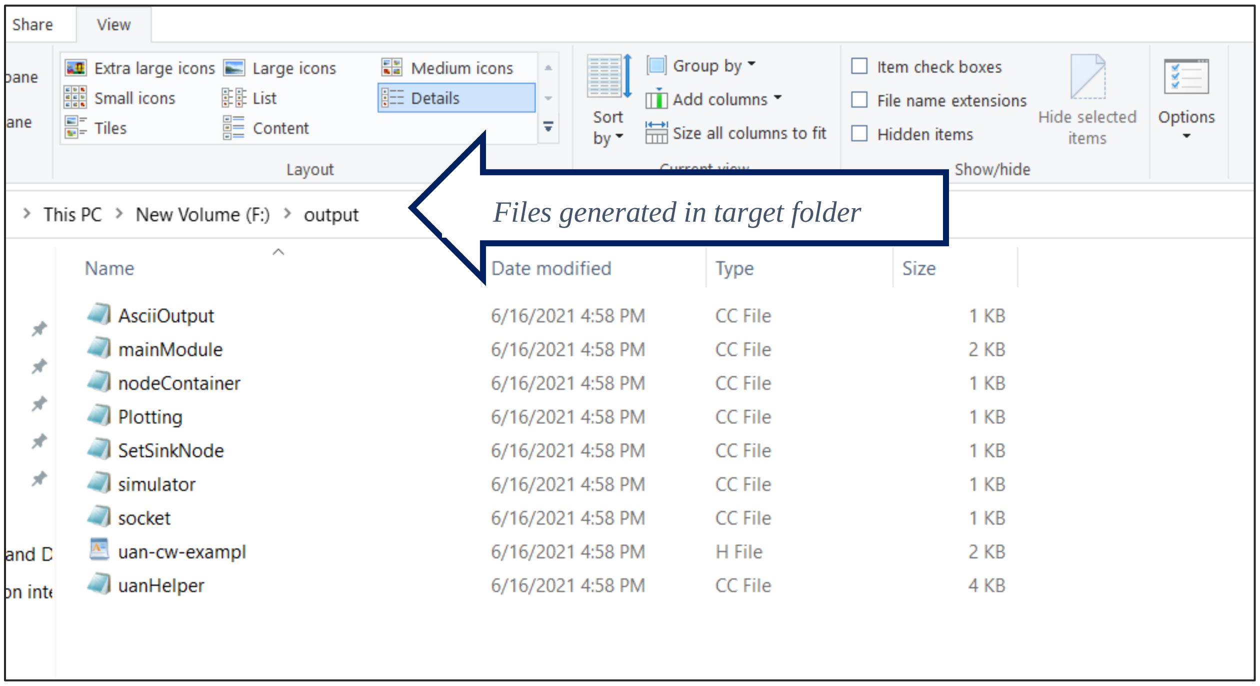
Destination Folder: F:\output

☑ Open Generated Files

Transformation Status

Model to Text Transformation completed

Figure 5.8: Input Screen with Transformation Status



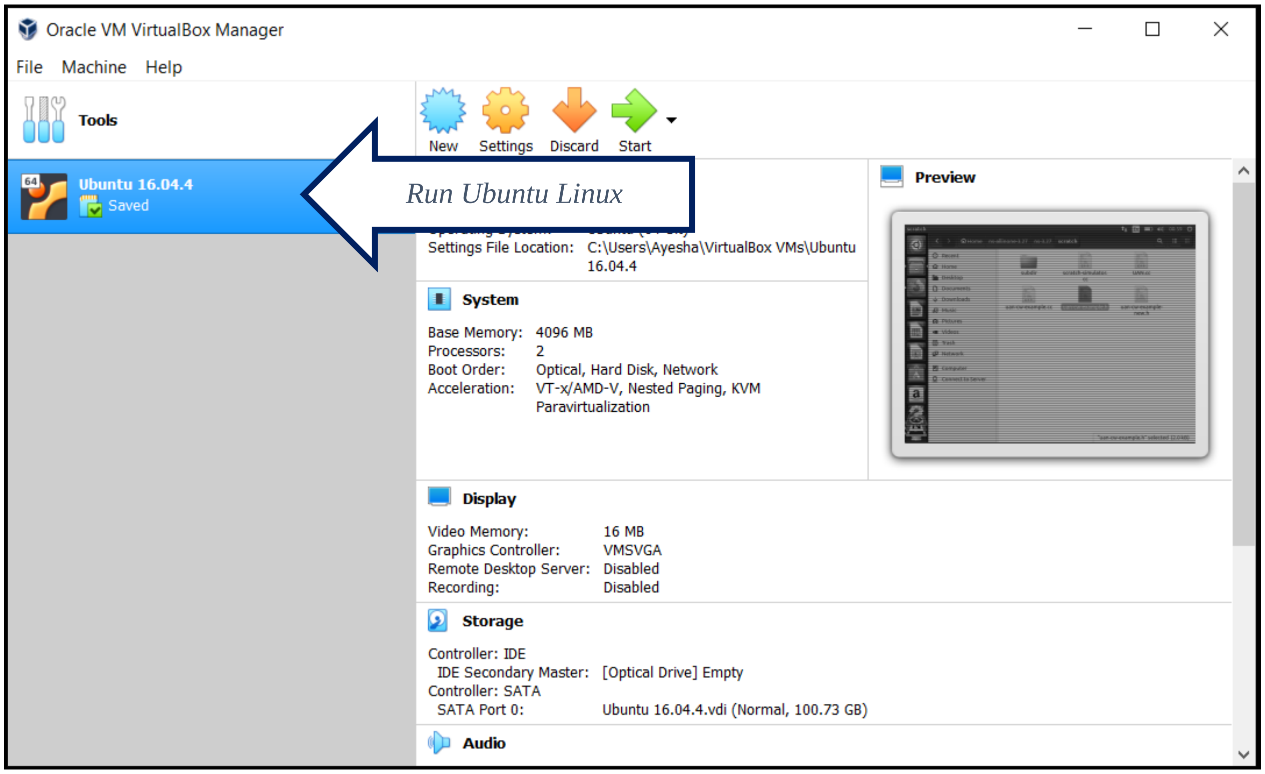
This PC › New Volume (F:) › output

Name	Date modified	Type	Size
AsciiOutput	6/16/2021 4:58 PM	CC File	1 KB
mainModule	6/16/2021 4:58 PM	CC File	2 KB
nodeContainer	6/16/2021 4:58 PM	CC File	1 KB
Plotting	6/16/2021 4:58 PM	CC File	1 KB
SetSinkNode	6/16/2021 4:58 PM	CC File	1 KB
simulator	6/16/2021 4:58 PM	CC File	1 KB
socket	6/16/2021 4:58 PM	CC File	1 KB
uan-cw-exampl	6/16/2021 4:58 PM	H File	2 KB
uanHelper	6/16/2021 4:58 PM	CC File	4 KB

Figure 5.9: Generated Files in Target Folder

5.1.4. Verification

The generated code from UWSNTE needs to be verified. Therefore, we have used AquaSim NS3 tool to perform simulation. Once transformation process is completed, UWSN files are run on Terminal in Ubuntu Linux by using the --run option in Waf as shown in Figure 5.10 and 5.11. Waf initially verifies that the code has been built correctly and, if necessary, does a build. Waf then runs the code, resulting in the compilation success illustrated in Figure 5.12. Figures 5.13 and 5.14 exhibit simulation results in which the CW (contention window size) parameter is adjusted during the simulation to highlight the variance in throughput as a result of CW. These findings show that there is a lower optimal throughput and a higher optimal CW value. This clearly demonstrates that MAC optimization in an underwater network of Offshore Oil & Gas Reservoirs is strongly dependent on the channel conditions, which are caused by the channel's environmental factors.



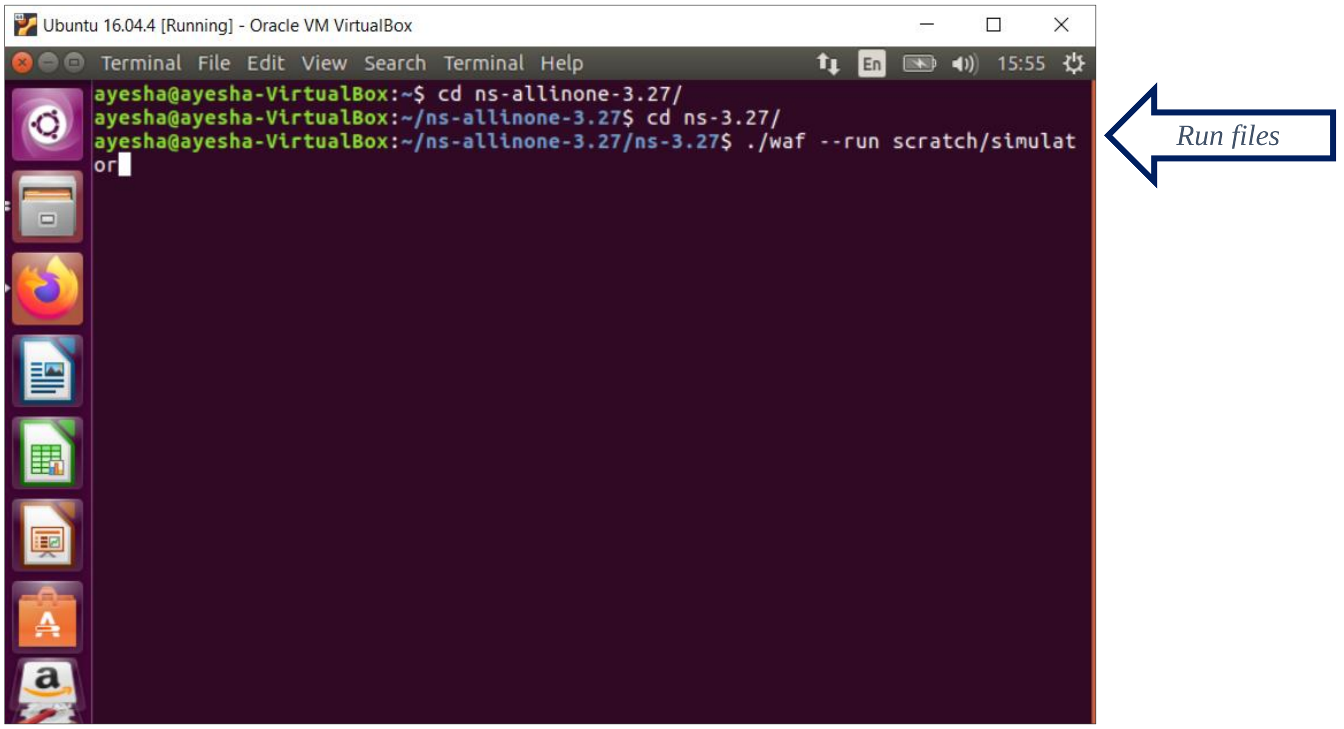
Settings File Location: C:\Users\Ayesha\VirtualBox VMs\Ubuntu 16.04.4

Base Memory: 4096 MB
Processors: 2
Boot Order: Optical, Hard Disk, Network
Acceleration: VT-x/AMD-V, Nested Paging, KVM Paravirtualization

Video Memory: 16 MB
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

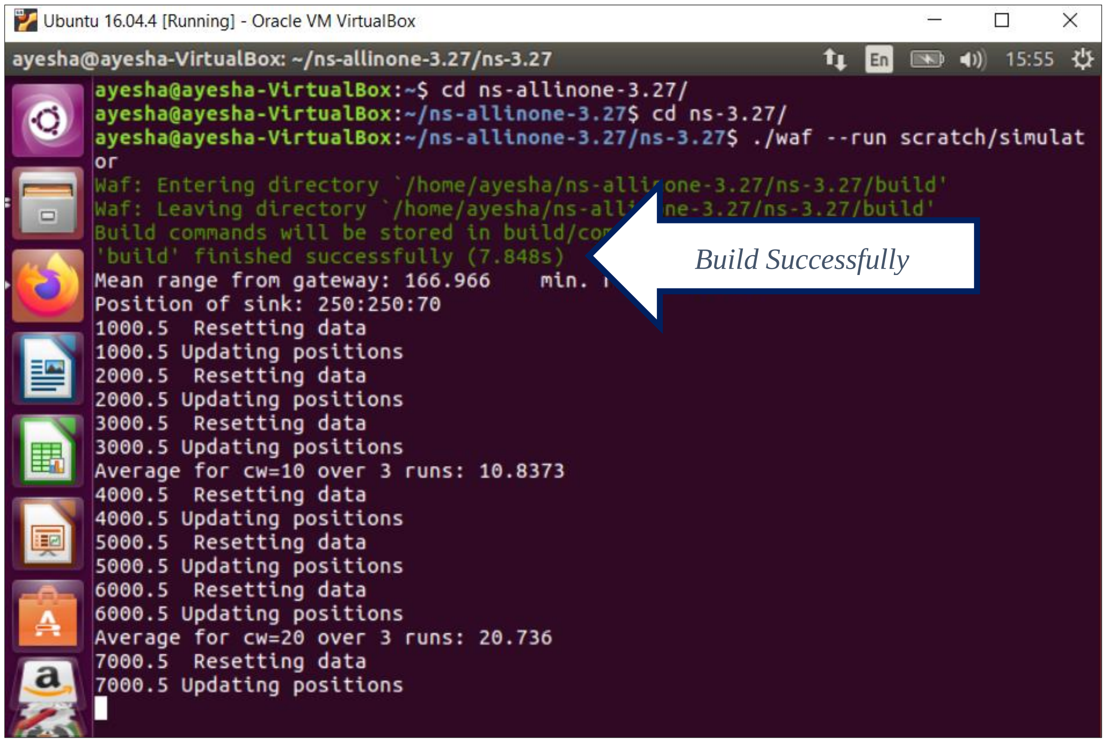
Controller: IDE
IDE Secondary Master: [Optical Drive] Empty
Controller: SATA
SATA Port 0: Ubuntu 16.04.4.vdi (Normal, 100.73 GB)

Figure 5.10: Run Ubuntu Linux for Verification


Ubuntu 16.04.4 [Running] - Oracle VM VirtualBox
Terminal File Edit View Search Terminal Help
15:55
Run files

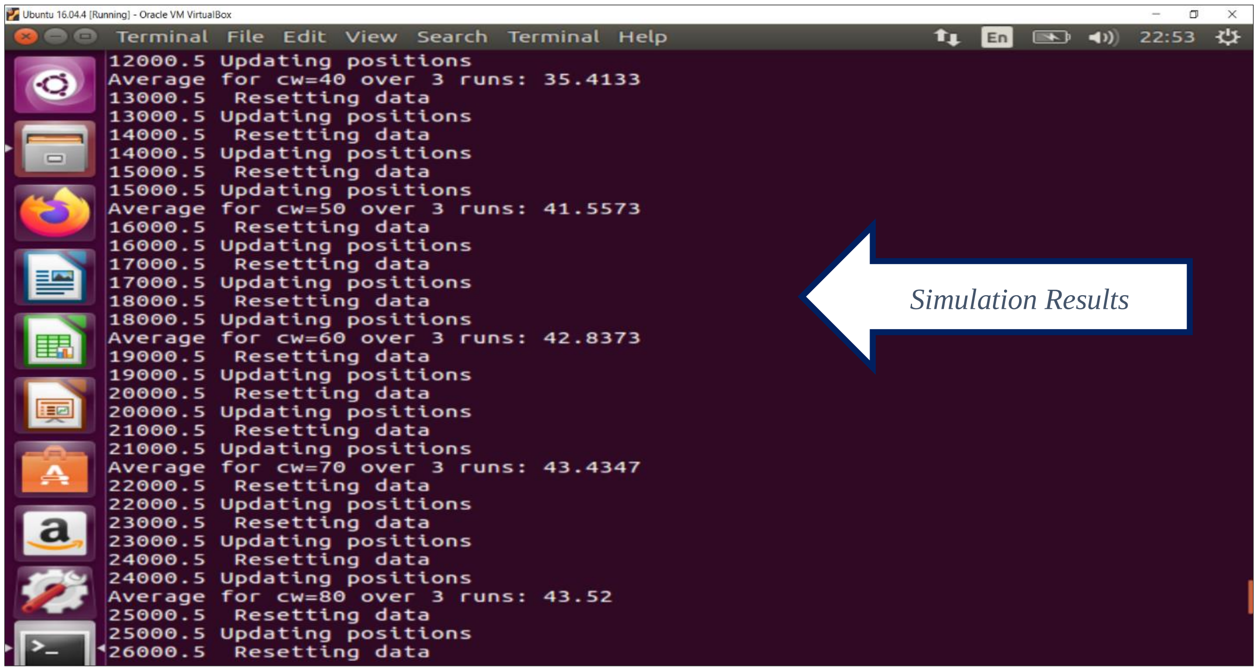
```
ayesha@ayesha-VirtualBox:~$ cd ns-allinone-3.27/
ayesha@ayesha-VirtualBox:~/ns-allinone-3.27$ cd ns-3.27/
ayesha@ayesha-VirtualBox:~/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/simulat
or
```

Figure 5.11: Running the Offshore Oil & Gas Reservoirs Files using Waf


Ubuntu 16.04.4 [Running] - Oracle VM VirtualBox
ayesha@ayesha-VirtualBox: ~/ns-allinone-3.27/ns-3.27
15:55
Build Successfully

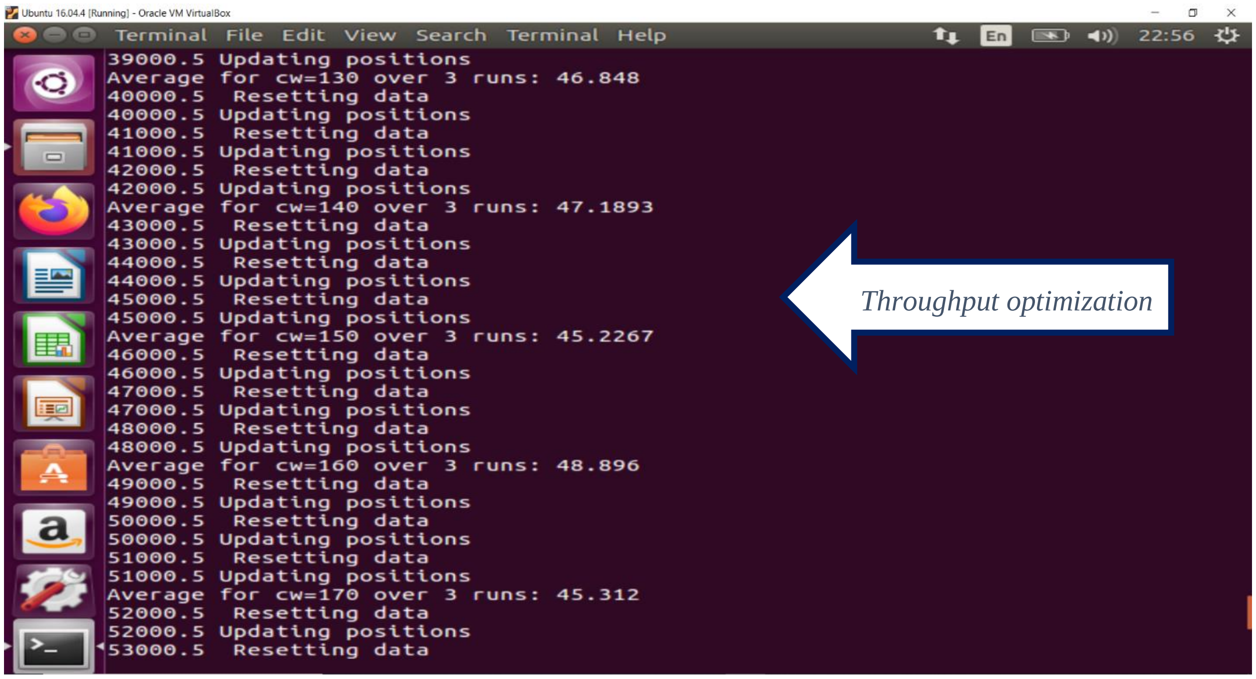
```
ayesha@ayesha-VirtualBox:~$ cd ns-allinone-3.27/
ayesha@ayesha-VirtualBox:~/ns-allinone-3.27$ cd ns-3.27/
ayesha@ayesha-VirtualBox:~/ns-allinone-3.27/ns-3.27$ ./waf --run scratch/simulat
or
Waf: Entering directory `/home/ayesha/ns-alli one-3.27/ns-3.27/build'
Waf: Leaving directory `/home/ayesha/ns-all ne-3.27/ns-3.27/build'
Build commands will be stored in build/co
'build' finished successfully (7.848s)
Mean range from gateway: 166.966    min.
Position of sink: 250:250:70
1000.5  Resetting data
1000.5 Updating positions
2000.5  Resetting data
2000.5 Updating positions
3000.5  Resetting data
3000.5 Updating positions
Average for cw=10 over 3 runs: 10.8373
4000.5  Resetting data
4000.5 Updating positions
5000.5  Resetting data
5000.5 Updating positions
6000.5  Resetting data
6000.5 Updating positions
Average for cw=20 over 3 runs: 20.736
7000.5  Resetting data
7000.5 Updating positions
```

Figure 5.12: Offshore Oil & Gas Reservoirs Files Build Status


Ubuntu 16.04.4 [Running] - Oracle VM VirtualBox
Terminal File Edit View Search Terminal Help
En
22:53
Simulation Results

```
12000.5 Updating positions
Average for cw=40 over 3 runs: 35.4133
13000.5  Resetting data
13000.5 Updating positions
14000.5  Resetting data
14000.5 Updating positions
15000.5  Resetting data
15000.5 Updating positions
Average for cw=50 over 3 runs: 41.5573
16000.5  Resetting data
16000.5 Updating positions
17000.5  Resetting data
17000.5 Updating positions
18000.5  Resetting data
18000.5 Updating positions
Average for cw=60 over 3 runs: 42.8373
19000.5  Resetting data
19000.5 Updating positions
20000.5  Resetting data
20000.5 Updating positions
21000.5  Resetting data
21000.5 Updating positions
Average for cw=70 over 3 runs: 43.4347
22000.5  Resetting data
22000.5 Updating positions
23000.5  Resetting data
23000.5 Updating positions
24000.5  Resetting data
24000.5 Updating positions
Average for cw=80 over 3 runs: 43.52
25000.5  Resetting data
25000.5 Updating positions
26000.5  Resetting data
```

Figure 5.13: Offshore Oil & Gas Reservoirs Simulation Results


Ubuntu 16.04.4 [Running] - Oracle VM VirtualBox
Terminal File Edit View Search Terminal Help
En
22:56
Throughput optimization

```
39000.5 Updating positions
Average for cw=130 over 3 runs: 46.848
40000.5  Resetting data
40000.5 Updating positions
41000.5  Resetting data
41000.5 Updating positions
42000.5  Resetting data
42000.5 Updating positions
Average for cw=140 over 3 runs: 47.1893
43000.5  Resetting data
43000.5 Updating positions
44000.5  Resetting data
44000.5 Updating positions
45000.5  Resetting data
45000.5 Updating positions
Average for cw=150 over 3 runs: 45.2267
46000.5  Resetting data
46000.5 Updating positions
47000.5  Resetting data
47000.5 Updating positions
48000.5  Resetting data
48000.5 Updating positions
Average for cw=160 over 3 runs: 48.896
49000.5  Resetting data
49000.5 Updating positions
50000.5  Resetting data
50000.5 Updating positions
51000.5  Resetting data
51000.5 Updating positions
Average for cw=170 over 3 runs: 45.312
52000.5  Resetting data
52000.5 Updating positions
53000.5  Resetting data
```

Figure 5.14: Offshore Oil & Gas Reservoirs Simulation Results

5.2. Smart Cities Beaches

This case study is discussed in four sections i.e., Requirement's specification for Smart Cities Beaches in **Section 5.2.1**, UML modeling of domain concepts with UWSNTE Profile applied in

Section 5.2.2 to present the system architecture, transformation results in the form of generated code in **Section 5.2.3**.

5.2.1. Requirements Specifications

The Internet of Things enabled Underwater and Wireless Sensor Networks (IoUT) is defined as a network of smart networked underwater items (I-UWSN s). Autonomous Underwater Vehicles (AUVs), buoys, ships, watchman nodes, and other smart objects are examples of smart objects. As a result, the I-UWSN architecture is a revolutionary type of IoT that is projected to serve a wide range of practical applications, including submarine exploration, environmental monitoring, and catastrophe mitigation. I-UWSN is regarded as one of the viable technologies for the creation of smart cities because of these uses. In I-UWSN, there are several approaches to deploy sensor nodes for the aforementioned goals. The nodes can be placed at random or in a grid and tree-like layout. To begin, each sensor node includes computing, communication, and intelligence skills to cope with a smart environment, such as in smart cities, in order to deliver reliable communication. As a result, I-UWSN is actively researching the design and implementation of routing protocols in order to support the smart city concept. Second, the routing protocol must ensure that data transmission from the source to the destination node is reliable and effective.

Our proposed method helps in the modeling of a routing protocol for dynamic topology, namely Time-Based Reliable Link (TBRL), which is intended to assist smart cities. The TBRL process is divided into three stages. It uses a topology discovery algorithm to discover the topology of each node in the network area in the first phase. The reliability of each formed link was determined in the second phase using a two-node dependable model for a smart environment. This dependability model minimizes the likelihood of horizontal and higher depth level communication between nodes while also identifying the next most dependable forwarders. All paths are assessed in the third step, and the most dependable path is chosen to transfer data packets.

Deployment Settings

The deployment settings of TBRL should be configured for boundary area size with total number of transmitting and sink nodes with their data packet size, frequency, data rate, depth, communication medium, number of topologies etc. It also includes transmission, receiving, idle and sleeping power.

Two Nodes Reliable Model (2N-RM)

The 2N-RM technique is used to construct stable networks by configuring three critical factors: distance, Expected Transmission Count (ETX), and residual energy. Each sensor node in Distance has a range limit that is set by the routing protocol. The range limit of TBRL nodes is set to 70 meters, and the Euclidean distance between the sources and the next forwarder is computed. The *Expected Transmission Count* is the likelihood of a successful packet delivery ratio and its acknowledgement via a network. The sensor node's energy state is calculated using residual energy. For sending and receiving data packets, all of the participating nodes must be alive and have enough energy. All alive nodes must be included in a reliable link from source to sink.

Receiving/forwarding Data Packets

The routing algorithm for smart cities should keep track of all packets which will be receive and forward. We calculate the optimum path based on the maximum number of nodes participating using tree attributes. The Iterative Depth First Search (IDFS) algorithm is used for this purpose. IDFS sorts the paths and recommends the best one. Data packets will be routed to the sink node after the optimal path has been chosen.

5.2.2. Modeling

This section describes the modeling of Smart Cities Beaches. The complete domain model is shown in Figure 5.15. It represents all structural elements of UWSN. Dynamic topology discovery, a two-node trustworthy model, and topology alterations of existing paths are the three fundamental concepts in this model. The developed model consists of several classes, as well as the stereotypes, attributes, and methods that are required for the system's proper execution.

In smart cities beaches model, <<Setup>> stereotype is applied to *Nodes Deployment* class which define network layout and properties of nodes. <<*Architecture*>> stereotype is mapped to Location, which assigns nodes to different depth levels based on their depth and water speed. The source node estimates the physical distance between itself and other nodes and compares their positions. <<Compute ETX >> class calculates link dependability based on the probability of successful packet delivery and acknowledgement over a link. The sensor node's energy status is calculated using *Residual Energy*. It ensures that all of the participating nodes are awake and have sufficient energy to send and receive data packets. <<Topology>> stereotype is mapped to *tree topology*, with the source node functioning as the root node and all sinks functioning as leaf nodes. Child nodes are multiple intermediary nodes that form a path from the root to the leaf nodes. <<Logging>> stereotype is mapped to *Send Message* which contains node id. Nodes receive packets and acknowledge them by sending an acknowledgment to the originating node. The receiver node id, position, and residual energy are all included in the acknowledgement packet structure. The resultant value of each link is saved in the *Links Queue*. In ascending order, all of the results are saved. *IDFS* class sorts the paths and chooses the best one. It chooses the path from source to sink node that has the most intermediate nodes. The length, height, and period of waves are used to define their properties.

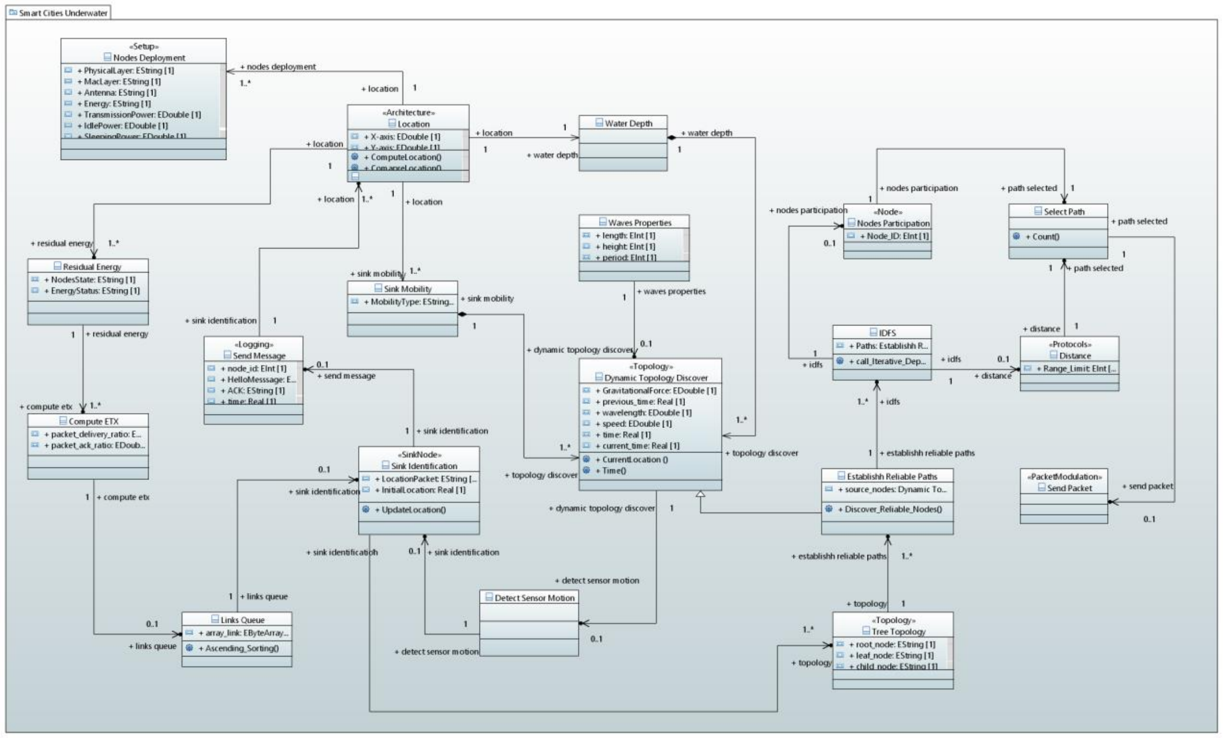


Figure 5.15: Domain Model of UWSN for Smart Cities Beaches (Structural Elements)

To this point we have modeled structural elements of UWSN for Smart Cities Beaches. UML state machine diagram is used for representing behavioral elements of UWSNs. Figure 5.16 provides the model i.e., state machine diagram, which is developed for representing actual behavior of identifying sink node method. In models PUWSN profile is applied and user can enter the values of properties defined in stereotypes as shown in Figure 5.17.

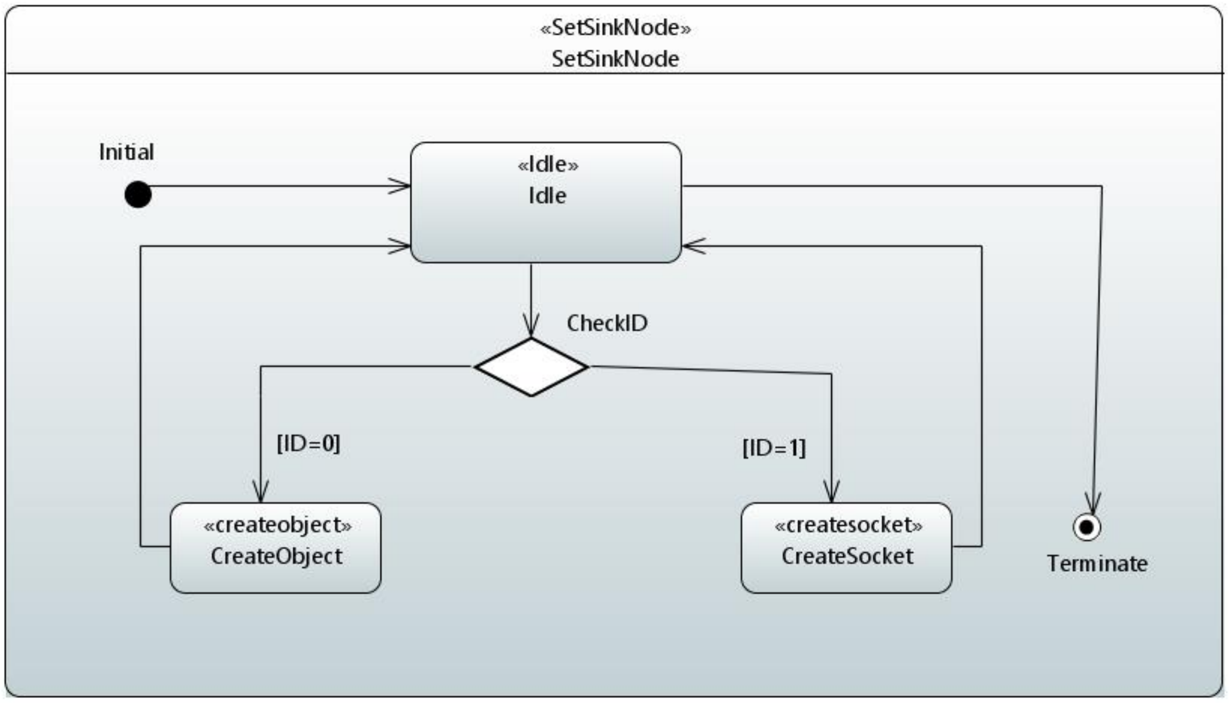
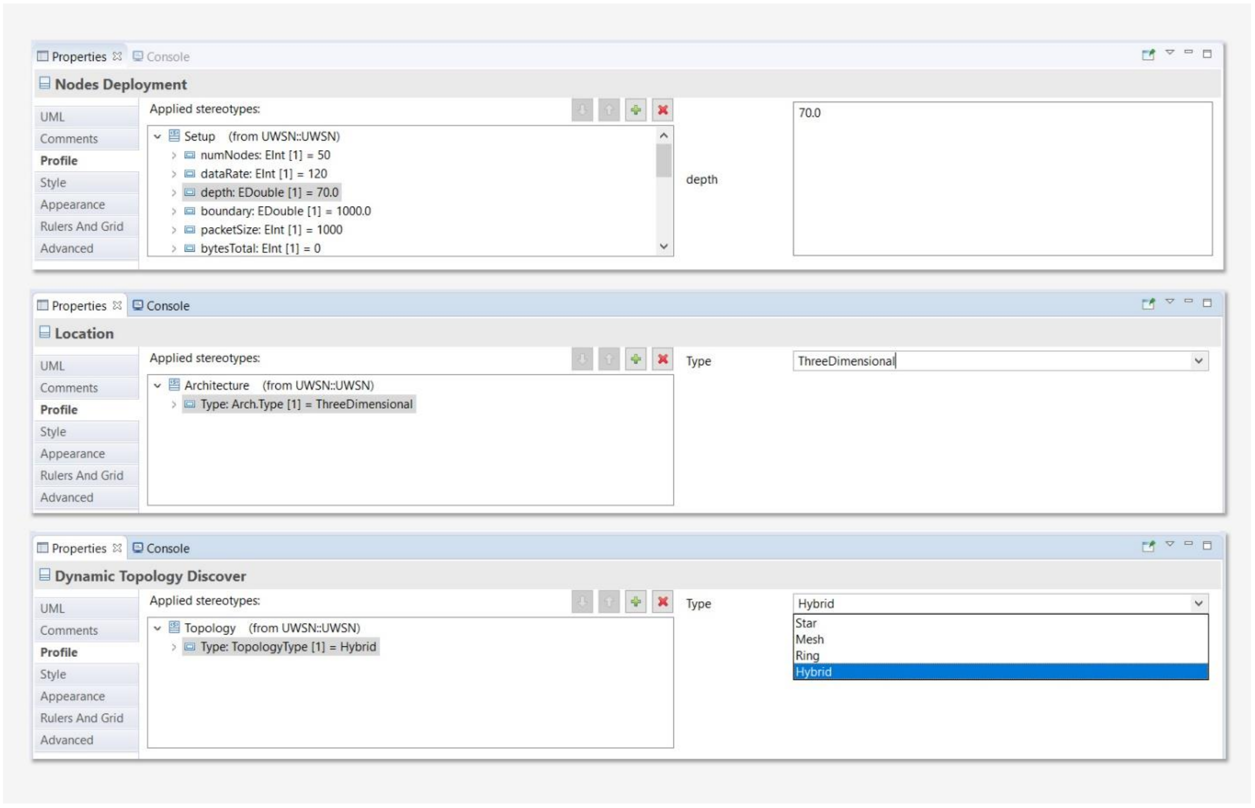


Figure 5.16: Behavioral Model of Identify Sink Node Function for Smart Cities Beaches



Nodes Deployment

Setup (from UWSN::UWSN)
- numNodes: EInt [1] = 50
- dataRate: EInt [1] = 120
- depth: EDouble [1] = 70.0
- boundary: EDouble [1] = 1000.0
- packetSize: EInt [1] = 1000
- bytesTotal: EInt [1] = 0

depth: 70.0

Location

Architecture (from UWSN::UWSN)
- Type: Arch.Type [1] = ThreeDimensional

Type: ThreeDimensional

Dynamic Topology Discover

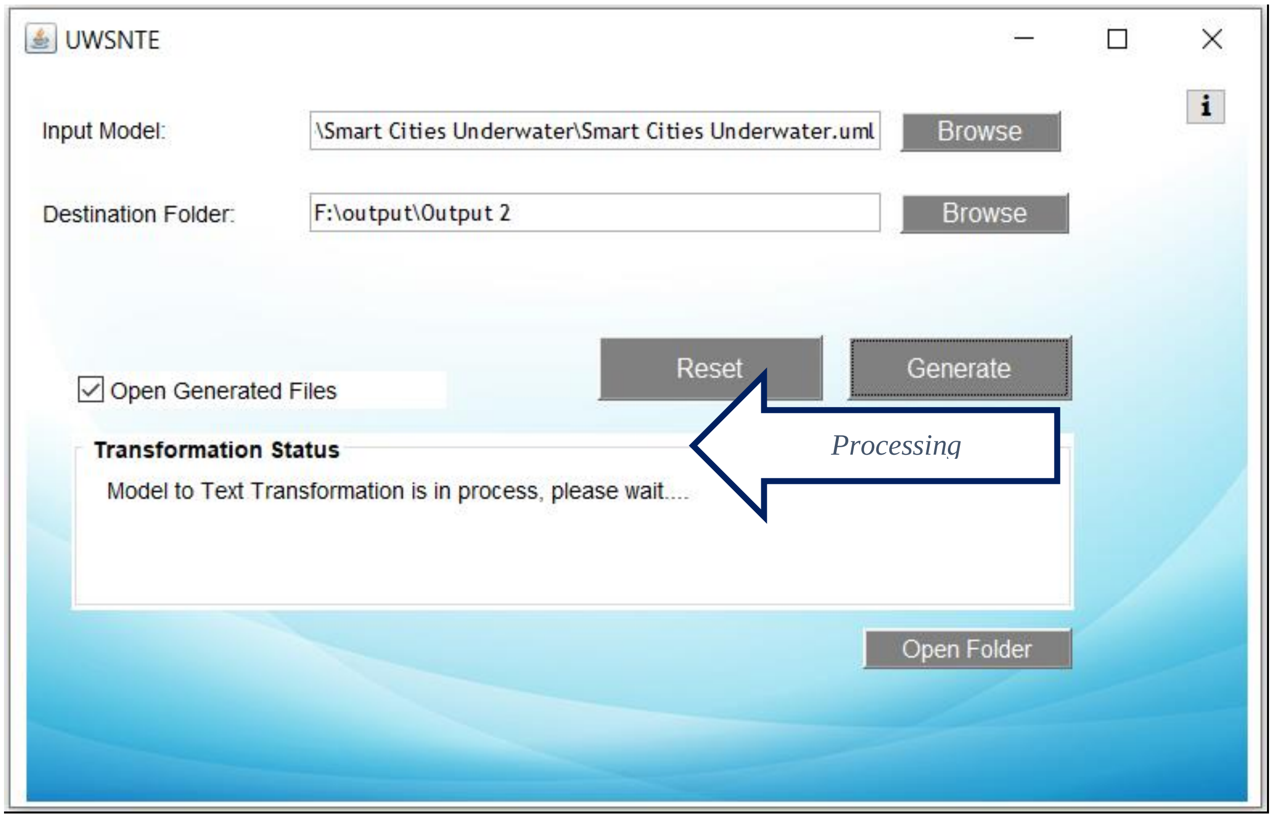
Topology (from UWSN::UWSN)
- Type: TopologyType [1] = Hybrid

Type: Hybrid
Star
Mesh
Ring
Hybrid

Figure 5.17: Assigning Tagged Values in Smart Cities Beaches Design Model

5.2.3. Transformation

This section highlights the code generation process from our proposed transformation engine "UWSNTE". It takes the Smart Cities Beaches domain model (.uml file) as an input and converts it to a.cc file (C++) using transformation rules. First of all, Smart Cities Beaches and destination folder are provided in interface. Transformation status is visible on the screen as shown in Figure 5.18 and 5.19. Once transformation process is completed, generated files can be seen in destination folder.



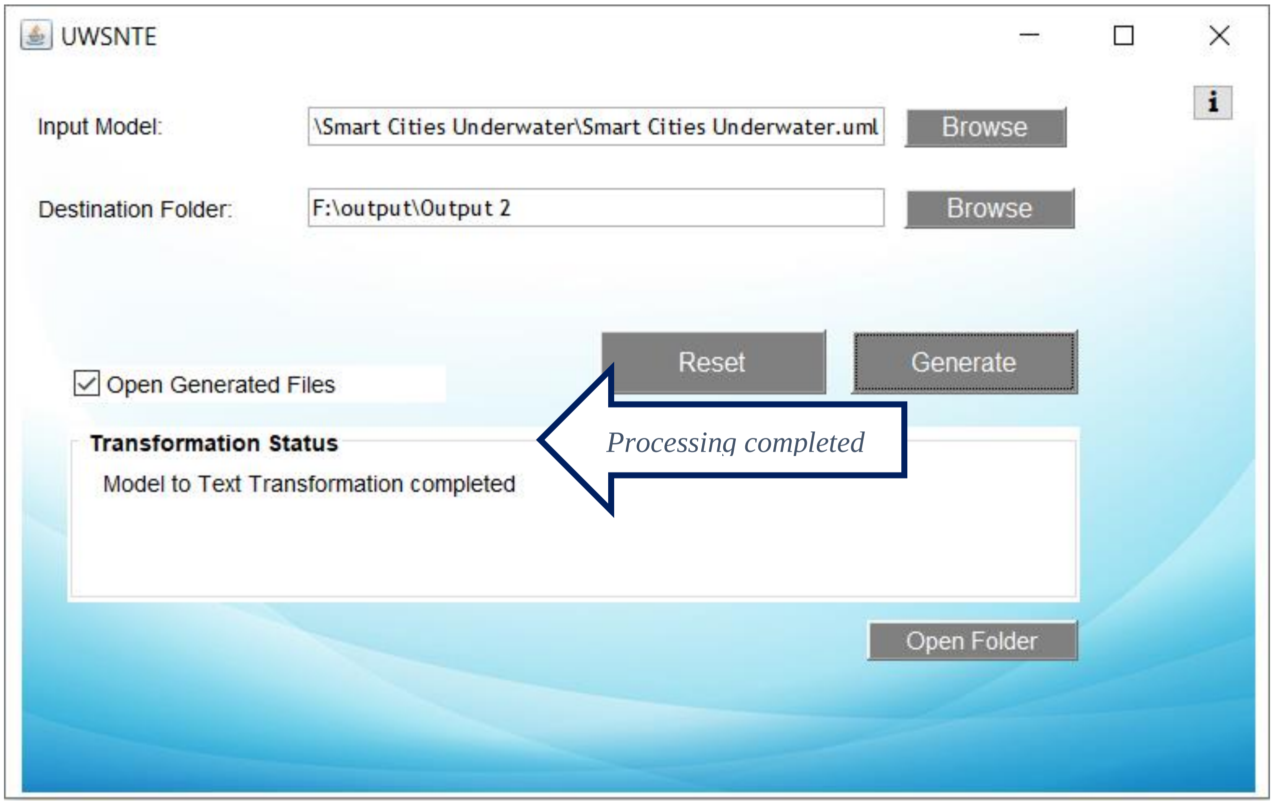
\Smart Cities Underwater\Smart Cities Underwater.uml

F:\output\Output 2

Transformation Status

Model to Text Transformation is in process, please wait....

Figure 5.18: Model Transformation of Smart Cities Beaches



Model to Text Transformation completed

Figure 5.19: Input Screen with Transformation Status

5.3. Quality of MFUWSN

The use of PUWSN to generate UWSN systems has been effectively reported. However, we must validate MFUWSN's benefits and actual use. MFUWSN offers numerous advantages in the field of underwater wireless network systems. It helps in reducing the gap between design and verification by providing higher abstraction level model for generating underwater system c++ code. After verifying MFUWSN with the help of two case studies, it is important to analyze the efficiency of this approach.

We must answer the following questions in order to validate the benefits.

1) In comparison to low-level programming, how easy is it to learn MFUWSN to develop underwater system c++ code?

2) How efficient is MFUWSN in comparison to AquaSim NS3 low-level programming? The entire time necessary to develop deployable C++ code is described as efficiency.

In order to analyze the quality of MFUWSN, we selected an industry expert i.e., Expert A who had a knowledge of underwater wireless networks and MDA. Expert A was provided the requirements of Oil & Gas Reservoirs and Smart Cites in textual form. Expert A was tasked with developing and verifying the underwater system c++ code for supplied case studies using AquaSim NS3 and keeping track of the total number of hours spent on the task as shown in Table 5.3**.** Expert A took 29 working hours to model and verify Oil & Gas Reservoirs and generate c++ code.

Table 5: Quality Evaluation of MFUT

CASE STUDY	**C++ code Manually**		**Proposed Approach**		**Transformation Errors**	**Efficiency**
	Working Hours for Design	**Working Hours for verification**	**Working Hours for Design**	**Working Hours for verification**	**Working Hours for corrections**	
Oil & Gas Reservoirs	10	19	9	11	2	(29-22)/22*100= 31.8%

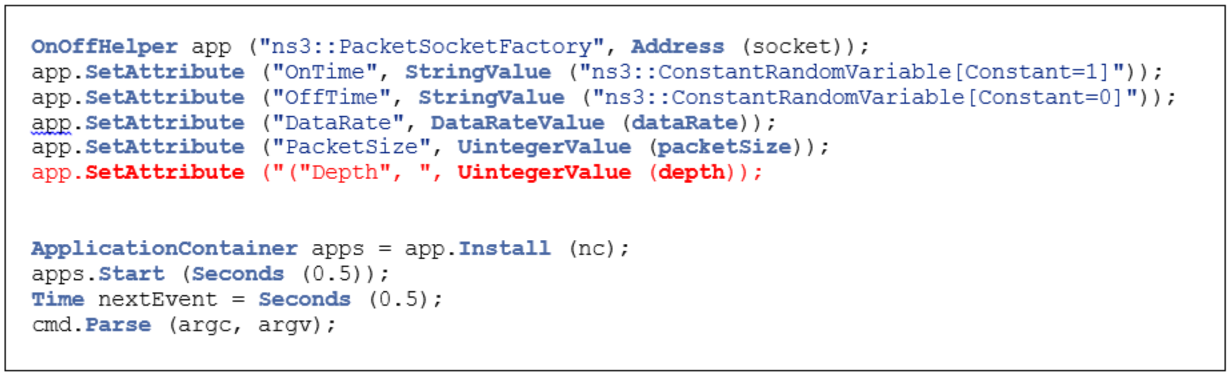
After successfully generating c++ code, Expert A manually wrote and verified c++ code for Oil & Gas Reservoirs in 29 hours respectively. Particularly, Expert A consumed 22 working hours to model and verify Oil & Gas Reservoirs underwater network system using MFUWSN and generated underwater system code with UWSNTE. It is evidently clear from Table 5.3 that MFUWSN has increased efficiency of generating underwater system code for Oil & Gas Reservoirs by 31.8% respectively. Hence this proves that MFUWSN is more efficient approach as compared to writing low level c++ code for underwater network system.

5.4. Transformation Losses

Transformation is the process of using mapping rules to convert models into desired output artifacts, such as another model or text. Because it is difficult to get 100% correct code throughout the model to text transformation process, there are usually some code losses. These losses are referred to as "transformation losses". In MFUWSN, transformation losses are low and can be quickly recovered with minimal effort. The transformation losses that occurred in our model to text transformation method will be discussed in this section.

5.4.1. Over Specifications

The code generated as a result of the transformation may have over specifications. Over specification is described as an extra line of code that has no bearing on the system's real operation. Over requirements identified in code created by UWSNTE are represented in Figure 5.20.



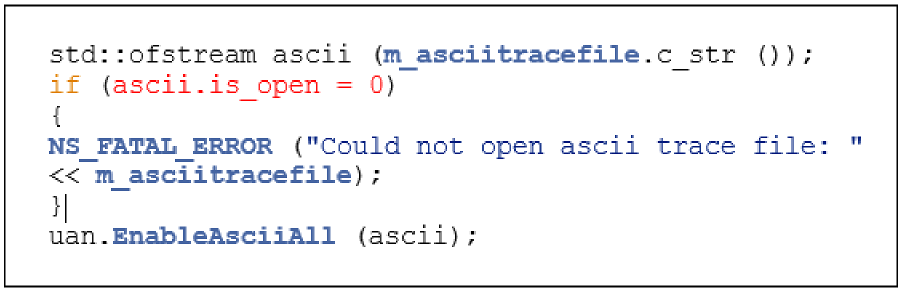
```
OnOffHelper app ("ns3::PacketSocketFactory", Address (socket));
app.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1]"));
app.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0]"));
app.SetAttribute ("DataRate", DataRateValue (dataRate));
app.SetAttribute ("PacketSize", UintegerValue (packetSize));
app.SetAttribute ("("Depth", ", UintegerValue (depth));


ApplicationContainer apps = app.Install (nc);
apps.Start (Seconds (0.5));
Time nextEvent = Seconds (0.5);
cmd.Parse (argc, argv);
```

Figure 5.20: Transformation Losses- Over Specifications

5.4.2. Syntax Errors

Syntax errors are defined as errors in a programming language's syntax. Following transformation, a few syntax problems were discovered in the generated code, as illustrated in Figure 5.21.

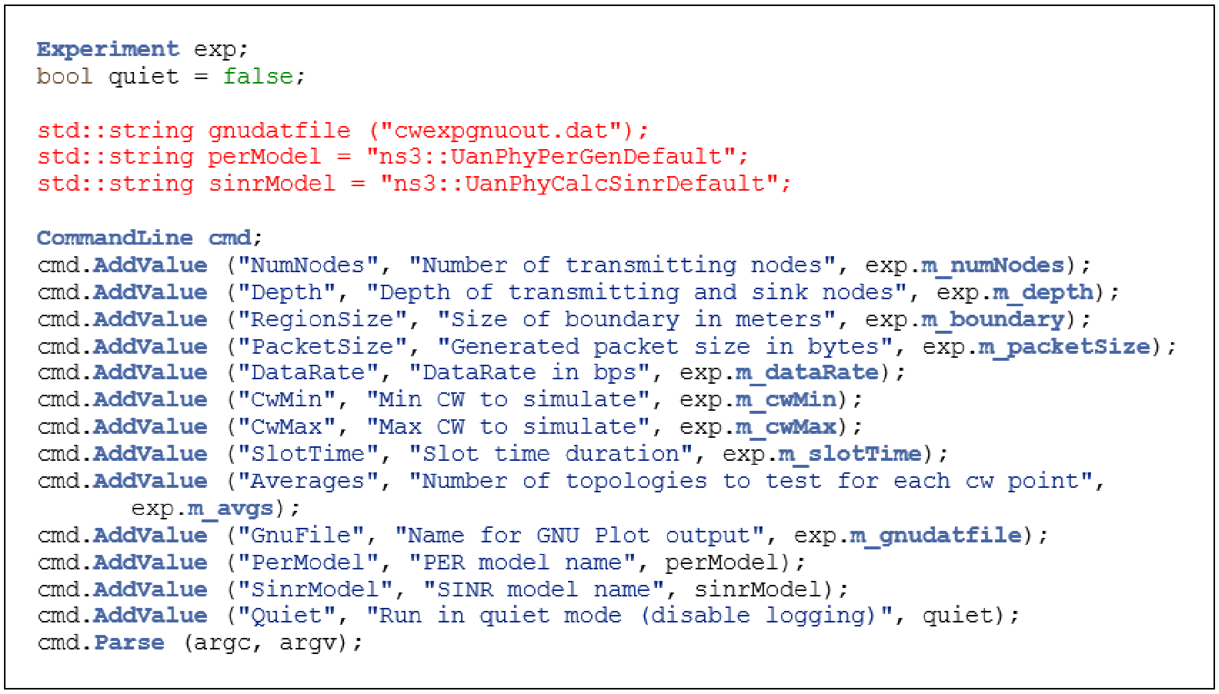


```
std::ofstream ascii (m_asciitracefile.c_str ());
if (ascii.is_open = 0)
{
NS_FATAL_ERROR ("Could not open ascii trace file: "
<< m_asciitracefile);
}
uan.EnableAsciiAll (ascii);
```

Figure 5.21: Transformation Loses- Syntax Errors

5.4.3. Actual Losses

Actual losses are those errors in generated code that result in a deviation from the system's intended functionality. The highlighted text in Figure 5.22 represents actual loss in our scenario. These lines of code are necessary for main function to work properly, but they were not created by UWSNTE.



```
Experiment exp;
bool quiet = false;

std::string gnudatfile ("cwexpgnuout.dat");
std::string perModel = "ns3::UanPhyPerGenDefault";
std::string sinrModel = "ns3::UanPhyCalcSinrDefault";

CommandLine cmd;
cmd.AddValue ("NumNodes", "Number of transmitting nodes", exp.m_numNodes);
cmd.AddValue ("Depth", "Depth of transmitting and sink nodes", exp.m_depth);
cmd.AddValue ("RegionSize", "Size of boundary in meters", exp.m_boundary);
cmd.AddValue ("PacketSize", "Generated packet size in bytes", exp.m_packetSize);
cmd.AddValue ("DataRate", "DataRate in bps", exp.m_dataRate);
cmd.AddValue ("CwMin", "Min CW to simulate", exp.m_cwMin);
cmd.AddValue ("CwMax", "Max CW to simulate", exp.m_cwMax);
cmd.AddValue ("SlotTime", "Slot time duration", exp.m_slotTime);
cmd.AddValue ("Averages", "Number of topologies to test for each cw point",
      exp.m_avgs);
cmd.AddValue ("GnuFile", "Name for GNU Plot output", exp.m_gnudatfile);
cmd.AddValue ("PerModel", "PER model name", perModel);
cmd.AddValue ("SinrModel", "SINR model name", sinrModel);
cmd.AddValue ("Quiet", "Run in quiet mode (disable logging)", quiet);
cmd.Parse (argc, argv);
```

Figure 5.22: Transformation Loses- Actual Loses

Chapter 6

Discussion and Limitation

CHAPTER 6: DISCUSSION AND LIMITATIONS

The **Section 6.1** presents a detailed discussion on proposed research work and limitations to the research are presented in **Section 6.2**.

6.1. Discussion

In this research, it has been analyzed that a very limited amount of work has been done on modeling of underwater wireless sensor networks using model driven approach. Only a partial meta-model has been proposed and no verification has been done. Available researches didn't work on model based design, analysis and verification of underwater networks and it's constraint in preliminary development period. And most of the work done in this area is domain specific. Our proposed work is a step towards modeling and verification with fully automated code generation of generic underwater wireless sensor networks using UML Profile.

Motivation behind this research is to provide an open source and generic underwater network modeling system that can tackle the cost and time efficient derivation of network structure and behavior from UWSN global behavior. Our proposed framework helps developers to generate a network architecture from a global definition and perform verification, which has been problematic in earlier solutions. It ensures the conformance of the design and derived behaviors to its specification before the actual deployment.

The proposed framework is based on model driven approach where UML Profile is used to introduce concepts of underwater wireless sensor network using stereotypes. User has to apply the specified stereotypes on required classes and set their values. They can instantiate as many nodes as they like hence lowering the cost of deployment. The proposed model represents the underwater networks shows higher abstraction of the system and is much smaller than expressed in code. It is less sensitive to changes as it is easier to understand the behavior of the system, manage changes, and maintaining the networks application at abstract level. A large number of functionalities i.e., routing protocols, ideal channels, throughput calculation, packet transmission errors, etc. can be added to the system in same amount of time resulting in minimizing the cost of time and a smaller number of people are required to build the network system using model driven approach. Furthermore, we intend to extend our UWSN system and add security components to deal with threats and malicious assaults that disrupt network communication and cooperation. The industrial simulation tools like TOSSIM, SeaWeb etc. provide variety of features for developing underwater wireless sensor networks. However, most of these simulators are not open source and are not freely available. These simulators causing the individuals to write low level code for design and properties at lower abstraction level. Due to large development and debugging efforts, writing low level code at a lower abstraction level takes a long time. Furthermore, it results in higher costs less user friendly and a longer development time. So, we have provided the higher level of abstraction with complete, open source, fully automated controlled, platform-independent and real time solution. It may have limited number of widgets dynamics at the moment, but we intend to add surface radio network and base station components to our proposed system.

Model Driven approach makes our proposed system less sensitive to changes in underwater wireless sensor networks verification technology due to its independent nature towards the platform. The model of our proposed work can easily be transformed in any of the language like like *oTcl, Tcl, C, C++, C#, Embedded C, Proto C* etc. depending on its transformation tool. Hence, this transformation is highly scalable and configurable, which allows to easily add new modules. It also leads to the high quality and less error-prone product.

Two case studies have been considered to validate our proposed framework. First case study, Offshore Oil & Gas Reservoirs, has detailed information about the required system which includes Mac layer, physical layer, channel models, packet modulation, 2D GNU plotting and Ascii output information. While second case study, Smart Cites Underwater, is based on smart environment for establishing reliable links between nodes and send data packets to most reliable path. The purpose of choosing case studies of different sizes is only to validate our proposed system.

6.2. Limitations

As we have taken the first step to network model generation for underwater wireless sensor networks, there are a few limitations to our work. PUWSN has a lot of potential but due to limited amount of time, we have provided a basic model of underwater networks with limited selected core elements and limited number of properties and behaviors. There are variety of other underwater wireless network concepts that can be added to the UWSN model such as security, surface radio network, base station components etc. Hence, there is need to improve applicability of UWSN model.

Chapter 7

Conclusion and Future Work

CHAPTER 7: CONCLUSION AND FUTURE WORK

The proposed framework is focused on underwater wireless sensor networks domain that allows us to design the network model in a platform independent way. It also provides a solution for automated code generation of UWSNs for early design verification. It is based on model-driven approach to provide simple, open-source, reusable, and wider applicable solution of early design verification. Our proposed UML profile, PUWSN (UML profile for Underwater Wireless Sensor Networks), is used for detailing the concepts, relationships and constraints between components of the system. The proposed model-driven approach MFUWSN (Model-Driven Framework for Underwater Wireless Sensor Networks) not only provides the early verification of system through models of system but it also provides surety of flexibility, quality of the end product and an early view of the system at initial stage by minimizing the overall cost and time of development. The proposed work also provides the Acceleo transformation engine, UWSNTE (Model-Driven UWSN Transformation Engine), for M2T transformation. It generates an executable code of the UWSNs from a model based on our defined transformation rules. The generated code artifacts have been verified and validated through two case studies i.e., Offshore Oil & Gas Reservoirs and Smart Cites Underwater. The results showed that the system worked in an expected way and was able to successfully verify the properties of underwater network system.

In future we tend to extend PUWSN by improving this approach in numerous directions. One possible option is to enrich the PUWSN by introducing security components, which will handle security threats and malicious attacks enforced by constraints of UWSNs and underwater acoustic channels. Other components i.e., surface radio network and base station can also be added to model the end user. Moreover, we planned to develop a graphical user interface that allows users to drag and drop UWSN elements, as well as provide real-time inspection of modules, variables, and event queues, as well as “step-by-step” and “run-until” execution capabilities.

REFERENCES

[1]. Muhammad Ayaz, Azween Abdullah, Ibrahima Faye, Yasir Batira, “An efficient Dynamic Addressing based routing protocol for Underwater Wireless Sensor Networks”, in journal Computer Communications (Volume 35, Issues 4), 2012.

[2]. Haitao Yu, Nianmin Yao, Jun Liu, “An adaptive routing protocol in underwater sparse acoustic sensor networks”, in journal Ad Hoc Networks (Volume 34), 2015.

[3]. Anjana P Das, Sabu M Thampi, “Fault-resilient localization for underwater sensor networks”, in journal Ad Hoc Networks (Volume 55), 2017.

[4]. Jun Liu, Zhong Zhou, Zheng Peng, Jun-Hong Cui, Michael Zuba, Lance Fiondella , “Mobi-Sync: Efficient Time Synchronization for Mobile Underwater Sensor Networks”, in journal IEEE Transactions on Parallel and Distributed Systems (Volume 24, Issues 2), 2013.

[5]. Rodolfo W.L. Coutinho, Azzedine Boukerche, Luiz F.M. Vieira, , Antonio A.F. Loureiro, “A novel void node recovery paradigm for long-term underwater sensor networks”, in journal Ad Hoc Networks (Volume 34), 2015.

[6]. Youngtae Noh, Uichin Lee, Paul Wang, Brian Sung Chul Choi, Mario Gerla, “VAPR: Void-Aware Pressure Routing for Underwater Sensor Networks”, in journal IEEE transactions on mobile computing (Volume 2, Issues 5), 2013.

[7]. Rodolfo W. L. Coutinho, Azzedine Boukerche, Luiz F. M. Vieira, Antonio A. F. Loureiro, “GEDAR: Geographic and Opportunistic Routing Protocol with Depth Adjustment for Mobile Underwater Sensor Networks”, in IEEE International Conference on Communications (ICC), 2014.

[8]. Youngtae Noh, Uichin Lee, Saewoom Lee, Paul Wang, Luiz F. M. Vieira, Jun-Hong Cui, Mario Gerla , Kiseon Kim, “HydroCast: Pressure Routing for Underwater Sensor Networks”, in journal IEEE Transactions on Vehicular Technology (Volume 65, Issues 1), 2016.

[9]. Manjula R. Bharamagoudra, SunilKumar S. Manvi, Bilal Gonen, “Event driven energy depth and channel aware routing for underwater acoustic sensor networks: Agent oriented clustering based approach”, in journal Computers & Electrical Engineering (Volume 58), 2017.

[10]. Hainan Chen, Xiaoling Wu, Guangcong Liu, Yanwen Wang, “A Novel Multi-Module Separated Linear UWSNs Sensor Node”, in IEEE Sensors Journal (Volume 16, Issues 11), 2016.

[11]. Mukhtar Ghaleb, Emad Felemban, Shamala Subramaniam, Adil A. Sheikh, Saad Bin Qaisar, "A Performance Simulation Tool for the Analysis of Data Gathering in Both Terrestrial and Underwater Sensor Networks", in journal IEEE Access (Volume 5), 2017.

[12]. Zhenghao Xi, Xiu Kan, Le Cao, Huaping Liu, Gunasekaran Manogaran, George Mastorakis, Constandinos X. Mavromoustakis, "Research on Underwater Wireless Sensor Network and MAC Protocol and Location Algorithm", in journal IEEE Access (Volume 7), 2019.

[13]. Inam Ullah, Yiming Liu, Xin Su, Pankoo Kim, "Efficient and Accurate Target Localization in Underwater Environment", in journal IEEE Access (Volume 7), 2019.

[14]. Bingbing Zhang, Yiyin Wang, Hongyi Wang, Xinping Guan, Zhaowen Zhuang, "Tracking a Duty-Cycled Autonomous Underwater Vehicle by Underwater Wireless Sensor Networks", in journal IEEE Access (Volume 5), 2017.

[15]. Inam Ullah ; Jingyi Chen ; Xin Su ; Christian Esposito ; Chang Choi, "Localization and Detection of Targets in Underwater Wireless Sensor Using Distance and Angle Based Algorithms", in journal IEEE Access (Volume 7), 2019.

[16]. Do Duy Tan, Tung Thanh Le, Dong-Seong Kim, "Distributed Cooperative Transmission for Underwater Acoustic Sensor Networks", in 2013 IEEE Wireless Communications and Networking Conference Workshops (WCNCW), 2013.

[17]. S. Mohamad Dehnavi, Moosa Ayati, Mohammad Reza Zakerzadeh, "Three Dimensional Target Tracking via Underwater Acoustic Wireless Sensor Network", in conference Artificial Intelligence and Robotics (IRANOPEN), 2017.

[18]. Jun Liu, Zhaohui Wang, Jun-Hong Cui, Shengli Zhou, Bo Yang, "A Joint Time Synchronization and Localization Design for Mobile Underwater Sensor Networks", in journal IEEE Transactions on Mobile Computing (Volume 15, Issues 3), 2016.

[19]. Priyatosh Mandal, Swades De, "New Reservation Multiaccess Protocols for Underwater Wireless Ad Hoc Sensor Networks", in IEEE Journal of Oceanic Engineering (Volume 40, Issues 2), 2015.

[20]. Zhangbing Zhou, Beibei Yao, Riliang Xing, Lei Shu, Shengrong Bu, "E-CARP: An Energy Efficient Routing Protocol for UWSNs in the Internet of Underwater Things", in IEEE Sensors Journal (Volume 16, Issues 11), 2016.

[21]. Zhuo Wang, Guangjie Han, Hongde Qin, Suping Zhang, Yancheng Sui, “An Energy-Aware and Void-Avoidable Routing Protocol for Underwater Sensor Networks”, in IEEE Access (Volume 6), 2018.

[22]. Yishan Su, Rong Fan, Xiaomei Fu, Zhigang Jin, “DQELR: An Adaptive Deep Q-Network-Based Energy- and Latency-Aware Routing Protocol Design for Underwater Acoustic Sensor Networks”, in IEEE Access (Volume 7), 2019.

[23]. Valerio Di Valerio, Francesco Lo Presti, Chiara Petrioli, Luigi Picari, Daniele Spaccini, Stefano Basagni, “CARMA: Channel-Aware Reinforcement Learning-Based Multi-Path Adaptive Routing for Underwater Wireless Sensor Networks”, in IEEE Journal on Selected Areas in Communications (Volume 37, Issues 11), 2019.

[24]. Zahoor Ali Khan, Muhammad Awais, Turki Ali Alghamdi, Adia Khalid, Aisha Fatima, Mariam Akbar, Nadeem Javaid, “Region Aware Proactive Routing Approaches Exploiting Energy Efficient Paths for Void Hole Avoidance in Underwater WSNs”, in IEEE Access (Volume 7), 2019.

[25]. Faiza Al Salti, N. Alzeidi, Bassel R. Arafeh, “EMGGR: an energy-efficient multipath grid-based geographic routing protocol for underwater wireless sensor networks”, in journal Wireless Networks (Volume 23, Issues 4), 2017.

[26]. Kun Hao, Haifeng Shen, Yonglei Liu, Beibei Wang, Xiujuan Du, “Integrating Localization and Energy-Awareness: A Novel Geographic Routing Protocol for Underwater Wireless Sensor Networks”, in journal Wireless Networks (Volume 23, Issues 5), 2018.

[27]. Sayyed Majid Mazinani, Hadi Yousefi, Mostafa Mirzaie, “A Vector-Based Routing Protocol in Underwater Wireless Sensor Networks”, in journal Wireless Personal Communications, 2018.

[28]. Abdul Wahid, Sungwon Lee, Dongkyun Kim, Kyung-Shik Lim, “MRP: A Localization-Free Multi-Layered Routing Protocol for Underwater Wireless Sensor Networks”, in journal Wireless Networks, 2014.

[29]. BalajiVijayan Venkateswarulu, Neduncheliyan Subbu, Sivakumar Ramamurthy, “An efficient routing protocol based on polar tracing function for underwater wireless sensor networks for mobility health monitoring system application”, in Journal of Medical Systems, 2019.

[30]. C Srimathi, Soo-Hyun Park, N Rajesh, "Proposed framework for underwater sensor Cloud for environmental Monitoring", in 2013 Fifth International Conference on Ubiquitous and Future Networks (ICUFN), 2013.

[31]. Charbel Geryes Aoun, Iyas Alloush, Yvon kermarrec, Joel Champeau, Oussama Kassem Zein, "A Mapping Approach for Marine Observatory Relying on Enterprise Architecture", in conference OCEANS - MTS/IEEE Washington, 2015.

[32]. Miaomiao Liu, Fei Ji, Quansheng Guan, Hua Yu, Fangjiong Chen, Gang Wei, "On-Surface Wireless-Assisted Opportunistic Routing for Underwater Sensor Networks", in 11th ACM International Conference on Underwater Networks & Systems, 2016.

[33]. Fatemeh Banaeizadeh, Abolfazl Toroghi Haghighat, "An energy-efficient data gathering scheme in underwater wireless sensor networks using a mobile sink" Springer, International Journal of Information Technology, March 2020.

[34]. Muhammad Faheem ; Rizwan Aslam Butt ; Basit Raza ; Hani Alquhayz ; Muhammad Waqar Ashraf ; Saleem Raza; MD. Asri Bin Ngadi ,"FFRP: Dynamic Firefly Mating Optimization Inspired Energy Efficient Routing Protocol for Internet of Underwater Wireless Sensor Networks" in IEEE Access, vol. 8, pp. 39587-39604, February 2020.

[35]. Mukhtiar Ahmed, Mazleena Salleh, M.Ibrahim Channa, "Routing protocols based on node mobility for Underwater Wireless Sensor Network (UWSN): A survey", in journal of Network and Computer Applications (Volume 78), 2017. https://doi.org/10.1016/j.jnca.2016.10.022

[36]. Mukhtiar Ahmed, Mazleena Salleh, M.Ibrahim Channa, "Routing protocols based on protocol operations for underwater wireless sensor network: A survey", in journal of Egyptian Informatics Journal (Volume 19, Issues 1), 2018. https://doi.org/10.1016/j.eij.2017.07.002

[37]. Mohammed Jouhari, Khalil Ibrahimi, Hamidou Tembine, Jalel Ben-Othman, "Underwater Wireless Sensor Networks: A Survey on Enabling Technologies, Localization Protocols, and Internet of Underwater Things", in journal IEEE Access (Volume 7), 2019.

[38]. Rodolfo W.L. Coutinho, Azzedine Boukerche, "Data Collection in Underwater Wireless Sensor Networks: Research Challenges and Potential Approaches", in 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems, 2017. https://dl.acm.org/doi/10.1145/3127540.3134267

[39]. Chithra, T.V., Milton, A. Energy Proficient Flooding Scheme Using Reduced Coverage Set Algorithm for Unreliable Links. Program Comput Soft 44, 381–387 (2018). https://doi.org/10.1134/S0361768818060117.

[40]. Sreenivasulu, D., Krishna, P.V. Deep Learning Based Efficient Channel Allocation Algorithm for Next Generation Cellular Networks. Program Comput Soft 44, 428–434 (2018). https://doi.org/10.1134/S0361768818060105.

[41]. Mohammad Furqan Ali, Dushantha Nalin K. Jayakody, Yury Alexandrovich Chursi, Soféine Affes, Sonkin Dmitry "Recent Advances and Future Directions on Underwater Wireless Communications", in Journal Archives of Computational Methods in Engineering, 2019. https://doi.org/10.1007/s11831-019-09354-8

[42]. Massobrio, R., Nesmachnow, S., Tchernykh, A. et al. Towards a Cloud Computing Paradigm for Big Data Analysis in Smart Cities. Program Comput Soft 44, 181–189 (2018). https://doi.org/10.1134/S0361768818030052.

[43]. Kozyrev, V.P. Estimation of the execution time in real-time systems. Program Comput Soft 42, 41–48 (2016). https://doi.org/10.1134/S0361768816010059.

[44]. Anwar, M.W., Rashid, M., Azam, F. et al. "A model-driven framework for design and verification of embedded systems through SystemVerilog", Des Autom Embed Syst (2019). https://doi.org/10.1007/s10617-019-09229-y

[45]. Kitchenham B. "Procedures for Performing Systematic Reviews" Elsevier, July 2004.

[46]. Kuznetsov, M.B. UML model transformation and its application to MDA technology. Program Comput Soft 33, 44–53 (2007). https://doi.org/10.1134/S0361768807010069.

[47]. Muhammad Waseem Anwar, Farooque Azam, Muazzam A Khan and Wasi Haider Butt The Applications of Model Driven Architecture (MDA) in Wireless Sensor Networks (WSN) - Techniques and Tools, In: Arai K., Bhatia R. (eds) Advances in Information and Communication. FICC 2019. Lecture Notes in Networks and Systems, vol 69. Springer, Cham

[48]. Konstantinos Skiadopoulos; Athanasios Tsipis; Konstantinos Giannakis; George Koufoudakis; Eleni Christopoulou; Konstantinos Oikonomou ; George Kormentzas and Ioannis Stavrakakis, "Synchronization of data measurements in wireless sensor networks for IoT application" IEEE Access., vol. 89, pp. 47-57, June 2019.

[49]. Sudeep Varshneya, Chiranjeev Kumara , Abhishek Swaroop, “Leach Based Hierarchical Routing Protocol for Monitoring of Overground Pipelines Using Linear Wireless Sensor Networks,” Elsevier, 6th International Conference on Smart Computing and Communications ICSCC, pp. 208–214, December 2017.

[50]. Jain U., Hussain M. “Underwater Wireless Sensor Networks, Handbook of Computer Networks and Cyber Security”, 2020 Springer, Cham. https://doi.org/10.1007/978-3-030-22277-2_9

[51]. A. Davis and H. Chang, "Underwater wireless sensor networks Oceans" 2012, pp. 1-5, doi: 10.1109/OCEANS.2012.6405141.

[52]. Nayyar A., Balas V.E., “Analysis of Simulation Tools for Underwater Sensor Networks (UWSNs).” International Conference on Innovative Computing and Communications. Lecture Notes in Networks and Systems, 2019, vol. 55. Springer, Singapore. https://doi.org/10.1007/978-981-13-2324-9_17

[53]. V. Khajuria and M. Kaur, "Underwater Wireless Sensor Network: Architecture, Applications and Challenges," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, pp. 939-944, doi: 10.1109/ICOEI.2018.8553903.

[54]. Guang Yang, Lie Dai, Guannan Si, Shuxin Wang, Shouqiang Wang, “Challenges and Security Issues in Underwater Wireless Sensor Networks”, Procedia Computer Science, 2019 , vol. 147, pp. 210-216, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2019.01.225

[55]. Hanjiang Luo, Kaishun Wu, Rukhsana Ruby, Yongquan Liang, Zhongwen Guo, and Lionel M. Ni. “Software-Defined Architectures and Technologies for Underwater Wireless Sensor Networks: A Survey”, Commun. Surveys Tuts. 20, 2018, pp. 2855–2888. DOI: https://doi.org/10.1109/COMST.2018.2842060

[56]. Hanjiang Luo, Kaishun Wu, Rukhsana Ruby, Feng Hong, Zhongwen Guo, and Lionel M. Ni., “Simulation and Experimentation Platforms for Underwater Acoustic Sensor Networks: Advancements and Challenges.” ACM Comput. Surv. 50, 2, Article 28, 2017, 44 pages. DOI: https://doi.org/10.1145/3040990

[57]. Alberto Rodrigues da Silva, “Model-driven engineering: A survey supported by the unified conceptual model”, Computer Languages, Systems & Structures, 2015 , vol. 43, pp. 139-155, ISSN 1477-8424, https://doi.org/10.1016/j.cl.2015.06.001.

[58]. Anas Abouzahra, Ayoub Sabraoui, Karim Afdel, “Model composition in Model Driven Engineering: A systematic literature review”, Information and Software Technology, 2020, vol. 125, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2020.106316.

[59]. https://www.eclipse.org/

[60]. https://www.eclipse.org/papyrus/

[61]. https://www.eclipse.org/acceleo

[62]. https://www.nsnam.org/releases/ns-3-29/

[63]. Nenad Medvidovic, David S. Rosenblum, David F. Redmiles, and Jason E. Robbins. “Modeling software architectures in the Unified Modeling Language” ACM Trans. Softw. Eng. Methodol. , 2002, 2–57. DOI: https://doi.org/10.1145/504087.504088

[64]. Aziz, K. M. "Evaluating Model Transformation Technologies-An exploratory case study." (2011).

COMPLETION CERTIFICATE

“It is certified that that the contents of thesis document titled “*A Model Driven Framework for Modeling and Verification of Underwater Wireless Sensor Networks*” submitted by Miss. Ayesha Tariq Registration No. 00000203649 have been found satisfactory for the requirement of degree”.

Thesis Advisor

(Dr. Farooque Azam)