A MODEL-DRIVEN FRAMEWORK FOR DESIGN AND ANALYSIS OF FOG BASED EHEALTH SYSTEM

Author

Rubia Anjum

Regn Number

274637

Supervisor

Dr. Farooque Azam

DEPARTMENT OF COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY

ISLAMABAD

JULY 2022

A Model-Driven Framework for Design and Analysis of Fog based EHealth System

Author

Rubia Anjum

Regn Number

274637

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 COMPUTER ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

JULY 2022

DECLARATION

I certify that this research work titled "*A Model-Driven Framework for Design and Analysis of Fog based EHealth System*" is my work. The work has not been presented elsewhere for assessment. The material that has been used from other sources has been properly acknowledged/referred.

Signature of Student

Rubia Anjum

FALL 2018-MS18(CSE)00000274637

LANGUAGE CORRECTNESS CERTIFICATE

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical, and spelling mistakes. The thesis is also according to the format given by the university.

Signature of Student

Rubia Anjum

FALL 2018-MS-18(CSE)00000274637

Signature of Supervisor

COPYRIGHT STATEMENT

ACKNOWLEDGEMENTS

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You set up in my mind to improve it. Indeed, I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am profusely thankful to my beloved parents who raised me when I was not capable of walking and continued to support me throughout every department of my life.

I would also like to express special thanks to my supervisor **Dr. Farooque Azam** and my co-supervisor, **Dr. Wasi Haider Butt**, for their constant motivation, patience, enthusiasm, and immense knowledge. Their guidance helped me throughout my 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 **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 **Dr. Arslan Shaukat** and **Dr. Umar Farooq** for being on my thesis guidance and evaluation committee. Their recommendations are very valued for improvement of the work.

I would like to pay special thanks to **Anam Amjad** for always providing me with continuous help and support throughout this thesis work.

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 parents and adored siblings whose tremendous support and cooperation led me to this wonderful accomplishment

ABSTRACT

The ehealth intervention is being used on a daily basis as the increase in the disease data such as diabetes, heart disease, lungs, neurological disorders, malignancies, and others. As a result, the volume of patient data is consuming a lot of storage space. However, advances in the field of fog computing are assisting in the storage of the massive amounts of data generated on a daily basis. Because the eHealth system is mission-critical, it is expected to be flawless. There is no room for error in the system. Furthermore, an early design and analysis of an ehealth system is required. State-of-the-art fog based ehealth techniques, on the other hand, are hard to come across in the literature. In the area of fog computing, there exist some tools (e.g., iFogSim), which provide simulation and analysis, however, it does not support modeling, which limits user's ability to design and analyze the system at early stages of SDLC. No code generation facility from the high-level model is available in fog computing tools. As a result, **M**odel-Driven **F**og based **eH**ealth System **F**ramework, an open-source framework for the construction and analysis of fog based ehealth systems, is proposed in this paper (MFeHF). **U**nified **M**odelling **L**anguage **P**rofile for **F**og based **eH**ealth system (UMLPFeH) is proposed in the proposed framework for modelling the requirements of health and fog, where the concepts of ehealth and fog are divided into four components that deal with ehealth data and data processing through fog computing. The MFeHF has created an open-source transformation engine that uses the Acceleo tool to convert UML models to JAVA language. Two case studies, (i) EEG tractor beam and (ii) smart glove, are used to validate the feasibility, cost-effectiveness, and scalability of the MFeHF for eHealth system analysis. The experimental results show that the suggested framework is a simple and reusable method for developing fog based ehealth systems.

Key Words: *Fog Computing, eHealth, Model-driven engineering, UML profile, model to text transformation/Acceleo.*

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

CHAPTER 1: INTRODUCTION

This section provides a detailed introduction to the research-related concepts. It's broken down into five components. The background study is described in **Section 1.1**, the research problem statement is presented in **Section 1.2**, the proposed methodology discusses in **Section 1.3**, the details about the research contribution are given in **Section 1.4**, and the thesis organization is presented in **Section 1.5.**

1.1 Background Study

This section's goal is to introduce the background study of several topics that were employed in this study. These ideas include the following.

1.1.1 Fog Computing:

Fog Computing is an arising standard that broadens calculation, correspondence, and storerooms toward the edge of an organization. Fog computing can uphold the delay-sensitive request for services from clients with lessened energy utilization and low blockage in comparison to traditional cloud computing [1]. Fog Computation bridges the gap between the cloud and end devices (e.g., IoT nodes) by enabling computing, systems administration, storage, and data management on network hubs within IoT gadgets' immediate vicinity. Cloud computing should be gained through the organization center, while fog computing can be obtained by related gadgets from the organization's edge to the organization's center. Furthermore, fog-based administrations do not require a constant Internet connection to function. [2]. Fog computing is not a replacement for cloud computing; although, it does lessen the requirement for data center connectivity. Regardless, both of them are employed in various applications [3] like Municipal Service [4], Smart Citizen [5], Smart Education [6], Ehealth [7], Smart Building [8], Smart Energy [9] and Smart Governance [10].

Municipal Service:

Smart cities combine advanced technologies into many municipal services, such as waste and water management (sewer and refuse), transit, parking, lighting, and so on, to improve living standards. Smart city infrastructure is used to develop municipal services. Even though each of these services is controlled by a separate municipality, they now operate independently. These services are incorporated into the smart municipal system to deliver increased services based on comprehensive knowledge.

Smart Citizen:

One of the fundamental criteria of a smart city is a smart citizen. A smart citizen is someone who uses information technology to prepare and think in order to save time and boost productivity. He is able to evaluate and improve his daily routines. Consider someone who is seeking instructions from others. Traditional methods are used to locate a location. As a result, questioning people and conducting trials will lead to the target, but it will not be optimum. When we talk about a smart citizen, we're talking about someone that uses information technology infrastructures to find their way of utilizing routing apps in an efficient and timely manner.

The smart citizen, on the other hand, does not have to be a technologist. In addition, smart citizen has a virtual identity that allows them to use smart city resources to exercise all of their citizenship rights. Furthermore, he can be a democratic participant, co-creator, or ICT user in a smart city. As a result, the smart citizen is a crucial idea in smart cities.

Smart Education:

One of the areas where technological advancements have a huge influence is education. Learning and teaching based on technology developments allow for lower costs, greater quality, and distant learning, allowing for simple access to high-quality education at any time and from any location. As a result, smart education initiatives have gotten a lot of coverage in recent years all around the world.

A precise definition of smart education is extremely difficult to obtain. Smart learners and a smart environment are two fundamental components that constitute this notion. For example, to enhance a smart education system, [11] discusses the smart environment, smart educational technologies, smart educational materials, a new generation of students, professional interconnected communities, and quick knowledge delivery to students. The role of technology in educational institutions is, nevertheless, emphasized in all of the associated publications.

eHealth:

Improving the healthcare system is a critical component of raising people's quality of life. The Internet of Things (IoT) provides the necessary infrastructure for this purpose, paving the way for smart healthcare. Remote patient monitoring, home care, emergency healthcare, and rehabilitation are all possible with today's healthcare systems. The benefits of smart healthcare include improving humanity's health and well-being by speeding diagnosis and personalized treatments, boosting efficiency, lowering management and treatment costs, and so on.

Smart Building:

The advent of a new concept known as the smart building is the result of changing lifestyles and rising comfort levels. Humans are predicted to be able to regulate their lives and work in an automated and efficient manner in the future. People are spending more time indoors [12]. A smart building includes features such as smart lighting, smart dimmers, intelligent cooling and heating systems, smart parking, intelligent surveillance and security systems, smart audio and video systems, electronic curtains, smart engine rooms, smart irrigation, intelligent fire systems, and more. Depending on the style of construction, all or some of these alternatives may be used. The intelligent building prototype in [13] is based on fog and cloud architecture, which allows for real-time local control. The energy consumption improves as well, even though the human effort to control and monitor is reduced.

Smart Energy:

One of the major issues in cities, particularly smart cities, is effective energy management. As a result, one of the most critical criteria of a smart city is the presence of an energy management system. An energy management system is an automated system that monitors, manages, and optimizes energy production, distribution, and consumption. Smart technology is used to create a smart energy management system that strives to achieve a low carbon, efficient distribution, and optimal usage. It caters to both home and business applications. The benefits of smart energy management systems include discovering energy-saving opportunities, analyzing upgrading alternatives, and locating financial incentives for future energy ecosystems [14].

Smart Governance:

Smart government has become a must for smart cities, and it aims to better serve inhabitants by utilizing information and communication technologies. Online services, electronic payments, the use of open data to enhance government transparency, online contact with local governments, and other aspects of government service delivery improve accountability and citizen participation. It also gives them the knowledge to help them plan and make better decisions [15].

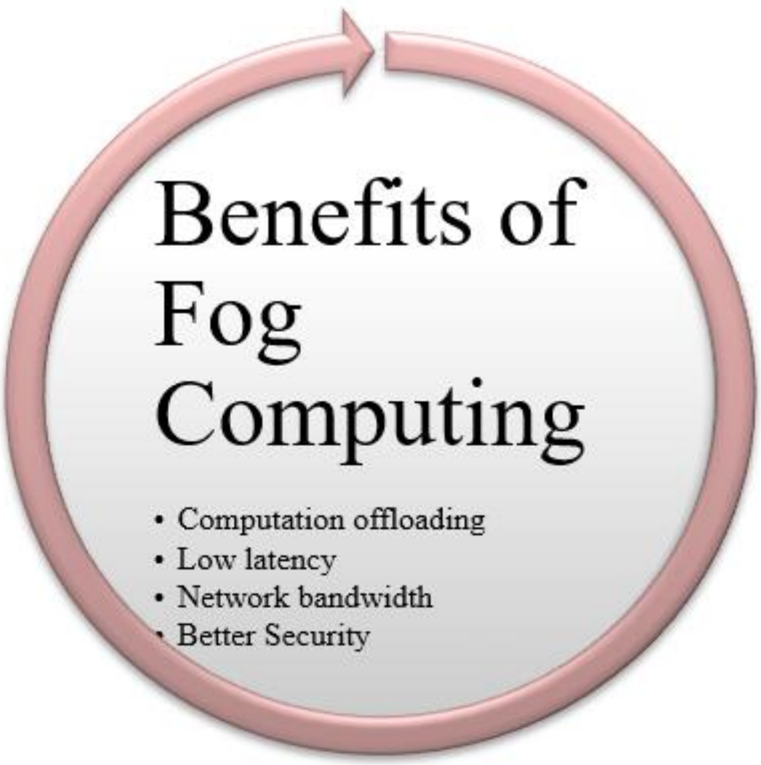


Figure 1: Benefits of Fog Computing

Computation Offloading One option for speeding up applications is to move the most computationally intensive areas (also known as hotspots or crucial regions) to more powerful processing devices, such as servers, and hardware accelerators (e.g., GPUs, FPGAs), supercomputing assets, or distributed computing frameworks. Computation offloading is the term for this type of computation transfer. [16].

Latency incorporates the over-the-air transmission delay, lining delay, preparing/figuring deferral, and retransmissions when required. Guaranteeing a full circle dormancy of 1 ms and inferable from the speed of light requirements (300 km/ms), the most extreme distance at which a collector can be found is around 150 km [17]. Latency is the deferral between a client's activity and a web application's reaction to that activity, regularly alluded to in systems administration terms as the absolute full circle time it takes for an information bundle to travel.

Network Bandwidth The benefit of delivering information between workers and customers at a specific moment in bits per second is known as bandwidth (bps). A large recurrence of the signal used in the transmission media is also known as bandwidth. In computer networks, bandwidth is commonly used as an equivalent for delivering data, which is the amount of data that can be transported from one place to the next in a certain time frame, usually measured in seconds. [18]. Fog computing can reduce both propagational latency and bandwidth use by placing fog nodes close to IoT devices and end-users. [19].

Security In the perception layer of IoT, sensor hubs have restricted computational force and low database limit, which make the recurrence bouncing transmission application and public-key encryption to guard the IoT gadgets inaccessible. Lightweight encryption innovation, which incorporates lightweight cryptographic calculation, is utilized for IoT gadgets. In the network layer, the IoT network has security issues, like man-in-the-center and fake assaults. The two assaults can catch and send counterfeit data to transmitting nodes in the network [18].

1.1.2 eHealth System:

eHealth is a widely used phrase in the healthcare industry, and it refers to the use of information and communication technology. In terms of standard procedure, eHealth can be reached through any sort of automated device or health monitoring system. That is utilized by practitioners in the healthcare discipline or by individuals to control or enhance their health [20]. The use and effectiveness of third-party servers for the storage and evaluation of health data raises security and privacy concerns that, if not addressed, could limit the adoption of the collaborative health standard [21]. In the following, we discuss the domains of eHealth that play an instrumental role.

Electronic Health Record (EHR) are computerized adaptations of each patient's healthcare records, these records are maintained by healthcare suppliers for a long period, that carries data associated with a patient's consideration, including socioeconomics, examination, operation, prescriptions, indispensable signs, vaccination, research facility results, and images of radiology [22]. EHRs contain comprehensive knowledge associated with disease diagnosis for extensive patient communities. Although the essential objective of the EHR is to record patients' consideration for repayment, a secondary purpose for the obtained data is to be used as a research platform [23].

Electronic Medical Records (EMR) guide people on how to avert diseases and enhance the healing rate. It gives a huge premise to healthcare organizations and pharmaceutical corporations and gives legitimate proof for healthcare carelessness and disputes [24].

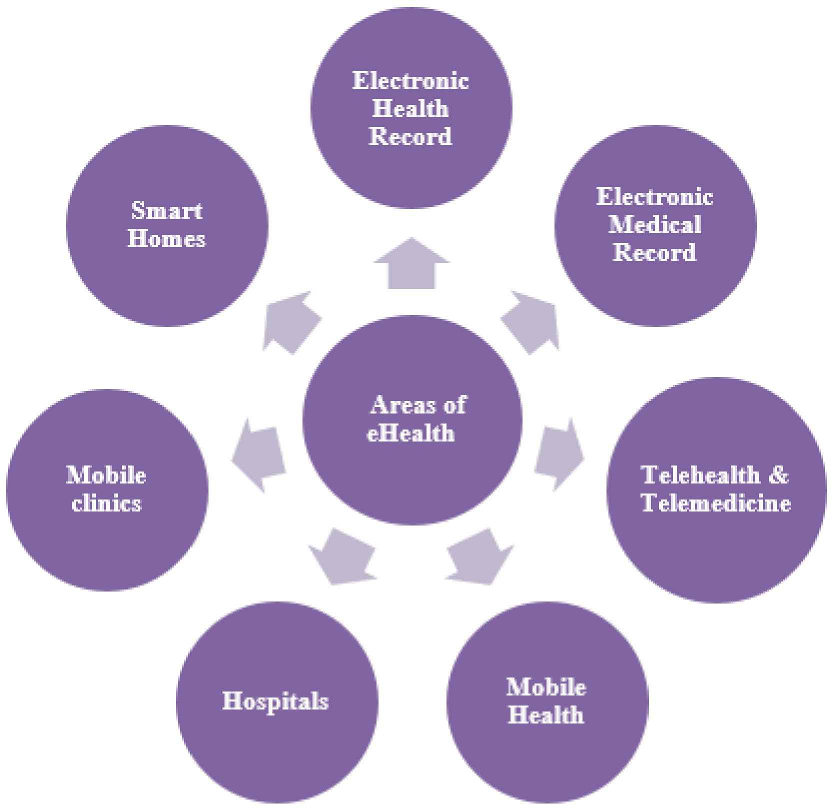


Figure 2: Areas of eHealth

Telehealth and Telemedicine Telehealth frameworks provide remote care for older people and people with less physical strength furthermore it provides remote medical procedures, therapies, and examinations [25]. Telemedicine is the theoretical term used to characterize clinical benefits conveyed through Information Technology and Telecommunications [26]. Telemedicine can conceivably change medical services conveyance in low-asset conditions by empowering the augmentation of clinical information to remote areas, in this way improving the productivity and adequacy of the bigger medical care framework [27].

Mobile Health (mHealth) Progressions in cell phones are significantly changing the universe of medical services. The accessibility of individual clinical or health gadgets is recommended as a potential way to minimize medical care costs in a developing world. For example, clients/patients can undertake at-home recovery with these devices, which allows them to recover in a familiar environment without the need for a professional [28].

Hospitals are growing larger with assets to handle, admit, control, or observe a developing amount of patients. They rely on technological advancements to complete their tasks efficiently. Smart ambulances, for example, would achieve quick analysis to the point where clinical professionals might formulate prospective plans before the patient arrived. With the integration of IoT, a situation like this can be created [29]. Smart ambulances will need a variety of dependable health sensors for detection, a secure communication link with the hospital, and innovative work management tools for emergency clinic arrangements, which might include a working space where a specialist knows the patient's condition ahead of time. Medical operations could become more context-aware and efficient as a result of IoT when data exchange between specialists, clinical staff, and clinical devices in the operating room is seamless. Also, one may foresee the use of IoT in various medical departments, such as specialized units, intensive care units, and care units [30].

Mobile Clinics All over the world there is growing interest in mobile health clinics[31]. Portable health centers, for example, have been shown in African countries to provide high-quality, low-cost care to unprotected communities in remote areas where residents lack access to basic clinical facilities. Portable medical centers, on the other hand, are vehicles with limited clinical capabilities. IoT could have a significant impact on the architecture of mobile clinics, which could collaborate with large emergency clinics to provide analysis and selection assistance to remote medical centers. As a result, mobile medical facilities may be able to expand their capacity to provide care [30].

Smart Homes are a cyber-physical framework where houses are furnished with interconnected sensors and smart devices which are managed across the Internet to keep track of the atmosphere and guarantee house and personal security and safety [32].

1.1.3 Fog based eHealth System:

A vast number of applications such as smart wearables, smart homes, smart mobility, and smart cities. Healthcare is becoming increasingly difficult to manage due to insufficient and less effective healthcare services to meet the increasing demands of the rising aging population with chronic diseases. e-Health applications often need to manage streaming-based transmissions where real-time requirements need to be considered. Consequently, considerable energy is dissipated during the transmission process. Reliability in e-Health applications is of utmost importance and even short system unavailability often cannot be tolerated [30]. healthcare applications are mission-critical so they must have energy efficiency, performance, reliability, interoperability, and security.

To address these difficulties, the Fog computing paradigm is recommended; It focuses on the integration of edge layer storage, data processing, data handling, management, and administration equipment, such that calculations previously performed in clouds are now partially offloaded to network edges. As a result, network latency between end-users and the cloud can be reduced greatly, and data processing results can be obtained more efficiently. Better service quality and data evaluation can be obtained with such a computing unit at the network's edge.

Fog is an intermediate computing layer between the cloud and end devices that complement the advantages of cloud computing by providing additional services for the emerging requirements. This intermediate layer is discussed in different terms in various articles, such as Mobile Edge computing [33], Micro-clouds [34], or simply just Edge Computing [35]. fog layer requires continuously handling a large amount of sensory data in a short time and responding appropriately concerning various conditions. This task becomes more important in medical cases by enabling the system to react as fast as possible in medical emergencies [36].

Fog based eHealth System Architecture

The layered architecture of fog-based eHealth systems is presented in **Figure 3.** It is an organization of operational elements all over the network from sensors to the cloud.

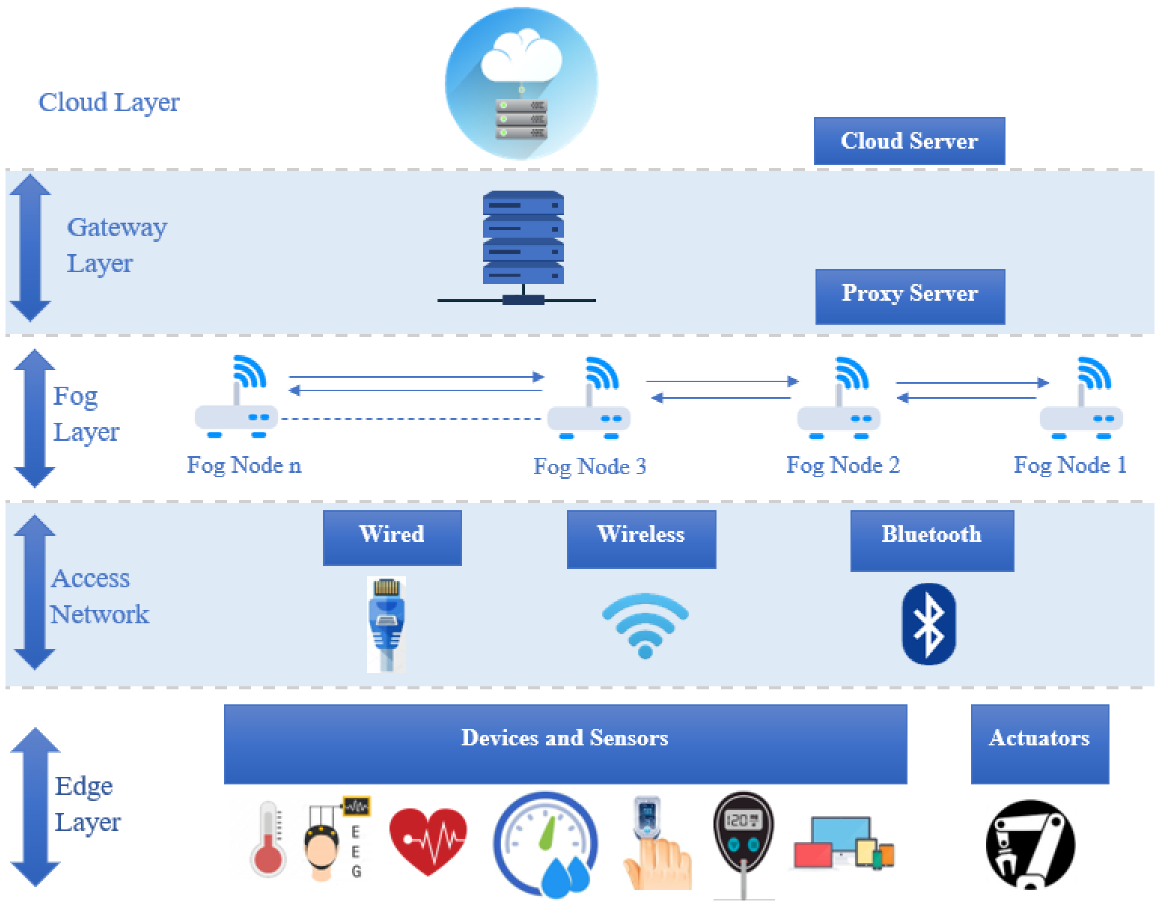


Figure 3: Fog based eHealth System Architecture

Sensors exist at the lowest level of the architecture and are mounted in different sites, detecting the surroundings, and releasing detected data to upper levels through gateways for further operations and refinement. Likewise, **Actuators** also exist at the lowest level and are in charge of regulatory an operation or the structure. The actuators are commonly intended to react to deviations in surroundings that are caught by sensors. **Data Values** are defined as the series of continuous values produced by sensors. **Access network** has different deployment settings based on the deployment budget. (a) The wired network functions as the access network when the infrastructure is somewhat extensive. (b) When a wired network is not fully covered, wireless network communication is the superior option due to its lower cost of implementation and system configurability. Any component in the fog network capable of hosting a portion of the application units is referred to as a Fog Device. Cloud resources made available on-demand from numerous data centers are likewise included in fog devices [37]. Fog Devices are connected to the cloud with **Gateway.** The **Cloud Layer** provides facilities like storage, forming a knowledge base to aid future decisions, perform analysis, and support GUI activities.

1.1.4 Model-Driven Engineering

Model-Driven Engineering (MDE) [38] or Model-Driven Development (MDD) [39] is a software development paradigm that emphasizes the creation of complex, dependable, and reusable systems. MDE uses a set of tools to design abstract-level domain models that depict concepts of a specific complex problem and then automatically transform these models into required platform-specific code. With the MDE approach, individuals need to focus on modeling software instead of handwriting and continuous debugging of low-level code. Domain models are created by a substantial collaboration between product managers, developers, designers, and application end users. Abstraction and automation are its key concepts. MDE helps to write and implement software applications quickly, effectively, and with little cost. MDE is already successfully practiced across a wide range of applications such as controller design, integrated circuit design, and other complex software systems.

MDD uses different languages to develop conceptual and metamodels of a complex problem of a specific domain. Among these languages, The Unified Modeling Language (UML) is commonly used. UML defines the modeling architecture as well as the design of a system by providing language constructs for system components, interfaces, objects, interactions, etc. UML conceptual models can be extended and customized by using UML Profile. UML

Profile allows adapting UML language according to the system requirements or customer needs. Another important aspect of MDD is Model Transformation which performs automatic transformation of source models into target models. Transformation is based on mapping rules that define how each source element needs to be transformed into the respective target element. Thus, for a given source model along with metamodels of both source and target models, applying transformation rules can automatically generate deployable target domain applications. This transformation can be from a platform-independent model to a platform-specific model (Model to Model) or from a platform-independent model to code (Model to Text) as described in **Figure 4**.

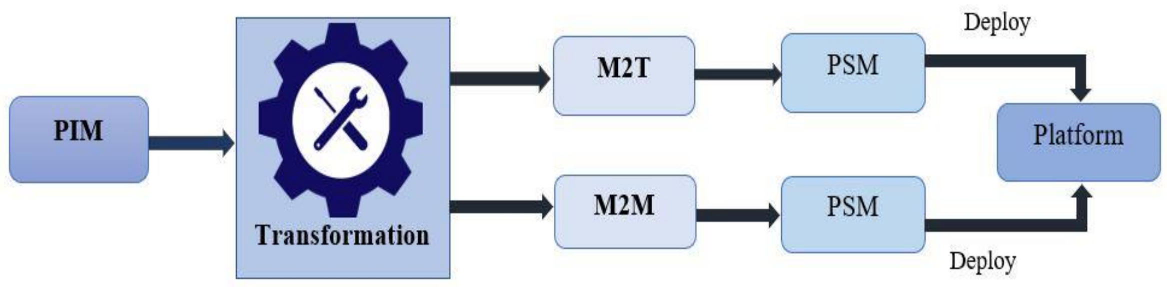


Figure 4: Model-Driven Engineering

1.2 Problem Statement

Producing immense volumes of information can devastate data-storing systems and intelligent applications. For these applications to work efficiently they need low latency communication, but the delay produced by conveying information to the cloud and then returning to the application back can affect their effectiveness. Therefore, fog-based architecture is required with seamless communication and minimal latency.

Moreover, In the area of fog computing, there exist some tools (e.g., iFogSim), which provide simulation and analysis, however, it does not support ***modeling***, which limits the user's ability to design & analyze the system at the early stages of SDLC.

Although there exist some studies that work on model-driven approaches but there is no code generation facility from the high-level model is available in fog computing tools.

Therefore, there is a strong need to develop a fog based framework that provides an easy and reusable solution for an eHealth system. We Provide a model-driven approach based on MDA standards using UML Profile. From a high-level model, we generate low-level complex code.

In this context, Model-Driven Engineering provides the ability to create models to represent fog based eHealth system requirements in early phases which can later be converted into low-level java code with minimum effort, hence aiding the implementation process.

Hence, by providing a modeling approach we achieve the following objectives for the designers/developer community: -

Early Analysis: They can analyze the system at the early stages of SDLC.

Reusability: They can reuse the design because it developed as a platform-independent model (PIM).

Low Complexity: They can generate the code through the high-level model, hence, most of the complex code will be generated.

1.3Proposed Methodology

The entire research is done in a systematic way **Figure 5** represents the flow of research step by step. In the first step, we identified the problem. Then proposed the ideal solution for the problem identified in the first step. We carried out a detailed and comprehensive literature review that helped us to identify the optimal solution for the problem. We reviewed the research carried out related to the proposed solution, analyzed it, and compared it. We also performed a comparative analysis of tools and techniques discussed for fog based eHealth systems earlier.

The **M**odel-driven **F**og based **eH**ealth **F**ramework **(MFeHF)** is introduced for the design and analysis of fog-based ehealth systems. MFeHF involves a **U**nified **M**odelling **L**anguage **P**rofile for **F**og-based **eH**ealth system **(UMLPFeH)** for modeling the requirements of ehealth and fog. The concepts of ehealth and fog are divided into four components that deal with the data of health and processing of data through fog computing. An open-source transformation engine is included in the MFeHF. The transformation engine is implemented using the JAVA programming language and the Acceleo tool, using a model-to-text transformation strategy. It automatically transforms models into target java file implementations. The MFeHF is demonstrated through two case studies that prove that the MFeHF is feasible, and scalable for the analysis of the eHealth system.

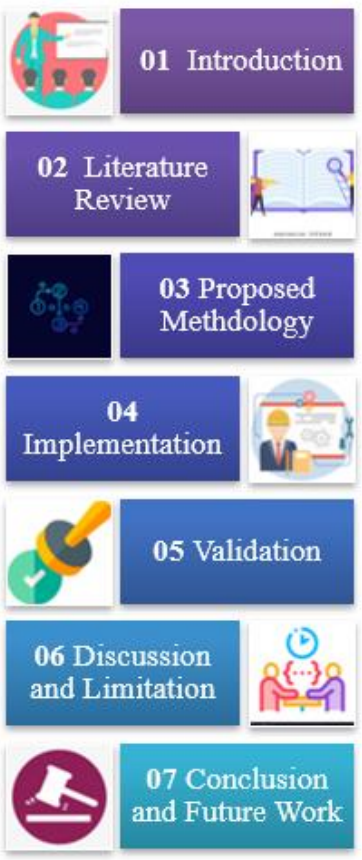


Figure 5: Research Flow

1.4 Research Contribution

A complete, open-source, and automated Fog-based eHealth System is proposed in this work. A detailed set of contributions of the proposed approach are as follows:

1. To create a fog based ehealth system, a model-driven framework is offered. It contributes to a greater level of system abstraction and decreases design complexity.
2. UML Profile for Fog based eHealth System (UMLFeH) is developed to design or develop an eHealth system for critical analysis.
3. A transformation tool is developed for transforming the input UML model into complete Health java code by using defined transformation rules.
4. The automatically generated development artifacts are deployable java code for the eHealth system (.java files).

1.5 Thesis Organization

The organization of the thesis is represented in**. CHAPTER 1:** offers a brief introduction containing the background study, problem statement, research contribution, and thesis organization. **CHAPTER 2:** gives a thorough assessment of the literature, highlighting the work that has been done in the domain of Fog Computing and eHealth. **CHAPTER 3:** covers the details of the proposed methodology used for the identification of the problem. **CHAPTER 4:** shows the architecture as well as the detailed implementation of the suggested method, profile, tool, and transformation engine. **CHAPTER 5:** provides two essential case studies for the validation of our suggested technique. The two case studies chosen for validation are from different domains and sizes to ensure that our proposed approach works in every situation. **CHAPTER 6:** contains a quick explanation of the study done as well as the research's limitations. **CHAPTER 7:** summarizes the study and makes recommendations for the future.

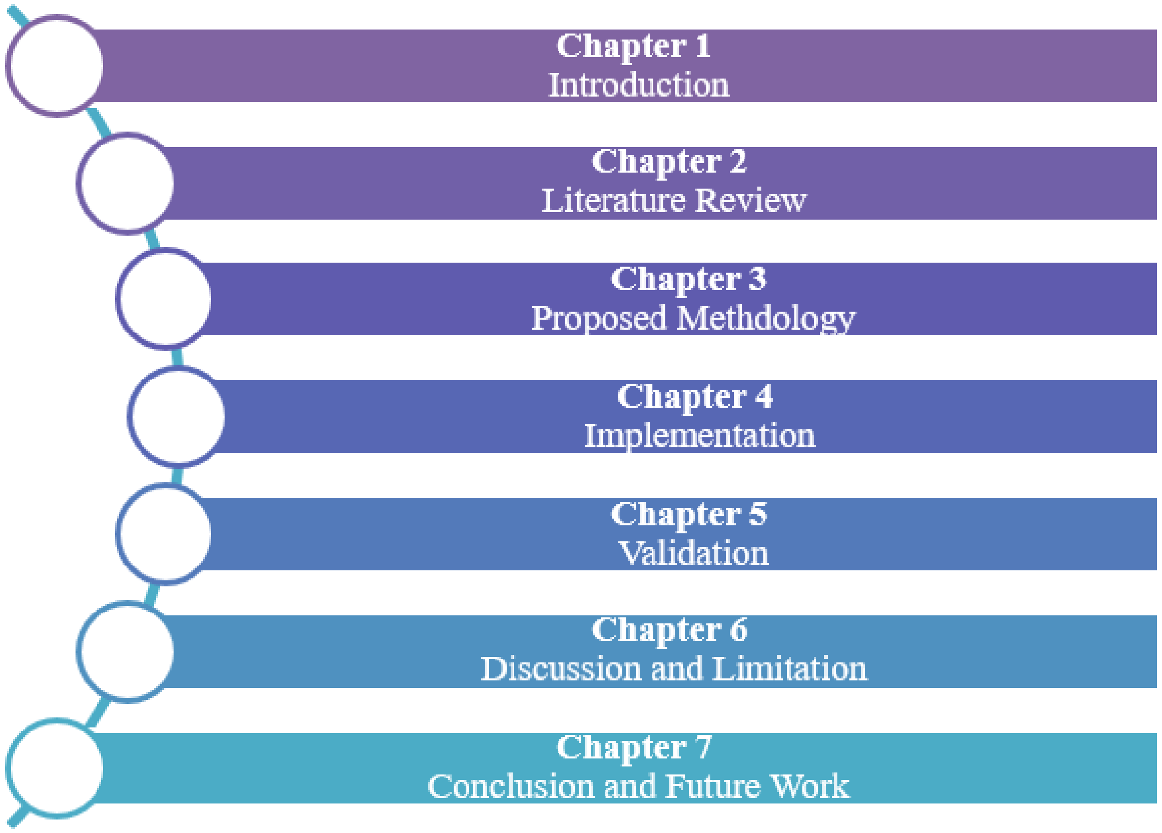


Figure 6: Thesis Outline

Chapter 2

Literature Review

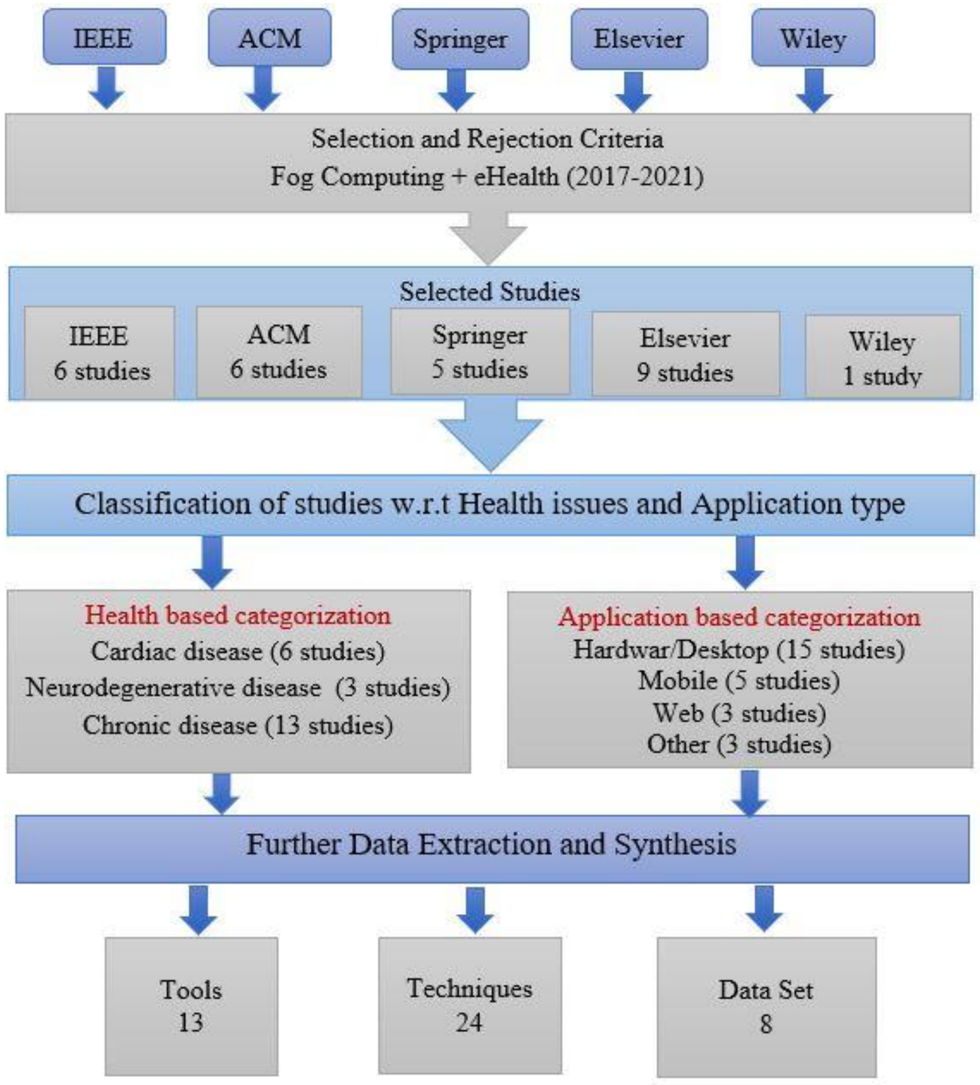
CHAPTER 2: LITERATURE REVIEW

The literature review on fog computing and eHealth studies is undertaken in this chapter. This research includes a systematic evaluation of the literature on the subject of ehealth and an analysis of past fog computing projects. Motivation, limits faced by researchers, and ideas provided to analysts for developing this crucial study field were the criteria and attributes that were studied for enhancing understanding of different relevant parts of this topic in literature. This study's key contribution is the categorization of studies and the identification of proposed approaches and tools. The datasets that are utilized to get the results.

Fog computing is a computing architecture that makes use of the collaborative aggregation of end-user clients or close customer end devices to eliminate a large amount of storage, management, control, communication, and configuration [40]. It is developing as an alluring solution to the issue of information handling in IoT. It is dependent on the devices at the system's edge, which have higher processing power than end devices and is closer to them than the more prominent cloud resources, resulting in lower application latency[41]. Fog computing permits the provisioning of sources and services outside the cloud, at the network edge, closer to end devices, or ultimately, at ranges designated by Service Level Agreement. Fog computing certifiably isn’t a replacement for cloud computing yet a strong complement. It facilitates processing at the endpoint while still allowing the opportunity to cooperate with the cloud [42].

With the developing utilization of innovation and digital instruments, it is clear that the healthcare area is likewise grasping this from multiple points of view incorporating eHealth. In terms of common practice, eHealth can be reached through any sort of automated device or health monitoring system. That is utilized by practitioners in the healthcare discipline or by individuals to control or enhance their health [43]. The use and effectiveness of third-party servers for the storage and evaluation of health data raises security and privacy concerns that, if not addressed, could limit the adoption of the collaborative health standard. This considerably increases the issue of whether the patient is certainly in care of his/her health reports [44].

Fog computing and eHealth can be a powerful combination for society. Because it can provide mobile patients with remote real-time monitoring, instruction, and supervision, it can improve their autonomy, and self-confidence, and save onsite expenditures. Patients will be able to maintain their normal daily activities [45] through customized healthcare solutions for example [46] wherever they are; low-cost cell phones and sensors can provide a low-cost foundation to exchange data with the environment and remote servers and accumulate valuable information from the usual circumstances of the patient or her/his surroundings such as climatic conditions, city information on traffic [47], pollution, routes, etc.



IEEE
ACM
Springer
Elsevier
Wiley
Selection and Rejection Criteria
Fog Computing + eHealth (2017-2021)
Selected Studies
IEEE 6 studies
ACM 6 studies
Springer 5 studies
Elsevier 9 studies
Wiley 1 study
Classification of studies w.r.t Health issues and Application type
Health based categorization
Cardiac disease (6 studies)
Neurodegenerative disease (3 studies)
Chronic disease (13 studies)
Application based categorization
Hardwar/Desktop (15 studies)
Mobile (5 studies)
Web (3 studies)
Other (3 studies)
Further Data Extraction and Synthesis
Tools 13
Techniques 24
Data Set 8

Figure 7: Research Overview

2.1 Category Definition:

Health:

We have classified the selected research into three categories based on health-related issues.

- *Cardiac Disease:* It affects the structure or function of health. Arrhythmias, aorta disease, coronary artery disease, heart attack, heart failure, heart muscle disease, heart valve disease, and vascular disease are some of the conditions involved.

- *Neurodegenerative Disease:* The nervous system, which includes the brain and spinal cord, is made up of neurons. Neurons do not ordinarily multiply or replace themselves, so when they are injured or die, the body is unable to replace them. Parkinson's disease, Alzheimer's disease, Huntington's disease, Dementia, Epilepsy, Mental Illness, Movement Disorder, and Stroke are all included.
- *Chronic disease:* This category involves chronic diseases like Diabetes, Cancer, Asthma, Respiratory, and Lung disease.

Applications:

The selected research is classified into four categories based upon their usage of platforms.

Hardware/ Desktop Application:

In the investigated studies, the health-related data collection was done by utilizing different hardware devices requiring desktop applications for the stimulation of fog devices. Due to this hardware requirement of desktop applications, both are being discussed under the same heading.

Mobile Application:

The investigated research consisted of some case studies which were validated by the usage of mobile applications gathering an individual’s health-related data and identifying the potential health issues. Based on the issues identified, these applications also provided some precautionary measures along with the recommendation of the concerned health specialist to seek medical care.

Web Application:

The validation of the case studies presented by the investigated research was being done through the utilization of web applications.

Other:

This category consists of applications that are either web desktop or mobile desktop or none among both.

ML Techniques:

There are several types of techniques. These techniques include many other techniques. For example, linear SVM. We are considering the main ML technique in this paper the other sub-categories fall under the major category. Some of the techniques identified are as follows.

- Artificial Neural Network (ANN)
- Support Vector Machine (SVM)
- Convolutional Neural Network (CNN)
- Naïve Bayes (NB)
- Flow Clustering and Analysis (FCA)
- K-means

This section will offer a systematic literature assessment of fog computing technologies in healthcare systems, following and examining the previous work.

2.2 Review Protocol:

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

2.2.1 Research Questions:

RQ1: What are the proposed fog computing techniques/frameworks in the area of ehealth?
RQ2: What fog computing tools are accessible in the area of ehealth?
RQ3: What are the key benefits and limitations of utilizing fog computing in ehealth?

2.2.2 Selection and Rejection Criteria:

The SLR is managed through the selection and rejection criteria mentioned below.

1. Only those studies are selected in which ehealth uses fog computing.
2. The papers that were chosen should be published in one of the five scientific repositories, which include: IEEE, Springer, ACM, Elsevier, and Wiley are some of the most well-known publishers.
3. The studies chosen should be published between 2017 and 2021.

The studies that can be selected must follow all the above-mentioned rules. If any of the above-mentioned rules are not followed it would lead to the rejection of the study. e.g. The studies that follow the first two rules but were published before 2017 should be rejected.

2.2.3 Search Process:

The search process is conducted on a well-defined selection rule. We only consider five recognized repositories IEEE, ACM, Springer, Elsevier, and Wiley. We apply various search terms in each repository and the "AND", and "OR" operators to get corresponding results as shown in **Table 1**. The year filter is applied (i.e. 2017–2021) to enforce the third selection rule during the search process.

Table 1: Search Process after applying filters

Search Term	**Results**				
	IEEE	**ACM**	**Springer**	**Elsevier**	**Wiley**
"fog" and "ehealth"	4	1631	63	80	0
Fog ehealth	4	1629	63	80	22
Fog computing ehealth	4	132993	55	76	20
"fog computing" OR "ehealth" OR "Model Driven"	3393	1975	8854	5515	1859
"fog computing" AND "Model Driven"	96	12	1977	57	8
"fog computing" OR "ehealth"	2029	813	4506	2671	1018
"fog computing" OR "health"	29106	16341	7097	753135	328759
"fog computing" OR "health" AND "model driven"	2056	649	2869	1878	614
("fog computing" OR "health") AND "model driven"	29	143	2869	770	255

Through sequential steps, the search process is conducted. For example, by reading the title we reject multiple search results. Similarly, many research studies are rejected after reading the abstract. Some studies are rejected after reviewing the entire content. Finally, we choose twenty-six (26) papers that strictly adhere to our selection and rejection criteria, as shown in **Fig 8**.

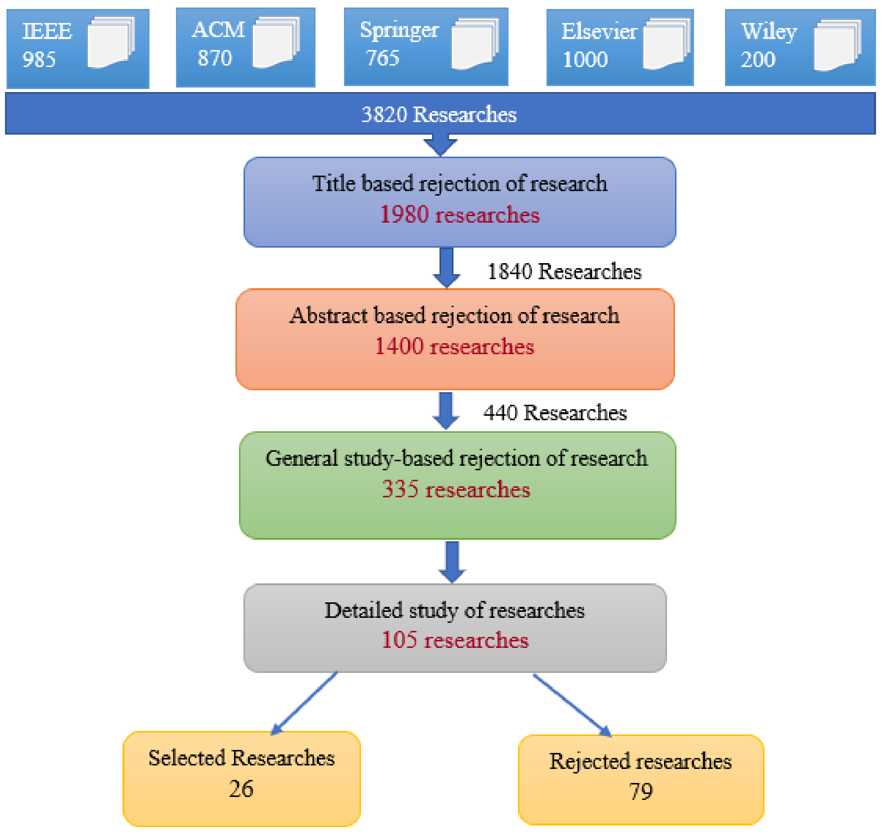


Figure 8: Research Process

2.2.4 Quality Assessment:

To enhance the relevance or to decrease the irrelevance and biasness towards the studies under review, the proposed SLR ensures reliability along with the quality of the results acquired. We apply the year filter between 2017-2020 so that we can get the latest research and analyze the latest trends of the study. We have selected 1 study from the year 2020, 9 studies from 2019, 11 studies from 2018, and 5 studies from 2017. The results are given in **Table 2**.

We ensure that high-impact studies are chosen to ensure that the SLR's results are credible. The databases chosen, for example, are authentic and widely recognized. Six studies from IEEE, six studies from ACM, five studies from Springer, eight studies from Elsevier, and one study from Wiley were chosen. The outcomes are presented in **Table 3**.

Table 2: Selected Studies yearly

Sr No.	Publication Year	Selected Studies	Total
1	2020	[53]	1
2	2019	[51] [54] [59] [60] [61] [62] [66] [69] [73]	9

3	2018	[48] [49] [50] [52] [55] [56] [57] [68] [69] [70] [71]	11
4	2017	[58] [63] [64] [65] [67]	5

Table 3: Selected studies distribution

Database	Type	Studies	Total
IEEE	Journal	[55] [56] [61] [71] [72]	6
	Conference	[57]	
ACM	Journal	[59] [60] [64]	6
	Conference	[58] [67] [70]	
Springer	Journal	[66] [69]	5
	Conference	[62] [65] [67]	
Elsevier	Journal	[48] [49] [50] [51] [52] [53] [63]	8
	Conference	[54]	
Wiley	Journal	[73]	1
	Conference	Nil	

2.2.5 Data Extraction:

For data extraction, a formal template, presented in **Table 4** is followed to collect all relevant and authentic information from selected studies.

Table 4: Data Extraction Template

Section	Details
Information	Title, Publication Year, Author
	Data Extraction
Summary	Objective, impact of the study
Assumption	Study Limitations
Validation	Experimental evaluation or other
	Data Synthesis
Health issues	Health issues are classified into 3 categories (Table 5)
Research studies	Research studies are classified into 4 categories (Table 6)
Proposed technique/ framework	The techniques/ frameworks proposed by researchers (Table 7)
Tools/ Simulators	Tools utilized (Table 8) (Table 11)

Machine Learning	The researchers utilized the ML Techniques (Table 9)
Data Set	Different types of datasets are identified (Table 10)

2.3 Results:

We analyze 26 studies and classify health-related issues into 3 categories and research studies into 4 categories. We discover the 17 techniques/ framework proposed by the researchers and 14 tools/simulators utilized. Some of the researchers used ML techniques in their research studies. 7 studies are identified that utilize ML techniques. 8 datasets are used in different studies.

Table 5:Classification of study based on health issues

Sr No	Health issues	Studies	Total
1	Cardiac disease	[48] [52] [53] [58] [65] [71]	6
2	Neurodegenerative (brain)	[49] [57] [72]	3
3	Chronic	[51] [54] [55] [56] [59] [60] [61] [64] [66] [67] [68] [69] [70]	13

The selected studies are classified into 2 different categories. These are classified based on health issues and application type. The health issues are further classified into 3 types. There are 6 studies related to cardiac disease, 3 studies related to neurodegenerative disease, and 13 studies related to the chronic category as given in **Table 5**.

The application types are further classified into 4 types. There are 15 studies related to hardware/desktop, 5 studies related to the mobile application, 3 studies related to web application, and 3 related to the other category as given in **Table 6**.

The categorization of studies given in **Table 5** and **Table 6** simplifies the analysis and synthesis process.

Table 6: Classification of Research Studies on the basis of application type

Sr No	Category	Studies	Total
1	Hardware/ Desktop	[48] [49] [50] [51] [53] [54] [57] [58] [65] [66] [68]	15

		[69] [70] [71] [72]	
2	Mobile Application	[52] [59] [60] [61] [67]	5
3	Web Application	[55] [62] [63]	3
4	Other	[56] [64] [73]	3

Table 7: Proposed Technique/ framework

Sr No	**Proposed Technique / Framework**	**Purpose**
	Techniques	
1	UTGate	Data compression, data fusion, Web socket server. [48]
2	DTW Android-based smart watch Healthcare 4.0	Used for the deployment and design of a patient-centric medical data analysis system. [52]
3	BBN Classifier	Classify an event's appearance into normal and abnormal and calculate their probabilities. [55]
4	UbeHealth	Dynamically manage the QoS and increase the network performance. [56]
5	Hierarchical computing architecture	Remote IoT-based patient monitoring system. [58]
6	PFHDS	Obtain data secrecy and fine-grained access control. [59]
7	Three-party AKA protocol with bilinear pairings	Security, Performance. [62]
8	Measurement methodology holistic approach	Selection of user equipment to fulfill communication requirements [63]
9	Taxonomy of SHM technique	Obtain awareness into a cyber-physical system design. [64]
10	Body Edge	Guarantee high flexibility, robustness, and adaptive service level. Reduce transmitted data and processing time. [72]
	Framework	
11	multi-device HAR framework	Fog computing is used to conduct sophisticated computations from the sensing layer to intermediary

		devices, and eventually to the cloud. [60]
12	e-healthcare framework LINDDUN	Deals with privacy issues. [61]
13	F2CDM	Low delay, distributed storage [65]
14	TIMON Cloud	Providing a cooperative framework for all users of the transportation ecosystem. [67]
15	IoT Fog Framework	To achieve good QoS, fog nodes must work together and manage resources and jobs optimally. [68]
16	Framework for cloud healthcare recommender service	Ensures privacy [69]
17	REDPFramework RCA (algo)	Improve the reliability of data transportation and processing speed. [71]

In **Table 7** we analyze different techniques and frameworks proposed by the researchers. In the context of fog computing, they were used for various ehealth objectives. This will help in data security, privacy, and transfer of data. These techniques/frameworks deal with both ehealth and fog computing.

Table 8: Available Tools/Simulators

Sr No	Tools/Simulators	Purpose	Study
1	Mongo DB	Database	[49] [65]
2	Ubuntu	OS for cloud computing to support Open Stack	[50] [65]
3	Raspberry Pi	Platform used for IoT developments	[51] [72]
4	Arduino	Easy to use hardware that designs single-board microcontrollers.	[54]
5	Amazon EC2	Member of Amazon’s Cloud Computing. Enable users to borrow virtual computers to drive their applications.	[55] [66]
6	Weka	Contains ML algorithms, and contain tools for data processing, distribution, regression, clustering, association rules, and visualization.	[55]
7	Icancloud	Model and simulate cloud computing systems.	[56]

	simulator		
8	iFogSim simulator	Modelling and simulation of resource management.	[57] [70]
9	Barreto-naehrig	A system to produce pairing-friendly elliptic curves over a prime field. Compute the performance of PFHDS.	[59]
10	SQL	Database	[61] [69]
11	Cloud Platform	Simulate cloud server and fog node	[62]
12	Open Air Interface	Support Mobile telecommunication systems. Develop access solutions for network, radio, and core networks.	[63]
13	AALHP	Combine current and cooperative smart house institutions into homecare items, study data, and then take action based on the findings.	[67]

In **Table 8** the tools given are available online and are easy to use. These tools ensure the quality of the results. The SLR is conducted to identify the best-used tools. The tools used for database purposes like Mongo DB and SQL are reliable for storing a large amount of data. Our analysis shows that some tools most frequently use for ehealth and fog computing include Ubuntu, Raspberry pi, and iFogSim.

Table 9: ML techniques utilized in studies.

Study	**ML Technique**					
	ANN	**SVM**	**CNN**	**NB**	**FCA**	**K-means**
[53]	✓	✓	✓	×	×	×
[55]	×	×	×	✓	×	×
[56]	✓	×	×	×	✓	×
[58]	×	✓	×	×	×	×
[59]	×	×	×	✓	×	×
[60]	×	✓	×	×	×	✓
[66]	×	×	×	✓	×	×

In **Table 9** Machine Learning techniques are given. Some researchers employ Machine Learning approaches to attain high data accuracy. The patient's information must be accurately predicted to identify the actual health issue with the patient.

Table 10: Datasets utilized for ehealth.

Sr No	Dataset	Open source	Studies	Feature
1	MIT-BIH Arrhythmia	Yes	[48] [54]	110000 beats
2	Beth Israel Hospital Arrhythmia	No	[52]	Nil
3	UCI	Yes	[56] [66] [71]	19908 data instances
4	ISPDSL-II and Waikato-VIII	Yes	[57]	93 million and 183 million packets
5	Long-Term St database	Yes	[58]	86 records
6	CSP	No	[61] [62]	Nil
7	Al-Nahrain University	No	[65]	Nil
8	Sporty Pal	No	[69]	36 activities

In **Table 10** different databases are given that are utilized by different researchers in different studies. In the given table UCI repository is mostly used. It deals with 19908 instances and helps the researchers in the correct identification of the disease.

Table 11: Tools/Simulators

Ref	Tool/ Simulator	Open Source	Language	Purpose		
				Modelling	Simulation	Analysis
[55]	Weka	✓	Java	×	✓	✓
[56]	Icancloud	✓	C++	×	✓	✓
[57][70]	iFogSim	✓	Java	×	✓	✓
[60]	Eclipse	✓	C, Java	✓	×	×
[69]	CloudSim	✓	Java	×	✓	✓

In **Table 11** different tools and simulators are given. They are differentiated through modelling, simulation, and analysis. Most of the tools/simulators support simulation and analysis. They do not support ***modeling***, which limits the user's ability to design & analyze the system at the early stages of SDLC.

2.4 Research Gap:

This section deals with the research gap and proposed solutions in fog-based ehealth systems. Fog Computing is spread widely, and it has a variety of devices. These devices are coupled to supply transmission, computation, and storage services. Many studies have been conducted about the techniques, architecture, and tools in the domain of fog computing and ehealth. However, to the best of our knowledge, no work has been done that analyses and summarizes eHealth methodologies and technologies in the context of fog computing.

The Intervention in ehealth has been applied on a various day-to-day basis increasing disease data such as diabetes, heart disease, lungs, neurodegenerative, cancers, and others. Accordingly, the amount of patients' information is obtaining huge storage space. However, advancements in the field of Fog computing are playing an effective role in storing the excessive amount of generated data on daily basis. Though the eHealth system is mission-critical that's why it is supposed to be perfect there is no chance of a mistake in the system. Furthermore, the healthcare systems are analyzed at the time of implementation to whether they are working perfectly or not. The state-of-the-art methodologies for fog based eHealth systems, on the other hand, are hard to come by in the literature. As a result, an easy-to-use, open-source framework for the creation and study of fog based eHealth systems is urgently needed. Following a thorough review of existing research on fog based eHealth systems, we concluded that the present solutions had the following issues:

- Healthcare systems are tested at the time of implementation.
- There is a lack of a high-level solution to lessen the difficulties of developing a healthcare system.
- There is a lack of open-source tools for developing healthcare systems.
- Lack of native java-based complete solution for fog based ehealth system.
- lacks the amount of work done in the fields comprising of model-driven approach, ehealth, and fog computing.

We suggested a solution based on a comprehensive, open-source java code to fill the gaps revealed following a thorough literature assessment of fog based ehealth development methodologies. We've integrated a model-driven approach into a fog based ehealth system, which reduces the complexity of the development process, provides a higher-level abstraction with fully automated end-to-end implementation, controlled development during early stages, and an open-source generic and real-time solution that can provide highly usable java code for ehealth systems. The advantages of the proposed method are as follows:

- A high-level solution that makes the development process easier.
- Java code that is ready to use.
- The developed fog-based ehealth models are reusable.
- Development time is cut in half.

Chapter 3

Proposed Methodology

Chapter 3: Proposed Methodology

This chapter contains a detailed description of the concepts involved in the proposed methodology. The recommended solution is derived using model-driven engineering that includes UML meta-model extension to impose the meta-level notations in modeling the necessities of Fog based eHealth Systems. The proposed UML Profile adds up to existing meta-models to draw new domain-specific concepts. Then M2T Transformation (Model-to-text) is performed on the domain model to generate resource utilization in *java* language for simulation of Fog based eHealth Systems. All the generated files are required to deploy in an actual environment for being fully operational i.e. we have simulated the generated artifacts in the iFogSim simulation tool. The required tools and techniques for this purpose can be seen in **Section 3.1** and **Figure 10**. The proposed framework i.e. MFeHTE (Model-Driven Fog-based eHealth System Transformation Engine), enables the domain experts to assess resource utilization for simulating the design and verification properties at the early development phase. The overview of the proposed framework is graphically presented in **Figure 9**. According to **O**bject **M**anagement **G**roup (OMG) [74], there are two UML-extension mechanisms i.e. meta-model extension and profiling, and both concepts are termed as *Profile*. We have extended the UML meta-model [75] to develop a profile that models Fog based eHealth system components. The description of concepts involved in fog based ehealth systems is presented with the help of UML-profile in **Section 3.2.**

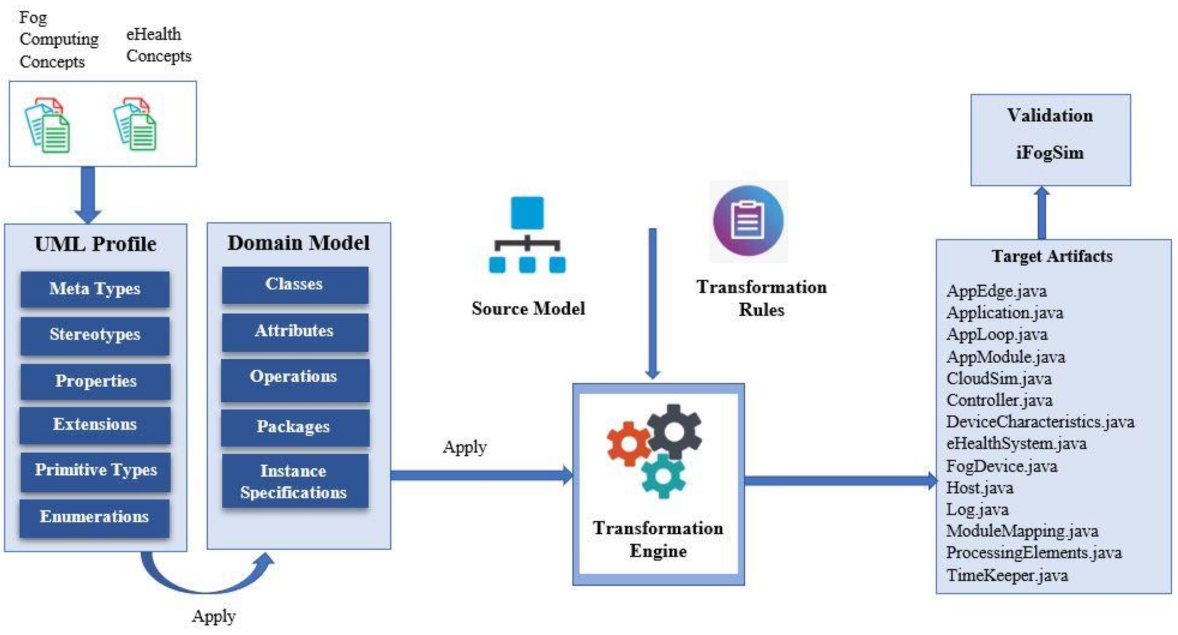


Figure 9: Model-Driven Framework for Fog based EHealth System

3.1 Architecture for Tools and Technique

Tool support is an essential aspect of increasing software development productivity. To support the framework, there is a tool support architecture. To speed up the software development process, good tool assistance is required, which is also very beneficial in terms of results. The tools and techniques utilized in the development of MFeHTE are shown in **Figure 10**. It is based on four layers i.e. Meta-modeling IDE, Modeling Editor, Model to Text Transformation (M2T) plugin, and Simulator. For meta-modeling, Eclipse-2018-12 is used, which is an open-source tool [76]. It acts as a framework with different variety of plugins for different intentions. Papyrus - Eclipse plugin for modeling is utilized to design the domain model and UML Profile [77]. Meanwhile, for M2T transformation, the Acceleo tool and Acceleo in Eclipse language are used [78]. The fourth layer contains a tool needed to debug and deploy the transformed or generated code. iFogSim [79] is a simulator designed for Fog and IoT environments to accomplish IoT amenities in a Fog Network. iFogSim is selected for validating the generated code from the transformation engine "MFeHTE".

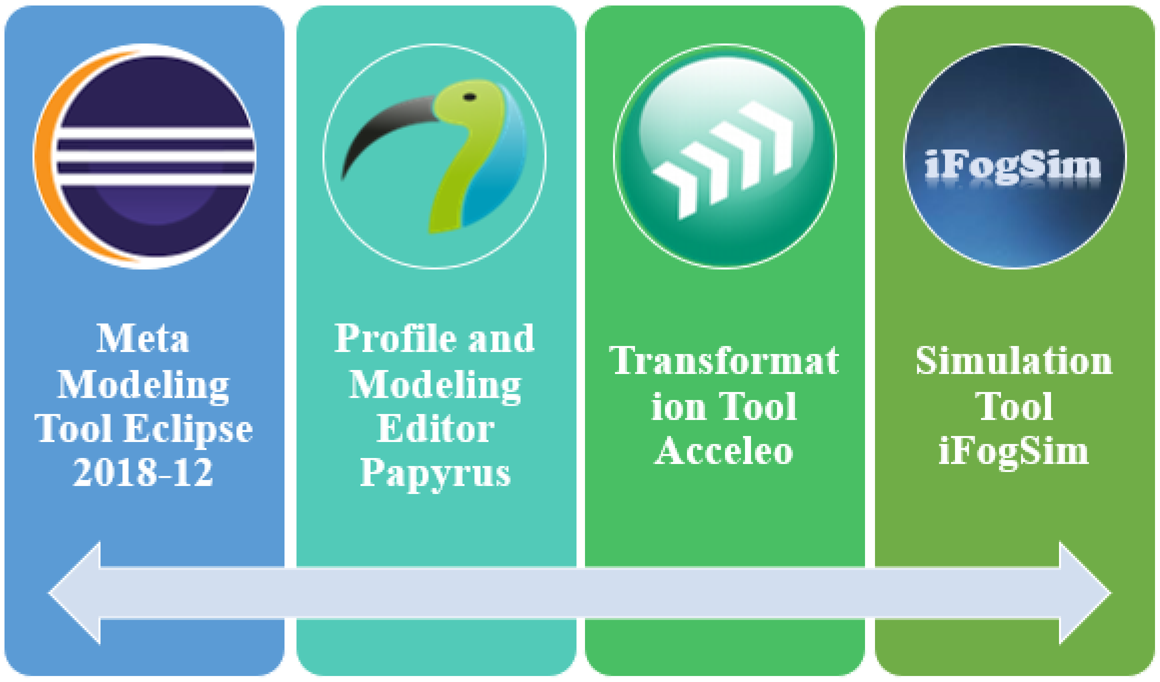

iFogSim
Meta Modeling Tool Eclipse 2018-12
Profile and Modeling Editor Papyrus
Transformat ion Tool Acceleo
Simulation Tool iFogSim

Figure 10: Tools and Techniques Architecture

3.2 UML Profile for Fog based eHealth System

The UML Profile for Fog-based eHealth System (UMLPFeH) is divided into four sub-profiles. These profiles are further classified into Fog and ehealth Concepts. Purple-colored stereotypes show the ehealth based concepts and green-colored stereotypes show the fog-based concepts.

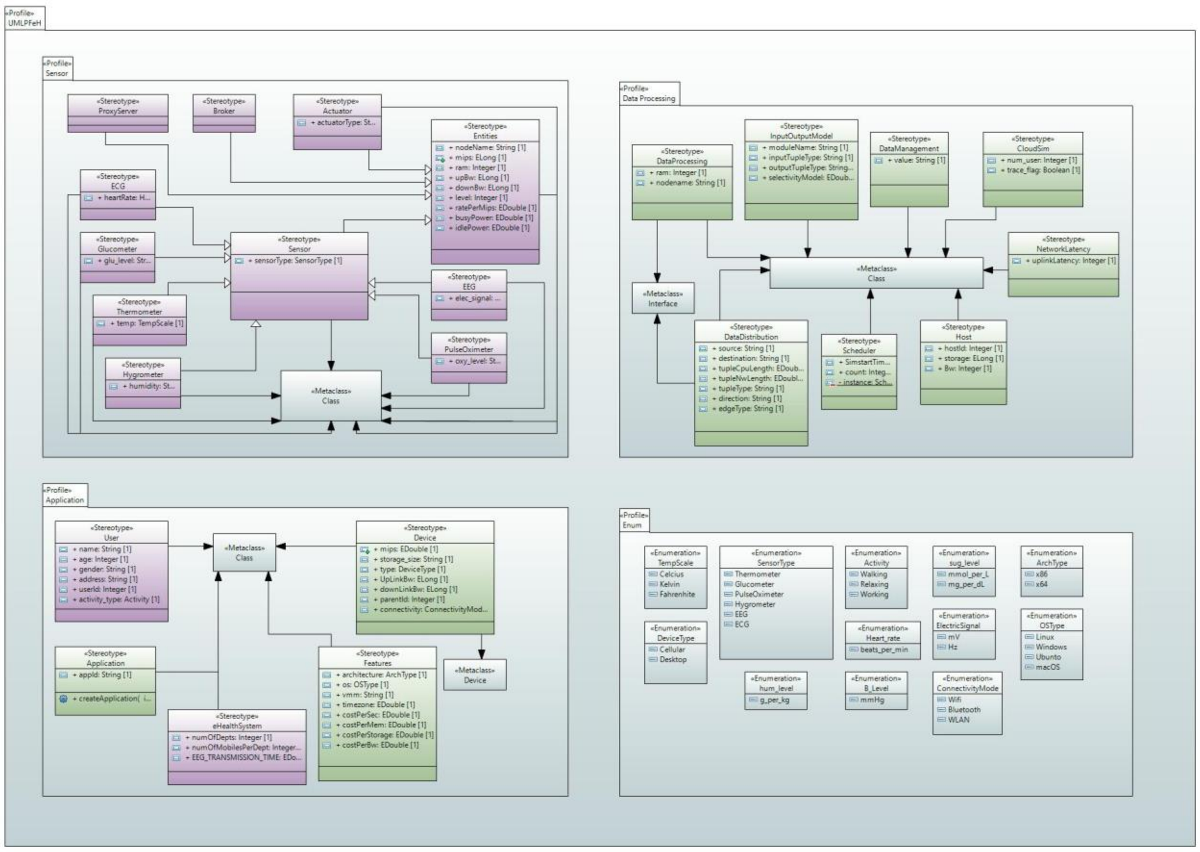


Figure 11: UML Profile for Fog based eHealth System (UMLPFeH)

In the realm of software engineering, the Unified Modeling Language (UML) is a general-purpose modeling language that provides a common approach to depicting the design of complex systems. The profiling technique can be used to enhance 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 fitter for a specific area (e.g. health, finance, wireless technology, etc.). It does not allow you to change existing metamodels; instead, it allows you to customize them.

The keyword <<profile>> is inserted before the name of the package in the profile notation. Stereotypes, constraints, and tagged values make up the majority of a UML Profile. The stereotype is the basic extension component of the profile. The stereotype is a profile class that outlines how a metaclass can be extended as part of the profile. The stereotype is written in class notation, with the keyword <<stereotype>> written before the name of the stereotype. Because stereotype is a class, it can have characteristics known as tag definitions. The values of the properties are referred to as tagged values when a stereotype is applied to a model element. We've created a UML profile that supports eHealth systems built on Fog.

By introducing domain-specific notions of Fog-based eHealth System, our proposed framework, UMLPFeH, provides a general extension mechanism that allows to adapt or alter metamodel (i.e. UML) utilizing stereotypes. The stereotype extends meta-classes of Class, Instance Specification, Property, and Slot, and is the fundamental extension construct of UMLPFeH. UMLPFeH is depicted in detail in **Figure 11**. UMLPFeH was created using the Papyrus modeling editor. It is a Java-based eclipse plug-in. Multiple stereotypes are included in the proposed profile to aid in the introduction of Fog Computing and eHealth-related domain ideas into UML modeling. These ideas are further broken down into sub-categories.

- eHealth based Sensor Sub profile
- Fog based Data Processing Sub profile
- Fog and eHealth based Application Sub profile
- Enumerations

These sub-profiles are based on the layered architecture of IoT.

3.2.1 eHealth based Sensor Sub Profile Concepts

This section represents the components of ehealth. This section of the UML profile contains 11 stereotypes i.e. Sensor, ECG, EEG, Glucometer, Thermometer, Hygrometer, Pulse Oximeter, Entities, Actuator Broker and Proxy Server. These stereotypes are used to provide the functionality of sensing, identification, actuation, and communication, and also scheduling the execution of tasks on appropriate fog nodes. With minimal human interaction, it includes technologies for sensing (gathering data from the environment and sending it to databases, data warehouses, or the Cloud), identification (identifying objects based on a unique identifier assigned to them), actuation (taking mechanical action based on sensed data), and communication (establishing connectivity among heterogeneous smart devices). It is defined by its ability to capture information from the real world and represent it digitally [80].

Description of Stereotypes

Sensor Stereotype

Description: sensing the environment and emitting observed values to upper layers via gateways for further processing and filtering. The *Sensor* stereotype represents the "sensing module" of the Sensor in ehealth system. It is being generalized from the super-class of *Entities* and utilizes the same properties as of superclass. It strictly depends on the type of system-to-be, different sensing mechanisms are associated with different sensors and consequently affect the cost, architecture, and outcomes of the system. A thorough literature review reveals different sensors in practice for the ehealth system that will be discussed in the Enumeration sub-profile.

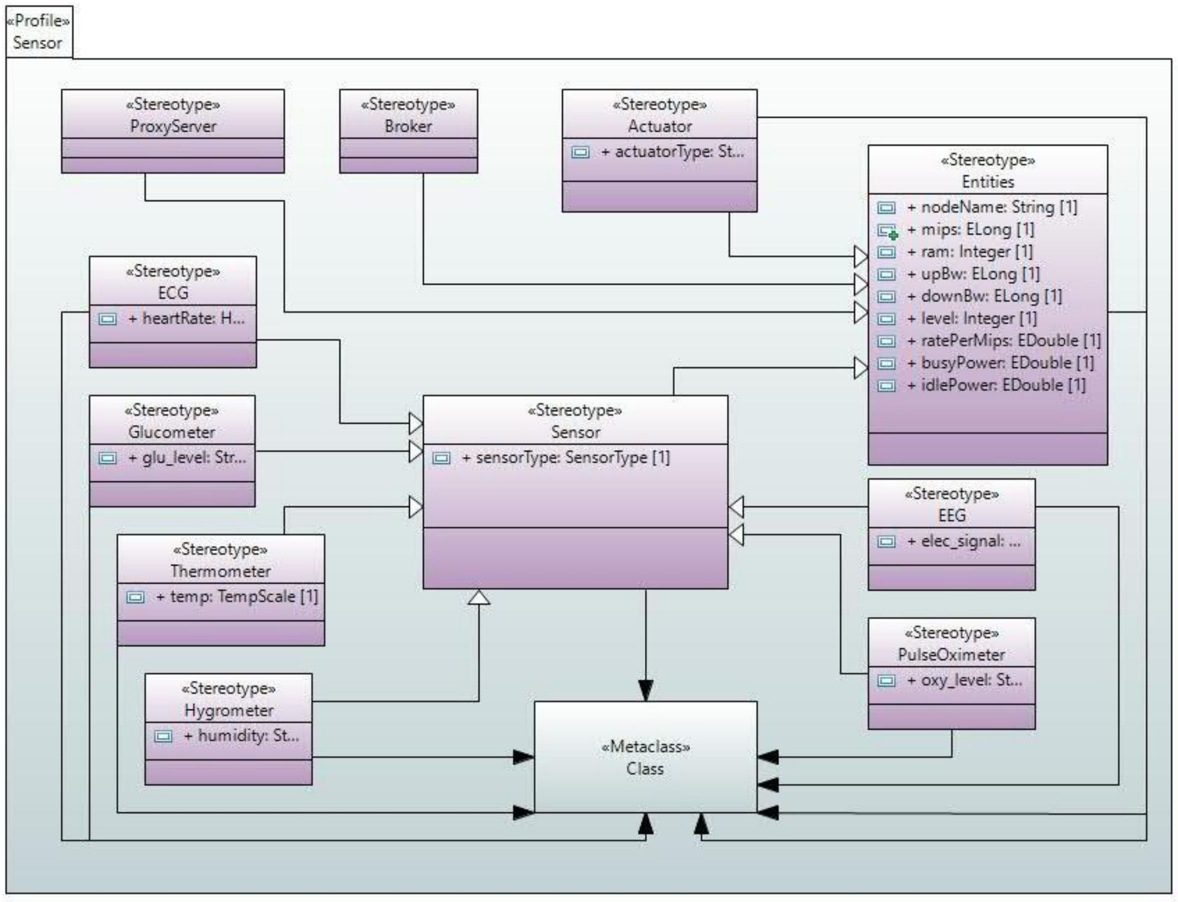


Figure 12: Sensor Sub-profile

Meta-Class: Class

Tagged Values: Senor Meta class has the following tag value:

sensorType: SensorType[1] which is known by fog device.

Actuator Stereotype

Description: designed to respond to changes in environments that are captured by sensors. The *Actuator* stereotype represents the "Display module" of the ehealth system.

An actuator is a component of a machine that is responsible for moving and controlling; that may be a result of some mechanism or process.

Meta-Class: Class

Tagged Values: Actuator Meta class has the following tag value:

actuatorType: String[1] which is known as a fog device.

Entities Stereotype

Description: This represents the data distribution between the "Nodes" concept of the ehealth system. The entities need to be initiated and their configurations are initialized. Sensors, gateways, and cloud virtual machines, as well as the linkages that specify how these entities are linked, fall into this category. This can be done via GUI or by java programming using iFogSim supporting classes. In our case, it would be based on a programmatic approach. However, there is only the need to define these in entities, iFogSim libraries will take care of the execution. It generalizes the stereotypes of devices/nodes like *display*, *Proxy Server*, *Broker*, *Actuator*, and *Sensor* and shares their common attributes of it.

Meta-Class: Class

Tagged Values:

nodeName: String[1] name of the device used in the simulation.

mips: ELong[1] million instructions per second

ram: Integer[1] main memory of the fog node.

upBw: ELong[1] uplink bandwidth.

downDw: ELong[1] downlink bandwidth.

level: Integer[1] hierarchy level of the device.

ratePerMips: EDouble[1] cost rate Per Mips used.

busyPower: EDouble[1] the amount of power consumed when the fog node is in busy state.

idlePower: EDouble[1] the amount of power consumed when the fog node is in an idle state.

Proxy Server Stereotype

Description: A proxy server is a computer system or router that works as a relay between the client and the server, preventing an attacker from accessing a private network and enabling the installation of a firewall. "A proxy server is a server application or appliance that acts as an intermediary for requests from clients seeking resources from servers that provide those resources."

Meta-Class: Class

Broker Stereotype

Description: This stereotype is based on the “Class” meta-class in the UML model. By processing data streams, the fog reduces network traffic and lowers the latency of time-sensitive applications (tuples). The fog broker is the most important part of the architecture, as it is responsible for scheduling task execution on appropriate fog nodes utilizing iFogSim's numerous scheduling policies.

Meta-Class: Class

ECG Stereotype

Description: This stereotype is based on an electrocardiogram (ECG or EKG) that **records the electrical signal from your heart** to check for different heart conditions.

Meta-Class: Class

Tagged Values: ECG Meta class has the following tag value:

heartRate: HeartRate[1] which is known as the fog device

EEG Stereotype

Description: This stereotype is based on an electroencephalogram (EEG) that detects abnormalities in your brain waves or the electrical activity of your brain.

Meta-Class: Class

Tagged Values: EEG Meta class has the following tag value:

elec_signal: ElectricSignal[1] which is known as a fog device.

Glucometer Stereotype

Description: a medical device for determining the approximate concentration of glucose in the blood.

Meta-Class: Class

Tagged Values: The glucometer Meta class has the following tag value:

glu_level: sug_level[1] which is known by the fog device.

Pulse Oximeter Stereotype

Description: estimates the amount of oxygen in your blood.

Meta-Class: Class

Tagged Values: Pulse Oximeter Meta class has the following tag value:

oxy_level: String[1] which is known as a fog device.

Hygrometer Stereotype

Description: measure the amount of water vapor in the air, in soil, or in confined spaces.

Meta-Class: Class

Tagged Values: The hygrometer Meta class has the following tag value:

humidity: hum_level[1] which is known by the fog device.

Thermometer Stereotype

Description: measuring the temperature of body and room.

Meta-Class: Class

Tagged Values: Thermometer Meta class has the following tag value:

temp: TempScale[1] which is known as a fog device.

3.2.2 Data Processing Sub Profile Concepts

This section represents the components of Fog computing. This section of the UML profile contains 8 stereotypes i.e. Data Distribution, Scheduler, Host, Network Latency, CloudSim, Data Management, Input Output Model, and Module. Its objective is to convey data collected by perception nodes to the information processing unit (or high-level decision-making units) for analysis, data mining, and data aggregation using wired or wireless communication channels [80].

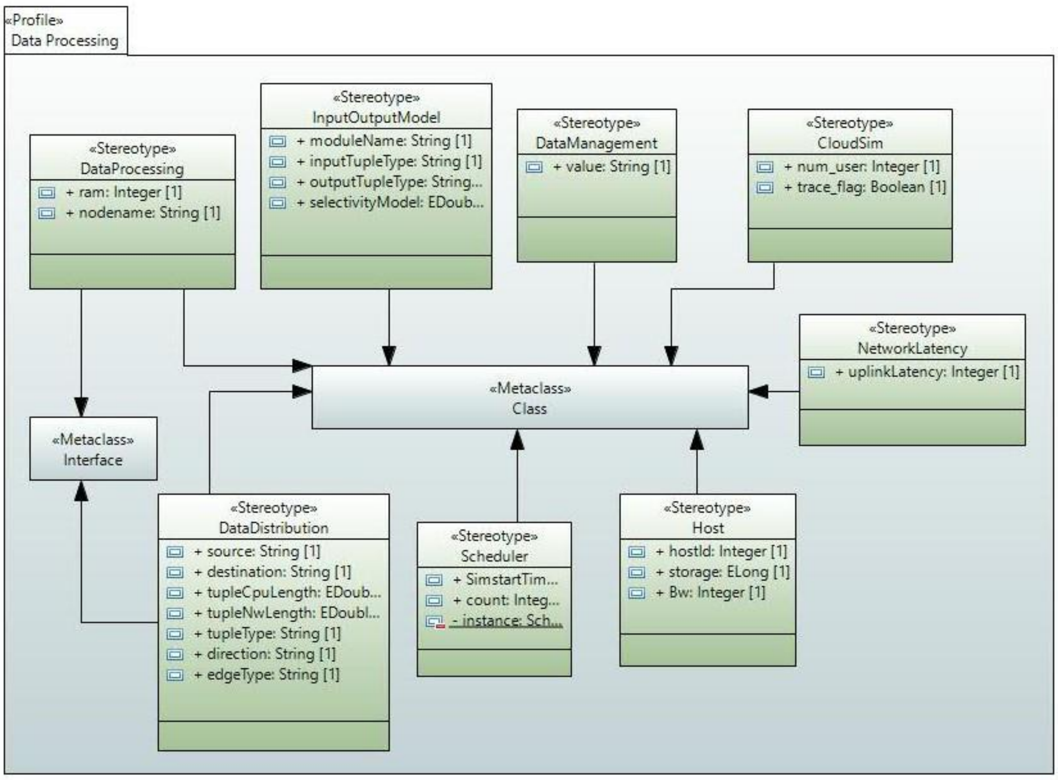


Figure 13: Data Processing Sub-profile

Description of Stereotypes

DataProcessing Stereotype

Description: This stereotype represents the *vertices* concept of the application network hierarchy (directed graph) and characterizes by name and the memory it occupied or consumes.

Meta-Class: Class, Interface

Tagged Values: Data Processing Meta class has the following tag value:

ram: Integer[1] main memory of the module.

nodename: String[1] name of the modules used in the simulation

DataDistribution Stereotype

Description: This stereotype adds a periodic edge to the application model following these attributes:

Meta-Class: Class, Interface

Tagged Values: Data Distribution Meta class has following tag value:

source: String[1] source node of the edge

destination: String[1] destination node of the edge

tupleCpuLength: EDouble[1] size of the info they carry for computation
tupleNwLength: EDouble[1] size of the info they carry for transmission
tupleType: String[1]

- **inputTupleType** (Type of tuples carried by the incoming edge)
- **outputTupleType** (Type of tuples carried by the output edge)

direction: String[1] upward, downward
edgeType: String[1] periodic, non-periodic

InputOutputModel Stereotype
Description: This stereotype represents the input-output relationship of the application modules for a given input tuple type. It is important to note that *Tuple* is " the fundamental unit of communication between entities in the Fog, and it is defined by its type as well as the source and destination application modules". Three different input-output models are provided by iFogSim to check whether an incoming tuple can generate an output tuple and are listed below as:
Fractional Selectivity: Generates an output tuple for an incoming input tuple with a fixed probability. In other words, the fixed probability of output tuple creation per incoming input tuple.
Bursty Selectivity: Generates an output tuple for every input tuple according to a bursty model. During the high burst period, all input tuples result in an output tuple. During a low burst period, no input tuples result in an output tuple.
Selectivity Model: It has specific methods to check whether the incoming tuple can generate an output tuple. (True/False)
Meta-Class: Class

DataManagement Stereotype
Description: respond to changes in environments that are captured by sensors.
Meta-Class: Class
Tagged Value: Data Management Meta class has the following tag value.
value: String[1]

CloudSim Stereotype
Description: iFogSim operates on the underlying architecture of CloudSim but utilizes its unique algorithms/policies for resource management, scheduling, and power

monitoring. Its purpose is to make network simulation possible in CloudSim. Furthermore, CloudSim's network models are deactivated.

Meta-Class: Class

Tagged Values: CloudSim Meta class has the following tag value:

num_user: Integer[1] number of users.

trace_flag: Boolean[1] trace events.

NetworkLatency Stereotype

Description: This stereotype indicates *uplink latency* for the hierarchical network design. The latency of the proxy server's connection to the cloud, the latency of the router's connection to the proxy server, and the latency of the router sensors are all routinely set in ms (milliseconds).

Meta-Class: Class

Tagged Values: Network Latency Meta class has the following tag value:

upLinkLatency: Integer[1] to measure the end-to-end latency.

Host Stereotype

Description: The *Host* stereotype performs tasks relating to virtual machine management (e.g., creation and destruction). The host has a defined memory and bandwidth provisioning approach, as well as a Pe (Processing Elements) allocation mechanism for virtual machines. A virtual machine can be hosted on a host that is connected to a data center.

Meta-Class: Class

Tagged Values: The Host meta class has the following tag value:

hostId: Integer[1] Id of the host.

storage: ELong[1] Data to be stored.

Bw: Integer[1] Bandwidth.

Scheduler Stereotype

Description: Manage Time.

Meta-Class: Class

Tagged Values: Scheduler Meta class has the following tag value:

SimstartTime: [1] Simulation start time.

count: Integer[1] Count the time.

instance: Scheduler[1] Instance of timekeeper.

3.2.3 Application Sub Profile Concepts

This section represents the components of Fog computing and ehealth. This section of the UML profile contains 5 stereotypes i.e. User, Application, eHealthSystem, Features, and Device. This sub profile’s goal is to manage and deploy applications on a global scale using data collected by the perception layer and processed by the data processing unit. It allows end-users across the network to obtain personalized services tailored to their specific requirements via a variety of mobile devices and terminal equipment [80].

Description of Stereotypes

User Stereotype

Description: sensing the environment and emitting observed values to upper layers via gateways for further processing and filtering.

Meta-Class: Class

Tagged Values: The user Meta class has the following tag value:

name: String[1] Name of the user.

age: Integer[1] Age of the user.

gender: String[1] Gender of the user.

address: String[1] Address of the user.

userId: Integer[1] Id of the user.

activity_Type: Activity[1] The type of activity in which the user is.

Application Stereotype

Description: This stereotype enables an application in the Distributed Dataflow Model. The data processing aspects of an application are represented as a collection of units. Data from one component may be utilized as input by another, creating data dependence. As a consequence, we may describe an application as a directed graph, with vertices representing modules and directed edges showing data flow between components. The architecture supports two models used for IoT applications.

1. Sense-Process-Actuate Model
2. Stream Processing Model

Meta-Class: Class

Tagged Values: Actuator Meta class has the following tag value:

actuatorType: String[1] which is known as a fog device.

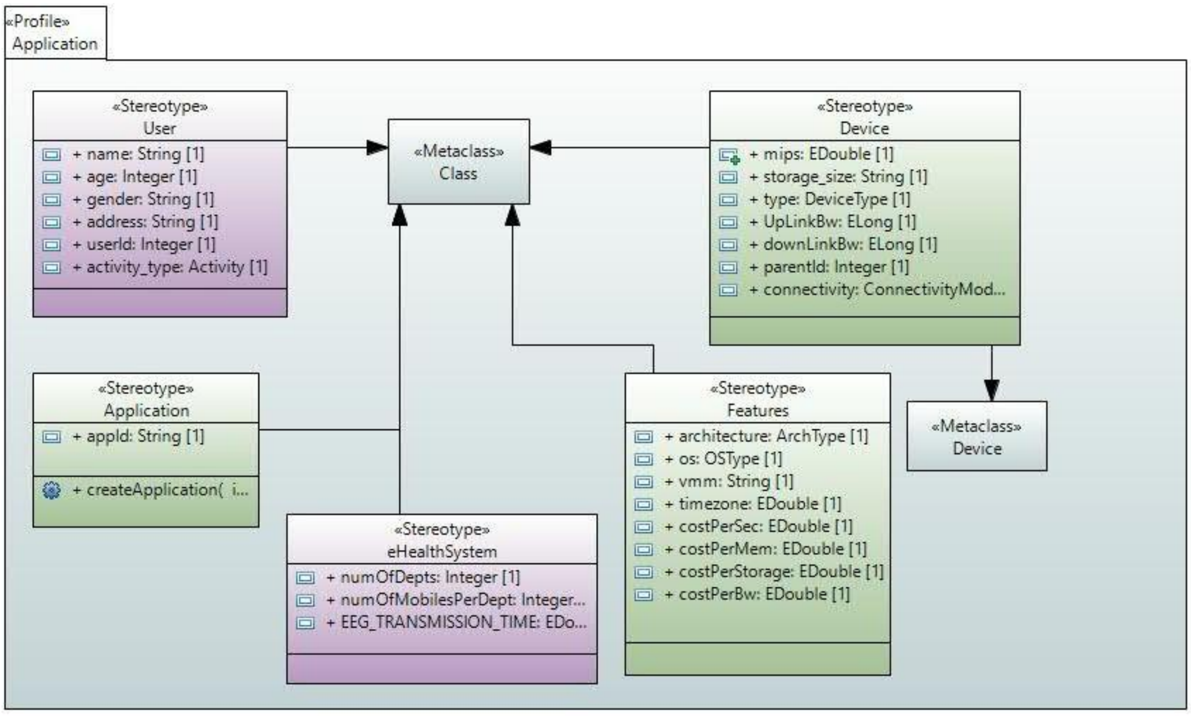


Figure 14: Application Sub-profile

eHealthSystem Stereotype

Description: The stereotype holds key significance in the proposed UML profile. It defines elementary components like *numofdepts*, *numofmobilesPerDept*, *and Transmission Time.* Variables like *numofdepts* and *numofmobilePerdept* are self explanatory, it is assumed that there would be four fog devices for every single department.

Meta-Class: Class

Tagged Values:

numofDepts: Integer[1] Number of departments.

numofMobilesPerDept: Integer[1] Number of Mobile per department.

Transmission_Time: EDouble[1] Time to transmit signals from the sensor.

Device Stereotype

Description:

Meta-Class: Class, Device

Tagged Values: The device Meta class has the following tag value:
mips: EDouble[1] which is known as a fog device.
storage_size: String[1] Size of the storage.
type: DeviceType[1] Type of the device.
UpLinkBw: ELong[1] uplink bandwidth.

downLinkBw: ELong[1] downlink bandwidth.
parentId: Integer[1] Id of the parent node.
connectivity: ConnectivityMode[1] Wifi or Bluetooth.

Features Stereotype

Description: This stereotype is also extended from the UML meta-class named Class and it is defined in the profile to specify the application module’s characteristics described as follows:

arch: ArchType[1] the architecture of a resource

os: OSType[1] the operating system used

vmm: String[1] the virtual machine monitor used.

time_zone: EDouble[1] time zone this resource located, should be of range [GMT-12 ... GMT+13].

cost: EDouble[1]the cost per sec of using processing in this resource.

costPerMem: EDouble[1] the cost of using memory in this resource.

costPerStorage: EDouble[1] the cost of using storage in this resource.

costPerBw: EDouble[1] the cost of using bw in this resource.

3.2.4 Enum Sub Profile Concepts

A thorough literature review is done before this project starts; got various notable practices and find them important to be listed here as enumerations, defined below and shown in **Figure 15.**

Description of Enumerations

TempScale Enumeration

Description: represents the types of temperature in which it is measured.

Enumeration Literal: TempScale has Celsius, Kelvin, and Fahrenheit enumeration literals.

DeviceType Enumeration

Description: represents the types of devices.

Enumeration Literal: DeviceType has Cellular, Desktop enumeration literals.

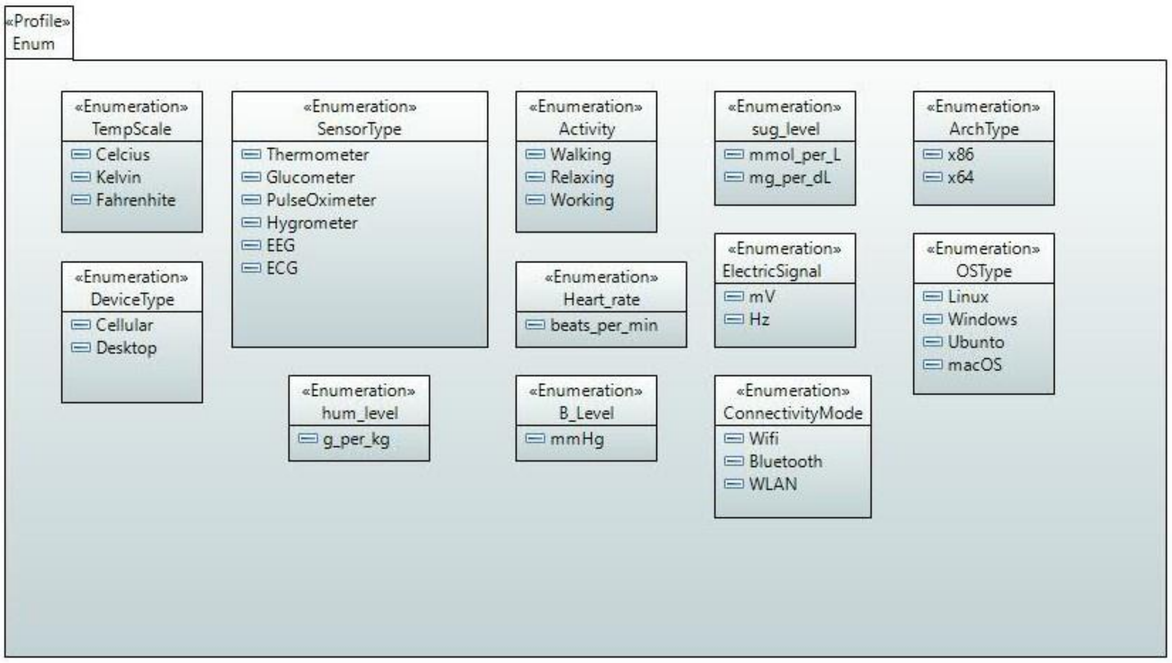


Figure 15: Enumerations

SensorType Enumeration

Description: represents the types of temperature in which it is measured.

Enumeration Literal: SensorType has Thermometer, Glucometer, PulseOximeter, Hygrometer, EEG, and ECG enumeration literals.

hum_level Enumeration

Description: represents the level of humidity in which it is measured.

Enumeration Literal: hum_level has g_per_kg enumeration literal.

Activity Enumeration

Description: represents the state of the Person in which it is.

Enumeration Literal: Activity has Walking, Relaxing, and Working on enumeration literals.

Heart_rate Enumeration

Description: represents the rate of heart in which it is measured.

Enumeration Literal: Heart_rate has beats_per_min enumeration literal.

B_level Enumeration

Description: represents the blood level in which it is measured.

Enumeration Literal: B_level has mmHg enumeration literal.

sug_level Enumeration

Description: represents the sugar level in which it is measured.

Enumeration Literal: sug_level has mmol_per_L, enumeration literals.

ElectricSignal Enumeration

Description: represents the electric signal in which it is measured.

Enumeration Literal: ElectricSignal has mv, Hz enumeration literals.

ConnectivityMode Enumeration

Description: represents the Mode of connection.

Enumeration Literal: ConnectivityMode has Wifi, Bluetooth, and WLAN enumeration literal.

ArchType Enumeration

Description: represents the type of Architecture that is used.

Enumeration Literal: ArchType has x86, x64 enumeration literals.

OSType Enumeration

Description: represents the type of Operating System.

Enumeration Literal: OSType has Linux, Windows, Ubunto, and macOS enumeration literals.

Chapter 4

Implementation

Chapter 4: Implementation

This chapter explains the development of our proposed transformation engine, Model-Driven Fog based eHealth System (MDFeH), which converts a UML Class Diagram (.uml) represented by UMLPFeH into java code for simulation. **Section 4.1** describes the architecture of our proposed transformation engine, while **Section 4.2** describes how to apply transformation rules to an input model to convert it to java code.

4.1 Transformation Engine Architecture

Figure 16 depicts the design of our transformation engine. MDFeH accepts input models and uses transformation rules to convert them to java code. It is written in the Java and Acceleo programming languages (for writing transformation rules). Model to Text Language (MTL) is the foundation of MDFeH. The MDFeH transformation engine is made up of two primary parts: the Tool User Interface and the Code Generator. Below is a detailed description of these components.

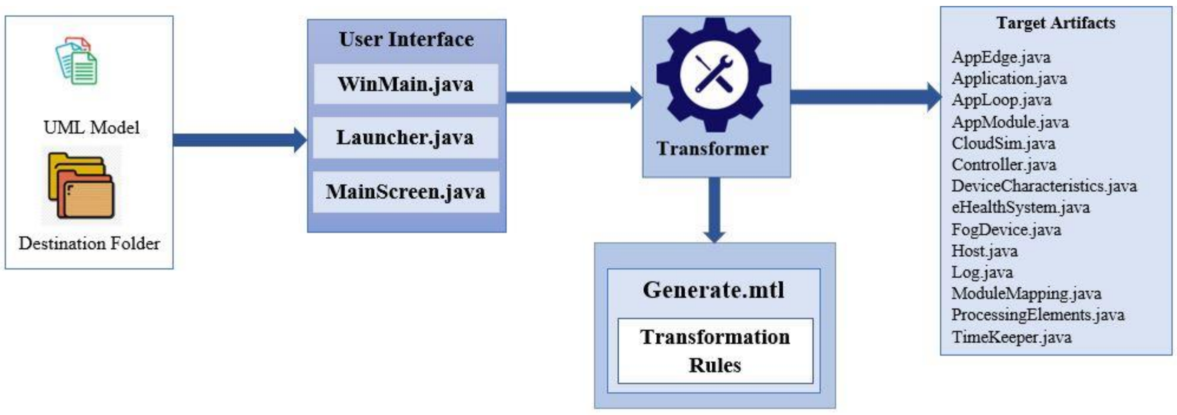


Figure 16: Architecture of Model-Driven Fog based eHealth Transformation Engine

4.1.1 Tool User Interface:

Figure 17 depicts the main interface of our tool MDFeH. It has two key features: a System Modeler and a Transformation Engine.

- *System Modeler* allows users to utilize the Papyrus tool to model java code.
- *Transformation Engine* option allows the user to supply an input model and have it transformed into java code using the Transformation Engine interface (**see Figure**

18). The options for this interface are Input Model, Destination Folder, Generate, Status, Reset, and Open Folder.

- *Input Model* allows the user to explore and select (.uml) files on his system. The Destination Folder option allows the user to specify the location of the created output files.
- *Generate* button generates java code files automatically based on the provided model.
- *Status* displays the tool's current state, such as whether files were successfully generated or if an error occurred.
- *Reset* users can utilize the Reset button to enter a new input model and destination folder.
- *Open folder* takes you to the location where the output files were created.

The Launcher, MainScreen, TextRefiner, and WinMain java classes are used to create MDFeH's user interface. 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. Text Refiner performs string processing in a format that can be used later.

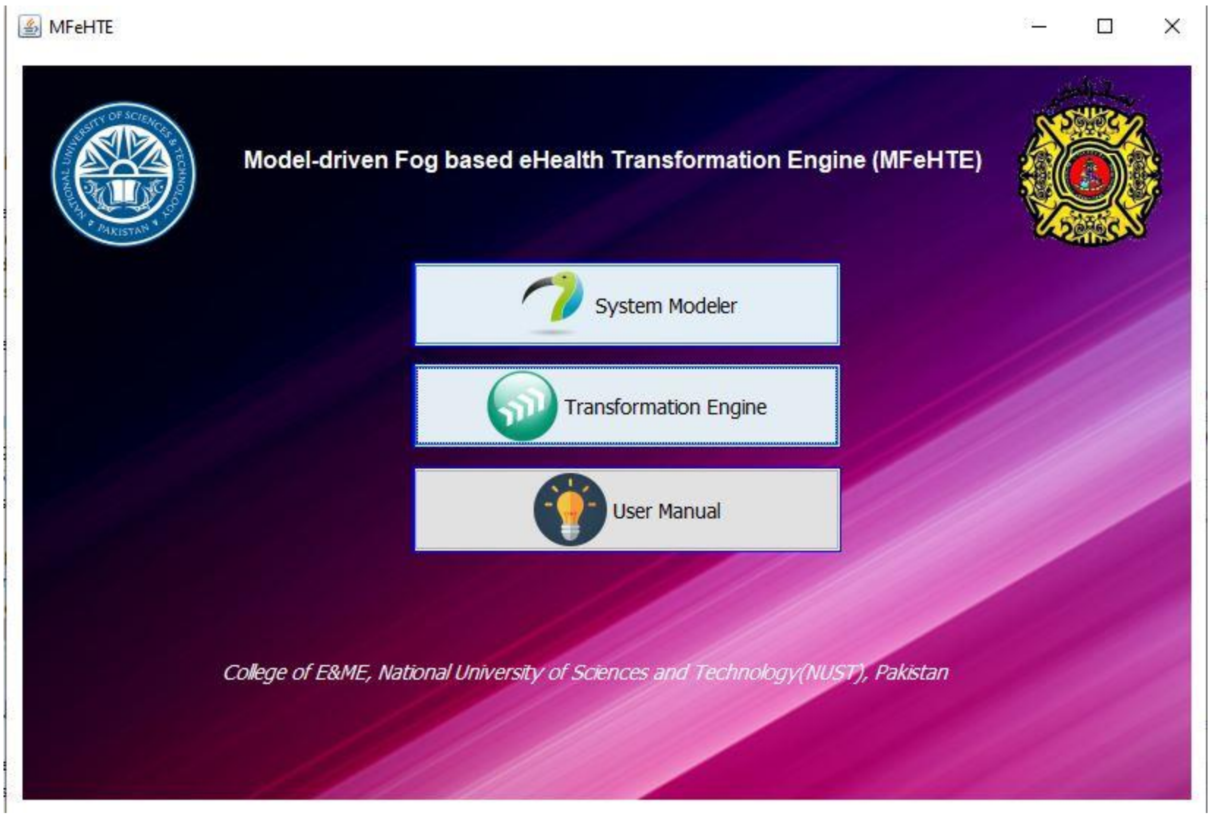

MFeHTE
Model-driven Fog based eHealth Transformation Engine (MFeHTE)
System Modeler
Transformation Engine
User Manual
College of E&ME, National University of Sciences and Technology(NUST), Pakistan

Figure 17: Main Screen of MFeHTE

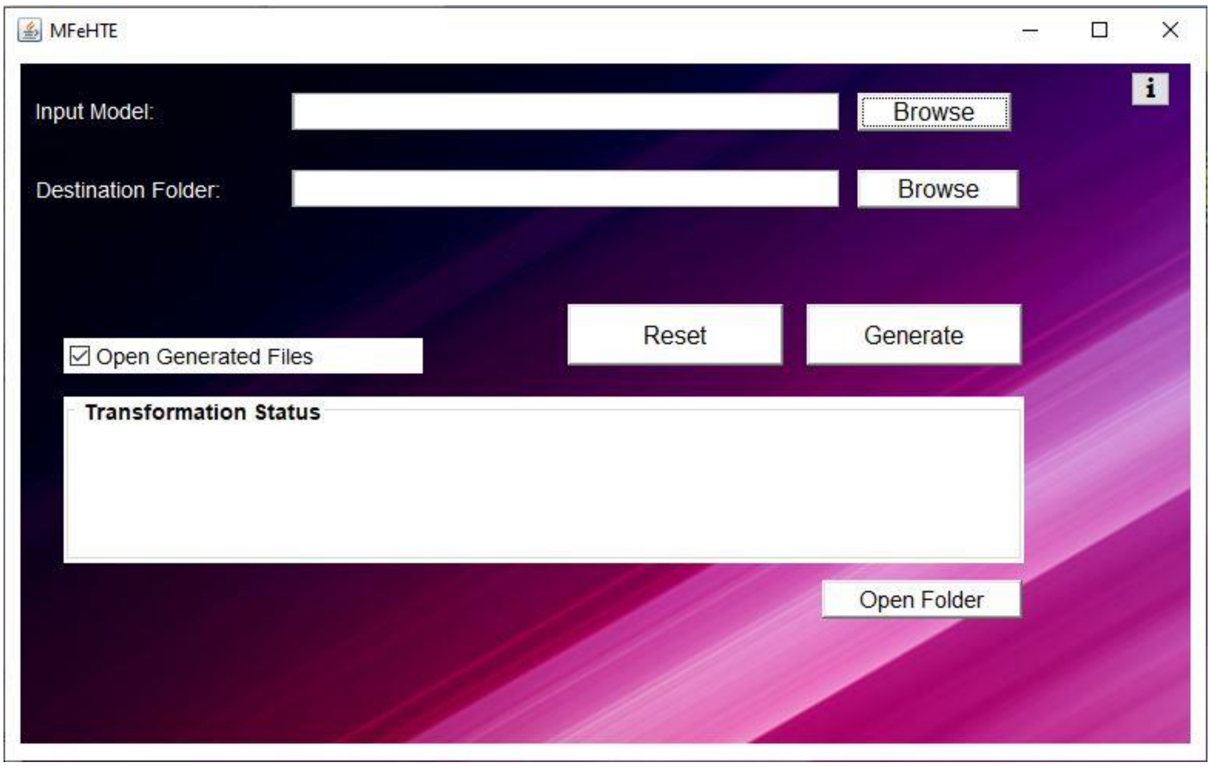


Figure 18: Input Interface of MFeHTE

4.1.2 Code Generator:

The user-specified input models are loaded into a code generator, which uses transformation rules to turn them into deployable java code. The code generator is made up of two primary parts.

1. Generate (Generate.java)
2. Template (generate.mtl)

The Template file is the main file in this module, and it provides references to sub-template files for implementing transformation rules. 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 output artifacts are based on java code that may be deployed.

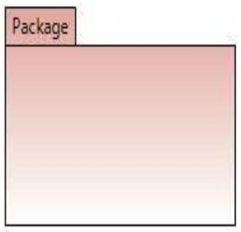
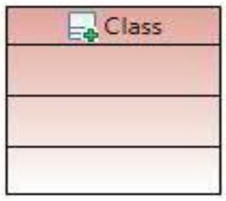
4.2. Transformation Rules

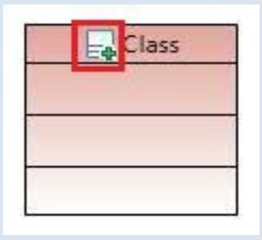
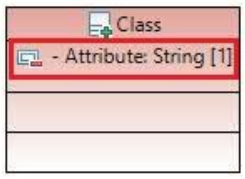
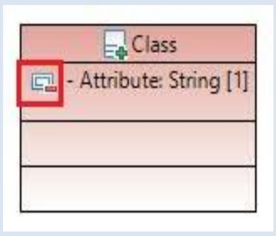
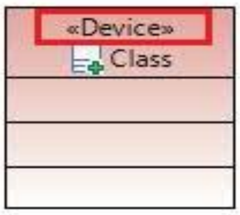
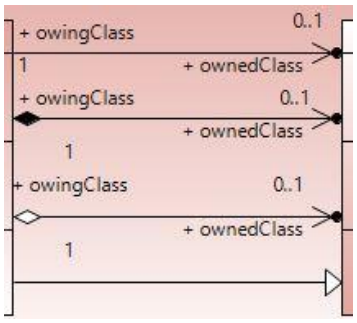
This section contains a full description of the transformation rules required for the Fog-based eHealth System's transformation process to yield desirable outputs. Model transformation is the process of creating output artifacts (models, code, text, documentation, and so on) from an input source model using transformation rules. The set of formal definitions that explain how one or more constructs in the source model language map to one or more constructs in the target model language is known as transformation rules [81]. The fundamental goal when creating transformation rules is to minimize total effort and information loss during the transformation process. This segment, as illustrated in **Figure 16**, describes the mapping rules that are used to convert UML models into required artifacts. A set of established rules that changes an input model into another needed model or textual artifacts is referred to as model transformation. Transformation rules are the names given to these predetermined rules. The basic elements for transforming UML models into appropriate java code are transformation rules. These rules ensure that the model remains consistent and that information is not lost.

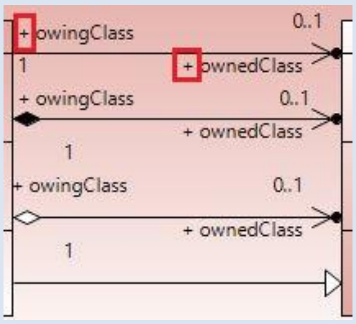
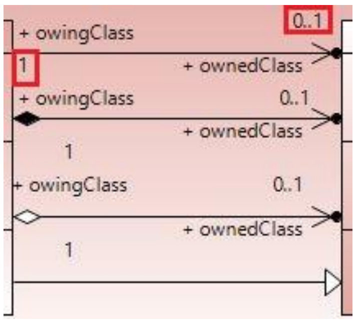
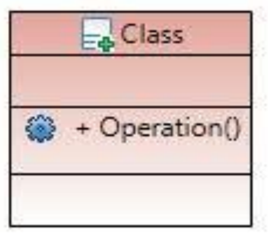
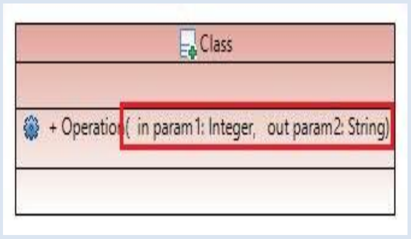
4.2.1. Transformation Rules for Code Generation:

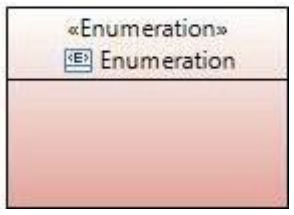
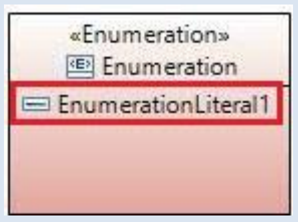
Our Code Generation transformation engine handles the process of generating code or text artifacts via Model to Text (M2T) transformation. The necessary transformation rules for M2T transformation are presented and discussed in **Table 12**.

Table 12: Transformation rules for standard modeling elements

Model Artifacts		**Code Artifacts**	**Mapping**
	Package		
Package Name		Folder	Package — Name → Folder Name
	Model Class		
Class Name			Model Class — Name → Class Name

Visibility		Class	Model Class — Visibility → Access Specifiers Class
Attribute			Model Class — Attribute → Owned Attributes
Attribute Visibility			Model Class — Attribute Visibility → Owned Attributes Access Specifiers
Applied Stereotype			Model Class — Applied Stereotype — Property → Owned Attributes
Association			
Member Ends	 + owingClass 0..1 1 + ownedClass + owingClass 0..1 + ownedClass 1 + owingClass 0..1 + ownedClass 1	Class Instance	Association — Member Ends → Owning Class/ Owned Class

Member Ends Visibility	 + owingClass 0..1 1 + ownedClass + owingClass 0..1 + ownedClass 1 + owingClass 0..1 + ownedClass 1	Public/ Private/ Protected	Association — Member Ends Visibility → Owned Instance Access Specifier
Member Ends Multiplicity	 + owingClass 0..1 1 + ownedClass + owingClass 0..1 + ownedClass 1 + owingClass 0..1 + ownedClass 1	List<Type>Instance Name OR Single Instance	Association — Member Ends Multiplicity/ Cardinality → List<Type> or Single Instance
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 Behaviour			Opaque Behaviour — Description → Body of Method
Enumeration			

Enumeration Name		enum Enumeration	Enumeration — Name → enum Name
Owned Literal		Enumeration Literal	Enumeration — Owned Literal → Enum Values

For modeling fog-based ehealth systems, Model Artifacts are based on parts of the Unified Modelling Language (UML). Code Artifacts are Android code elements that are used to translate UML model elements into text. Mapping connects UML model elements to their code counterparts. A package is a UML element that puts together other packageable items and gives the model a hierarchical view. The name of a package is mapped to the name of a folder containing code files throughout the transformation process.

In UML, a Model Class specifies a system's static structure. The prefix keyword class is used to transfer the name of a Model Class to the name of a class in code. By enabling four sorts of accessibility options, such as public, private, protected, and package, visibility restricts the use of a named element. By prefixing it to the class keyword, the visibility of a Model Class is transferred to the access specifier of a coding class. In code, model class attributes, attribute visibility, and applied stereotypes property is mapped to class-owned attributes and attribute visibility. The Member Ends of the Owned Class are translated to the instances of the Owned Class as properties of an Owning Class in code artifact, providing a semantic relationship between classifiers in UML. The access specifiers and cardinality of the instances of the Owned Class are mapped to Member Ends Visibility and Multiplicity, respectively. The cardinality '1' corresponds to a single instance, while the cardinality '0..*' corresponds to a List instance. In UML, operations define a Model Class's behavioral features, and its name is transferred to a class's method name in code. The name of Owned Parameters is mapped to the name of class parameters, and the direction of Owned Parameters is mapped to the type of method parameters, such as in, out, in-out, and return.

Like a block of code, opaque behavior is dependent on the implementation of particular semantics in UML. The body of the method or the method declaration in code is mapped to the description of opaque behavior. In UML, an enumeration is a user-defined set of named elements. By prefixing the name with the enum keyword, the Name of Enumeration is mapped to the name of Enum in code. In code, Enumeration Literals are mapped to named Enum values.

Chapter 5

Validation

Chapter 5: Validation

With the use of two case studies, this part discusses the application and validity of our suggested framework. The flow of the validation procedure is depicted in **Figure 19**. It also serves as a summary of our findings. UMLPFeH is built on three sorts of ideas: fog computing, health, and hybrid ideas. Following profile modeling, our suggested transformation engine MFeHTE transforms the domain model (Class Diagram) into java code. Finally, the proposed framework is evaluated and verified using two case studies, namely the eHealth System and case studies, which demonstrate the framework's applicability and utility. The eHealth System case study is discussed and validated in **Section 5.1**, and the case study is discussed and validated in **Section 5.2**.

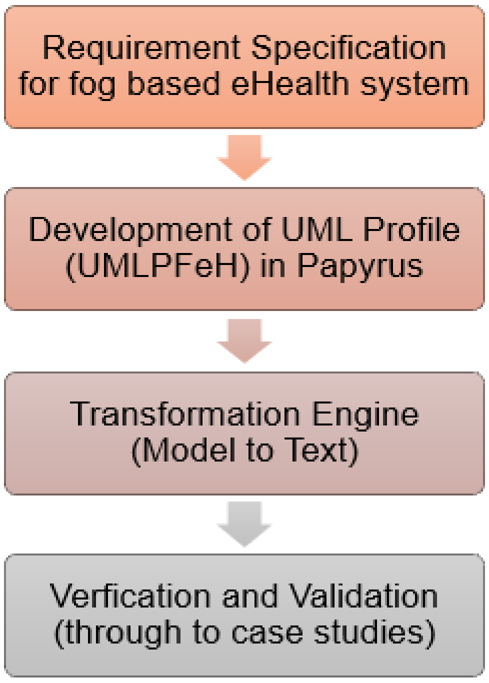

Requirement Specification for fog based eHealth system
Development of UML Profile (UMLPFeH) in Papyrus
Transformation Engine (Model to Text)
Verfication and Validation (through to case studies)

Figure 19: Flow towards validation

5.1 EEG Beam Tractor Case Study

This case study is explained and validated by dividing it into four sections**. Section 5.1.1** describes the Requirement Specification for the Home Automation System. **Section 5.1.2** demonstrates UML domain modeling with applied Profile UMLPMH of this case study in Eclipse plugin Papyrus. **Section 5.1.3** presents the transformation of the domain model into android code as result. Finally, verification of the case study is performed in **Section 5.1.4**

5.1.1 Requirement Specification

A large number of IoT-based medical devices, such as wearables, sensors, and smartphones, enable patient-driven healthcare systems to track patients' health in real-time. Any personal computer or cell phone can gather real-time health information and securely link it with the cloud eHealth platform. The patient's data is collected via a variety of wearables or sensors, such as a smartwatch or smart eyewear. All they needed was a fog gateway connection with an appropriate communication protocol. For proper connection in this scenario, a large number of personal area network (PAN) and wireless sensor network (WSN) protocols are employed, including WIFI for remote connections and Bluetooth low energy (BLE) for short-distance connections.

In the case study, there is a latency-critical application called EEG Beam Tractor, which involves a lot of brain-computer contacts. The patient must wear a Patient Concentration Device (PCD) remote EEG headset that is connected to the Fog Device to play the EEG Beam Tractor. On a mobile, the game runs as an Android application. The program handles the EEG signals detected by the EEG headset in real-time and determines the patient's brain status.

The game displays all of the patients on a ring encircling an objective article on the app's showcase. Following his level of concentration, the patient might apply an appealing power to the target. To win the game, a patient should try to pull the goal toward himself by practicing attention and denying other players the opportunity to snatch the goal.

5.1.2 Modelling

The application EEG Beam Tractor comprises three significant modules which perform preparing Patient, Concentration Calculator, and Coordinator.

1. Patient: The sensor communicates with the patient module, which receives raw EEG signals. It looks for any discrepancies in the received signal values and discards any that appear to be inconsistent. If the observed signal value is constant, it is sent to the Concentration Calculator module, which uses it to calculate the user's concentration level from the signal. It displays the concentration level after receiving it by transmitting the value to the DISPLAY actuator.

2. Concentration Calculator: The concentration calculator module is in charge of calculating the concentration level and determining the patient's brain state based on the sensed EEG signal values. This module updates the game status of the patient on the display by informing the Patient module about the measured concentration level.

3. Coordinator: The coordinator operates at a global level, coordinating the game amongst numerous patients who may be located in different parts of the world. The Coordinator distributes the current state of the game to all linked patients' Patient modules regularly.

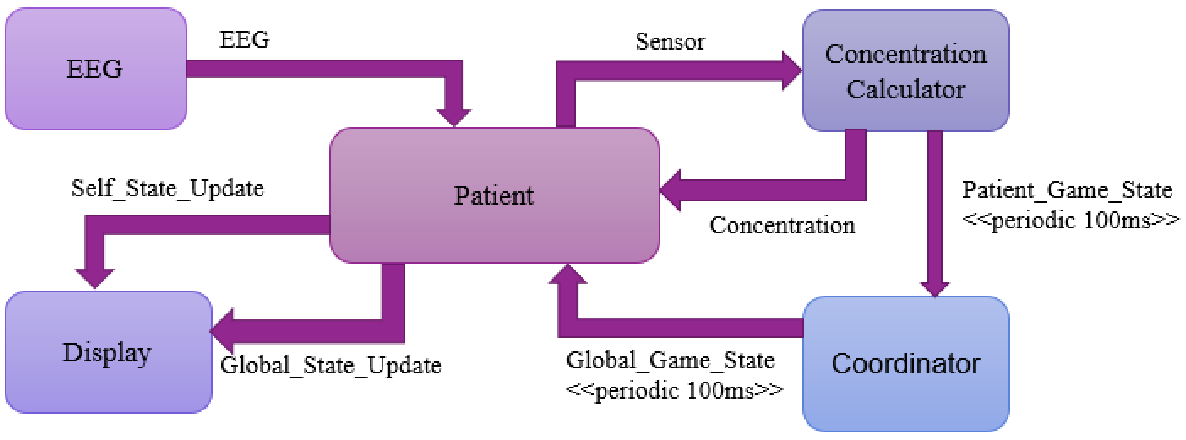


Figure 20: Application model of EEG Tractor Beam Case Study

The application model of the eHealth system is shown in **Figure 20**. The developed model constitutes numerous classes with the obligatory applied stereotypes, attributes, and methods for accurate execution as shown in **Figure 21**. This model contains thirty classes that are defined as below:

The major operations of the system are handled by the application class in the Distributed Dataflow Model or graphically in the form of a directed acyclic graph (DAG). Based on obtained data, it is used to define various modules, edges, and the tuples generated by these. The *AppLoop* class is used to define the process-control loops that the operator is interested in. The iFogSim calculates the control loops' end-to-end latency. An AppLoop instance is just a list of components starting at the loop's origin and ending at the loop's finish. The *AppModule* class represents an application module, which contains the processing parts of the iFogSim application model. *AppEdge* is a class that represents application edges, which connect modules and represent data dependencies. Conceptual components include Patient, Concentration Calculator, and Coordinator.

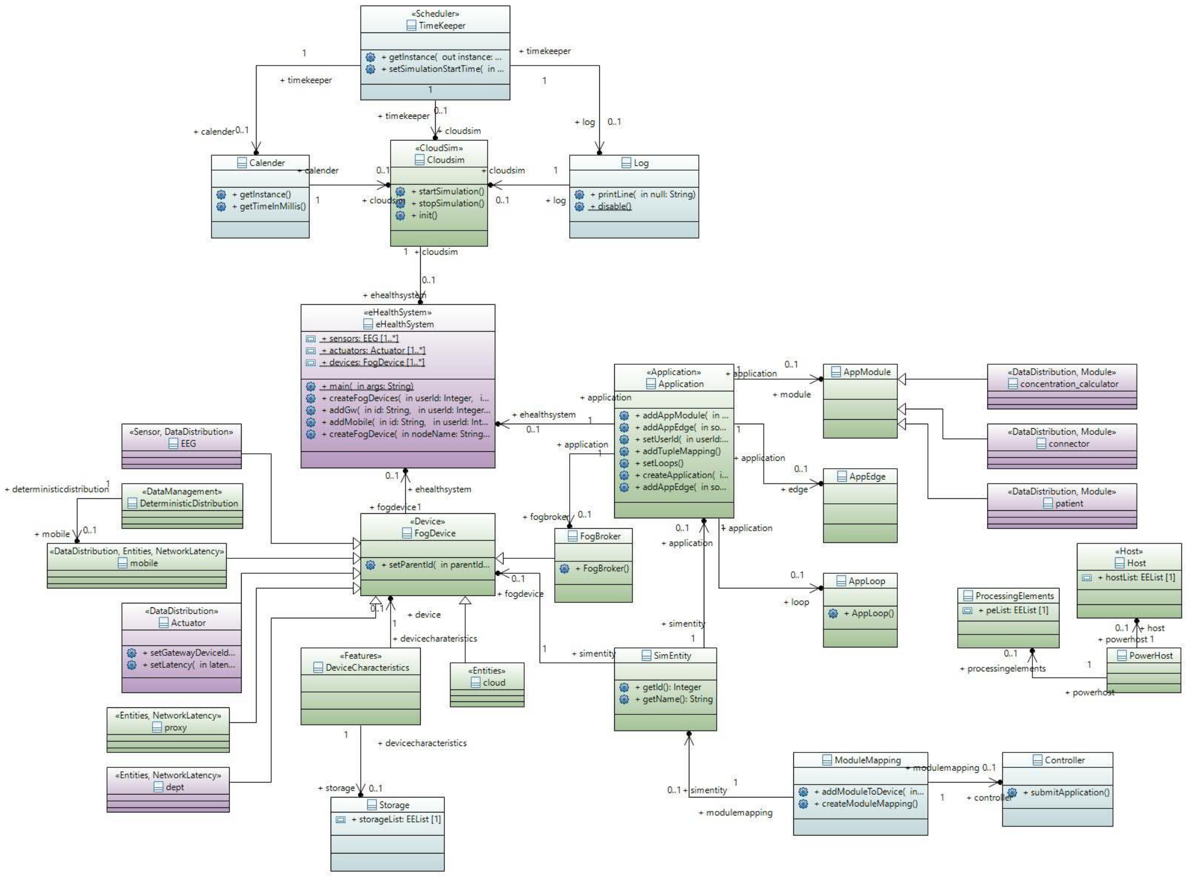


Figure 21: Design Model of Fog based eHealth System

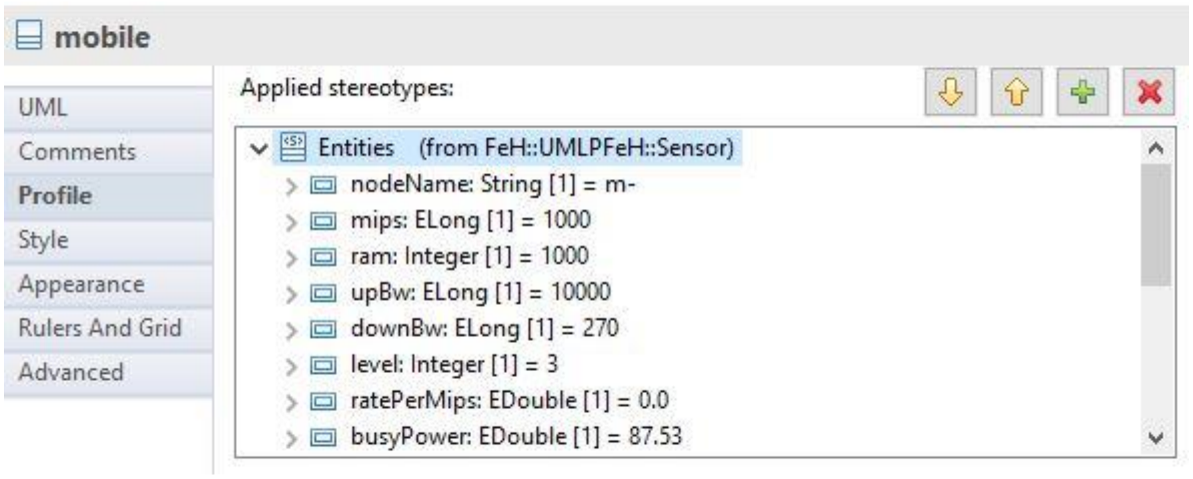
eHealth System is the main class in which certain functions initiate the number of cloud users, application devices, and the number of departments for EEG patients. Initializing controller and module mapping as well. *FogDevice* all the devices involved in building fog-based ehealth systems are termed fog devices. *Sensor, Actuator, Fog Broker, Proxy,* and *Cloud* are the same as discussed in **section 3.2.1.** *SimEntity* class represents a Simulation Entity that manages different actions and can communicate these actions to other entities by pre-defined methods.

Deterministic Distribution is the same as discussed in **section 3.2.2.** In a Data Cloud, *storage* represents the expected capabilities of a storage system. By setting the storage capability and the maximum transfer rate, the classes that implement this storage mechanism should imitate the properties of various storage systems. The transfer rate is defined as " the time required to perform a few typical actions on the storage, e.g. storing a file, retrieving a file, and erasing a file". *Features* same as discussed in **section 3.2.3.** *Controller* deals with output mechanisms either in console or in graphical results. *Module Mapping* deals with mapping from one node to the other instances to be launched.

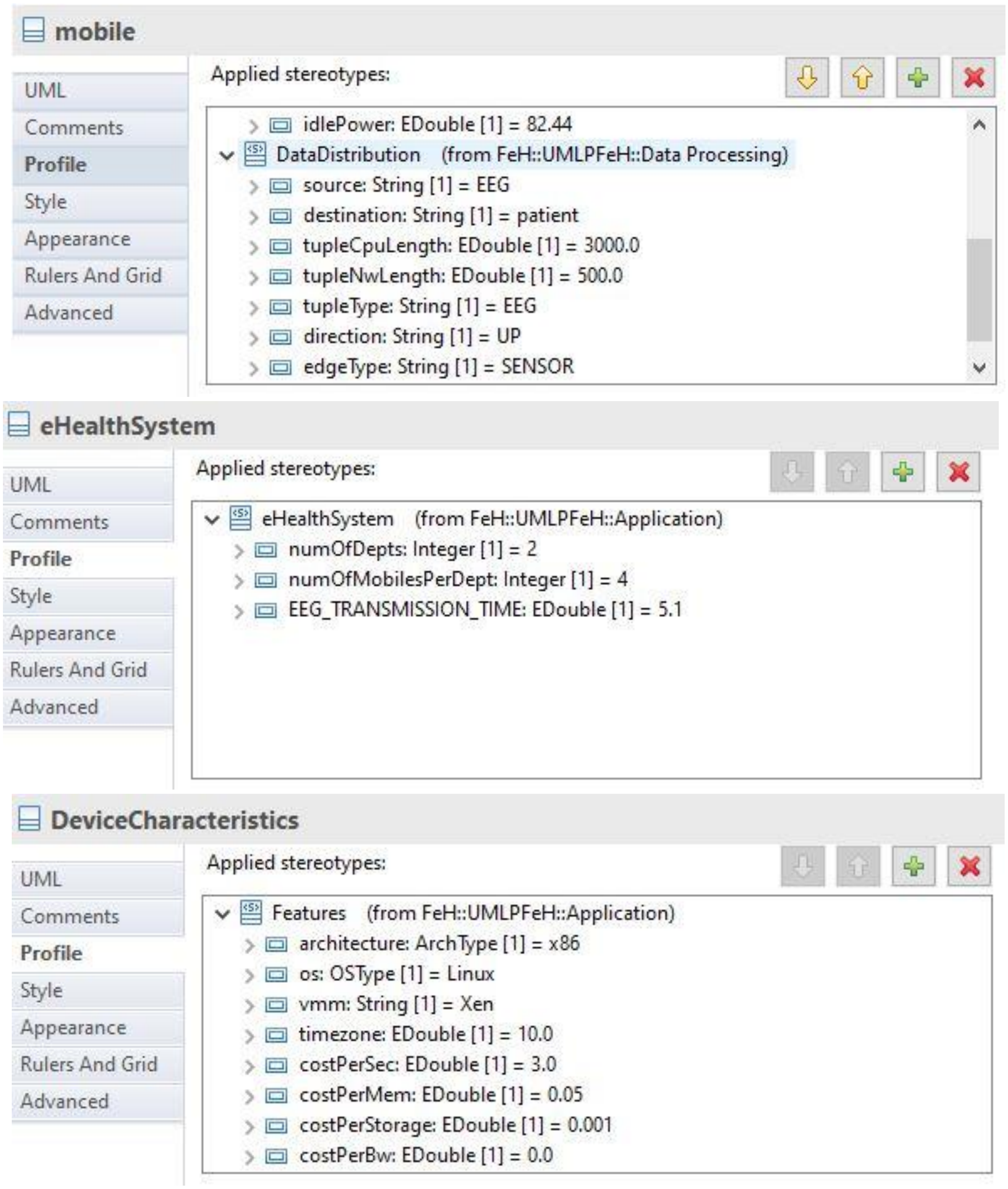
Processing Elements class represents the CPU component, which is expressed in MIPS (Millions of Instructions Per Second). The *host* is in charge of all actions related to virtual machine management (e.g., creation and destruction). A host has a defined memory and bandwidth provisioning approach, as well as a Pe allocation method for virtual machines. A data center is linked to a host.

It can host virtual computers.*PowerHost* class permits the simulation of power-aware hosts. *CloudSim* enhances CloudSim Core's capabilities to make network simulation easier in CloudSim. Furthermore, it disables all of CloudSim's network algorithms to provide a networking simulation. The Log class is used to run the simulation procedure's logging. It provides the ability to change the output stream. *Calendar* class is a java utility class that offers approaches for transforming among a particular instant of time and calendar fields such as Year, Month, Day_of_Month, Hour, and so on, and for operating the calendar fields, such as attaining the date of the next week. *TimeKeeper* is the iFogSim utility class that computes and provides time in milliseconds. Also, this class have functions like *setSimulationStartTime()* to set simulation start time.

It is important to note that the UML profile requires the assignment of tagged values as defined in UMLPFeH. It is important to define the entities' characteristics (i.e. nodes) represented by *nodeName, mips, ram, upbw, downbw, level, ratePerMips, busyPower, and IdlePower*. as shown in **Figure 22**. Moreover, Connection Edges are defined as the information that flows between nodes represented by source, destination, tupleCpuLength, tupleNwLength, tupleType, direction, and edgeType as demonstrated in **Figure 22**.



Entities (from FeH::UMLPFeH::Sensor)
nodeName: String [1] = m-
mips: ELong [1] = 1000
ram: Integer [1] = 1000
upBw: ELong [1] = 10000
downBw: ELong [1] = 270
level: Integer [1] = 3
ratePerMips: EDouble [1] = 0.0
busyPower: EDouble [1] = 87.53



idlePower: EDouble [1] = 82.44
DataDistribution (from FeH::UMLPFeH::Data Processing)
source: String [1] = EEG
destination: String [1] = patient
tupleCpuLength: EDouble [1] = 3000.0
tupleNwLength: EDouble [1] = 500.0
tupleType: String [1] = EEG
direction: String [1] = UP
edgeType: String [1] = SENSOR

eHealthSystem (from FeH::UMLPFeH::Application)
numOfDepts: Integer [1] = 2
numOfMobilesPerDept: Integer [1] = 4
EEG_TRANSMISSION_TIME: EDouble [1] = 5.1

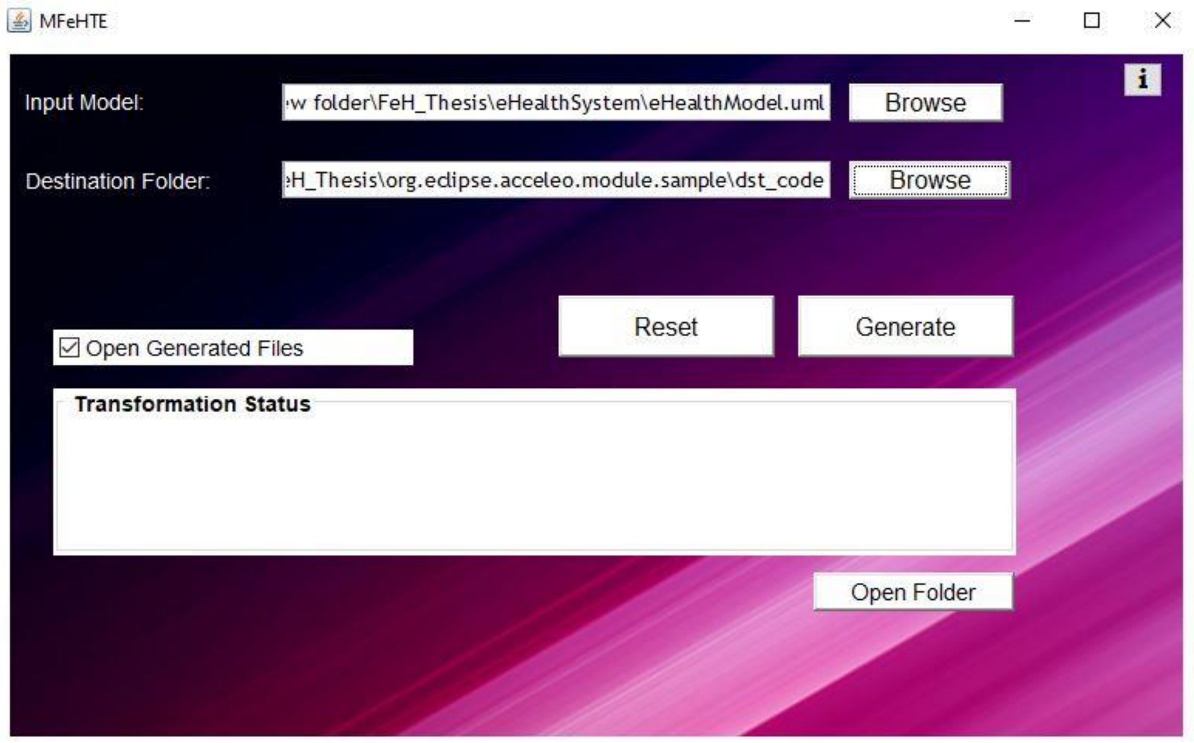
Features (from FeH::UMLPFeH::Application)
architecture: ArchType [1] = x86
os: OSType [1] = Linux
vmm: String [1] = Xen
timezone: EDouble [1] = 10.0
costPerSec: EDouble [1] = 3.0
costPerMem: EDouble [1] = 0.05
costPerStorage: EDouble [1] = 0.001
costPerBw: EDouble [1] = 0.0

Figure 22: Assigning tagged values in Domain Model

5.1.3 Transformation

In the Transformation engine "**MFeHTE**" the domain model (.uml file) of fog based ehealth is fed as an input and then execute the transformation set of rules to convert the input domain model (.uml file) into java code, as shown in **Figure 23**. Initially, the UML model and destination folder are specified in GUI. Transformation status can be seen on the screen in

Figures 23 and **24**. Once the transformation is done, generated java files can be seen in the destination folder in **Figure 25.**



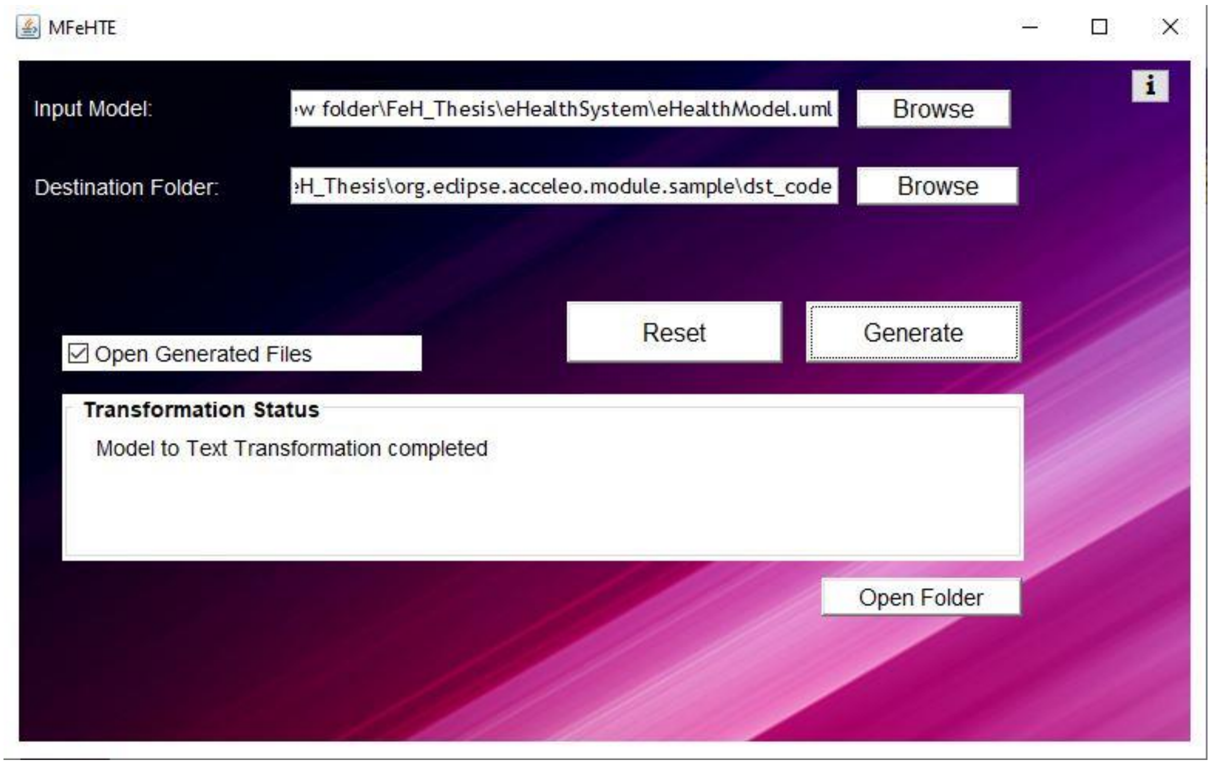
Input Model: w folder\FeH_Thesis\eHealthSystem\eHealthModel.uml

Destination Folder: H_Thesis\org.eclipse.acceleo.module.sample\dst_code

☑ Open Generated Files

Transformation Status

Figure 23: Model Transformation of Case Study



Input Model: w folder\FeH_Thesis\eHealthSystem\eHealthModel.uml

Destination Folder: H_Thesis\org.eclipse.acceleo.module.sample\dst_code

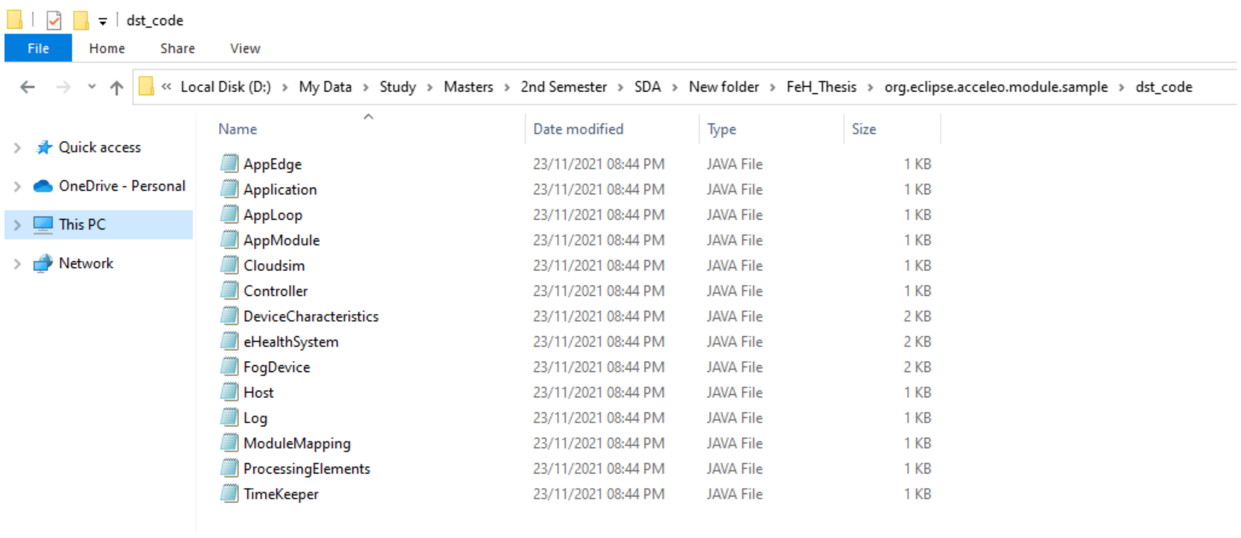
☑ Open Generated Files

Transformation Status

Model to Text Transformation completed

Figure 24: Input Screen with transformation status

The generated output can be seen in the Target Folder in Figure.


dst_code
File
Home
Share
View
Local Disk (D:) › My Data › Study › Masters › 2nd Semester › SDA › New folder › FeH_Thesis › org.eclipse.acceleo.module.sample › dst_code
Quick access
OneDrive - Personal
This PC
Network

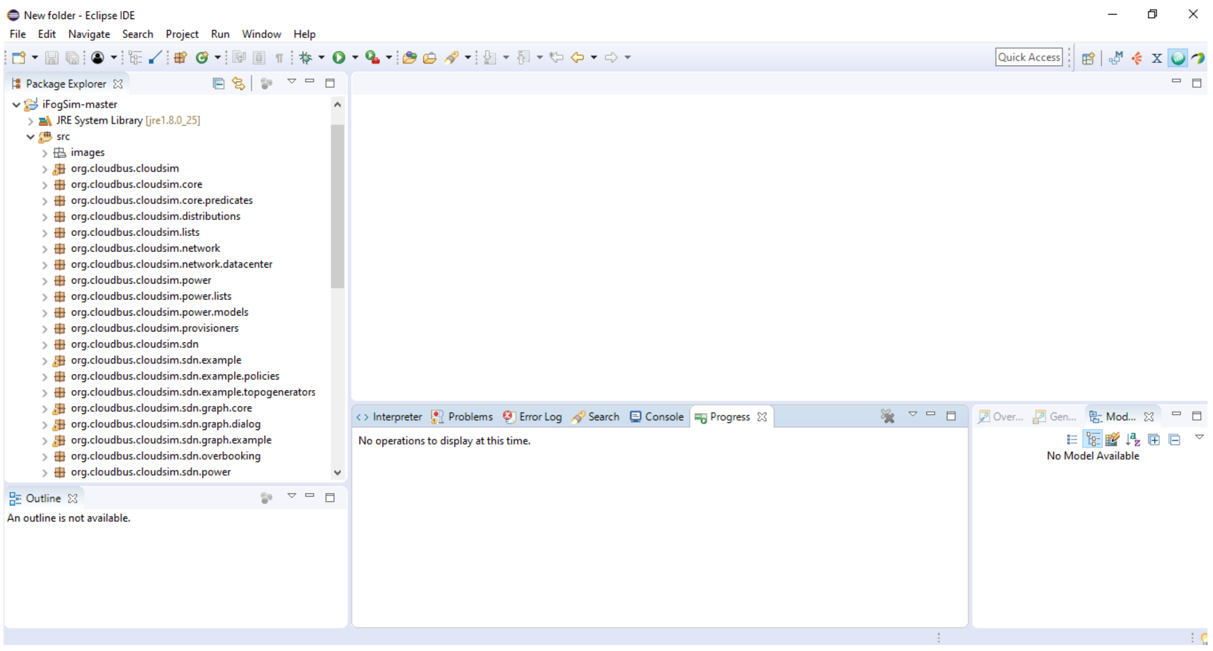
Name	Date modified	Type	Size
AppEdge	23/11/2021 08:44 PM	JAVA File	1 KB
Application	23/11/2021 08:44 PM	JAVA File	1 KB
AppLoop	23/11/2021 08:44 PM	JAVA File	1 KB
AppModule	23/11/2021 08:44 PM	JAVA File	1 KB
Cloudsim	23/11/2021 08:44 PM	JAVA File	1 KB
Controller	23/11/2021 08:44 PM	JAVA File	1 KB
DeviceCharacteristics	23/11/2021 08:44 PM	JAVA File	2 KB
eHealthSystem	23/11/2021 08:44 PM	JAVA File	2 KB
FogDevice	23/11/2021 08:44 PM	JAVA File	2 KB
Host	23/11/2021 08:44 PM	JAVA File	1 KB
Log	23/11/2021 08:44 PM	JAVA File	1 KB
ModuleMapping	23/11/2021 08:44 PM	JAVA File	1 KB
ProcessingElements	23/11/2021 08:44 PM	JAVA File	1 KB
TimeKeeper	23/11/2021 08:44 PM	JAVA File	1 KB

Figure 25: Generated files in the target folder

5.1.4 Early Resource Analysis

The generated code from MFeHTE needs to be verified. Therefore, we have used the iFogSim tool to perform simulation. Once the transformation process is completed, java files are selected and imported into the iFogSim tool as shown in **Figure 26**. After the import is completed, java code needs to be compiled in iFogSim as shown in **Figure 27**. After successful execution as shown in **Figure 28.** simulation results are presented. **Figure 29** provides the simulation results in the console. It demonstrates the calculated values of:

- Execution Time:
- Application Loops Delay:
- CPU Execution Delay:
- Patient_Game_State, EEG, Concentration, Sensor and Global_Game_State.
- Energy Consumed: cloud, proxy-server, department, and mobile sensors
- Cost of execution in the cloud: Total network usage:

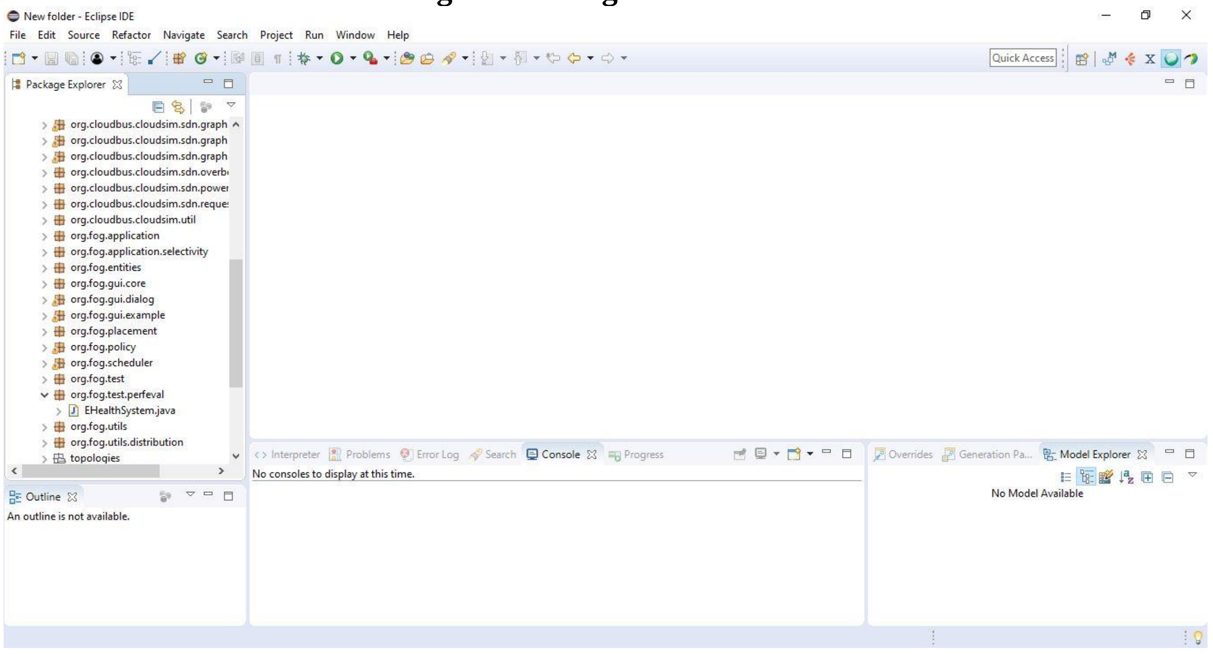

New folder - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Package Explorer
iFogSim-master
JRE System Library [jre1.8.0_25]
src
images
org.cloudbus.cloudsim
org.cloudbus.cloudsim.core
org.cloudbus.cloudsim.core.predicates
org.cloudbus.cloudsim.distributions
org.cloudbus.cloudsim.lists
org.cloudbus.cloudsim.network
org.cloudbus.cloudsim.network.datacenter
org.cloudbus.cloudsim.power
org.cloudbus.cloudsim.power.lists
org.cloudbus.cloudsim.power.models
org.cloudbus.cloudsim.provisioners
org.cloudbus.cloudsim.sdn
org.cloudbus.cloudsim.sdn.example
org.cloudbus.cloudsim.sdn.example.policies
org.cloudbus.cloudsim.sdn.example.topogenerators
org.cloudbus.cloudsim.sdn.graph.core
org.cloudbus.cloudsim.sdn.graph.dialog
org.cloudbus.cloudsim.sdn.graph.example
org.cloudbus.cloudsim.sdn.overbooking
org.cloudbus.cloudsim.sdn.power
Outline
Interpreter Problems Error Log Search Console Progress
Over... Gen... Mod...

An outline is not available.

No operations to display at this time.

No Model Available

Figure 26: iFogSim Interface

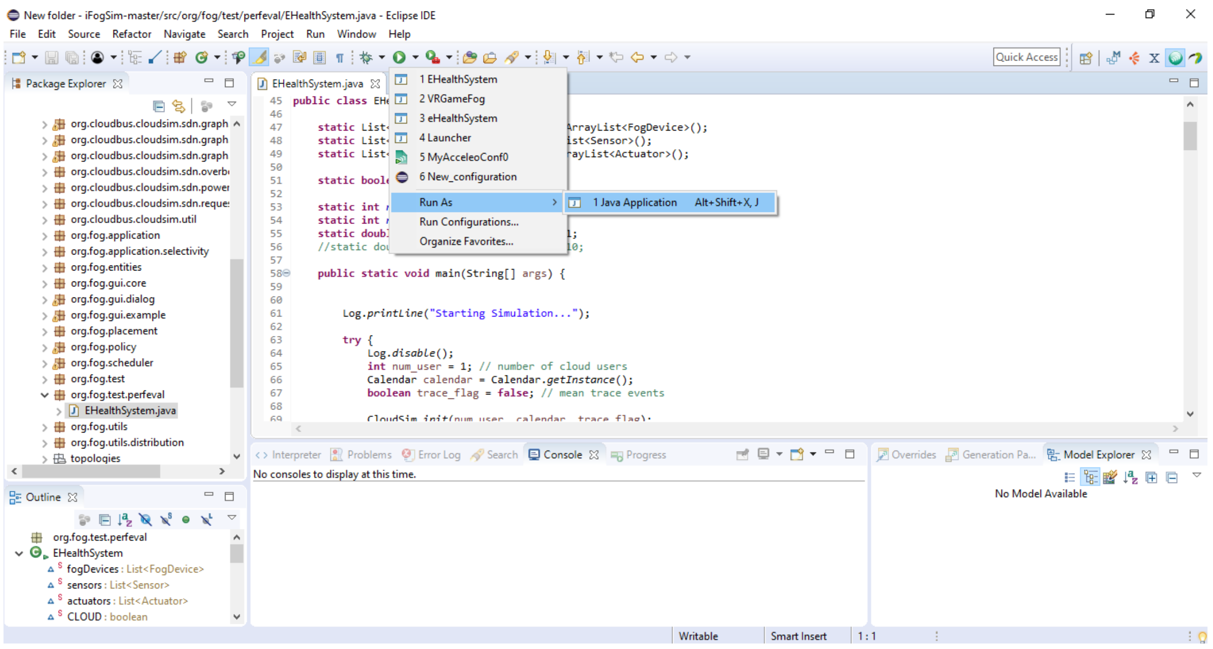

New folder - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
org.cloudbus.cloudsim.sdn.graph
org.cloudbus.cloudsim.sdn.graph
org.cloudbus.cloudsim.sdn.graph
org.cloudbus.cloudsim.sdn.overb
org.cloudbus.cloudsim.sdn.power
org.cloudbus.cloudsim.sdn.reque
org.cloudbus.cloudsim.util
org.fog.application
org.fog.application.selectivity
org.fog.entities
org.fog.gui.core
org.fog.gui.dialog
org.fog.gui.example
org.fog.placement
org.fog.policy
org.fog.scheduler
org.fog.test
org.fog.test.perfeval
EHealthSystem.java
org.fog.utils
org.fog.utils.distribution
topologies
Outline
Interpreter Problems Error Log Search Console Progress
Overrides Generation Pa... Model Explorer

An outline is not available.

No consoles to display at this time.

No Model Available

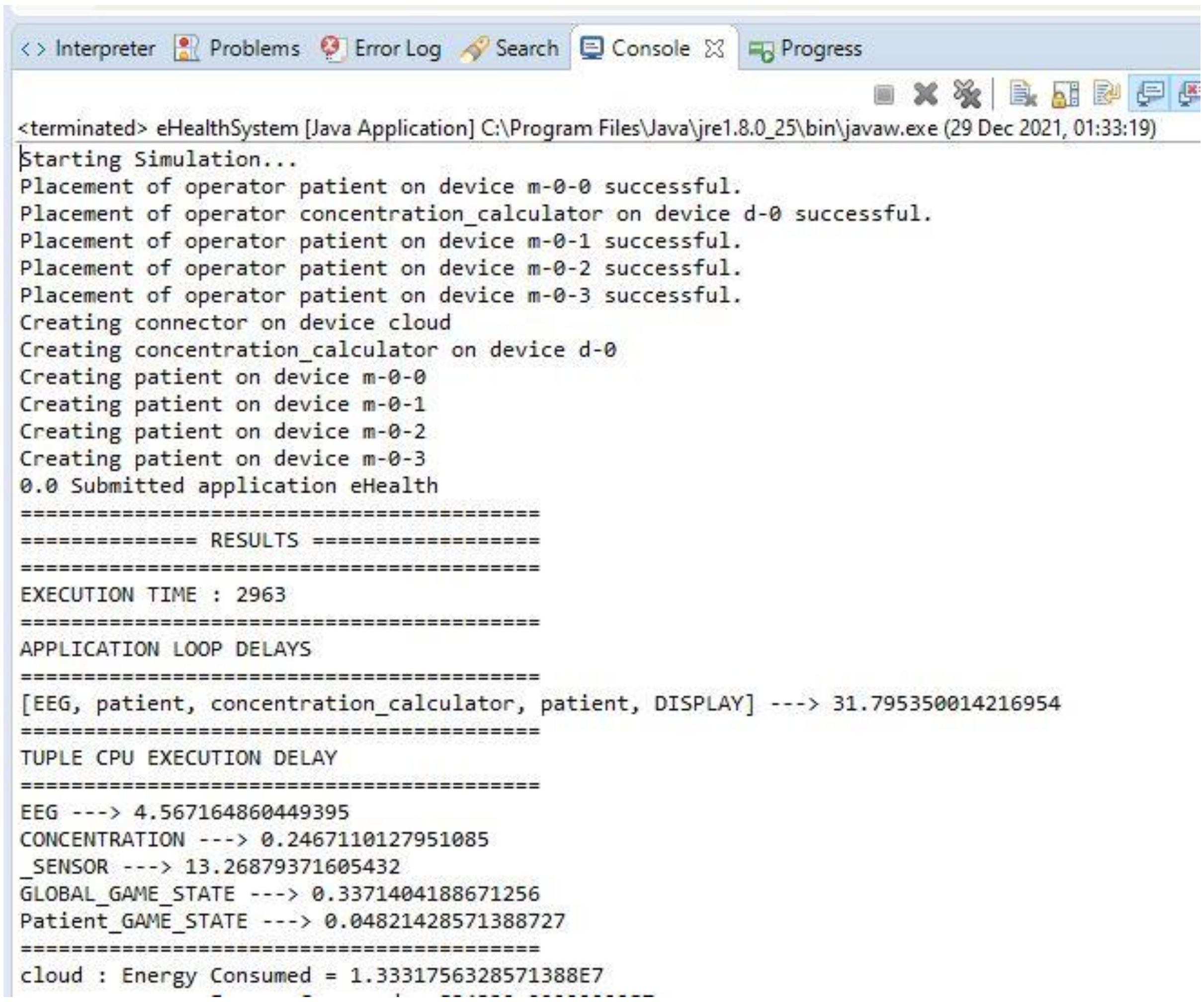
Figure 27: Importing java file in iFogSim



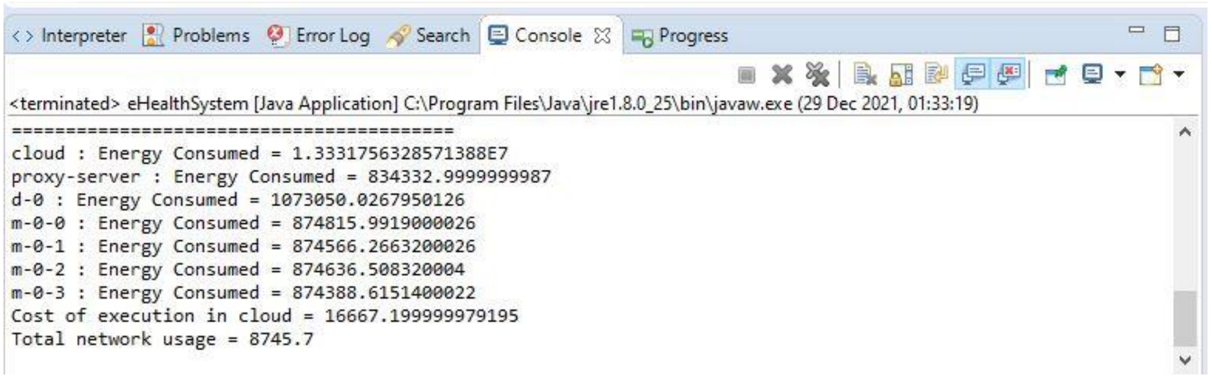
```
public class EHe
    
    static List<                                    ArrayList<FogDevice>();
    static List<                                    ist<Sensor>();
    static List<                                    rayList<Actuator>();
    
    static boole
    
    static int r
    static int r
    static doubl                                    1;
    //static dou                                    10;
    
    public static void main(String[] args) {
    
    
        Log.printLine("Starting Simulation...");
        
        try {
            Log.disable();
            int num_user = 1; // number of cloud users
            Calendar calendar = Calendar.getInstance();
            boolean trace_flag = false; // mean trace events
            
            CloudSim.init(num_user, calendar, trace_flag);
```

```
No consoles to display at this time.
```

Figure 28: Executing java file in iFogSim



```
<terminated> eHealthSystem [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (29 Dec 2021, 01:33:19)
Starting Simulation...
Placement of operator patient on device m-0-0 successful.
Placement of operator concentration_calculator on device d-0 successful.
Placement of operator patient on device m-0-1 successful.
Placement of operator patient on device m-0-2 successful.
Placement of operator patient on device m-0-3 successful.
Creating connector on device cloud
Creating concentration_calculator on device d-0
Creating patient on device m-0-0
Creating patient on device m-0-1
Creating patient on device m-0-2
Creating patient on device m-0-3
0.0 Submitted application eHealth
=========================================
============== RESULTS ==================
=========================================
EXECUTION TIME : 2963
=========================================
APPLICATION LOOP DELAYS
=========================================
[EEG, patient, concentration_calculator, patient, DISPLAY] ---> 31.795350014216954
=========================================
TUPLE CPU EXECUTION DELAY
=========================================
EEG ---> 4.567164860449395
CONCENTRATION ---> 0.2467110127951085
_SENSOR ---> 13.26879371605432
GLOBAL_GAME_STATE ---> 0.3371404188671256
Patient_GAME_STATE ---> 0.04821428571388727
=========================================
cloud : Energy Consumed = 1.3331756328571388E7
```



```
<terminated> eHealthSystem [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (29 Dec 2021, 01:33:19)
=========================================
cloud : Energy Consumed = 1.3331756328571388E7
proxy-server : Energy Consumed = 834332.9999999987
d-0 : Energy Consumed = 1073050.0267950126
m-0-0 : Energy Consumed = 874815.9919000026
m-0-1 : Energy Consumed = 874566.2663200026
m-0-2 : Energy Consumed = 874636.508320004
m-0-3 : Energy Consumed = 874388.6151400022
Cost of execution in cloud = 16667.199999979195
Total network usage = 8745.7
```

Figure 29: Fog based ehealth Framework (console) Results in iFogSim

5.2 Smart Glove Case Study

This case study is explained and validated by dividing it into four sections**. Section 5.2.1** describes the Requirement Specification for the Smart Glove case study. **Section 5.2.2** demonstrates UML domain modeling with applied Profile UMLPFeH of this case study in Eclipse plugin Papyrus and presents the transformation of the domain model into android code as result.

5.2.1 Requirement Specification

A smart glove is employed in this case study to meet the telemedicine requirements for Parkinson's disease [82]. The pointing finger and thumb of the smart glove are sewn with flex sensors to detect motor symptoms in the hand such as tremors, rigidity, and slowness of movement. The Arduino 101 Smart Glove has an Intel Curie processor for onboard processing and Bluetooth low-energy connection with the local network. The voltage across the flex sensors is read by the Curie Chip, which then converts it to resistance. We used the finger-tapping motor task from the Unified Parkinson's Disease Rating Scale (UPDRS) [83], which is one of the examinations. The frequency of tapping is one of the major aspects evaluated by neurologists. Neurologists, for example, track the rate of fluctuation in tapping speed and provide a clinical score for the job. The finger tapping activity was chosen because it has a higher clinical value and is simple for patients to complete at home.

The patients must tap their pointing fingers and thumb 10 times to complete the job. As a result, we included a healthy person in our study who did five rounds of finger-tapping. In each round, the individual was asked to alter the rate of finger tapping frequency. To determine the intensity of the tapping on each occasion. The patient must wear a Smart Glove that is connected to the Fog Device to tap their pointing fingers and thumb, we used a signal calculator.

We consider a signal to be a peak if its amplitude exceeds a threshold and is greater than both previous and subsequent samples. We also added a temporal threshold because each pinch has a few micro motions or minor variations that cause multiple peaks to appear at once. As a result, we do temporal filtration to ensure that if the time interval between two peaks is too short, we do not treat them as different peaks for different pinches.

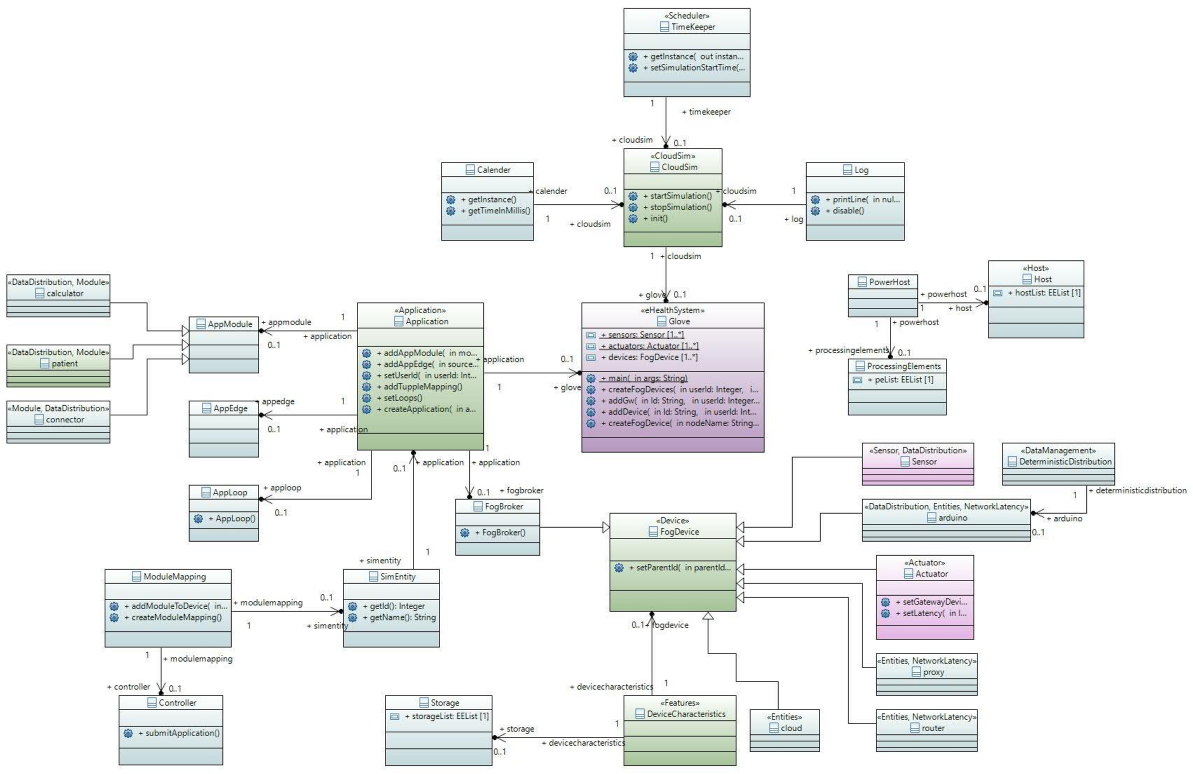
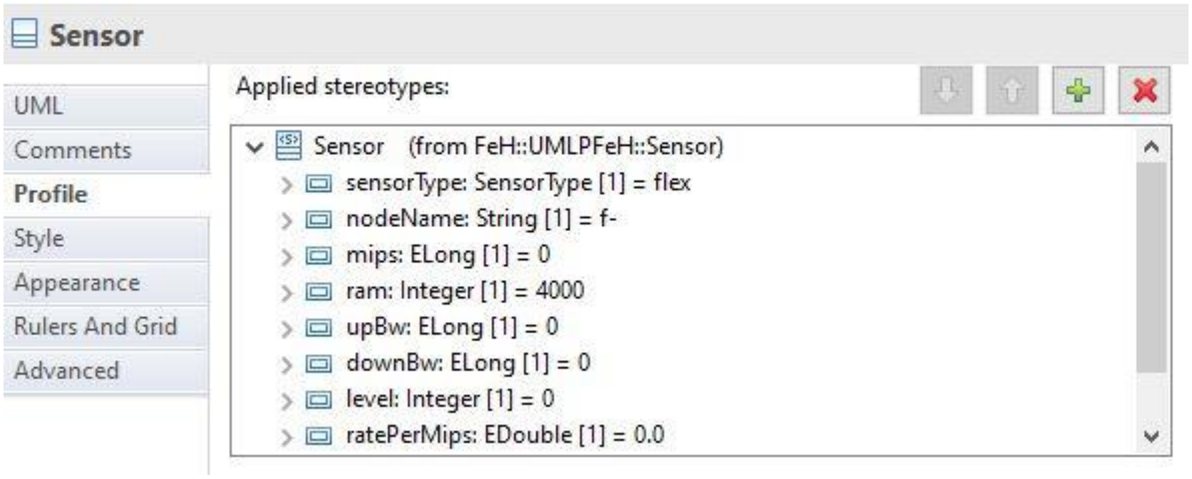


Figure 30: Design model of Smart Glove

5.2.2 Modeling

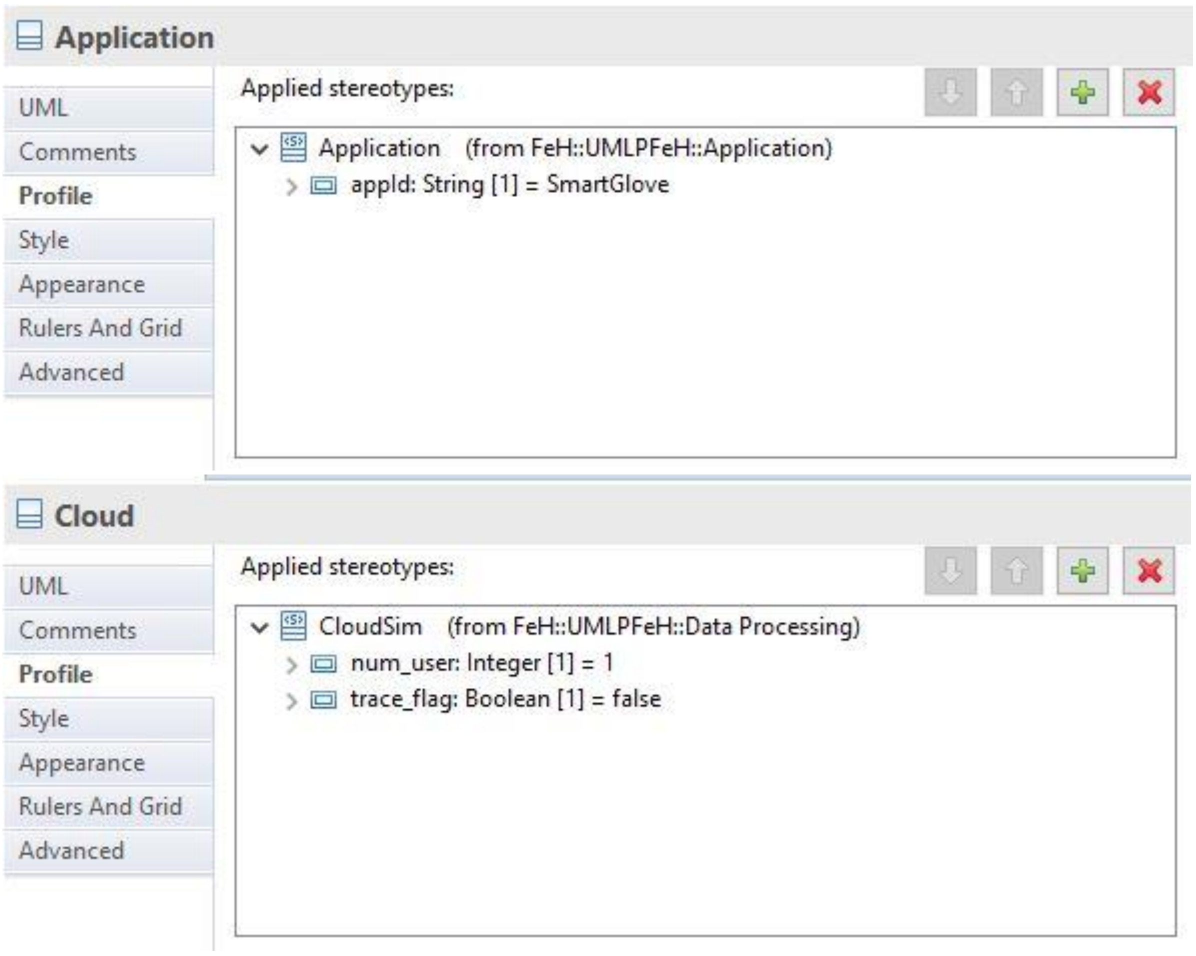
As previously stated, the MFeH provides a generic approach for writing Java code. As a result, the domain model mentioned in the preceding case study can be applied here as well.


Sensor

UML
Comments
Profile
Style
Appearance
Rulers And Grid
Advanced

Applied stereotypes:

Sensor (from FeH::UMLPFeH::Sensor)
sensorType: SensorType [1] = flex
nodeName: String [1] = f-
mips: ELong [1] = 0
ram: Integer [1] = 4000
upBw: ELong [1] = 0
downBw: ELong [1] = 0
level: Integer [1] = 0
ratePerMips: EDouble [1] = 0.0



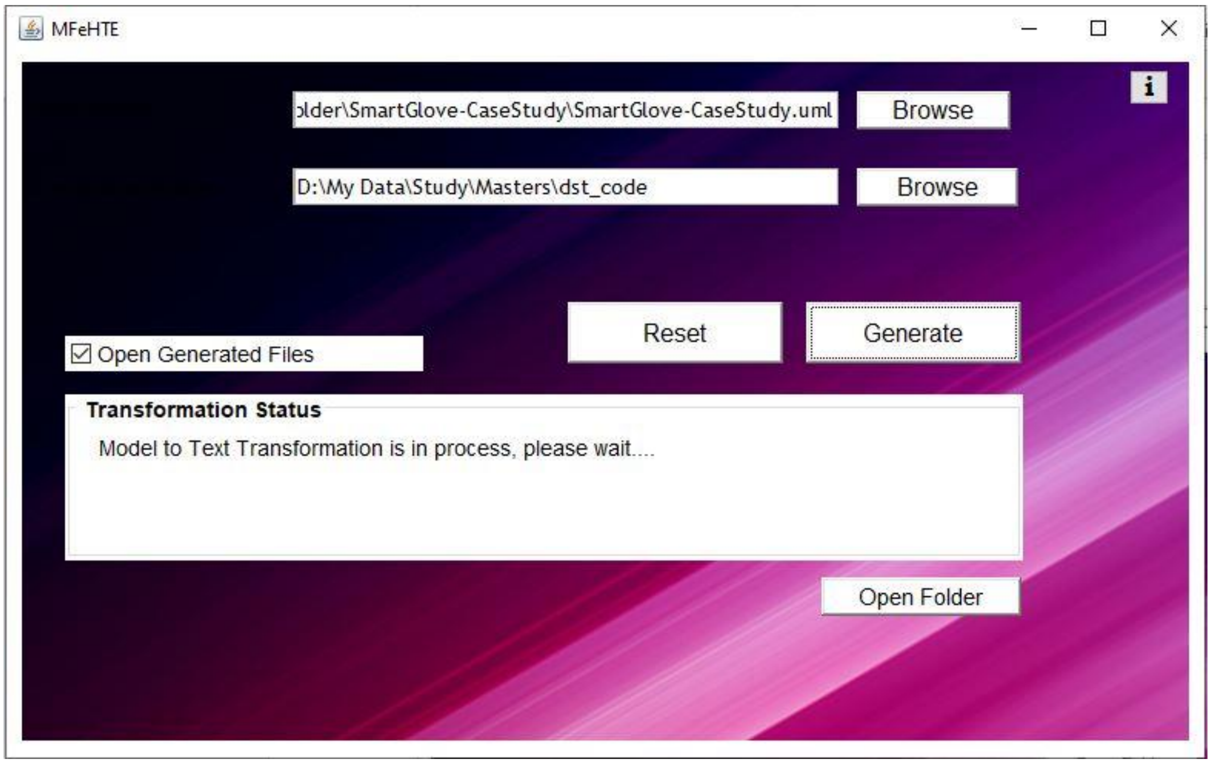
```
Application    (from FeH::UMLPFeH::Application)
    appId: String [1] = SmartGlove
```

```
CloudSim    (from FeH::UMLPFeH::Data Processing)
    num_user: Integer [1] = 1
    trace_flag: Boolean [1] = false
```

Figure 31: Assigning Tagged Values

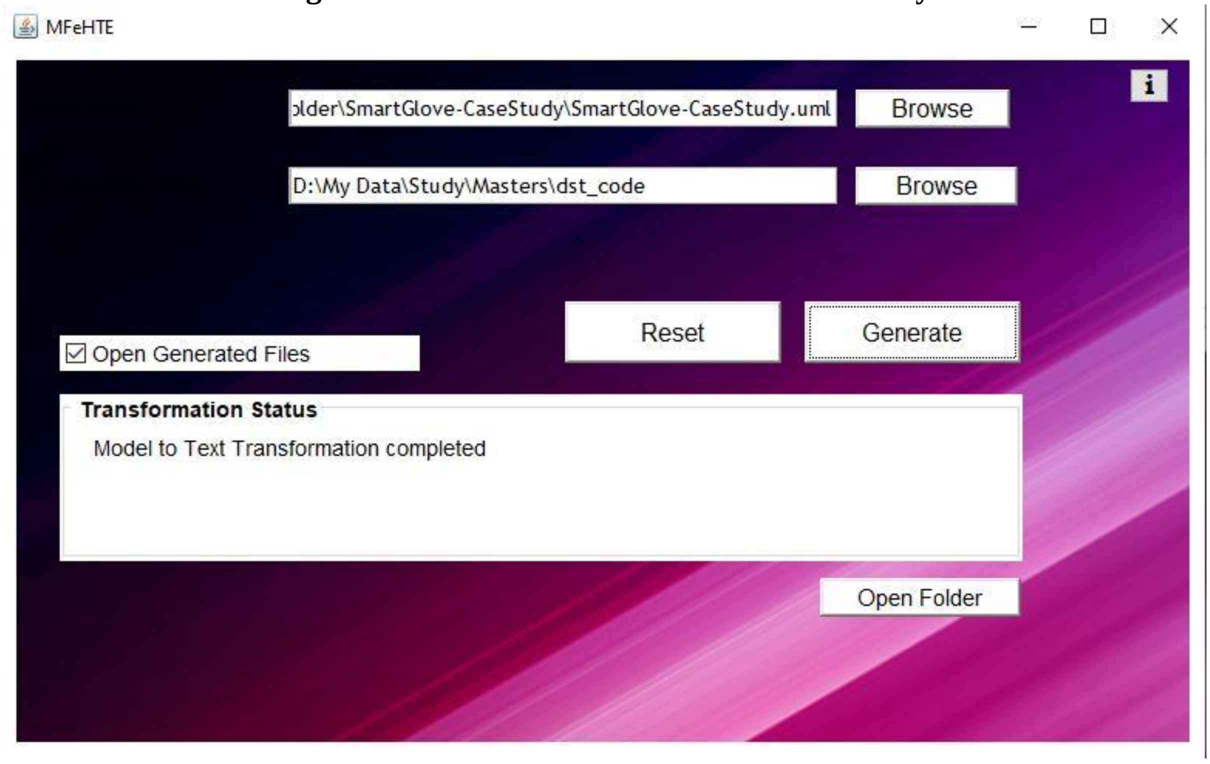
5.2.3 Transformation:

In the Transformation engine "**MFeHTE**" the domain model (.uml file) of Smart Glove is fed as an input and then execute the transformation set of rules to convert the input domain model (.uml file) into java code. Initially, the UML model and destination folder are specified in GUI. Transformation status can be seen on the screen in **Figures 32** and **33**.


MFeHTE
ɔlder\SmartGlove-CaseStudy\SmartGlove-CaseStudy.uml
Browse
D:\My Data\Study\Masters\dst_code
Browse
Reset
Generate
☑ Open Generated Files
Transformation Status

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

Figure 32: Model Transformation of Case study


MFeHTE
ɔlder\SmartGlove-CaseStudy\SmartGlove-CaseStudy.uml
Browse
D:\My Data\Study\Masters\dst_code
Browse
Reset
Generate
☑ Open Generated Files
Transformation Status

Model to Text Transformation completed

Figure 33: Model Transformation of Case Study

Chapter 6

Discussion and Limitations

Chapter 6: Discussion and Limitations

Section 6.1 contains a detailed discussion of the proposed research work and **Section 6.2** deals with the limitation of the research.

6.1 Discussion

IoT-based services are growing increasingly popular. The number of linked devices reached 9 billion in 2020, and this figure is anticipated to rise to 24 billion by 2025. With such a rapid rise in the number of different devices and the data created as a result, efficiently managing the data generated, power used, and bandwidth consumed is not difficult. In this case, a localized micro data center would be located near the local nodes to delegate duties and filter the raw data. Fog or Edge is the name given to the tiny data center.
By including better receptiveness, the fog paradigm helps to reduce delays and improve service quality, making multimedia streaming and other delay-sensitive applications possible. When the cloud and IoT are combined, multimedia data is generated.

The cost of latency is too high for eHealth, which is one of the most basic IoT applications. It is unquestionably necessary to be able to analyze and follow up on time-sensitive data and conditions. As a result, relying on the traditional cloud architecture and design to collect and evaluate patient-important clinical information, vital signs, and bio-signals across a large geographical region in the face of varying environmental conditions is not practical. Using fog or edge computing to take cloud computing and administrations to a higher level is the most efficient approach to deal with this issue. A fog node is a device that has computational, storage, and networking capabilities. We analyze time-sensitive data in the eHealth platform and make extremely time-sensitive decisions on fog nodes. These nodes are the ones that are closest to the medical equipment that generates the data. The rest of the data, on the other hand, is sent to the cloud as the primary storage and computational resource. Another significant difficulty with IoT eHealth is the need to conserve network capacity. An EEG device, for example, can create many GBs of raw time-series data in a single day. However, sending these massive amounts of data from thousands of patients to the cloud is neither possible nor necessary. As a result, fog nodes are a viable option for processing, filtering, and compressing data sent between medical equipment and the cloud.

Model-driven engineering (MDE) enables the creation of UML models to express fog based ehealth system needs as stereotypes. With minimal effort, these models can be translated into low-level code. Because of its independence from the platform, the MDE method makes our suggested system less susceptible to technological changes. The model of the proposed work can easily be transformed into any of the languages like Java, C#.Net, C++, etc.

In this research, an **MFeHSTE** (**M**odel-Driven **F**og-based **eH**ealth **S**ystem **T**ransformation **E**ngine) is presented. Particularly, a Unified Modeling Language (UML) Profile (i.e. **U**ML **P**rofile for **F**og-based **eHe**alth **S**ystem - **UPFeHS**) is developed by extending UML-meta-model to adapt to the concepts of fog based ehealth system framework. This allows modeling of fog based ehealth system functions along with resources and their characteristics. Furthermore, a transformation engine **MFeHSTE** (**M**odel-Driven **F**og-based **eH**ealth **S**ystem **T**ransformation **E**ngine) is developed, that converts the high-level UPFeHS models into low-level java code for eHealth Systems. The transformation engine is developed by following the model-to-text approach by using the Acceleo tool. The generated code from the transformation engine not only supports eHealth system development but also provides early resource assessment capabilities. Particularly, the generated code can be integrated into the iFogSim tool to perform resource analysis. Lastly, the framework is validated by two benchmark case studies i.e., EEG Tractor Beam and Smart Glove. In addition, the experimental results indicate that the offered framework delivers an easy and reusable solution for a fog-based eHealth system where major functions along with resource estimation characteristics can be managed simultaneously at a higher level of abstraction with reduced complexity and cost.

6.2 Limitations

This approach leads to the automated code generation for Fog based eHealth Systems in an automated way, but there exist some limitations as well. We have validated our proposed framework using two benchmark case studies but the implementation of this system in the real world has not yet been carried out. We have provided a very basic type of components of the eHealth system with a limited number of their properties. The data is not real-time for EEG Monitoring. The current method involves modeling complexity as an operator has to apply stereotypes by himself to define each artifact. Therefore, a proper editor is required to develop this system.

Chapter 7

Conclusion and Future Work

Chapter 7: Conclusion and Future Work

The proposed framework provides a solution for a Fog-based eHealth System where resource assessment characteristics can be generated simultaneously at a higher level of abstraction with reduced complexity and cost. It is based on a model-driven approach to provide simple, open-source, and reusable solutions for early design analysis.

Particularly, a Unified Modeling Language (UML) Profile (i.e. **U**ML **P**rofile for **F**og-based **eH**ealth **S**ystem - **UPFeHS**) is developed by extending UML-meta-model to adapt to the concepts of fog based ehealth system framework. This allows to model fog based ehealth system functions along with resources and their characteristics. Furthermore, a transformation engine, named **M**odel-Driven **F**og-based **eH**ealth **S**ystem **T**ransformation **E**ngine (**MFeHSTE**), is developed to convert high-level UPFeHS models into low-level java code for ehealth systems. The transformation engine is based on the model-to-text (M2T) approach and developed using the Acceleo tool. The generated code from the transformation engine not only supports ehealth system development but also provides early resource assessment capabilities. Particularly, the generated code can be integrated into the iFogSim tool to perform resource analysis. Lastly, the framework is validated by two benchmark case studies i.e. EEG beam tractor and smart glove. The experimental outcomes indicate that the offered framework provides an easy and reusable solution for fog based ehealth systems where major functions along with resource estimation characteristics can be managed simultaneously at a higher level of abstraction with reduced complexity and cost.

In the future, we are intended to extend UPFeH to add concepts that may help to compute real-time data. In addition, this research may be extended to deal with the confidentiality and security concerns in the eHealth system and emphasize data dissemination among different nodes and at different levels of the network. We also are concerned about enhancing the real-time data/information.

References

[1] M. Mukherjee, L. Shu and D. Wang, "Survey of Fog Computing: Fundamental, Network Applications, and Research Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1826-1857, thirdquarter 2018, doi: 10.1109/COMST.2018.2814571.

[2] Yousefpour, Ashkan, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. "All one needs to know about fog computing and related edge computing paradigms: A complete survey." *Journal of Systems Architecture* 98 (2019): 289-330.

[3] Javadzadeh, Ghazaleh, and Amir Masoud Rahmani. "Fog computing applications in smart cities: A systematic survey." *Wireless Networks* 26, no. 2 (2020): 1433-1457.

[4] Shivashankar, Karthik, and Venkat Bakthavatchaalam. "Smart Municipal Governance System Using Confluence of Technologies in a Developing Country Scenario." In *14th International Conference on Theory and Practice of Electronic Governance*, pp. 535-537. 2021.

[5] Cardullo, Paolo, and Rob Kitchin. "Being a 'citizen'in the smart city: Up and down the scaffold of smart citizen participation in Dublin, Ireland." *GeoJournal* 84, no. 1 (2019): 1-13.

[6] Singh, Harpreet, and Shah J. Miah. "Smart education literature: A theoretical analysis." *Education and Information Technologies* 25, no. 4 (2020): 3299-3328.

[7] Granja, Conceição, Wouter Janssen, and Monika Alise Johansen. "Factors determining the success and failure of eHealth interventions: systematic review of the literature." *Journal of medical Internet research* 20, no. 5 (2018): e10235.

[8] Dong, Bing, Vishnu Prakash, Fan Feng, and Zheng O'Neill. "A review of smart building sensing system for better indoor environment control." *Energy and Buildings* 199 (2019): 29-46.

[9] Thellufsen, Jakob Zinck, Henrik Lund, P. Sorknæs, P. A. Østergaard, M. Chang, D. Drysdale, Steen Nielsen, S. R. Djørup, and K. Sperling. "Smart energy cities in a 100% renewable energy context." *Renewable and Sustainable Energy Reviews* 129 (2020): 109922.

[10] Tomor, Zsuzsanna, Erico Przeybilovicz, and Charles Leleux. "Smart governance in institutional context: An in-depth analysis of Glasgow, Utrecht, and Curitiba." *Cities* 114 (2021): 103195.

[11] Tikhomirov, Vladimir, Natalia Dneprovskaya, and Ekaterina Yankovskaya. "Three dimensions of smart education." In *Smart Education and Smart e-Learning*, pp. 47-56. Springer, Cham, 2015.

[12] H. Kazmi, F. Mehmood and M. Amayri, "Smart Home Futures: Algorithmic Challenges and Opportunities," *2017 14th International Symposium on Pervasive Systems, Algorithms and Networks & 2017 11th International Conference on Frontier of Computer Science and Technology & 2017 Third International Symposium of Creative Computing (ISPAN-FCST-ISCC)*, 2017, pp. 441-448, doi: 10.1109/ISPAN-FCST-ISCC.2017.60.

[13] J. Dutta and S. Roy, "IoT-fog-cloud based architecture for smart city: Prototype of a smart building," 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence, 2017, pp. 237-242, doi: 10.1109/CONFLUENCE.2017.7943156.

[14] P. Dongbaare, S. O. Osuri and S. P. Daniel Chowdhury, "A smart energy management system for residential use," 2017 IEEE PES PowerAfrica, 2017, pp. 612-616, doi: 10.1109/PowerAfrica.2017.7991296.

[15] N. V. Lopes, "Smart governance: A key factor for smart cities implementation," 2017 IEEE International Conference on Smart Grid and Smart Cities (ICSGSC), 2017, pp. 277-282, doi: 10.1109/ICSGSC.2017.8038591.

[16] Cardoso, João MP, José Gabriel F. Coutinho, and Pedro C. Diniz. "Chapter 5-Source code transformations and optimizations." *by João MP Cardoso, José Gabriel F. Coutinho, and Pedro C. Diniz. Boston: Morgan Kaufmann* (2017): 137-183.

[17] M. Bennis, M. Debbah and H. V. Poor, "Ultrareliable and Low-Latency Wireless Communication: Tail, Risk, and Scale," in *Proceedings of the IEEE*, vol. 106, no. 10, pp. 1834-1853, Oct. 2018, doi: 10.1109/JPROC.2018.2867029.

[18] P. Putra, S. Wijaya, Z. A. Wirahaditenaya, R. Jayadi and T. Mauritsius, "Predicting Network Bandwidth Usage: Case Study at PT. Bank ABC," *2020 IEEE International Conference on Communication, Networks and Satellite (Comnetsat)*, 2020, pp. 227-233, doi: 10.1109/Comnetsat50391.2020.9328970.

[19] R. Yu, G. Xue and X. Zhang, "Application Provisioning in FOG Computing-enabled Internet-of-Things: A Network Perspective," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, 2018, pp. 783-791, doi: 10.1109/INFOCOM.2018.8486269.

[20] A. Konda, "An Analysis of eHealth Modes, Usage Levels, Enablers, and Obstacles," *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, 2018, pp. 1-4, doi: 10.1109/HealthCom.2018.8531087.

[21] K. Edemacu, B. Jang and J. W. Kim, "Collaborative Ehealth Privacy and Security: An Access Control With Attribute Revocation Based on OBDD Access Structure," in *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2960-2972, Oct. 2020, doi: 10.1109/JBHI.2020.2973713.

[22] Xiong, Haoyi, Jinghe Zhang, Yu Huang, Kevin Leach, and Laura E. Barnes. "Daehr: A discriminant analysis framework for electronic health record data and an application to early detection of mental health disorders." *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, no. 3 (2017): 1-21.

[23] Yadav, Pranjul, Michael Steinbach, Vipin Kumar, and Gyorgy Simon. "Mining electronic health records (EHRs) A survey." *ACM Computing Surveys (CSUR)* 50, no. 6 (2018): 1-40.

[24] Sun, Jin, Xiaomin Yao, Shangping Wang, and Ying Wu. "Blockchain-based secure storage and access scheme for electronic medical records in IPFS." *IEEE Access* 8 (2020): 59389-59401.

[25] G. Márquez, H. Astudillo and C. Taramasco, "Security in Telehealth Systems From a Software Engineering Viewpoint: A Systematic Mapping Study," in *IEEE Access*, vol. 8, pp. 10933-10950, 2020, doi: 10.1109/ACCESS.2020.2964988.

[26] Mohamed, Walaa, and Mohammad M. Abdellatif. "Telemedicine: An IoT Application For Healthcare systems." In *Proceedings of the 2019 8th International Conference on Software and Information Engineering*, pp. 173-177. 2019.

[27] Chandwani, Rajesh, and Neha Kumar. "Stitching Infrastructures to Facilitate Telemedicine for Low-Resource Environments." In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pp. 1-12. 2018.

[28] C. Crema, A. Depari, A. Flammini, E. Sisinni, A. Vezzoli and P. Bellagente, "Virtual Respiratory Rate Sensors: An Example of A Smartphone-Based Integrated and Multiparametric mHealth Gateway," in *IEEE Transactions on Instrumentation and Measurement*, vol. 66, no. 9, pp. 2456-2463, Sept. 2017, doi: 10.1109/TIM.2017.2707838.

[29] Ruiz-Zafra, Angel, Kawtar Benghazi, Constandinos Mavromoustakis, and Manuel Noguera. "An iot-aware architectural model for smart habitats." In *2018 IEEE 16th International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 103-110. IEEE, 2018.

[30] Farahani, Bahar, Farshad Firouzi, Victor Chang, Mustafa Badaroglu, Nicholas Constant, and Kunal Mankodiya. "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare." *Future Generation Computer Systems* 78 (2018): 659-676.

[31] Attipoe-Dorcoo, Sharon, Rigoberto Delgado, Aditi Gupta, Jennifer Bennet, Nancy E. Oriol, and Sachin H. Jain. "Mobile health clinic model in the COVID-19 pandemic: lessons learned and opportunities for policy changes and innovation." *International Journal for Equity in Health* 19, no. 1 (2020): 1-5.

[32] Shuhaiber, Ahmed, and Ibrahim Mashal. "Understanding users’ acceptance of smart homes." *Technology in Society* 58 (2019): 101110.

[33] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," in *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180.

[34] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir and E. Rivière, "On Using Micro-Clouds to Deliver the Fog," in *IEEE Internet Computing*, vol. 21, no. 2, pp. 8-15, Mar.-Apr. 2017, doi: 10.1109/MIC.2017.35.

[35] Satyanarayanan, Mahadev. "The emergence of edge computing." *Computer* 50, no. 1 (2017): 30-39.

[36] J. Granados, A. Rahmani, P. Nikander, P. Liljeberg and H. Tenhunen, "Towards energy-efficient HealthCare: An Internet-of-Things architecture using intelligent gateways," *2014 4th International Conference on Wireless Mobile Communication and Healthcare - Transforming Healthcare Through Innovations in Mobile and Wireless Technologies (MOBIHEALTH)*, 2014, pp. 279-282, doi: 10.1109/MOBIHEALTH.2014.7015965.

[37] H. Hong, "From Cloud Computing to Fog Computing: Unleash the Power of Edge and End Devices," *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2017, pp. 331-334, doi: 10.1109/CloudCom.2017.53.

[38] D. C. Schmidt, "Guest Editor's Introduction: Model-Driven Engineering," in *Computer*, vol. 39, no. 2, pp. 25-31, Feb. 2006, doi: 10.1109/MC.2006.58.

[39] Pastor, Oscar, Sergio España, José Ignacio Panach, and Nathalie Aquino. "Model-driven development." *Informatik-Spektrum* 31, no. 5 (2008): 394-407.

[40] Huang, Dijiang, and Huijun Wu. *Mobile cloud computing: Foundations and service models*. Morgan Kaufmann, 2017.

[41] Dastjerdi, A., Harshit Gupta, R. Calheiros, and S. Ghosh. "Chapter 4—fog computing: Principles, architectures, and applications. InInternet of Things: Principles and Paradigms, ed. R. Buyya, and AV Dastjerdi, 61–75." (2016).

[42] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow and P. A. Polakos, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416-464, Firstquarter 2018, doi: 10.1109/COMST.2017.2771153.

[43] Konda, Anuja. "An Analysis of eHealth Modes, Usage Levels, Enablers, and Obstacles." In *2018 IEEE 20th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1-4. IEEE, 2018.

[44] K. Edemacu, B. Jang and J. W. Kim, "Collaborative Ehealth Privacy and Security: An Access Control With Attribute Revocation Based on OBDD Access Structure," in *IEEE Journal of Biomedical and Health Informatics*, vol. 24, no. 10, pp. 2960-2972, Oct. 2020, doi: 10.1109/JBHI.2020.2973713.
[45] García-Valls, Marisol, Christian Calva-Urrego, and Ana García-Fornes. "Accelerating smart eHealth services execution at the fog computing infrastructure." *Future Generation Computer Systems* (2018).
[46] Qi, Jun, Po Yang, Geyong Min, Oliver Amft, Feng Dong, and Lida Xu. "Advanced internet of things for personalised healthcare systems: A survey." *Pervasive and Mobile Computing* 41 (2017): 132-149.
[47] Ahmed, Syed Hassan, and Shalli Rani. "A hybrid approach, Smart Street use case and future aspects for Internet of Things in smart cities." *Future Generation Computer Systems* 79 (2018): 941-951.
[48] Rahmani, Amir M., Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach." *Future Generation Computer Systems* 78 (2018): 641-658.
[49] Farahani, Bahar, Farshad Firouzi, Victor Chang, Mustafa Badaroglu, Nicholas Constant, and Kunal Mankodiya. "Towards fog-driven IoT eHealth: Promises and challenges of IoT in medicine and healthcare." *Future Generation Computer Systems* 78 (2018): 659-676.
[50] García-Valls, Marisol, Christian Calva-Urrego, and Ana García-Fornes. "Accelerating smart eHealth services execution at the fog computing infrastructure." *Future Generation Computer Systems* (2018).
[51] Vilela, Pedro H., Joel JPC Rodrigues, Petar Solic, Kashif Saleem, and Vasco Furtado. "Performance evaluation of a Fog-assisted IoT solution for e-Health applications." *Future Generation Computer Systems* 97 (2019): 379-386.
[52] Kumari, Aparna, Sudeep Tanwar, Sudhanshu Tyagi, and Neeraj Kumar. "Fog computing for Healthcare 4.0 environment: Opportunities and challenges." *Computers & Electrical Engineering* 72 (2018): 1-13.
[53] Farahani, Bahar, Mojtaba Barzegari, Fereidoon Shams Aliee, and Khaja Ahmad Shaik. "Towards collaborative intelligent IoT eHealth: From device to fog, and cloud." *Microprocessors and Microsystems* 72 (2020): 102938.

[54] Debauche, Olivier, Saïd Mahmoudi, Pierre Manneback, and Abdessamad Assila. "Fog IoT for Health: A new Architecture for Patients and Elderly Monitoring." *Procedia Computer Science* 160 (2019): 289-297.

[55] Verma, Prabal, and Sandeep K. Sood. "Fog assisted-IoT enabled patient health monitoring in smart homes." *IEEE Internet of Things Journal* 5, no. 3 (2018): 1789-1796.

[56] T. Muhammed, R. Mehmood, A. Albeshri and I. Katib, "UbeHealth: A Personalized Ubiquitous Cloud and Edge-Enabled Networked Healthcare System for Smart Cities," in IEEE Access, vol. 6, pp. 32258-32285, 2018, doi: 10.1109/ACCESS.2018.2846609.

[57] P. H. Vilela, J. J. P. C. Rodrigues, L. R. Vilela, M. M. E. Mahmoud and P. Solic, "A Critical Analysis of Healthcare Applications Over Fog Computing Infrastructures," 2018 3rd International Conference on Smart and Sustainable Technologies (SpliTech), 2018, pp. 1-5.

[58] Azimi, Iman, Arman Anzanpour, Amir M. Rahmani, Tapio Pahikkala, Marco Levorato, Pasi Liljeberg, and Nikil Dutt. "HiCH: Hierarchical fog-assisted computing architecture for healthcare IoT." *ACM Transactions on Embedded Computing Systems (TECS)* 16, no. 5s (2017): 1-20.

[59] Tang, Wenjuan, Ju Ren, Kuan Zhang, Deyu Zhang, Yaoxue Zhang, and Xuemin Shen. "Efficient and Privacy-preserving Fog-assisted Health Data Sharing Scheme." *ACM Transactions on Intelligent Systems and Technology (TIST)* 10, no. 6 (2019): 1-23.

[60] Concone, Federico, Giuseppe Lo Re, and Marco Morana. "A fog-based application for human activity recognition using personal smart devices." *ACM Transactions on Internet Technology (TOIT)* 19, no. 2 (2019): 1-20.

[61] R. Saha, G. Kumar, M. K. Rai, R. Thomas and S. -J. Lim, "Privacy Ensured e{e} - Healthcare for Fog-Enhanced IoT Based Applications," in IEEE Access, vol. 7, pp. 44536-44543, 2019, doi: 10.1109/ACCESS.2019.2908664.

[62] Jia, Xiaoying, Debiao He, Neeraj Kumar, and Kim-Kwang Raymond Choo. "Authenticated key agreement scheme for fog-driven IoT healthcare system." *Wireless Networks* 25, no. 8 (2019): 4737-4750.

[63] Garcia-Perez, Cesar, Almudena Diaz-Zayas, Alvaro Rios, Pedro Merino, Kostas Katsalis, Chia-Yu Chang, Shahab Shariat, Navid Nikaein, Pilar Rodriguez, and Donal Morris. "Improving the efficiency and reliability of wearable based mobile eHealth applications." *Pervasive and Mobile Computing* 40 (2017): 674-691.

[64] Bhuiyan, Md Zakirul Alam, Jie Wu, Guojun Wang, Jiannong Cao, Wenjun Jiang, and Mohammed Atiquzzaman. "Towards cyber-physical systems design for structural health

monitoring: Hurdles and opportunities." *ACM Transactions on Cyber-Physical Systems* 1, no. 4 (2017): 1-26.

[65] Al-Joboury, Istabraq M., and Emad H. Al-Hemiary. "F2CDM: Internet of Things for Healthcare Network Based Fog-to-Cloud and Data-in-Motion Using MQTT Protocol." In *International Symposium on Ubiquitous Networking*, pp. 368-379. Springer, Cham, 2017.

[66] Bhatia, Munish, and Sandeep K. Sood. "Exploring temporal analytics in fog-cloud architecture for Smart Office HealthCare." *Mobile Networks and Applications* 24, no. 4 (2019): 1392-1410.

[67] Cankar, Matija, Eneko Olivares Gorriti, Matevž Markovič, and Flavio Fuart. "Fog and Cloud in the Transportation, Marine and eHealth Domains." In *European Conference on Parallel Processing*, pp. 292-303. Springer, Cham, 2017.

[68] Al-Khafajiy, Mohammed, Lee Webster, Thar Baker, and Atif Waraich. "Towards fog driven IoT healthcare: challenges and framework of fog computing in healthcare." In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, pp. 1-7. 2018.

[69] Elmisery, Ahmed M., Seungmin Rho, and Mohamed Aborizka. "A new computing environment for collective privacy protection from constrained healthcare devices to IoT cloud services." *Cluster Computing* 22, no. 1 (2019): 1611-1638.

[70] Mahmud, Redowan, Fernando Luiz Koch, and Rajkumar Buyya. "Cloud-fog interoperability in IoT-enabled healthcare solutions." In *Proceedings of the 19th international conference on distributed computing and networking*, pp. 1-10. 2018.

[71] K. Wang, Y. Shao, L. Xie, J. Wu and S. Guo, "Adaptive and Fault-Tolerant Data Processing in Healthcare IoT Based on Fog Computing," in IEEE Transactions on Network Science and Engineering, vol. 7, no. 1, pp. 263-273, 1 Jan.-March 2020, doi: 10.1109/TNSE.2018.2859307.

[72] P. Pace, G. Aloi, R. Gravina, G. Caliciuri, G. Fortino and A. Liotta, "An Edge-Based Architecture to Support Efficient Applications for Healthcare Industry 4.0," in IEEE Transactions on Industrial Informatics, vol. 15, no. 1, pp. 481-489, Jan. 2019, doi: 10.1109/TII.2018.2843169.

[73] Hartmann, Morghan, Umair Sajid Hashmi, and Ali Imran. "Edge computing in smart health care systems: Review, challenges, and research directions." *Transactions on Emerging Telecommunications Technologies* (2019): e3710.

[74] https://www.omg.org/

[75] https://www.uml-diagrams.org/uml-meta-models.html

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

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

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

[79] https://github.com/Cloudslab/iFogSim

[80] Gupta, Brij B., and Megha Quamara. "An overview of Internet of Things (IoT): Architectural aspects, challenges, and protocols." *Concurrency and Computation: Practice and Experience* 32, no. 21 (2020): e4946.

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

[82] L. Plant, B. Noriega, A. Sonti, N. Constant and K. Mankodiya, "Smart E-textile gloves for quantified measurements in movement disorders," 2016 IEEE MIT Undergraduate Research Technology Conference (URTC), 2016, pp. 1-4, doi: 10.1109/URTC.2016.8284077.

[83] Martínez-Martín, Pablo, Carmen Rodríguez-Blázquez, Mario Alvarez, Tomoko Arakaki, Víctor Campos Arillo, Pedro Chaná, William Fernández et al. "Parkinson's disease severity levels and MDS-Unified Parkinson's Disease Rating Scale." *Parkinsonism & related disorders* 21, no. 1 (2015): 50-54.