



TAMPEREEN TEKNILLINEN YLIOPISTO  
TAMPERE UNIVERSITY OF TECHNOLOGY

HANNING ZHAO  
USER INTERFACE DESIGN OF META MODEL REPOSITORY  
FOR IOT DEVICES

Master of Science Thesis

Examiners: Prof. Kaisa Väänänen,  
Bilhanan Silverajan  
Supervisors: Prof. Kaisa Väänänen,  
Bilhanan Silverajan  
Examiners and topic approved by the  
Council of the Faculty of Computing  
and Electrical Engineering on 27th  
September 2017

## ABSTRACT

**HANNING ZHAO:** User interface design of meta model repository for IoT devices  
Tampere University of technology  
Master of Science Thesis, 70 pages, 6 Appendix pages

Master's Degree Programme in Information Technology  
Major: User Experience  
Examiner: Professor Kaisa Väänänen, Bilhanan Silverajan

**Keywords:** Data models, Internet of Things(IoT), user-centered design, usability, user interface design, collaboration platforms

Internet of Things (IoT) has become prevalent in recent years. IoT works as a gigantic network in which the vast set of devices are integrated and interconnected. These devices include sensors, gateways and other smart objects. Accordingly, plenty of data models are produced to define and describe IoT devices by various organizations and manufacturers. Those data models significantly help in device management. However, it seems that the sharing and presenting of data models is not so effective. For a variety of organizations have different standardized ways exist to manage and present data models. Particularly, one device may have multiple data models. They are generated as diverse data formats of defining a device and distributed in different platforms. Consequently, to facilitate developers and enterprises' work with data models, existing practices of data model management still need to be upgraded.

This master's thesis proposes a user interface design solution for a meta model repository for IoT devices. Based on the exploration of various collaboration platforms, it analyses the selected platform with its aspects that make it easy to share the models while allow collaboration. Additionally, the study includes the research work on exploring current state of data models. The analysis of different collaboration platforms is also reported. Throughout the design process, user-centered design (UCD) methodology was applied to help create a usable repository in terms of both its user interface and its functionality. In this regard, two rounds of usability testing (6 and 7 participants, respectively) were conducted, which aimed to collect insights and requirements from users. The relevant results are presented and discussed.

The outcome of this thesis is a functional meta model repository which has been designed iteratively during user tests. It starts with support of Lightweight machine to machine (LWM2M) data models. With more data models appear, the repository will be more valuable and significant. More importantly, the repository can be extended to machine to machine communication in future. Therefore, the result of this thesis also demonstrates perceptions about the possibilities of the repository in future use.

## PREFACE

This master thesis work was carried out during autumn 2017 and spring 2018 at the Laboratory of Pervasive Computing at Tampere University of Technology. The goal of the thesis was to design a usable user interface for meta model repository for IoT devices. It aimed to enhance a better management of data models through designing a repository.

Firstly, I would like to express my sincere gratitude to my supervisor Professor Kaisa Väänänen and Doctor Bilhanan Silverajan for constant guidance, inspiration and all constructive comments during my thesis. Especially, Prof. Kaisa Väänänen provided insightful suggestions related to user experience, user interface design and user centered design. Doctor Bilhanan Silverajan also assisted me in the work of IoT field in terms of data models and device management. Both of them are always being willing to provide supports and share their knowledge whenever I got stuck in my thesis.

I also warmly thank my colleague and other participants, who spent time attending the user tests, without their participant, the implemented repository could not achieve a high usability level. Especial thanks to them for providing invaluable feedbacks and comments that contribute to the future development of the repository.

Furthermore, I wish to thank all warm friends and schoolmates for encouragements. Finally, I greatly appreciate my family for immense supports and encouragements in my entire life.

Tampere, on 19<sup>th</sup> of March 2018

Hanning Zhao

## CONTENTS

1.	INTRODUCTION .....	1
1.1.	Terminology .....	2
1.2.	Motivation .....	3
1.3.	Research goal and questions .....	4
1.4.	Structure of the Thesis .....	5
2.	RELATED WORK OF IOT DATA MODELS .....	7
2.1.	Introduction to IoT data models .....	7
2.1.1.	LWM2M and IPSO Objects .....	7
2.1.2.	Bluetooth Low Energy Profile .....	9
2.1.3.	Unified Configuration Interface .....	10
2.1.4.	Other data models .....	11
2.2.	Exploration of existing data model management tools .....	12
2.2.1.	Open Mobile Alliance for LWM2M .....	12
2.2.2.	IPSO Alliance .....	14
2.2.3.	Eclipse Vorto project .....	15
2.2.4.	BLE-IPSO project .....	16
2.3.	Summary of various data models .....	17
3.	WEB BASED COLLABORATION PLATFORMS .....	18
3.1.	Content management system (CMS) .....	18
3.2.	Version control system (VCS) .....	18
3.3.	Content sharing web platforms .....	19
3.3.1.	Custom built platforms .....	19
3.3.2.	Online CMS platforms .....	20
3.3.3.	Online VCS platforms .....	20
3.4.	Comparison of various web platforms .....	22
4.	UX DESIGN AND EVALUATION METHODS .....	25
4.1.	User experience .....	25
4.2.	User centered design .....	26
4.3.	User interface design .....	28
4.4.	UX design methods .....	30
4.4.1.	High-fidelity prototyping .....	31
4.5.	UX evaluation methods .....	32
4.5.1.	Task based usability testing and interview .....	32
4.5.2.	Heuristic evaluation .....	34
4.5.3.	Data gathering and analyzing .....	35
5.	DESIGN AND EVALUATION OF USER INTERFACE FOR IOT META MODEL REPOSITORY .....	37
5.1.	Selected platform for the repository .....	37
5.2.	Overall process of research work .....	38
5.2.1.	UCD process .....	38

5.2.2.	Two rounds of user tests .....	40
5.3.	Design of the first version of the UI.....	42
5.4.	Iterative evaluation .....	45
5.4.1.	First user tests.....	46
5.4.2.	Second user tests .....	49
5.5.	The final design.....	53
5.5.1.	Re-design of UI.....	53
5.5.2.	New features .....	57
5.6.	Summary .....	60
6.	DISCUSSION .....	61
6.1.	Reflections of the research questions .....	61
6.2.	Limitations .....	62
6.3.	Future research .....	63
7.	CONCLUSION.....	65
	REFERENCES.....	66

APPENDIX A: CONSENT FORM

APPENDIX B: USER BACKGROUND QUESTIONNAIRE

APPENDIX C: USER SATISFACTION QUESTIONNAIRE

APPENDIX D: TASKS IN FIRST USER TESTS

APPENDIX E: TASKS IN SECOND USER TESTS

APPENDIX F: QUESTIONS IN INTERVIEWS

## LIST OF FIGURES

<i>Figure 1-1: The overview architecture of LWM2M</i> .....	1
<i>Figure 2-1: The structure of a LWM2M data model</i> .....	8
<i>Figure 2-2: The example of IPSO temperature object</i> .....	9
<i>Figure 2-3: The structure of a single BLE profile</i> .....	10
<i>Figure 2-4: The configuration file of UCI model</i> .....	11
<i>Figure 2-5: The example of Demo Router YANG model</i> .....	12
<i>Figure 2-6: LWM2M online editor of data models</i> .....	13
<i>Figure 2-7: LWM2M objects in object and resource registry</i> .....	13
<i>Figure 2-8: The homepage of the public IPSO Repository</i> .....	14
<i>Figure 2-9: The web interfaces of Vorto Repository</i> .....	15
<i>Figure 2-10: A list of BIPSO Objects</i> .....	16
<i>Figure 4-1: Activities of user-centered design (ISO 13407)</i> .....	27
<i>Figure 5-1: The overall process of research work</i> .....	39
<i>Figure 5-2: High-fidelity prototypes of desktop and mobile</i> .....	39
<i>Figure 5-3: The process of a single user test</i> .....	41
<i>Figure 5-4: The overall structure of the user interface design</i> .....	43
<i>Figure 5-5: The homepage of the mockups</i> .....	43
<i>Figure 5-6: The model list page</i> .....	44
<i>Figure 5-7: The page of multiple data models for gateway system object</i> .....	45
<i>Figure 5-8: The results of grades by same participants in two rounds user tests</i> .....	52
<i>Figure 5-9: The new design of UI structure</i> .....	53
<i>Figure 5-10: The design of new navigation bar</i> .....	54
<i>Figure 5-11: The design of homepage</i> .....	55
<i>Figure 5-12: The design of model list with the pagination</i> .....	56
<i>Figure 5-13: The page of multiples data models for one object</i> .....	57
<i>Figure 5-14: The design of search bar, which above the model list</i> .....	57
<i>Figure 5-15: The design of download function of xml file</i> .....	58
<i>Figure 5-16: The page of contribute</i> .....	59

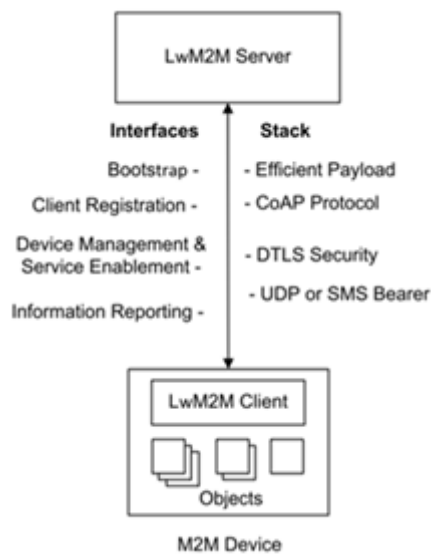
## LIST OF SYMBOLS AND ABBREVIATIONS

BLE	Bluetooth Low Energy
CI	Continuous Integration
CoAP	Constrained Application Protocol
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IP	Internet Protocol
IPSO Alliance	IP Smart Object Alliance
ISO	International organization for standardization
LWM2M	Lightweight machine to machine
OCF	Open connectivity foundation
OMA	Open Mobile Alliance
REST	Representational State Transfer
RAML	RESTful API Modeling Language
UCI	Unified Configuration Interface
UCD	User-Centered Design
UI	User interface
UX	User experience

# 1. INTRODUCTION

Internet of Things (IoT) refers to the vast network of interconnected devices. It includes smart devices such as smart mobile phones, tablets as well as wearable devices, and other physical objects with sensors. Indeed, IoT intends to get all real world “objects” connected to the network, and then to interconnect the real world with the virtual world. In doing so, the objects and their states in the real world will be reflected in the virtual world (Giner et al., 2008). Specifically, diverse devices interconnect to gather and exchange data, such as sensors, actuators, gateways as well as other IoT devices. All these physical devices can be defined in various formats according to different protocols and standards.

To help developers manipulate IoT devices, data models are defined to describe smart objects for device management. More precisely speaking, data models could identify managed smart objects competently by comprising implementation-specific and protocol-specific details. They generally indicate how developers can manage and maintain IoT devices. Due to the diversity of data models, the practices of device management vary widely. For example, Lightweight Machine to Machine (LWM2M) is one example of a device management protocol, which is a standard from the Open Mobile Alliance (OMA). LWM2M specifications defines how the LWM2M client communicate with LWM2M server in the application layer. The main target of LWM2M specification is for device management and service enablement for the M2M devices (Elgazzar, 2015). Figure 1-1 below displays the architecture of LWM2M.



*Figure 1-1: The overview architecture of LWM2M*



Overall, LWM2M provides a simple and reusable object model with a set of interfaces for managing constrained devices, covering bootstrap, information reporting, client registration, service enablement and device management (Tracey & Sreenan, 2017). Through these four logical interfaces, LWMW enables the devices management more effectively, in particular, for constrained devices. Moreover, LWM2M has its own data models to define devices and applies them in device management.

Apart from LWM2M, other approaches of device management exist based on different concepts and data models. While some of data models come from Standards Developing Organizations (SDOs), others may be non-standard or proprietary models. Currently, all these data models vary in terms of formats and locations. There is no unified platform available for sharing and presenting these data models. That leads to an extremely challenging task for both developers and organizations to search or organize these data models for IoT devices. Even one device can also have multiple data models. Therefore, the management of these data models is a significant component that makes IoT work properly and efficiently. Since the manipulation of IoT devices is mainly based on understanding these data models, through sharing and organizing data models via a collaboration platform, IoT devices could be discovered and managed effectively. Therefore, sharing and presenting multiple data models in a more appealing and organized way can contribute to a better device management. On the other hand, it undoubtedly facilitates the work of both individual developers and organizations.

## 1.1. Terminology

The relevant terms and concepts are presented below to avoid the confusion; some of terms are specifically defined for this thesis work instead of using general definition. All terms and concepts were used and concerned throughout the research work.

*Data model* – is generally used to define IoT devices with more details in terms of implementation and protocol. A data model contains the information of how data is represented and operated. Data models vary widely due to different SDOs. There can be multiple data models for one device and each data model also supports versioning.

*Information model (IM)* – is the abstract level of defining IoT devices. It manages objects at a conceptual level, without any details of implementation or protocols. Normally there is confusion between data model and information model. The degree of specificity (or detail) of the abstractions of a IM depends on the modeling needs of its designers. To make an explicit overall design, an IM is supposed to hide all protocol and implementation details (Pras & Schoenwaelder, 2003).

*Device management* – means the management of IoT devices, including manipulation, maintenance and operation of devices. Even the communication between multiple devices is involved.

*Data model management* – refers to the management of various data models with relevant files and versions. Mainly for sharing and organizing diverse data models. It also includes presenting data models in a more interactive way. Moreover, data model management can contribute to data model discovery and distribution.

*Collaboration platform* – serves as a platform specifically for managing various data models. It allows collaborative work of sharing and organizing data models. The platform develops with time, with more contributions of data models from different parties.

*Repository* – a collaboration platform used for data model management. It also stores all relevant resources of multiple data models, including files, code snippets and different versions. It is a proposal of solution of data model management in this thesis.

## **1.2. Motivation**

With the advent of increasingly interconnected devices, the IoT domain has begun to get congested with heterogeneous services and applications by using various data models and communication protocols (Bandyopadhyay & Bhattacharyya, 2013). A very large number of nodes are recording, transmitting, storing data and similarly large numbers of nodes are retrieving, analyzing and consuming the information with intermediate nodes transforming and integrating data to information (James et al., 2009). Currently, to promote an efficient and competent IoT device management, various organizations such as the All-Seen alliance, OpenIoT alliance and IPSO Alliance have developed different standardization of communication protocols to provide interoperability between diverse vendors' silos. These approaches will be discussed in the following chapter to provide an explicit overview of current practices.

Generally, data models help identify managed smart objects and IoT devices. Typically, through comprising implementation-specific and protocol-specific details, they could demonstrate the way to operate devices. In fact, there are already several alternatives to manage data models based on different standards. However, the sharing of data models remains in a way of inefficiency and obscure. This is mainly due to the fact that data models are presented and systematized differently according to the preferences of various organizations. Therefore, it costs much effort for developers when considering that one device may have multiple different data models. Moreover, some data models can be distributed in different online ways, for example, in the form of a rough list or documentation. By contrast, others can be presented via websites with poor user interfaces.

Currently, the research related to sharing and organizing multiple data models is quite restricted, especially for non-standard or proprietary data models. Those are usually from either developers or organizations who develop their own IoT devices. For the majority of literature work, the aim is to study data models from the various standards perspectives. They rarely attempt to explore the solution of managing multiple data models. Since there

is no well-organized method of gathering and managing data models of devices, the work in device management become more time-consuming and inefficient. This provides the motivation of undertaking these challenges by developing engaging and usable solutions. Based on current practices, a design solution of organizing and sharing multiple data models more efficiently is highly demanded.

User interfaces are an important part of the design solution of the managing data model. They are supposed to be designed properly to help in use of the system. With a usable user interface, users can search and share data models easily and efficiently. In addition to functionality, a product with appealing visual design can contribute to a positive attitude of users. It promotes the motivation of continued use of the system. Meanwhile, a usable and attractive user interface enables users to achieve tasks quite successfully and enjoy a delightful user experience.

### **1.3. Research goal and questions**

Based on the barriers and weaknesses in current practices of sharing and discovering data models, the goal of this thesis is to implement a repository to create a better solution of data model management. More precisely, the research primarily focuses on two questions as follows:

1. How to share and distribute standard and non-standard data models for IoT devices?
2. How to design a usable user interface for data model repository for developers, contributors and enterprise users?

The first question will be approached by starting with the exploration of the related work, for example, investigation of the current data model management tools. Afterwards, a suitable collaboration platform for repository is selected based on the analysis of existing platforms. In order to share and distribute the data models properly, there still some requirements based on that. Primarily, it should contain multiple data models, and make it possible for sharing new data models. Meanwhile, the platform is supposed to encourage collaboration and distribution. Therefore, the ultimate solution is to implement a functional repository, which can benefit three user groups as follows:

- Developers, mainly those who have demands for searching and browsing data models for implementation work.
- Contributors, who need to share or present the data models for their own devices. A contributor can be either an individual developer or a company (organization) who wants to share data models.
- Enterprise users, who intend to have their own platform to present and maintain data models. They are mostly companies and organization who want a customized platform for their own use instead of just sharing data models.

Moreover, the repository can be extended for machine-to-machine (M2M) communication. In the LWM2M context, M2M communication can allow a LWM2M server to query the repository for unknown data models. However, this part of work is out of scope of this thesis and will be addressed in future.

After achieving functional requirements, the data model repository is expected to provide good user experience for all end users. More concretely, developers should be able to effortlessly search and browse data models. As for contributors, they can share data models easily and reliably. Additionally, enterprise users can simply take the repository for their own business without exhaustive maintenance work.

Based on the diversity of user groups, the repository should be simple and easy to learn in a general way. The overall design of user interface is supposed to be clean and elegant. In addition, the interaction between user and the repository is supposed to be straightforward. That can help users pay attention to the information itself without any distrusting from improper user interface.

#### **1.4. Structure of the Thesis**

This thesis work is organized in seven chapters. It begins with the background of IoT and provides the terminology of relevant concepts. Then, it explains the existing problems related to devices management, which indicates research questions and objectives.

Chapter 2 explores the related work of data management in IoT world. More precisely, it is divided into two subsections. Firstly, various data models are introduced, and then the exploration work of existing platforms for data models is described with a brief introduction of each platform.

Chapter 3 covers the description of various web-based collaboration platforms. By introducing different technologies for collaboration platforms, it discusses several examples of web platforms for content sharing and presents a comparison among different web-based collaboration platforms.

Chapter 4 goes through the theoretical background of research work by introducing the fundamental concepts and main methodologies that are applied in this thesis work. In addition, it describes the principles and process related to these methods and the reasons of use is clarified.

Chapter 5 demonstrates the design process of the meta-model repository and it contains four subsections. Initially, an overall process and phase of design work is presented. It contains both user-centered design process and two rounds of user tests. Then the first version of the user interface is introduced. The third part reported the related iterative evaluation based on two rounds usability testing. Finally, it provides the final design of the repository in terms of redesign part and new features.

Chapter 6 includes the discussion session, mainly about reflections of the research questions and limitations of the thesis work. Additionally, it discusses the future research. A conclusion is given in the last Chapter 7.

## 2. RELATED WORK OF IOT DATA MODELS

This chapter mainly describes the related work of exploring current data models. It consists of the introduction of various data models and relevant management tools of data models. The primary emphasis is to figure out different data formats and how they are being created and presented currently. By discovering existing approaches, more insights and ideas related to the repository can be generated.

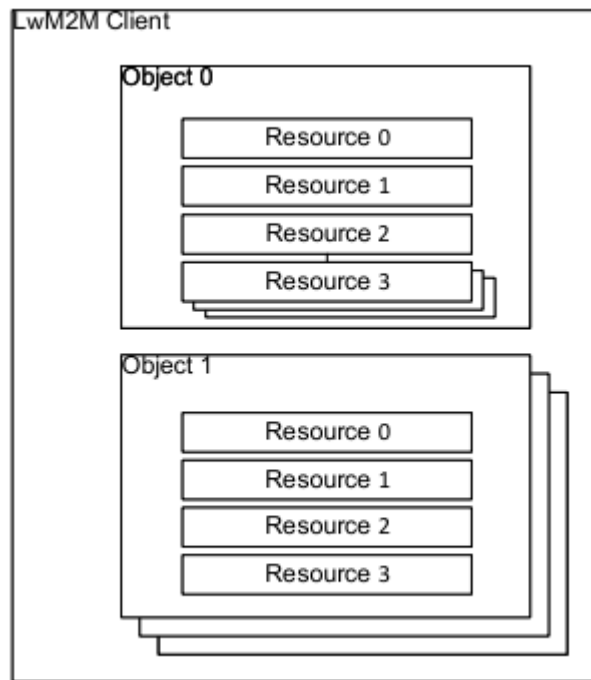
### 2.1. Introduction to IoT data models

Due to different technical and commercial reasons, several SDOs define their own device management methods and data models for use in IoT. Each SDO's data models are distributed differently. Meanwhile, there is a huge chance that one IoT device may have multiple data models, to cater to each SDO. Several data models of different SDOs are introduced as follows.

#### 2.1.1. LWM2M and IPSO Objects

LWM2M is a standard from the Open Mobile Alliance; it aims to develop an efficient deployable specification of client-server to provide machine to machine (M2M) services and device management (Rao et al., 2015). Based on four logical interfaces which were shown in Figure 1-1, LWM2M provides an efficient device management. The communication model within the LWM2M relies on the COAP methods, mainly by GET, PUT, POST and DELETE. That significantly makes LWM2M an easy and efficient management of constrained devices.

Apart from device management, LWM2M defines a data model in which data sources are defined by resources. A resource can be seen as an atomic piece of information that contains multiple instances. A resource can be operated such as written, read or executed. Multiple resources are logically grouped into one single Object and a LWM2M Client can have one or multiple Object Instances (Robles & Jokela, 2015). Figure 2-1 displays the structure of the LWM2M data object.



**Figure 2-1: The structure of a LWM2M data model**

As shown, the object model is a collection of resources and all these resources are used to present the objects, and all the objects can define the IoT device. Moreover, the theory behind the device management concept is that the LWM2M server can manipulate different clients through operating a series of resources within objects.

Another SDO is the IPSO Alliance. It is an organization for promoting the use of the Internet Protocol (IP) technology for communications between smart objects (<http://www.ipso-alliance.org/>). In general, IPSO focuses on the definitions and interoperability of smart objects. More concretely, the IPSO objects are basically based on LWM2M, following the LWM2M data model format. The object model is associated with a uniform resource identifier (URI) to be accessed; it also contains metadata which can be operated in implementation work. According to Jimenez, Koster and Tschofenig (2016), IPSO Smart Objects provide the same design pattern as LWM2M, which is an object model, to enhance a high-level interoperability between different smart object devices. More concretely, the data model for IPSO Smart Objects consists of four parts:

- Object Representation
- Data Types
- Operations
- Content Formats

IPSO Smart Objects cover a range of entities, including basic sensors and actuators. These basic objects are represented using a simple common data model and resource template in LWM2M (Tracey & Sreenan, 2017). Moreover, there are possibilities for IPSO basic

data models to form composite objects with a more usable design. Figure 2-2 below displays an example of IPSO Temperature object.

**Object Info:**

Object	Object ID	Object URN	Multiple Instances?	Description
IPSO Temperature	3303	urn:oma:lwm2m:ext:3303	Yes	Temperature sensor, example units = Cel

**Resources:**

Resource Name	Resource ID	Access Type	Multiple Instances?	Mandatory	Type	Range or Enumeration	Units	Descriptions
Sensor Value	5700	R	No	Mandatory	Float			Last or Current Measured Value from the Sensor
Units	5701	R	No	Optional	String			Measurement Units Definition e.g. "Cel" for Temperature in Celsius.
Min Measured Value	5601	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The minimum value measured by the sensor since power ON or reset
Max Measured Value	5602	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The maximum value measured by the sensor since power ON or reset
Min Range Value	5603	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The minimum value that can be measured by the sensor
Max Range Value	5604	R	No	Optional	Float	Same as Measured Value	Same as Measured Value	The maximum value that can be measured by the sensor
Reset Min and Max Measured Values	5605	E	No	Optional	Opaque			Reset the Min and Max Measured Values to Current Value

**Figure 2-2: The example of IPSO temperature object**

Each IPSO object model has a unique ID and object uniform resource name (URN) as the LWM2M data model. The general description of the data model is provided, and the resources are used to define the object specifically through various properties. Apart from presenting in the form of document, each ISPO object has the XML file, which contains both the object information and resources fields.

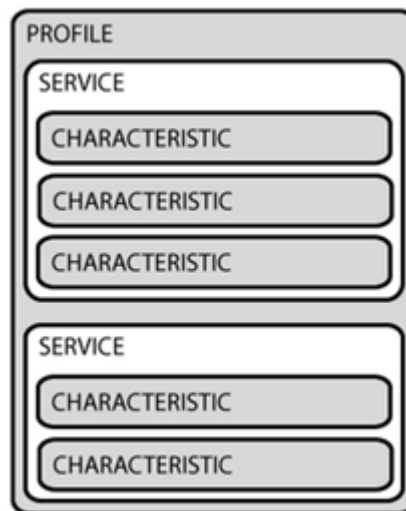
### 2.1.2. Bluetooth Low Energy Profile

Bluetooth wireless technology is an open specification that enables short-range and low-power wireless connections (Yu, Xu & Li, 2012). Within the Bluetooth wireless technology, a low energy feature is developed to serve for Bluetooth Smart devices. Bluetooth Low Energy (BLE) is an emerging wireless technology, it developed by the Bluetooth Special Interest Group (Bluetooth SIG) for short-range communication (Gomez, Oller &



Paradells, 2012). The BLE standard aims to transmit small packets of data between devices with saving energy.

In BLE system, a BLE profile contributes to the communication among different BLE devices. It enables the transfer of data based on services and characteristics. More specifically, the BLE profile includes a standard data format constituted by the Bluetooth SIG. It mainly consists of many types of different profiles with various relevant functions (Profile, 2016). The structure of a single BLE profile is displayed in Figure 2-3.



*Figure 2-3: The structure of a single BLE profile*

Generally, a single profile can have multiple services while one service may include several characteristics. According to Bluetooth 5.0 specification, a profile can be seen as a virtual collection of services. Services are mainly grouping of real data associated with the operation for specific functionalities and features. Every service contains several characteristics with their descriptors.

Each service is assigned with a unique identifier, called a UUID. Within services, each characteristic also includes a value with related information of configuration. Moreover, it contains properties and operations to manipulate the value. For example, the value can be read, write, indicate and notify. The BLE profile defines the way of the operation and communication between the devices.

### **2.1.3. Unified Configuration Interface**

The Unified Configuration Interface (UCI) aims to centralize the configuration of a Linux distribution for embedded devices and access points, called OpenWRT. It serves as OpenWRT's main configuration user interface for the most important settings related to sthe

system. Typically, these settings are crucial for proper device functioning. Examples including logging functionality, wireless settings, network interface configuration, and remote access configuration (<https://openwrt.org/>). OpenWRT provides the general information of a UCI file and contains a list of UCI files for users. Figure 2-4 shows the example of UCI file.

This is the default configuration for this section:

```
config 'httpd'
    option 'port' '80'
    option 'home' '/www'
```

The httpd section contains these settings:

Name	Type	Required	Default	Description
c_file	string	no	/etc/httpd.conf	Path to configuration file.
home	string	no	/www	Path to the document root directory.
port	integer	no	80	Port number the web server should listen on.
realm	string	no	hostname	Authentication realm to be presented to the user. Defaults to system.@system[0].hostname

*Figure 2-4: The configuration file of UCI model*

Generally, a UCI model contains the configured file and rules of setting of system. The information of configuration is basically presented as plain text table. The UCI system makes the configuration become easy and straightforward.

#### 2.1.4. Other data models

Apart from the data models discussed above, more standardized data models exist in IoT device management. Another two well-known data models are YANG and RESTful API Modeling Language (RAML) models from Open connectivity foundation (OCF).

YANG is a language for data modeling, which is generally used to model configuration and state data that manipulated by the Network Configuration Protocol (NETCONF). A YANG module mainly defines a hierarchy structure of data. It can be used for NETCONF-based operations, includes state data, configuration, Remote Procedure Calls (RPCs), and notifications (Bjorklund, 2010). Figure 2-5 shows an example of Router YANG model.

```

- Demo Router YANG Model

module router {
  yang-version 1;
  namespace "urn:sdnhub:odl:tutorial:router";
  prefix router;

  description "Router configuration";

  revision "2015-07-28" {
    description "Initial version.";
  }

  list interfaces {
    key id;
    leaf id {
      type string;
    }
    leaf ip-address {
      type string;
    }
  }
  container router {
    list ospf {
      key process-id;
      leaf process-id {
        type uint32;
      }
    }
    list networks {
      key subnet-ip;
      leaf subnet-ip {
        type string;
      }
      leaf area-id {
        type uint32;
      }
    }
  }
  list bgp {
    key as-number;
    leaf as-number {
      type uint32;
    }
    leaf router-id {
      type string;
    }
    list neighbors {
      key as-number;
      leaf as-number {
        type uint32;
      }
      leaf peer-ip {
        type string;
      }
    }
  }
}

```

**Figure 2-5: The example of Demo Router YANG model**

In general, a YANG module contains several types of content. It consists of Header information, Type definitions, Imports & Includes, Action & Notification declarations and Configuration & Operational data declarations. Based on these attributes, it provides another possibility for data modeling and sharing.

The other one is RAML, which is used by OCF. To manage IoT devices, OCF has resource specifications, and it mainly uses RAML as a standard language for the APIs and JavaScript Object Notation (JSON) schemes as the representations of resources. Alternatively, a RAML data model induces the APIs of operations of resources and can be in different file format such as eXtensible Markup Language (XML), JSON.

Overall, a variety of data models is created for IoT device management. They have different data formats and the way of use vary. In addition to standardized data models, there are also non-standards data models from different companies and enterprises.

## 2.2. Exploration of existing data model management tools

Based on various data formats and organizations, different data model management tools are provided to facilitate the work of device management. This section mainly presents existing management of data models from several organizations.

### 2.2.1. Open Mobile Alliance for LWM2M

To support LWM2M and manage data models more effectively, OMA provides other possibility for developers and enterprise users. In addition to the definition of protocol and object models, OMA allows developers to register their own LWM2M data models

by providing an online editor. The XML schema used by LWM2M Editor is included to present the example of XML of data model. The user interface of online editor is shown in Figure 2-6.

**LWM2M Object Name** [Edit]

Name

**Description** [Edit]

**Object definition**

Operations	Name	Object ID	ObjectVersion	LWM2MVersion
[Edit]	Name	0		
	Object URN		Instances	Mandatory
			Multiple	Optional

**Resource definitions**

Operations	ID	Name	Operations	Instances	Mandatory	Type	Range or Enumeration	Units	Description

[Add new resource]

[Edit Additional Definition]

**Figure 2-6: LWM2M online editor of data models**

Furthermore, OMA also provides object and resource registry, which shares plenty of LWM2M data models of different objects for IoT use cases. Figure 2-7 displays the LWM2M object and resource registry.

**LwM2M Objects**

**oma-label Objects Produced by OMA**

URN: urn:oma:lwm2m:oma-objectID:version | range (0 - 1,023)

URN / Version	ObjectID / xml	LwM2M Editor	Object Name	Technical Specification	Eclipse Vorto Link	Description
urn:oma:lwm2m:oma:0	0	0	LWM2M Security			It provides the keying material of a LWM2M Client appropriate to access a specified LWM2M Server. One Object Instance SHOULD address a LWM2M Bootstrap <a href="#">---more---</a>
urn:oma:lwm2m:oma:1	1	1	LWM2M Server			It provides the data related to a LWM2M Server. A Bootstrap Server has no such an Object Instance associated to it <a href="#">---more---</a>
urn:oma:lwm2m:oma:2	2	2	Access Control			It is used to check whether the LWM2M Server has access right for performing a operation.
urn:oma:lwm2m:oma:3	3	3	Device			It provides a range of device related information which can be queried by the LWM2M Server, and a device reboot and factory reset function. <a href="#">---more---</a>

**Figure 2-7: LWM2M objects in object and resource registry**

Concretely speaking, each data model is registered by Object ID and it has a unique URN. For one object model, there could be multiple versions. The registry can be regarded as a database containing different object models, mainly for developers or enterprises who use LWM2M in device management.

Generally, all LWM2M data models are shared and constructed with the OMA online editor and registry. The object registry is quite informative for developers who are searching for the data models for implementation work. More resources such as technical specifications, API or other documentation could be found there as well. Moreover, some open source LWM2M projects exist. For instance, Eclipse Leshan provides a Java-based LWM2M server and client while Eclipse Wakaama provides them in C (<https://www.eclipse.org/leshan/>).

## 2.2.2. IPSO Alliance

The IPSO Alliance also provides its own management of IPSO object. It develops the public IPSO Repository for object models. It uses GitHub, and the main page introduces object representation and validation. It also provides other document and support for developers. Figure 2-8 displays the user interface of the homepage of the repository.

**IPSO Smart Objects**

[View the Project on GitHub](#)  
IPSO-Alliance/pub

Download ZIP File    Download TAR Ball    View On GitHub

**Welcome to the public IPSO Repository.**

Here you can find the XML templates of the [IPSO Smart Object Registry](#) with the current object set. The registry is intended for developers that are building products based on IPSO Objects, it is not intended to be used at runtime by applications.

Some of the objects are generic in nature, such as voltage, altitude or percentage, while others are more specialized like the Color Object or the Gyrometer Object. Actuators and Controllers are also defined such as timer or buzzer and Joystick and Level. All of these objects were found to be necessary on a variety of use case domains.

**Object Representation**

Objects and resources are implicitly mapped into the URI path hierarchy by following the OMA LWM2M object model, in which each URI path component sequentially represents the Object Type ID, the Object Instance ID and the Resource Type ID. More precisely the structure consists of three unsigned 16-bit integers separated by the character '/':

```
<object ID>/<object instance ID>/<resource ID>
```

For example, a temperature sensor example URI would be 3303/0/5700:

```
3300 -> Temperature Sensor
0 -> instance 0 of a Temperature Sensor
5700 -> resource having the current value or recent reading
```

**Object Validation**

**Figure 2-8: The homepage of the public IPSO Repository**

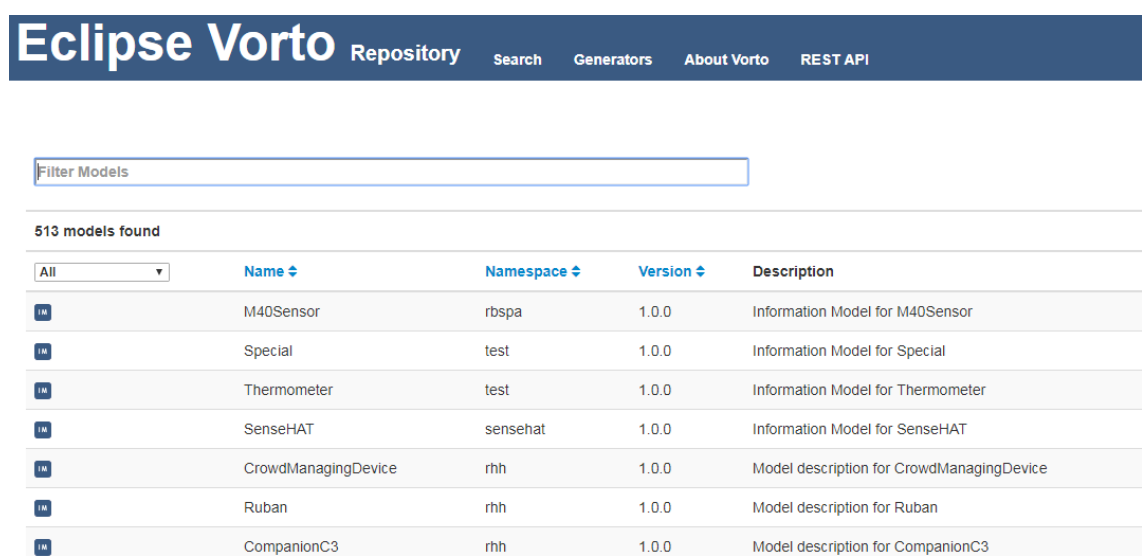
Primarily, the IPSO Smart Object Registry is intended for developers that are building products based on IPSO Objects (<https://ipso-alliance.github.io/pub/>). It is a repository on GitHub that contains a list of XML files of smart objects. Each file includes a definition of the object models and clarifies the resources elements. The registry contains an introduction of the IPSO smart objects and an XML file template of a data object.

### 2.2.3. Eclipse Vorto project

Eclipse Vorto (<http://www.eclipse.org/vorto/>) is an open source tool that helps users to create and manage abstract device descriptions, so called information models. In general, Eclipse Vorto is an advanced open source project. The foremost contribution of Eclipse Vorto is to define the capabilities of smart devices in entirety – by using information model. All these information models are stored in a central repository, by which different vendors can build reusable solutions. The generators can create automatic code which can ensure the model is in appropriate code representation and can be reuse and interoperability (Tayur & R, 2017). Through compiling and manage information models, it aims to achieve interoperability for connected IoT devices, especially for those use different platforms and technologies.

The purpose of Vorto is to serve for different target groups in IoT development scenarios, including various devices manufactures, platform operators, and different communion protocols as well as standardization organizations. Eclipse Vorto offers the value in terms of four main features: Vorto Toolset, Vorto Code Generators, Vorto repository and meta model. All these functions help in IoT development work. They also support collaboration among developers and organizations.

Unlike LWM2M and IPSO, what is special about the Eclipse Vorto is it focuses on information models of devices. In Eclipse Vorto, information models are mainly managed and shared in through the Vorto Repository. Generally, users can upload models, search the model they need or reuse the information models based on requirements. The repository is hosted by Eclipse Vorto, and users can complete most tasks through the web interfaces. Figure 2-9 shows the web interfaces of Vorto repository. There is a list of information models is provided to be managed and reviewed.



The screenshot shows the Eclipse Vorto Repository web interface. At the top, there is a navigation bar with the title 'Eclipse Vorto Repository' and links for 'Search', 'Generators', 'About Vorto', and 'REST API'. Below the navigation bar is a search input field labeled 'Filter Models'. Underneath, it indicates '513 models found'. A table lists several information models with columns for 'Name', 'Namespace', 'Version', and 'Description'. Each row also has a small 'IM' icon on the left.

	Name	Namespace	Version	Description
IM	M40Sensor	rbspa	1.0.0	Information Model for M40Sensor
IM	Special	test	1.0.0	Information Model for Special
IM	Thermometer	test	1.0.0	Information Model for Thermometer
IM	SenseHAT	sensehat	1.0.0	Information Model for SenseHAT
IM	CrowdManagingDevice	rh	1.0.0	Model description for CrowdManagingDevice
IM	Ruban	rh	1.0.0	Model description for Ruban
IM	CompanionC3	rh	1.0.0	Model description for CompanionC3

**Figure 2-9: The web interfaces of Vorto Repository**

For organizations or device manufactures, they can provide their information models for developers to browse or download. Moreover, it allows developers to create platform-specific source code through the code generator. This code generator helps developers integrate IoT devices into different platforms. That significantly reducing the amount of development work and ease the implementation work of IoT players. In addition to code, users are able to discuss about the information models by adding comments to a model. Nevertheless, the information model in the repository also contains specific details of devices. Therefore, it is tough to strictly separate the information model and data model in real IoT use case.

Overall, Eclipse Vorto provides a practical solution of managing and sharing of information models. Through Vorto, the IoT devices from different manufactures can be defined in information models, and the Vorto repository allows an effective data sharing and reuse. Additionally, its code generators have prompted a solution to developers of integrating information models with any other devices and platforms.

#### 2.2.4. BLE-IPSO project

BLE-IPSO (BIPSO) is another project aims to define BLE models as IPSO objects to solve this problem of consistency and compatibility for BLE applications. It defines a set of BLE Characteristics that follows the IPSO Smart Object Guideline for developers to build their applications with a unified data model (<https://bluetooth.github.io/bipso/#/>).

BIPSO consists of webpages hosted on the GitHub with clean user interfaces. It contains the information, motivation and introduction of the mapping of data models. In BIPSO, with a well-defined Characteristic Value, IPSO Smart Object can be mapped to a BLE Characteristic. More specifically, the BLE data model has its own properties and format to define the objects. The characteristics work as resources to define the smart object in detail. Therefore, the Characteristic Value is used within mapping process; since it is mostly a piece of data presents resources in an IPSO Object. Figure 2-10 illustrates the components of the BIPSO model.

Object ID	Char. UUID	Possible Fields in Char. Value	Description
dIn (3200)	0xccc00	id(uint8), flags(uint8), dInState(boolean)[], counter(uint8), dInPolarity(boolean), debouncePeriod(uint16), edgeSelection(uint8), counterReset(buffer), appType(string), sensorType(string) ]	digital input
dOut (3201)	0xccc01	id(uint8), flags(uint8), dOutState(boolean)[], dOutPolarity(boolean), appType(string) ]	digital output
aIn (3202)	0xccc02	id(uint8), flags(uint8), aInCurrValue(float)[], minMeaValue(float), maxMeaValue(float), minRangeValue(float), maxRangeValue(float), resetMinMaxMeaValues(buffer), appType(string), sensorType(string) ]	analogue input
aOut (3203)	0xccc03	id(uint8), flags(uint8), aOutCurrValue(float)[], minRangeValue(float), maxRangeValue(float), appType(string) ]	analogue output

*Figure 2-10: A list of BIPSO Objects*

Apart from the definition of object models, BIPSO provides an online tool, which mainly helps developers generate a snippet of C code in terms of defining the BLE characteristics. By choosing smart objects through web interface, the sample code that contains definition of models will be provided accordingly. Therefore, developers could use those code snippets in the implementation work of software applications.

Overall, BIPSO is serving as a mapping tool to generate BLE model in IPSO format. However, there are not many data models available in the repository. Instead, the purpose of the repository aims to indicate the concept of BIPSO, since it focuses on introducing the technique side of the mapping rather than sharing data models. Moreover, the tool primarily facilitates the developers' work in terms of implementation.

### **2.3. Summary of various data models**

This chapter indicated some popular ways for IoT device management. It introduced various formats of data models in different organizations. Moreover, each organization has its own solution to share and manage data models. In fact, there are more existing data model management approaches which were not discussed, such as non-standard data models. They all have different data formats, and the platforms of sharing data models vary.

Overall, there are diverse approaches in defining and manipulate IoT devices. Consequently, this variety of presenting and sharing data models causes inconvenience for both individual users (such as developers, contributors) and organizations. Although all standardized data models can be accessed quite effortlessly, it leads to frustration in the case of dealing with non-standard data models.

To solve this problem, the ultimate goal of this thesis work is to develop a meta model repository, that intends to serve as a platform for presenting and sharing multiple data models for IoT device management. Moreover, the repository aims to allow developers browse data models in a more interactive way. At the same time, it is able to help enterprises (organizations) to manage data models on their own platforms and demonstrate them publicly in a more appealing way.



## **3. WEB BASED COLLABORATION PLATFORMS**

This chapter discusses various web-based collaboration platforms. It starts with the introduction of existing technologies of collaboration in software development. Moreover, more examples and solutions of web platforms for content sharing are described. In the last part, it presents a comparison among these different web-based collaboration platforms.

### **3.1. Content management system (CMS)**

When it comes to managing digital content, content management system (CMS) is the prevalent technology for setting up to generate and customize specific information. A CMS can be also seen as a software for creating digital content. Specifically, it is a platform hosting on a web server. Generally, it allows users to manage the digital content including pages, online articles and blogs effortlessly. Users can update the content efficiently. Also, CMS services enables multiple editors to maintain online content without any conflicting occurs (Goroshko, 2014). Some prominent advantages have resulted in the widespread use of CMS. Firstly, it allows users to create a powerful website without a high level of programming skills, and it can manage and organize different digital content based on the need of user. Moreover, the design work of the website, including layout, appearance and even structure of the site can be easily modified and updated. Often, the types of content management system vary widely. Various CMSs have different emphases and a user can choose a suitable CMS according to the purpose of use.

### **3.2. Version control system (VCS)**

A Version control system (VCS) is typically used to track changes and updating in software development. It enables the acceleration and simplification of the software development process and enables new workflows. VCSs keep track of files and their history and have a model for concurrent access (Otte, 2009). There are two types of VCSs: centralized and distributed. The centralized VCS has one central repository, which is basically used to track files and history. Generally, a VCS allows developers to constant track the iterative changes to the code. As a result, developers can experiment with new ideas and changes, while they always have the option to revert to a specific past version in the development (Blischak, Davenport & Wilson, 2016). Meanwhile, it also records the changes and commits from other developers in the project. By using a VCS, all the changes of the code can be tracked and saved. That significantly enhances group collaboration.

### 3.3. Content sharing web platforms

This section mainly introduces three different solutions of web platforms for digital content sharing. It presents the general concepts of different web-based platforms. Moreover, some relevant examples of various platforms are given with the description of their features and usages.

#### 3.3.1. Custom built platforms

Generally, a web-based platform can be installed from scratch and customized according to different purposes. Companies and organizations can develop their own platform to support business or for enterprise use. To build a successful and stable web-based platform, several technical components should be considered. Firstly, a server installation is needed which contains a suitable operating system. Then, a software implementing the functionality of a web server is required. For example, many distributions of Linux are used for server platforms, while one widely used web server is Apache which uses open source code. Apart from the server, some programming languages can be used in building a web platform such as PHP and JavaScript. PHP is a server-side language for web development and PHP code is able to combine with other web content management systems. JavaScript can help create interactive web pages. It is also a high-level programming language and work for web development with Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). After determining the server and language framework, developers can implement the platform according to customized content and features.

Furthermore, rather than develop the entire platform by programming, CMSs can be used to create a platform more easily and efficiently. For example, Drupal is an open source platform written in PHP for building flexible websites. It provides thousands of free themes (templates) or even paid product with professional design. Additionally, users are able to expand it with different content and features according to needs. It also has a large community that provides developers with support, testing and documentation related to the website. Likewise, Joomla is another popular CMS for making highly interactive web sites speedily such as online media, portals, blogs and E-commerce services and applications (Patel et al., 2011). These CMSs can help build a web platform in a short period. Another possibility is using a VCS. Currently, various open source VCS existing to facilitate collaborations in software development. For instance, Subversion (SVN; <http://subversion.tigris.org>) is a widely used open source version-control system that facilitates distributed file sharing. SVN resorts a centralized architecture. It is mainly a single central server and hosts all metadata of projects (Lanubile et al., 2010).

As for hosting, it can be decided by the requirements of platform. One approach is the developers can build their own hardware and network based on their technical skills. Oth-

erwise, a Cloud hosting service can be utilized, such as Amazon Web Service or Microsoft Azure. However, a customer-built platform takes time and costs, which depends on the selected server, programming language and the way of hosting. Moreover, it requires some technical knowledge and experience in terms of operating. Even if the platform is built on a CMS, there are still costs and other additional plugin features need to be considered.

### **3.3.2. Online CMS platforms**

A variety of online CMS platforms is available for helping in managing digital content. One widely used CMS is WordPress, which is open source. It was initially developed as a blogging platform, but in the last several years, WordPress has repositioned itself as a practical content management system. It provides a large number of plugins released by other independent developers (Patel et al., 2011). That could especially contribute to the creation of a more sophisticated website. In addition, it contains the most number of freely available templates and extension plugins. It also has premium themes for more professional purpose. A WordPress community forum exists to provide advice and enhance communication among users so that developers do not encounter difficulties with selecting the correct plugins for specific purposes. Moreover, it is easy to move a WordPress website to a new host or change to a new domain name.

Another well-known platform is Blogger, which is a platform for publishing blogs-provided by the tech giant Google. It is free, reliable and quite enough to publish the users' content on the web (Stuff, 2018). Moreover, the platform has quite easy and usable user interfaces for bloggers to use. It also provides free templates for users with clean design. Users can customize their website by modifying the design through built-in tools. However, Blogger has quite limited tools which can only allow users to perform specific tasks online. It is not suitable for advanced and professional bloggers because of lack of high-end features. Compare to WordPress, Blogger has more limitations and the site cannot be moved easily and safely.

Generally speaking, different CMS platforms support different levels of services. They aim to help those users without any technical knowledge to make a promising website effortlessly. Also, they provide users with a selection of features based on the purpose of the platform. By using a CMS platform, users can create a satisfying website efficiently without struggling with the programming work. Meanwhile, numerous plugins can be utilized to implement a wide variety of features. More importantly, the website can go live in a short time and update immediately.

### **3.3.3. Online VCS platforms**

Online VCS platforms are today for collaborative work, particularly for open source software development. One example of an online web-based VCS platform is SourceForge.

It contributes a lot to control and manage open source projects by providing various helpful features. Every software project hosted on Sourceforge has a source code repository, which supports discussion boards, issue tracker, and a wiki for documentation. Developers can easily develop, download and publish projects. However, because it contains lots of feature and contents, it can take a long time for novice users to get started and make a good use of it.

Bitbucket is a web-based VCS platform owned by an Australian software company called Atlassian. Bitbucket provides repository-hosting services for software projects. It can also integrate with other software such as Jira, which is an application for project and issue managing from Atlassian. Bitbucket is a Git-based platform for code and code review. It supports many attractive features for developers and organizations to host their software projects. For example, it supports pull requests and code reviews, branch comparisons and tracking commit history. Bitbucket offers free unlimited private repositories as long as the amount of members of a team is under five.

In addition to Sourceforge and Bitbucket, GitHub is another web-based collaborative platform for software development. It is already known as one of the largest open source communities in world (Jiang et al., 2017). It generally manages software projects in the form of repository. A repository refers to a folder comprising all tracked code and files as well as their history. Each project can be a repository, with multiple core members. Other developers, once they want to submit changes or fix bugs, can forks the repository firstly and then send pull requests to merge the commits to the original repository. In the workflow, the forked repository appears as a copy of the original repository. Each developer can modify and submit new implemented features without damaging the original repository or causing any conflicts. In addition, core members can evaluate all pull requests before deciding whether to merge or reject them. In a way, GitHub provide a safe and encouraging environment for collaborations. There is a chance for projects to be implemented by various developers from the worldwide. GitHub also aims to bring a large amount of projects to a community and facilitate all the collaborations rather than just for one specific project.

Compared to SourceForge and Bitbucket, GitHub works more like a social coding site. It prompts a social network through connecting developers with different activities happening in their projects. Indeed, this social effect allows GitHub to be more widely used in fields other than software development. Currently, more organizations and users have started to use it as the general collaboration platform instead of narrowing in software projects. For instance, some repositories have been used to store the other tracked resources such as documentation, and conference papers. Based on its powerful versioning, GitHub is being widely used as a collaboration platform for doing more than just managing the code files in the software development.

Another attractive feature is GitHub pages, which are simple and free websites hosted by GitHub. Users are able to generate their own website or blog related to specific projects and host it on GitHub. The generator of the static site is called Jekyll. The web pages can be customized by integrating with other web frameworks such as Bootstrap (Perez-Riverol et al., 2016). GitHub pages can be customized according to the needs of different individuals or organizations. Furthermore, since GitHub aims to encourage the collaboration in software projects, some continuous integration (CI) tools exist to facilitate the development work. One popular tool is Travis CI, which is a hosted, distributed CI service. It can be used to develop and deploy software projects hosted on GitHub. Generally, it helps automates the process of merging new code from different developers. There are also other CI tools that work seamlessly with GitHub such as Jenkins and Circle CI.

### 3.4. Comparison of various web platforms

Based on the discussions in the previous sections regarding different web collaboration platforms, Table 1 displays a comparison among these platforms according to several dimensions. Based on their features and popularity, WordPress and GitHub have been selected to represent online CMS and CVS platforms respectively.

*Table 1. The comparison of different web collaboration platforms*

	Custom online CMS or VCS	WordPress	GitHub
Cost	Cost of hardware, software (server), hosting service. Time for setting up hardware, depends on the technical skills of developers. Time for developing.	Free themes for use. cost of professional template. Time for setting up webpage. Time for maintaining database.	Free themes for GitHub Pages. Time for setting up webpage.
Effort	Depends on the requirement of platform, may need some implementation work for collaborative features.	Depends on the requirement of platform, may need some implementation work for collaborative features.	Less effort since some collaborative features exist.

User base	Organizations, enterprises.	Organizations, individual, bloggers, enterprises.	Developers, individual, bloggers, enterprises, organizations.
Collaborative features	Depends on the requirements and all features need to be implemented by developers.	Supports social networking. Provides discussion forum, collaborative editing for digital content. Other Plugins such as project management, collaboration email.	Provides pull request for contribution, issue tracking, wiki and forking etc. Supports social networking.
Mobile usage	Have to develop the platform to fit the mobile usage.	Automatic fit, no extra effort.	Automatic fit, no extra effort.

As shown in Table 1, building a web platform on GitHub definitely takes much less cost and effort than other twos. Firstly, developers do not have to consider about the hardware and software such as hosting services. They can concentrate on the content of platform. GitHub pages provides the creation of webpages free with appealing themes. All these design themes are responsive and work smoothly on mobile devices. When users prefer to have a customized website, they can update web pages by implementing source code and deploying the code in the repository.

As for WordPress, it could be used for creation of websites for hosting digital content according to different user cases. Based on a variety of plugins and templates, it can set up an interactive website speedily and successfully. However, it might take extra time and effort to develop plugins if the platform has special collaborative requirements. With the updating of content, the maintenance of database could be another cost.

By contrast, GitHub has more collaborative features can be utilized directly. For example, pull requests have already enabled contributions and collaborations from others. Even in that way the quality and accuracy of data can be guaranteed based on the evaluation before merging the pull request. Since every user can login with their own GitHub account, it saves a lot work for developers when implementing the platform. They do not have to develop new user profile for users. Therefore, developers do not have to implement any extra plugin-ins as needed in WordPress. They can take advantage of all existing features of GitHub to help optimize the platform. Other features such as issues tracking, wiki,

allowing developer or contributor to follow the repository transparently, can contribute towards more collaborations among different users.

Another advantage is GitHub repository seems more practical in terms of storing files since it does not require any databases. It already provides the public repository for data storing with support of versioning. Any kinds of data can be stored in the repository by simply uploading. Additionally, the sources code of the web pages can be stored in the repository. The owner of the repository can modify these files easily to allow the webpage can be updated in a straightforward way at any time as needed.

More significantly, GitHub is the only web collaboration platform that enables the easy migration and replication of platform among different users. Through the forking feature, every user is able simply take the built platform for their own use. Since all the source code is available in the repository, they can modify it according to their own business. When building the platform on GitHub, the primary work will be concentrated on the design of the platform while GitHub delivers almost all collaborative functions for use.

## 4. UX DESIGN AND EVALUATION METHODS

This chapter presents the main theories studied in the research process. With explaining the concepts of user experience, other relevant methods such as user centered design and user interface design are also clarified. Moreover, it presents the main design methods and evaluation methods which applied in the design work of this thesis.

### 4.1. User experience

The definition of User experience (UX) varies widely within Human-Computer Interaction (HCI) community. According to ISO 9241-210 standards, UX can be defined as a *“person's perceptions and responses that result from the use and/or anticipated use of a product, system or service.”* It concerns that UX more subjectively by including all the users' perceptions, emotions and preference as well as anticipation. Hassenzahl and Tractinsky (2006) also summarized the UX as follows, *“UX is about technology that fulfils more than just instrumental needs in a way that acknowledges its use as a subjective, situated, complex and dynamic encounter. UX is a consequence of a user's internal state--, the characteristics of the designed system-- and the context within which the interaction occurs—”*. Generally, UX reflects the users' feeling/experience at the beginning, during and after of use of system or product. Another similar definition is from Hassenzahl (2008), he defined UX as a momentary, primarily evaluative feeling (good-bad) while interacting with a product or service. By that, UX shifts attention from the product and materials (i.e., content, function, presentation, interaction) to humans and feelings – the subjective side of product use. Thus, UX reveals what feedback and experience the product can provide from a perspective of user. That mostly drives from the subjective experiences of users when interacting with product.

Another relevant concept is usability, which refers to *“the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”* (ISO 9241-11). UX is occasionally incorrectly seen as synonym of usability. However, UX is not as same as usability, instead, usability can be an aspect contributing to the overall UX as perceived by the users (Roto et al., 2011). More specifically, usability can be regarded as one dimension of UX when users interact with products. It is a more concrete measurement of UX based on its criteria. Nevertheless, the range of UX covers more and it concentrates on creating the overall outstanding experience instead of just preventing usability problems. Moreover, it manifests more from subjective impression of users.

Good UX enables users to achieve their goal efficiently when interacting with product along with high satisfaction. Moreover, it even provides users with unexpected, delightful



experience. Norman (1988) also clarified that a product with good UX, especially with attractive visual design, makes users more creative and accomplish tasks patiently. Products with satisfying UX do not only fulfill user needs by designed services, but also focus on other factors which affecting UX. For example, the context of use, inner state of users (emotions, experience, motivation and expectation). With increasingly digital products appear, UX gathers more attention and plays a vital role in product competition.

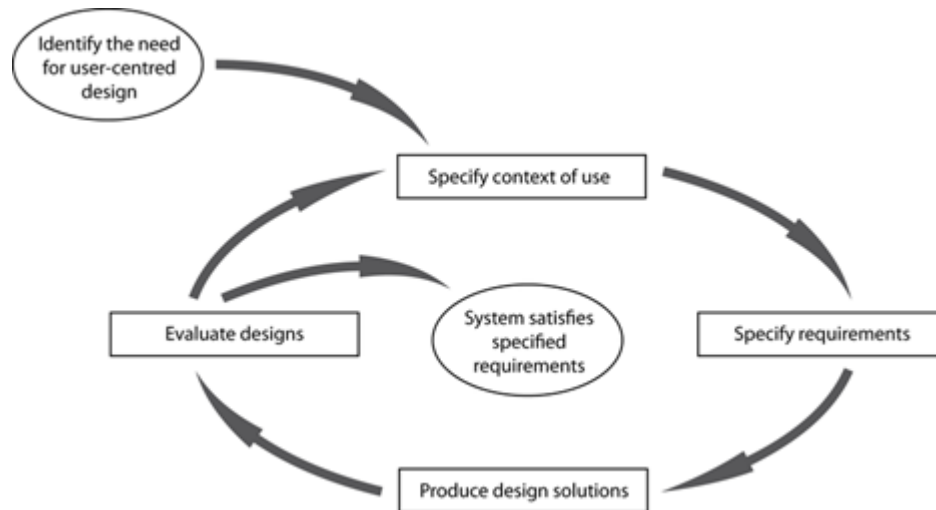
## 4.2. User centered design

User-centered design (UCD) is a general term refers to a philosophy and methods. It focuses on involving users in the design of computerized systems (Abrams et al., 2004). In brief, the end-user influences how the final product takes shape in user-centered design. Another working definition is ‘UCD is herein considered, in a broad sense, the practice of the following philosophies, the active involvement of end users for a clear understanding of users and requirements of tasks, iterative design and evaluation, and a multi-disciplinary approach’ (Vredenburg et al., 2002). This definition reflects the holistic perspective of the UCD.

In general, the essential principle of the UCD is to involve user throughout the entire design process. The user could only be end user who will use the final product in future. During design process, the user becomes a core part of the development processes. Their involvement contributes to more efficient, effective and safer products and leads to the acceptance and success of products (Preece, Rogers & Sharp, 2002). Additionally, ISO 9241-210 standard has defined six principles of the UCD method as below:

- The design is based upon an explicit understanding of end users, needs, tasks and environments.
- Users are involved throughout design and development.
- The design is driven and refined by user-centered evaluation.
- The process is iterative.
- The design addresses the whole user experience.
- The design team includes multidisciplinary skills and perspectives.

More than uncover the principles of designing a usable and appealing product, UCD also indicates the design process for inspiring and learning. According to ISO 13407, a general UCD process consists of main four activities, which is illustrated in Figure 4-1:



**Figure 4-1: Activities of user-centered design (ISO 13407)**

The main activities in a UCD process can be briefly described as follows:

- Understand and specifying the context of use.
- Specifying the user requirements.
- Producing design solution
- Evaluating the design

Knowing the user is a high priority task during design process, especially to understand the target user group, what they need and the context of use. Since that is the only way of product can be actual built based on the understanding of real use cases. Moreover, it is significant that involving the user in the evaluation stages. Since the real user is the only one who can evaluate the design, as a real user can tell what they expect and if the design is practical and user-friendly. Although the users may not speak everything they have in mind, it is not unrealistic to explore feedbacks and attitudes by other approaches, for example by observing, interviewing, survey and other methods in user research.

To be involved with user in design work, a variety selection of methods can be applied in UCD process, inducing usability testing, heuristic evaluation, participatory design and discount evaluation (Abrams et al., 2004). Through connecting user, more information that is valuable and profound insights could be revealed. That could help in determining the functionality side of design; meanwhile, it can suggest the designers of the overall user experience the product can provide. As a result, the product can be developed accurately on the behalf of users. In that way, there are higher chance to design the satisfying and delightful product. Therefore, UCD is an effective method can support of making digital products that are more usable and attractive. It guides the designers focus on the users instead of the product itself. It is the need of users shapes the final product. Admittedly, the business goals also affect the product designing; However, user-centered development still plays a key role in designing for good user experiences. Through understand real users' needs and values before starting designing and evaluating solution (Väänänen-

Vainio-Mattila, Roto & Hassenzahl, 2008). Moreover, keeping user in mind helps designers realize the potential desire, which can generate more ideas for the product in future development.

### 4.3. User interface design

User interface (UI) as another important component, determines if a product is usable and understandable. The user interface is that part of the computer system through which users can use for undertaking and achieving their goals (Stone et al., 2005). Therefore, the quality of user interfaces, such as if it is easy to understand and use, directly affecting the users' understanding of system and the use of system as well. Good UI enables users can control the system effortlessly. It also helps user concentrate on their tasks and accomplish them effectively. Moreover, proper interface design will provide a combination of well-design input and output mechanisms that satisfy the user's needs, capabilities. It helps users perform tasks most effectively (Galitz, 2007). It is the essence of a product providing good user experience.

User interface design can be known as a subset of HCI. In fact, user-centered design is a good approach to help in user interface design and development by involving end users throughout the design and development process (Stone et al., 2005). UI plays a critical role in UCD, especially in iterative design phase, a user interacts with system based on UI. User interface design significantly benefits from the iteration design process by validating UI from end users. In general, user interface design (UID) utilizes the same process of UCD, as they both have to focus on the users in order to create a product with usable and satisfying user interfaces. UID still starts with the understanding of real users, including who are they, what they need and what they expect. Afterwards, users interface is design iteratively based on feedback and comments from users. By resorting UCD method in which users shape the design, the product can be created with appealing user interfaces.

Although different products have dissimilar requirements of UI design, some widely used principles of UI can be concerned during design. According to Constantine and Lockwood (1993) in their usage-centered design, these are six general principles as follows:

- *The structure principle*  
Design should organize the user interface purposefully based on clear, consistent models that are apparent and recognizable to users.
- *The simplicity principle*  
The design should make simple, common tasks easy, communicating clearly and simply in the user's own language.
- *The visibility principle*  
The design should make all needed options and materials for a given task visible without distracting the user with extraneous or redundant information.

- *The feedback principle*  
The design should keep users informed of actions or interpretations, changes of state or condition.
- *The tolerance principle*  
The design should be tolerant and flexible, reducing the cost of mistakes and misuse by allowing undoing and redoing. It also prevents errors.
- *The reuse principle*  
The design should can be reused both internal and external components and behaviors, maintaining consistency to avoid users' rethink and remember.

Another type of user interface is mobile user interface. According to different use context and environment, mobile user interfaces have some special principles while they follow the general ones. However, user interface design of mobile application is more restrictive than desktop ones. Mobile devices' screens are typically smaller than their desktop counterparts. They have less processing power, and communicate in low-bandwidth environments, always-changing context. Mobile applications must be carefully designed when considering these limitations (Tarasewich, 2003). Accordingly, Gong and Tarasewich (2004) proposed a set of practical design principles for mobile device interfaces design. These guidelines also summarized general user interfaces design principle that modified based on mobile use context. A list of most frequently used principles is displayed in Table 2.

**Table 2.** Guidelines of mobile user interface design (Gong & Tarasewich, 2004)

<b>Guideline</b>	<b>Description</b>
Enable Frequent Users to Use Shortcuts	The system should always keep users informed about feedback, such as what is going on, through appropriate feedback within reasonable time.
Design Dialogs to Yield Closure	The number and pace of interactions should be reduced as the frequency of user increases.
Support Internal Locus of Control	Some feedback should be provided to every user action.
Consistency	Sequences of actions should be organized into groups. Users should obtain satisfaction of accomplishment and completion.
Reversal of actions	Users can in charge of the system and have the system respond to their actions, rather than feeling that the system is controlling them.

Error prevention and simple error handling	The design should be the same across multiple platforms and devices. Elements of mobile interfaces such as names, color schemes, and dialog appearances should be the same as their desktop counterpart.
Reduce short-term memory load	Mobile applications should rely network connectivity as little as possible.
Design for multiple and dynamic contexts	Nothing potentially harmful should be triggered by too simple an operation (e.g., power on/off).
Design for limited and split attention	Rely on recognition of function choices instead of memorization of commands.
Design for “top-down” interaction	Users can configure output to their needs and preferences. Allow for single- or no-handed operation. Have the application adapt itself automatically to the user’s current environment.
Design for speed and recovery	Provide sound and tactile output options.
Design for enjoyment	Provide enough information and let users decide whether or not to retrieve details

All these guidelines are basically deriving from the traditional guidelines for desktop applications. They were collected from different published researches. In addition, more user interfaces principles exist to provide guidance to design work based on different platforms, such as Human interface guidelines from Apple and material design of Google. All these guidelines and design frameworks can contribute to a useful and delightful product.

#### 4.4. UX design methods

To make a usable product, some design methods can be applied to help improve the user experiences. In fact, User centered design is widely utilized in the UX design to develop a product based on real users. During UX design, other approaches such as prototyping is discussed as follow. Generally, prototyping is involved in UCD method to guarantee the good user experience of the final design.

#### 4.4.1. High-fidelity prototyping

In user-centered design, prototype could be utilized to test the ideas, evaluate the usability of the product or help collect insights from both stakeholders and users. More specifically, the UCD process started with user and task analysis, including user characteristics, goal, tasks, and environment. These analyses formed a conceptual model. The model then evolved into prototype, consisting of low-fidelity prototype (paper prototype) and high-fidelity prototype (prototype with data flow and interface) (Anindhita & Lestari, 2016). Basically, the term prototype comes from the aggregation of two ancient Greek words: proto—meaning first, and typos—meaning gestalt, or shape. Thus, it can be seen as a preview version of a product that have not yet entirely implemented. In early design stage, it presents the ideas and concept of the product (Bähr & Möller, 2006). To evaluate these abstract concepts and ideas related to the product, prototype is widely used as a tool to demonstrate the product in a verifiable way. Moreover, it works as the bridge connecting between user and designer.

Another advantage of prototyping can be seen as a test vehicle especially in usability testing, which usually recommended to be conducted at the early stage of design process. Prototypes can be produced and tested by end users in the design processes. Meanwhile, designers are supposed to pay attention to the evaluations since they will help identify measurable usability criteria (Abrams et al., 2004). Having user tests with prototypes definitely saves the cost and time of design work to avoid the wrong direction. Meanwhile, it helps collect the feedbacks and perceptions from real users, which could lead to the improvement in further development.

There is no restriction of the selection of different prototypes. It is mainly based on the project. The widely used methods are low-fidelity prototype, high-fidelity prototype and paper prototypes. In some projects, only one type of prototype is chosen while others may use multiple prototypes through the project. Mostly, compare to the low-fidelity prototype, which are generally the form of wireframes with limited functionality and interaction, high-fidelity prototypes trade off speed for accuracy. They typically cost more time and effort than low-fidelity prototypes, but they faithfully represent the interface to be implemented in the product (Rudd, Stern & Isensee, 1996).

Although the high-fidelity prototype is commonly more time-consuming to create, it can outline the final product by filling in more details. Therefore, via testing with the high-fidelity prototype, more usability problems in terms of real design (e.g. color, user interface designs etc.) can be revealed. In addition to that, high prototypes work as a very good educational and guidance tool for programmers and information designers. The programmers can use the prototype as a living specification which contains functional and operation requirements (Rudd, Stern & Isensee, 1996). There is a variety of professional online tools available for creating high-fidelity prototypes, for example, Invision

(<https://www.invisionapp.com/>), UXpin (<https://www.uxpin.com/>) and even the Microsoft Office can do the work of prototyping when needed.

Prototyping is not only valid for the development of standard computer software, it also can be used in mobile applications. However, designing a user interfaces for mobile apps including dealing with several additional challenges that derive from changing of use contexts (Bähr & Möller, 2006). Both use cases of laptop and mobile phone have been involved in the usability testing during design process.

## **4.5. UX evaluation methods**

Even though UX concerns more about subjective quality, it is still measurable and can be evaluation through specific methods. UX evaluation methods (UXEMs) play an significant role in ensuring that the product's or service's development phase is going in the right direction. Therefore, the final product will meet the expectations of users and provide appropriate positive experience during use (Rajeshkumar, Omar & Mahmud, 2013). Through evaluation, the level of usability and UX of product can be reflected and improved. More importantly, it provides the hints of how the product is used and learned. Therefore, during evaluation phase, the methods should preferably allow comparative and repeatable studies in an iterative manner (Roto, Obrist & Väänänen-Vainio-Mattila, 2009). This chapter mainly introduced two methods adopted in the thesis work. However, other evaluation methods (such as online surveys, field studies, etc.) can also be applied to measure UX based on real users.

### **4.5.1. Task based usability testing and interview**

As the most common used evaluation method in web domain, usability testing is a very effective technique to identify the usability problems, especially for those related to the graphical user interfaces. Moreover, it helps in understanding real users by observing their behaviors during the test. Technically speaking, Usability testing is a usability evaluation method. A representative number of end users are requested to interact with the system voluntarily. During this test, users have to perform a set of predefined tasks using the software product to be tested. Indeed, a usability test during early stages helps to reduce the impact on costs since changes would not be difficult to implement (Paz et al., 2015).

While user is interacting with the system by performing the task, the usability problems can be identified through observation. Meanwhile, users may have more thoughts about the system to share. The general target of usability testing, according to Dumas & Redish (1993), usability testing generally is aims to achieve the following five goals:

- improve usability of products
- always involve real users in the testing
- give the users real tasks to accomplish

- enable testers to observe and record the actions (behaviors) of the participants
- enable testers analyze the data obtained and make changes accordingly

In general, usability testing focuses on users' tasks and needs, uses empirical measurement, and iterative design (Nielsen, 1994). Especially, usability testing for website also takes a user-centered approach, and designers can concentrate on the needs of end users (Norman, 1988). With conducting usability testing with real user, the actual opinions and problems could be figured out, and the final product will be shaped according to the feedback from users.

A usability testing normally starts with a brief introduction, and then it provides a consent form with background information to ask user's permission of attending the test. In addition, there is a background questionnaire for participant, which intends to collect demographic information. Afterwards, the participant is required to complete a list of tasks; all these tasks cover the main features of the target system, and they are given one by one. Once tasks are finished, there can be a semi-structure interview and a user satisfaction form need to be filled. During the user test, a series of relevant information has to be collected and restored for later analysis.

- **Semi-structure interview**

After performing tasks, there is a semi-structure interview for each participant. Generally, semi-structured interviews are merely conversations helps interviewers know the information they want to find out. They typically contain a couple of questions which cover topics and ideas. However, the conversation can be vary and is changing substantially between participants (Miles & Gilbert, 2005). Moreover, according to the Chauncey Wilson (2014), the semi-structured interviewing could be used to do the following:

- Gather attitudes, facts and opinions
- Gather data on relevant topic while allow users to raise new issues that are important to them through open-end questions
- Gather data when interviewers cannot observe behavior directly because of timing, hazards, privacy or other factors
- Understand user goals and motivations

Therefore, the target of the semi-interview is not only to discuss the mockups summarily, but also to explore more perspectives of the users. It can prompt more open ideas, for participants are encouraging to share and comment as much as they can. Admittedly, all these voice from real user are recorded as notes during interviews, while part of them also retrieved from reviewing the video.



### 4.5.2. Heuristic evaluation

Heuristic evaluation is referred as guidance throughout the whole design process. Particularly when designing prototypes and analyzing the results from usability testing. In general, heuristic evaluation is an inspection method, in which 3 or 5 usability specialists judge whether each element of a graphical user interface meets established usability principles, called heuristics (Nielsen, 1994). The principles which are frequently used in this method is a set of rules proposed by the same author of the technique. These guidelines are known as the ten usability heuristics of Nielsen (1995). All these usability heuristics provides a solid base to evaluate any digital product with graphic user interfaces. The statement of ten heuristics is displayed in Table 3 below.

*Table 3. Ten heuristics (Nielsen, 1995)*

Visibility of system status	The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.
Match between system and the real world	The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.
User control and freedom	Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.
Consistency and standards	Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.
Error prevention	Even better than good error messages are a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.
Recognition rather than recall	Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use	Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.
Aesthetic and minimalist design	Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.
Help users recognize, diagnose, and recover from errors	Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution
Help and documentation	Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.

In fact, the ten heuristics can serve as guidance to be considered in terms of user interface. It can illuminate the work of user interface design by these guidelines. When defining and analyzing the usability problems discovered in user tests, the heuristics can also help in figuring out the theory behind them. Meanwhile, it provides the principles of how to improve the design to achieve a more appropriate level of usability.

#### 4.5.3. Data gathering and analyzing

During evaluation phase, the data mainly come from the user tests. It can be divided into four portions, usability problems, findings, filled forms and video recording. In addition, findings mainly include suggestions and ideas, other thoughts from users and discoveries from observation during test and review of videos. After each test, various completed forms will be collected, such as consent form, background questionnaire and user satisfaction questionnaire. Particularly, the data from background and user satisfaction questionnaire will be analyzed after each test; consequently, relevant results can be recorded and reported.

Once the user test finished, the priority is to go through the usability problems. Based on the recording from the tests, each usability and UX problem can be rearranged with different labels. More specifically, there is an evaluation of the severity of the problem according to the extent to which the problem impedes the use of the repository; mostly could

be seen through observation and comments from users. Considering the scale of the system, there are just three categories of severity of problems as follows:

- Severe problem, which significantly affects the use of repository and should be repaired immediately.
- Major problem, prevents the users from using the repository in easy way or cause.
- Minor problem, complicates the use of repository, or others could be improved to make the use to be as pleasant as possible.

Therefore, all usability problems will be detailed according to severity and frequency as well. This effectively guides the future design work in terms of establishing a priority. Apart from problems, the background information of participants is documented by the results from the background questionnaire. Meanwhile, other findings will be reported after each round of user tests. For example, the suggestions and ideas can be discussed.

Moreover, there are other valuable results from the user satisfaction form, which directly reflects the perspective of users. All these data and results is displayed in next chapter. As for all related recordings, such as videos, forms, they will be destroyed after finishing the thesis work.

## **5. DESIGN AND EVALUATION OF USER INTERFACE FOR IOT META MODEL REPOSITORY**

The design and evaluation work of the meta model repository is described in this section. It starts with the explanation of the selected platform for the repository. Then it presents the overall process and phases of the design work. Moreover, this chapter describes the process of user interface design associated with iterative evaluation. Specifically, more details of two rounds of user tests and relevant findings are presented. Furthermore, it briefly outlines a new version of the repository with interactive UI and additional functionalities.

### **5.1. Selected platform for the repository**

Based on comparison in Chapter 3 and consideration of the research goals, GitHub is selected as the collaboration platform to manage multiple data models for IoT devices. Users can easily set up a public repository in GitHub while others can fork the repository for their own use. Moreover, they can even contribute back to the original repository by pull request (McDonald & Goggins, 2013). For the meta model repository, GitHub not only provides a public repository but also delivers GitHub pages to enable a satisfying use experience in terms of browsing data models. By presenting data models in web interfaces, developers and contributor can browse and search data models in a more interactive way compare to view data models in documentation.

With support of GitHub, the meta model repository can benefit three user groups through a public repository in addition to GitHub pages. It can contain various data models. More concretely, all files and resources of data models are stored in repository while they are demonstrated on the webpages. Therefore, users can visit the repository to seek for resources; alternately, it is encouraging to browse data models through webpage as well.

Another advantage is for contributors. GitHub provides support for contribution through pull requests, which allows developers can contribute flexibly and effortlessly (Gousios, Pinzger & Deursen, 2014). Therefore, external individual or parties can submit new data models through a pull request. Operator of repository has rights to either accept or reject merging the pull request by evaluating the data models. That guarantees the quality of data model. Additionally, the profile of contributor can be tracked easily through GitHub, and operator can discuss with contributor even in a pull request by leaving comments. This significantly facilitates the communication in a community without using any external channel. Moreover, the transparency of the platform enhances the collaborations.

Other features of GitHub also make it a suitable platform for meta model repository. For instance, the public repository in GitHub stores all the object definitions and code files, which could be retrieval and download. Especially it can support the different version of files and provide all the history of committing files, which can be effortlessly tracked through web interfaces of GitHub. Moreover, through pull requests, it is easy to update data models or make any changes related to the files. As a social platform, GitHub also integrates many other functionalities, including wiki, code review and issue tracking etc. More importantly, all these activities are visible for every developer who participant in the project, which prompts a developer-friendly environment.

Furthermore, the third user group can benefit from the GitHub. In GitHub, developers are encouraged to fork other repositories and modify the forked repository without asking for permission (Jiang et al., 2017). This approach could be applied for the enterprise users. More specifically, when they want to have their own platform for data models, they could simply fork the repository, and then modify the repository according to their own business. With working on the source code of the meta model repository, enterprises (organizations) can manage data models and demonstrate them publicly in more appealing way. Additionally, the collaboration between enterprise and developer/contributor could be enhanced with the pull request.

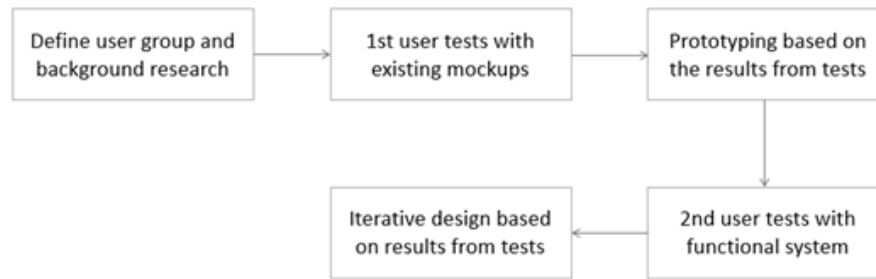
Overall, as both an open sources service and social networking platform, GitHub is the most appropriate platform for the meta model repository serves for three user groups. Meanwhile, with its advantageous features of GitHub, the repository can be implemented speedily and effectively without extra cost on other functionalities. Instead, the primary work can be engaged on design of GitHub pages to ensure a good user experience provided to users. Moreover, the social element of the platform allows the extension of the repository with merging more various data models.

## **5.2. Overall process of research work**

This chapter mainly reports the overall process of the UCD design work. It also discusses the two rounds of user tests. Moreover, it introduces the information of participants in user tests and the procedure of two rounds of user tests.

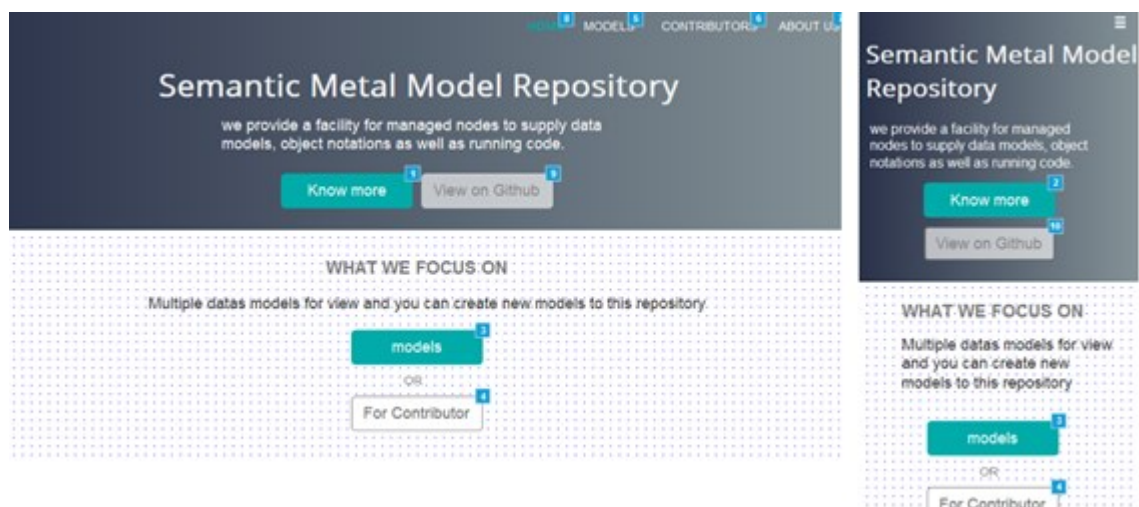
### **5.2.1. UCD process**

Generally, a UCD process is simply adopted in the research work. The research process of UCD was completed in a short period of three months. It started from September and ended in November. By considering the motivation of designing the repository, especially who have desire to use the repository, the target user group was defined as three categories. Throughout the entire design work, two rounds of usability testing have been conducted with the original mockups and functional repository, respectively. Figure 5-1 presents a brief progress of the research process.



**Figure 5-1: The overall process of research work**

During design work, the used prototyping tool was Axure RP(<https://www.axure.com/>). It is a widely used software and it allows to generate prototypes with different degree of fidelity. Most importantly, the notable new feature for mobile designer is Adaptive Views. Adaptive Views is Axure’s implementation of support for responsive design, moreover, Axure allows us to get a touchable, interactive prototype on an iPhone or Android device without having to worry about Objective-c or Java code (Hacker, 2013). Therefore, the prototypes include both laptop and mobile version. Figure 5-2 illustrates the designed prototypes in design process.



**Figure 5-2: High-fidelity prototypes of desktop and mobile**

Afterwards, the final repository was built as webpages hosted on GitHub, and it developed by using web technologies, mainly including HTML5, CSS and JavaScript, and it also used a Bootstrap UI framework. Due to the limitation of the contacting with users at early age, in second round user tests, it was the implemented repository that used to be evaluated. Based on results from two rounds of user tests, the repository was re-designed and updated twice during the entire design process.

### 5.2.2. Two rounds of user tests

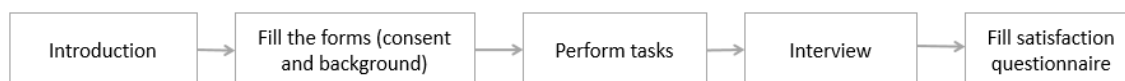
During the UCD process, two rounds of user tests were conducted in total. Each test was designed for different target systems. All user tests were done without a formal lab, instead, they were either in random meeting room or quite coffee area, with a portable device (iPad) to record the video. Another significant element in usability testing is the participant, which is the representative of end user. It is apparent that having user test with real users is significant since it will provide valuable information about how users interact with system (Nielsen, 1993). Nevertheless, it was quite tough to get access to end users because of schedule issues, and other limitations, such as they are in various locations. Some of these participants come from different companies while others work in university. Some of them are both student and employee. Table 4 displays the accurate data of participants. Basically, this information was collected by the background questionnaires (see Appendix B).

**Table 4. The background information of participants**

Participant	Age	Gender	Occupation	Education	Working experience
1	30 - 40	Male	Student, Employee	Master	6 - 10 years
2	30 - 40	Male	Employee	Master	6 - 10 years
3	20 - 30	Male	Student, Employee	Bachelor	1 - 5 years
4	20 - 30	Male	Employee	Master	6 - 10 years
5	40 - 50	Male	Employee	Master	More than 10 years
6	20 - 30	Male	Employee	Master	6 - 10 years
7	20 - 30	Male	Employee	Bachelor	1 - 5 years

As it shown in Table 3, almost all participants' age ranged from 20 to 40 years, and they are male. Based on their profile, they could be regarded as expert users since they are all skilled users of computers and familiar with technology. Majority of them have several years of working experience. On the other hand, all of them were selected based on the requirement of having work experience with data models, at least with LWM2M models. In that case, they are supposed to understand the concept of the repository with its value. In user tests, six of participants took part in both rounds while another one only participated in the second round.

More specifically, altogether six user tests were performed firstly in three consecutive weeks in September 2017. The pilot test was done in the recreation room of the Laboratory of Pervasive Computing of Tampere University of Technology. Apart from finding out usability problems, the main purposes of this test were to measure the approximate time of one test and to see if the process works well. Then, the rest of the five tests were completed in EIT Digital Helsinki Node Open Innovation House in Espoo. Additionally, the test procedure is shown in the Figure 5-3, which was applied in both rounds of user tests.



**Figure 5-3: The process of a single user test**

During the test, the participant was first introduced to a brief background, which contained the purpose of the usability testing and the general process. The participant was required to sign a consent form (see Appendix A) with background information, and then a background questionnaire. Afterwards, the participant was asked to complete seven tasks (see Appendix D), which covered the main features of the mockups. The tasks were given one by one, and if it seemed that the participant had problems finishing it, a hint would be provided. Meanwhile, some relevant questions would be asked, for instance, “do you think it is easy to find it”, “how you feel about it” or “any suggestions about it”. By discussing the problems during the task, there were more chances to figure out extra usability problems and the reasons behind them. In addition, it is more efficient to understand and record the thoughts from participant right away after specific tasks, since the participant might forget the detail of the task in the later phase of the test.

After finishing tasks on a laptop, two of seven tasks were selected to be done on mobile phone. When the whole task session was over, an interview was provided to participant, which consisted of eight questions (shown in Appendix F) about the mockups. They contain different aspects of the targeted system, for example, the design of the mockups, functions, other opinions or ideas, even one question about the test itself. Since it was a semi-structured interview, participants were encouraged to share and comment as much as they can during discussion, not only be limited to the design or functionality of the mockups, but also the feedbacks about the user tests. Eventually, the participant was asked to fill the user satisfaction form (see Appendix C). After all user tests, all filled forms were collected and reviewed; relevant notes and results were analyzed and wrote as report.

In the first round, the time for six user tests varied from 16 to 42 minutes, which corresponded to their background, previous experience with usability testing, also the understanding of the concept of repository mattered. In particular, the participant in the pilot test spent the least time since he had used the mockups previously and knew it quite well.



However, the pilot test had provided perceptions and insights from the real user. Meanwhile, it helped to realize to what extent the test is comprehended by participants, especially for those who have never been in a usability testing before.

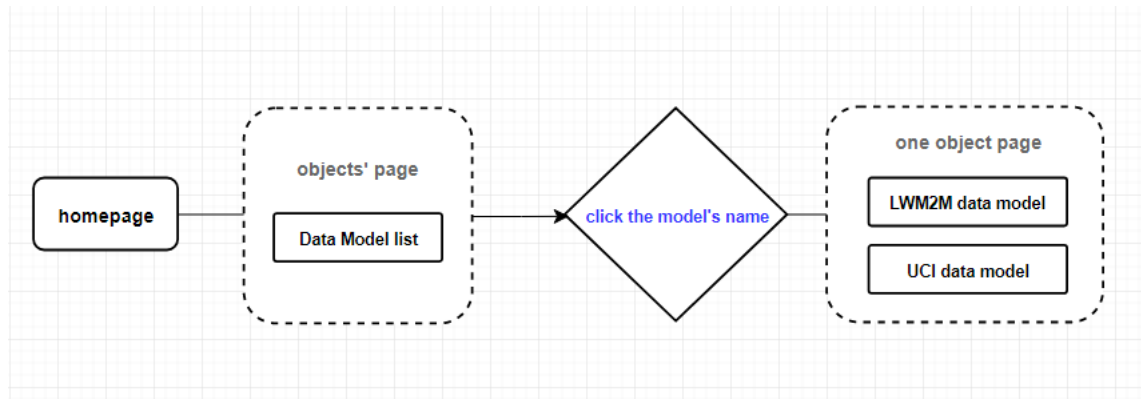
As for the second round, all seven tests were done in the following month after the first user tests. They were in the same place as last round, one in Tampere University of Technology while others in Espoo. Moreover, the test target in second round was the implemented repository; however, the process remained the same as previous one, except there was no need for some participants to fill the background questionnaire again. Another update was a new list of tasks (see Appendix E). Due to the additional functions, which were iteratively designed according to the results from the first user tests, three supplementary tasks were added to test. They aimed to test the new features such as contribution part, search function and download files.

Apart from the test on laptop, two tasks were given to test on mobile device. However, the list of questions in interview was the same as last one. In the second user tests, time of user tests were between 17 minutes and one hour. Overall, the user tests went more smoothly since nearly all participants have more experienced with usability testing after the first round. Accordingly, all relevant forms and results were collected and restored for later analysis.

Furthermore, the purpose of both rounds of the user test primarily focused on two user groups, which are developers and contributors. However, the test for enterprise users was excluded, since the main usage for them is simply forking the repository from GitHub. Therefore, no user interface design involved at this moment except a hyperlink of the document of maintenance work in the contribute page.

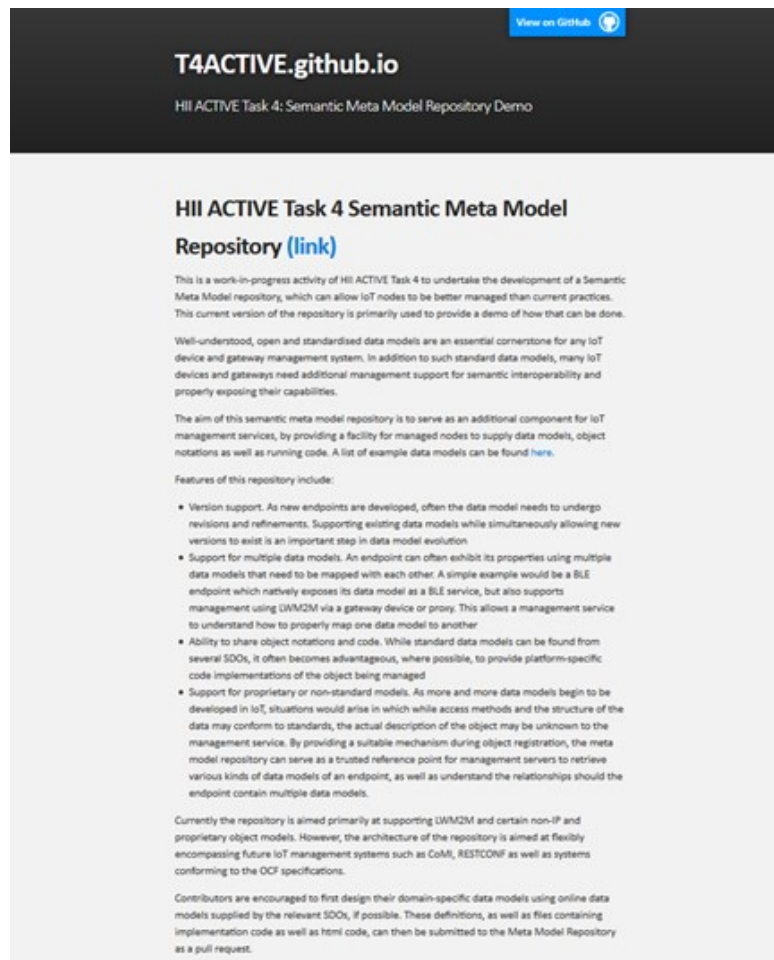
### **5.3. Design of the first version of the UI**

The first version of the repository is basically the mockup as GitHub pages. It included primary pages to illustrate the idea of the data model repository. Since it was hosted on GitHub, apart from interactive pages; some relevant files were stored in repository of GitHub, for example, the XML files, JSON files and other code snippets for implementation work etc. Moreover, a few smart objects of IoT devices were included in the mockups. The overall structure of the user interface design of the mockups is shown in Figure 5-4.



**Figure 5-4: The overall structure of the user interface design**

Generally, the main tasks for a user is to browse the data models of specific objects. Users get access to a data model list through main page, and by clicking the name of the object, users can browse all data models of this object, basically the LWM2M data models and UCI data models if it has. Below Figure 5-5 shows the homepage of the mockups.



**Figure 5-5: The homepage of the mockups**

All the information is displayed on the main page and the only entrance of the page of model list is hidden in the text. On data model list page, a set of smart objects are simply shown in a table as in Figure 5-6.

## ACTIVE Data Model Repository

Welcome to the ACTIVE Data Model Repository. Reusable objects and object definitions are listed below.

Object Name	Description
<a href="#">Gateway System</a>	Example Gateway Application Object describing base system
<a href="#">Gateway Wireless</a>	Example Gateway Application Object for representing wireless IP interfaces
<a href="#">PAN Interface</a>	Generic object for modelling endpoint or gateway PAN interfaces, such as Bluetooth, BLE and Zigbee
<a href="#">BLE GATT</a>	Generic object for proxies to expose BLE services
<a href="#">EMM Integrator</a>	Object for defining the components of an Energy Measurement Module
<a href="#">Log Control</a>	Controls the log handling for the sensor data
<a href="#">Device Binding</a>	Connects the EMM to the device it measures

**Figure 5-6: The model list page**

By clicking the name of object, for example, gateway system, the link goes to the multiple models' page of the object. More specially, there are two data models as shown in Figure 5-7, LWM2M data model and UCI model. The LWM2M part is initially expanded once the page is opened.

**HII ACTIVE Task 4 Demo**  
Semantic Object Models

**Gateway System Object**

**Description**  
This Object defines the basic settings for a generic gateway.

**Data Models**

Object Version: urn:oma:lwm2m:ext:1022:1

**Resource Fields**

- Hostname**

Resource ID	Access Type	Multiple Instances	Mandatory	Type	Range or Enumeration	Units	Description
0	R,W	No	Optional	String			Hostname of the System
- Timezone**

Resource ID	Access Type	Multiple Instances	Mandatory	Type	Range or Enumeration	Units	Description
1	R,W	No	Optional	String			The timezone used in the system
- DNS Server List**

Resource ID	Access Type	Multiple Instances	Mandatory	Type	Range or Enumeration	Units	Description
2	R,W	No	Optional	String			List of DNS Servers used by this system
- NTP Server List**

Resource ID	Access Type	Multiple Instances	Mandatory	Type	Range or Enumeration	Units	Description
3	R,W	No	Optional	String			List of NTP Servers used by this system

**Example Usage**

- XML
- JSON
- Code

• UCI

**Figure 5-7: The page of multiple data models for gateway system object**

The part of LWM2M data model is quite informative. The resource field and several tables are displayed to show the resources of the data model. Apart from the resources, there are other content involved, for example, checksum, example usage and code part for XML, JSON and others. The one collapsed below code session is the UCI model. As for the two folders under the black horizontal bar, they are not working properly, a zip file of entire the code of the mockups will be downloaded once either of them being clicked.

Generally, the mockups follow a hierarchical path for both navigation and structure in GitHub repository. The home page displays all groundwork of the repository. Additionally, the entrance of model list only appears in homepage, otherwise, users have to navigate by the “go back” button provided by browser.

## 5.4. Iterative evaluation

This chapter primarily presents the evaluation work during the design work of repository. The evaluation is based on two rounds of the user tests. Through collecting results and feedbacks from user tests, the discovered usability problems were analyzed, and other inspiring findings were concerned when designing the user interface of repository. Therefore, the repository was designed and implemented iteratively based on these qualitative data from real users.

### 5.4.1. First user tests

In the first round of tests, the results were collected through recording in tests and reviewing the video. Overall, the results are divided into four sessions to be presented more explicitly. Primarily, usability and UX problems is revealed in Table 5. Meanwhile, the positive findings and the results from interview session is illustrated in Table 6 and Table 7 respectively.

As one of the most significant outcomes from the user test, all usability problems were labeled according to their severity. Another applied value is the frequency, which points out how many times the problem appeared during use of system. Table 4 below mainly demonstrates the existing problems.

*Table 5. The usability and UX problems found in the first user tests*

<b>Usability and UX problems</b>	<b>Severity</b>	<b>Frequency</b>
The home page is full of text and hard to follow, no sub-headings for specific information	Severe	4
UCI model is hard to find out and takes time	Severe	3
The model list button is hidid in text and not obvious	Severe	3
Information of contribute takes time to be seen	Severe	3
The main blue link does not work and causes confusion	Severe	3
Layout of the object model is not appealing	Severe	2
Resource field seems crowded and hard to read	Major	3
Hard to navigate among different data models	Major	2
Navigation is not so clear	Major	2
Unfamiliar terms on page, what is SDOs	Minor	2

From all user tests, the most severe problems were caused by a crowded homepage with disordered information. Specifically, three participants tended to click the blue link when they performed the task 3, which is finding data models list. Then it turned out a fake link. After they found the correct link, most commented it is hidden there. Therefore, all participants in tests suggested to make the significant information more obvious, or even use subheadings. Similarly, half of participants spent time searching the information for

the contribution section and they even went through the text information from the beginning. What is worse, one participant could not find it until he got hints. It seemed that the navigation and organization of information should be improved for more ease of use.

The similar problems appeared in the page of multiple data models. Due to the improper design, 3 out of 6 participants missed the UCI model. Only two participants succeeded in this task, and the reason behind that is one has used the system before. The other participant just used the search function of Google Chrome and found it.

Meanwhile, half of participants claimed the design of multiple models is not appealing, since it is extremely long and occupied full width of screen. Moreover, for the resource field within LWM2M data model, it is quite hard to read. Furthermore, two participants suggested to keep the LWM2M model collapsed initially. In that way, it will be more usable and easy to navigate. As for the term which related to the specialized field, it is always better to be self-explained, either by opening abbreviation or provide relevant hyperlink, since different participants share diverse background and experience.

Apart from existing problems, other suggestions had emerged during tests and interviews. All these thoughts provided the possibilities and guidance for the design work. More importantly, they outlined users' actual needs and potential requirements of repository as well. Below is the table of the results.

**Table 6. The mentioned ideas and suggestions during interviews**

Ideas/Suggestions	Frequency
Search function is needed when more data models come	6
More obvious button or link for entrance of model list	5
Provide more information or a guidance about contribute	3
More clear navigation for the repository	3
Provide simple way for people to contribute	3
Display the resource field of data models in better way	2
Allow to download the files within data models	2
Re-organized the information and make page clearer	2
Make the data models shown in more explicit way	1

The most significant suggestions for improvement were addressed on the search function, and the more obvious presentation of information. When it comes to contribution, more

information is expected to be provided since there were some participants who are unfamiliar with the GitHub and pull request. It turned out that they do not know how it works. The rest of the suggestions were proposed based on the working experience with data models of participants.

In addition to the problems and relevant suggestions, there were encouraging findings from the tests described below, which displayed the positive side of the mockups. They also reflected the repository in a way which participants expect. Accordingly, the new version of the repository was created by repairing the problems while learning on the optimistic design.

*Table 7. The positive findings and comments found from interviews*

<b>Positive findings</b>	<b>Frequency</b>
The information of data models is valuable, and I will use it again	6
The overall appearance of repository is nice and simple	4
Code part in data models is useful for implementation work	3
The mobile version works better	2

From the perspective of users, the idea of gathering of the multiple data models is quite valuable. Another amusing finding is there were two diametrically opposed opinions of mobile version, one participant felt that it is painful to browse information on phone. In contrast, other two thought it is even better look on phone than laptop. The theory behind this phenomenon might be they use different smart phones, which cause distinct views. Certainly, the mobile version should be designed appropriately. Since there could be usage of mobile use, it is supposed to serve a pleasant user experience.

According to six interviews, participants came up with some ideas and suggestions while they mentioned some positive findings of the mockups. More concretely, all participants thought the information of data models is valuable. Most of participants (4 out of 6) comments the overall appearance of repository is nice and simple. Half participants complimented the code part of data model is very useful and glad to see it. However, some participants wanted the search and download function for data models. Moreover, some participants tended to need a clearer navigation and more obvious entrance for data model instead of a small hidden link. Another suggestion is about contribution, some participants needed more information of how to contribute while others asked an easier way to contribute. Additionally, some mentioned it would be better to re-organize the full text on homepage. They also thought the mobile version work fine as two of them believed it is

even better than laptop one. All the participants would like to use the system again. Additionally, the last result from the user satisfaction forms which filled during inter-view session. It is illustrated in table 8 below and reflects the grades from all participants.

**Table 8. The grades from user satisfaction forms in the first round of user tests**

Participants	P1	P2	P3	P4	P5	P6
Grades	4.5	3	3	4	4	4

**Average grades: 3.75**

According the findings and grades from participants, usability problems which uncovered previously definitely affected the use of the system. It led to improvement work of the repository. Based on all results obtained from the first user tests including the interviews, the new version of repository was invoked over iteration rounds, and the finalized repository is revealed in chapter 5.4 which implemented by taking the results from two rounds of user tests into account.

#### **5.4.2. Second user tests**

The results from second user tests were gathered in the same way as the first round. Following an overview of results from the user tests, the results are described precisely according to three categories, first one is the suggestions and ideas found from tests and interviews (See Table 9), and then positive findings (See Table 10), the third one is the data from user satisfaction forms which shown in Table 11.

As the repository was designed based on the results from the first user tests, the second round was performed to figure out if it is usable and well understood, especially if it achieves the expectation of users. Through these seven user tests, all participants completed these tasks quite successfully and efficiently without any hints. For task 9, which asks to contribute data models, since it requires the user to interact with GitHub platform, it took more time to achieve but still got quite positive outcome based on users' performance. Just one participant does not have GitHub account, so the task just skipped. However, he still agreed with the idea of contributing and argued it is a good point.

In general, based on these user tests and interviews, no severe usability problem was revealed except one problem related to UX writing, which is about designing the words to support users' interaction with system. More especially, the navigation is re-designed for the repository, it seems a general navigation bar used in most website, and inside one item called contributor, which is for the page contains information of how to contribute new data models. It turned out the word 'contributor' causes significant confusion in consecutive three user tests, all these users thought it is the page about who are the contributor



of repository, for example, a list of people. Therefore, the term was modified immediately, and it worked well in following user tests. Otherwise, no other obvious errors are found. Instead, participants turned to come up with more suggestions and their preferences during the second-round tests. As for the mobile use, all participants tended to browse through the repository instead of just finishing two tasks. Almost all participants provided positive feedback and complimented is quite nice and clean. Only one mentioned he never searches data models by mobile, so he does not mind what it looks like.

Furthermore, more ideas and suggestions came from interviews, in addition, there were valuable thoughts discovered during performing tasks. All these results were collected to illuminate the next iteration design. Table 9 below lists the all ideas with the frequency of these mentioned by participants.

**Table 9. The gathering of ideas and suggestions from interviews**

<b>Ideas/Suggestions</b>	<b>Frequency</b>
More advanced search function (search in deeper level) will be great	3
Group or categories the data models, or put labels	2
Model list could be presented in another way, such as photo gallery	2
More automatic contribute way could be considered	2
Navigation bar should be narrower	1
Display the multiple versions in data models	1
Display the contributors' name in data models list	1
Download all files of one data models	1
Clearer information (screenshots) of how to contribute is needed	1
The banner on homepage occupies lot of space in mobile version	1
The description and URL in UCI model should be collapsed as others	1

Apart from those thoughts, there were other positive findings and comments from the tests, which also presented the users' perspectives.

**Table 10. The positive findings and comments found from interviews**

<b>Positive findings</b>	<b>Frequency</b>
--------------------------	------------------

The design is clean and easy to use	6
The repository works quite nice on mobile phone	5
It is easy to perform task on mobile phone	4
Search function works quite nice, since you do not have to click any button	4
The instruction of how to contribute is clear and helpful, it is much better than last time	3
Everything is clear in the repository	3
The code part in data models is impressive and helpful	3
I like this design style, because I like box a lot	1

From the interviews in the second round, mostly different ideas showed up based on the design of repository. While additional features, some participants suggested different ideas related to the layout of the data models, other information need to be labeled. In general, nearly all participants agreed with the design of the repository and everything worked well and clear. Compare to the first test, participants had better feedback in terms of navigation and contribute page. They thought these features were helpful. They solved usability problems according to the success rate of task. Therefore, during interviews, participants tended to play with new open ideas rather than struggle with drawbacks of the repository. At this stage, every participant showed the different preferences.

Furthermore, another result was the grades gathered from the user satisfaction form and demonstrated as below.

*Table 11. The grades from user satisfaction forms in the second tests*

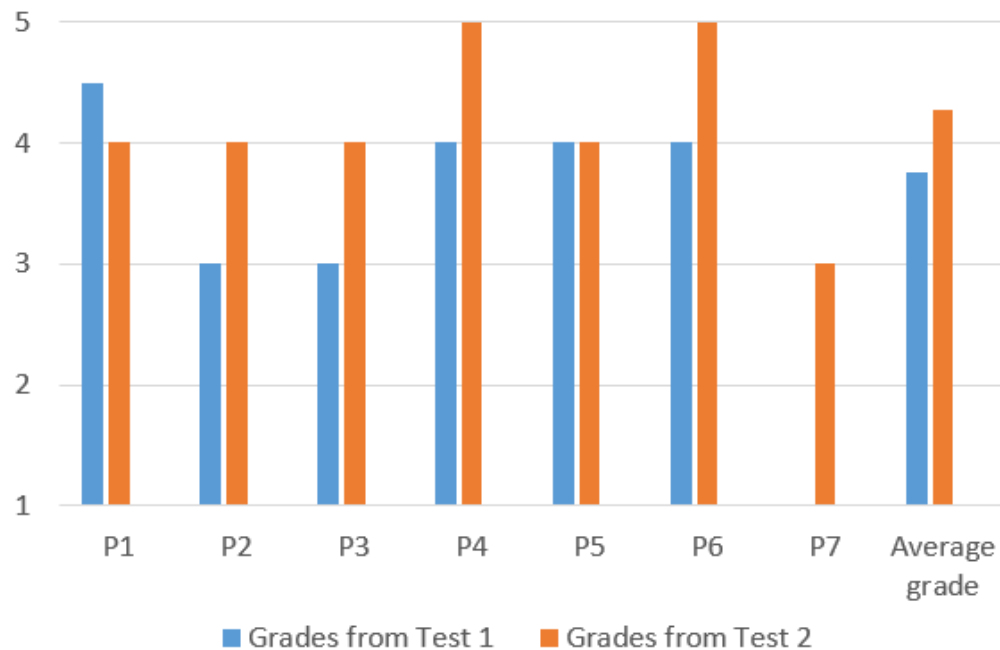
Participants	P1	P2	P3	P4	P5	P6	P7
Grades	4.9	4	4	5	4	5	3

**Average grades: 4.27**

During filling the form, one participant marked the repository as “not easy to use”, then he kindly explained it is only because he is very new to pull request, so he needs more information about contribution. By contrast, other participants did not feel it is problematic to send pull request.

On the other hand, two participants only have experience with one standard data model (LWM2M) in work, so one found difficulty in comprehending the value of the repository. Likewise, another thought is more effective for him to search on official website for models. Despite that, they still completed the task quite smoothly and believed the repository is easy to use and clean in term of user interfaces. Moreover, due to various experience, users may have different understanding and opinion about the repository. For example, one participant has a preference of the website which full of information, so at some point he reckoned the white space of repository should be occupied with more information.

Based on the experience with two tested systems, the same six participants gave the grades respectively while the seventh participant only attended the second one. The grade can be given from one to five, which based on the overall using experience from participants; the results from two rounds of user test are shown in Figure 5-8.



**Figure 5-8: The results of grades by same participants in two rounds user tests**

It is apparent that the grades in second user tests is improved compare to the first one. two participants gave full grade in the second tests while most marked at 4. The average grades are 3.75 and 4.27 respectively. As for the last participant, he only attended the second test and he marked the grades as three. However, he explained that it is merely because there should be more information about how to contribute, for he is not familiar to the pull request, otherwise, he would like to give a better grade. As for most participants, the functional repository is more practical and attractive, which provides better user experience compare to mockups.

In conclusion, according to the presented results and comparison of grades in user tests, the second version of the repository mockup is more usable than the previous mockups, for both user interface and functionality. More specifically, the new repository is easier to use because of a more explicit navigation and overall visual design. The design of repository is consistently regard as clean and straightforward. Due to a better user interface design, users can perform tasks effortlessly and efficiently. Moreover, the additional features facilitate the users' work with data models.

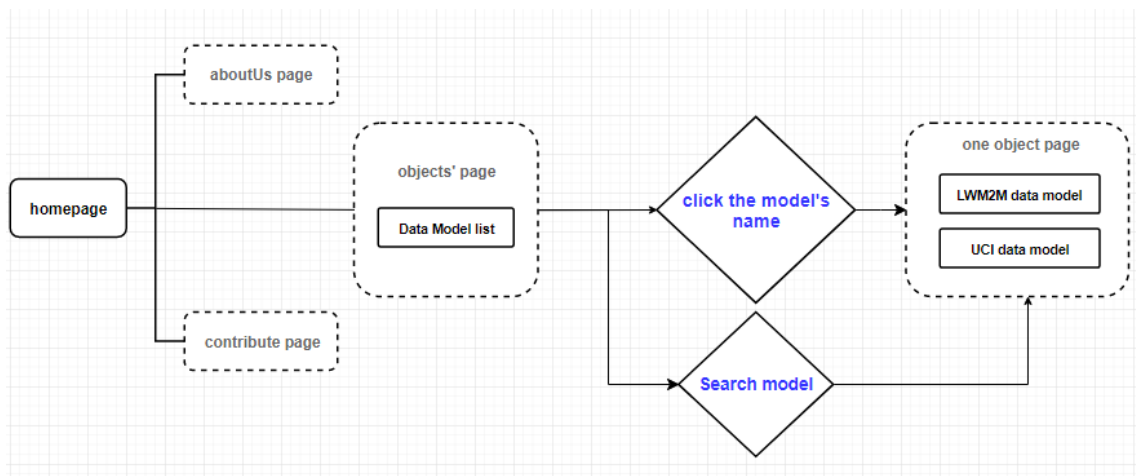
More importantly, most participants thought the repository works well on mobile phone, and they agreed it is easy to perform tasks and it looks nice. Almost all participants complimented the mobile version, except one mentioned he never searches data models by mobile, so he does not mind what it looks like. Moreover, majority of participants appreciated the appearance of repository and thought it is appealing and attractive.

## 5.5. The final design

The new functional meta model repository was designed based on the results from the user tests. This chapter mainly introduces the final design of the repository, includes of the improvement of the navigation, and other re-design part. All these design work have been evaluated in the second round of user tests. Moreover, this chapter discusses the additional features with the reasons behind it.

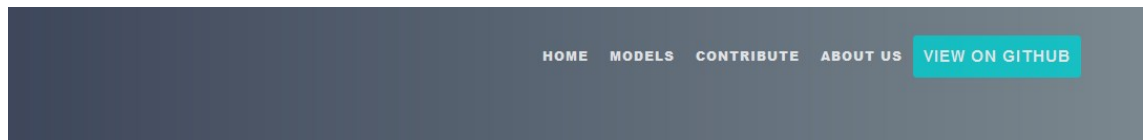
### 5.5.1. Re-design of UI

The most significant improvement started from the navigation part, according to the feedback from users, a new navigation is applied in the repository to make. It is the widely used one in most websites; therefore, it does not take much effort for users to learn. Figure 5-9 illustrates the changes of the structure of the repository.



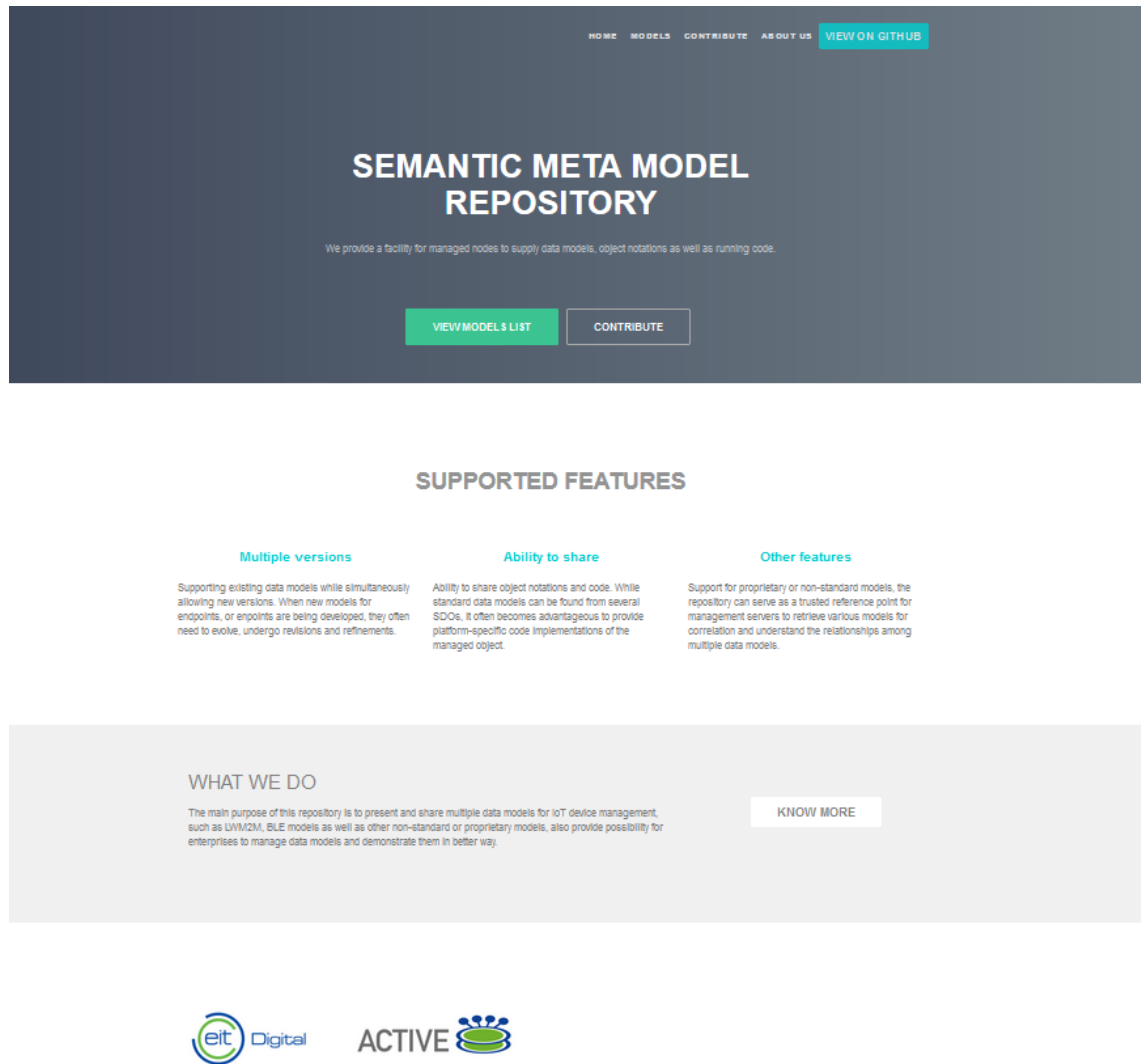
*Figure 5-9: The new design of UI structure*

It turns out that the most flow was kept as in mockups while the navigation part was modified. More specially, two more pages were introduced in the repository to help users with more information. A search function was added to the repository and download of files is allowed within in data models (which not shown in the Figure 5-9). Therefore, the most prominent updated is navigation part, which shown in Figure 5-10.



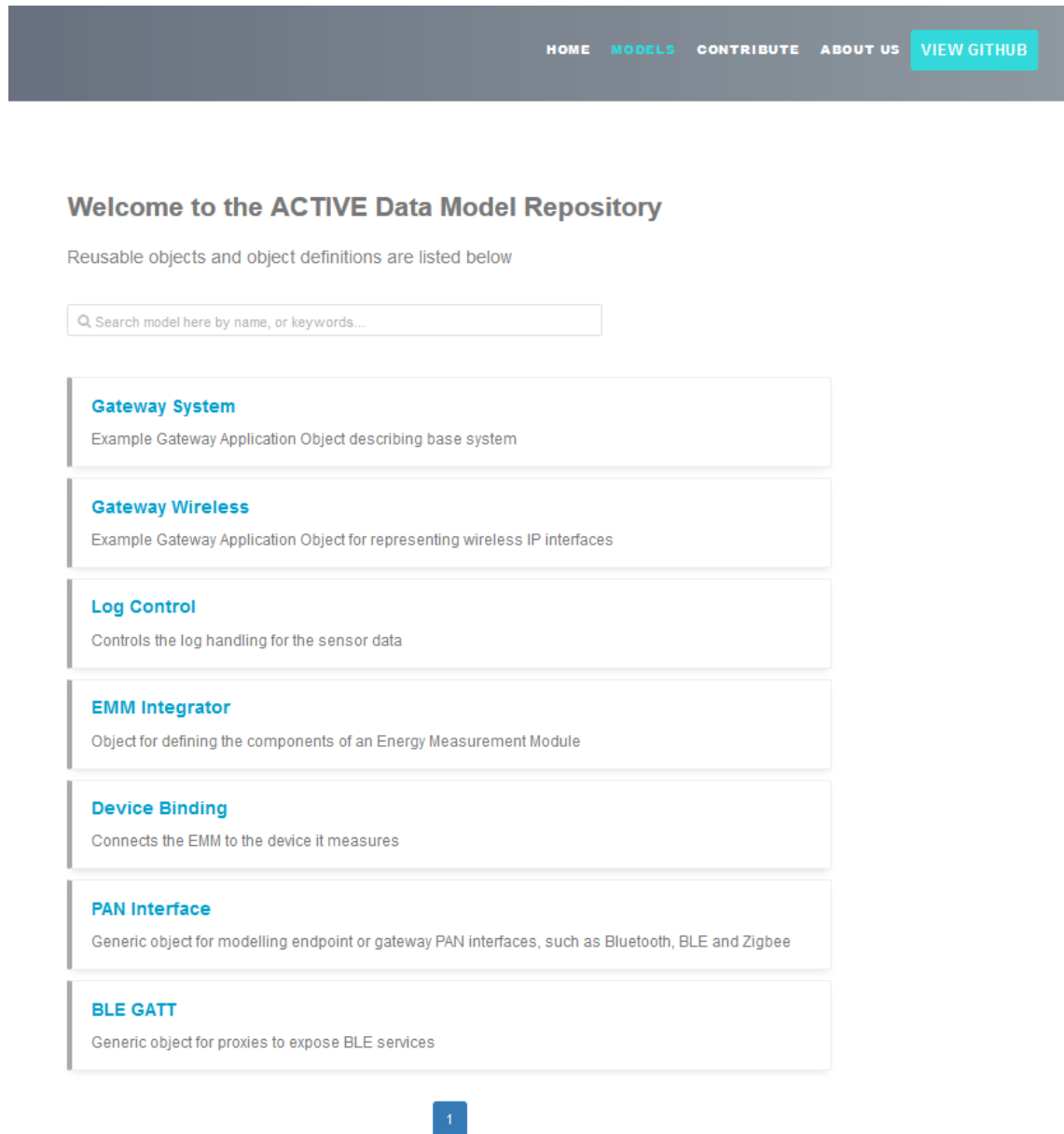
***Figure 5-10: The design of new navigation bar***

As it shown above, the new repository consists of several pages by reorganizing the information of the previous homepage. Compare to the mockups, there are only a few entrances of these pages stay in the homepage instead of showing everything without priority. In addition, the homepage displays the main features of the repository. Figure 5-11 presents the design of homepage.



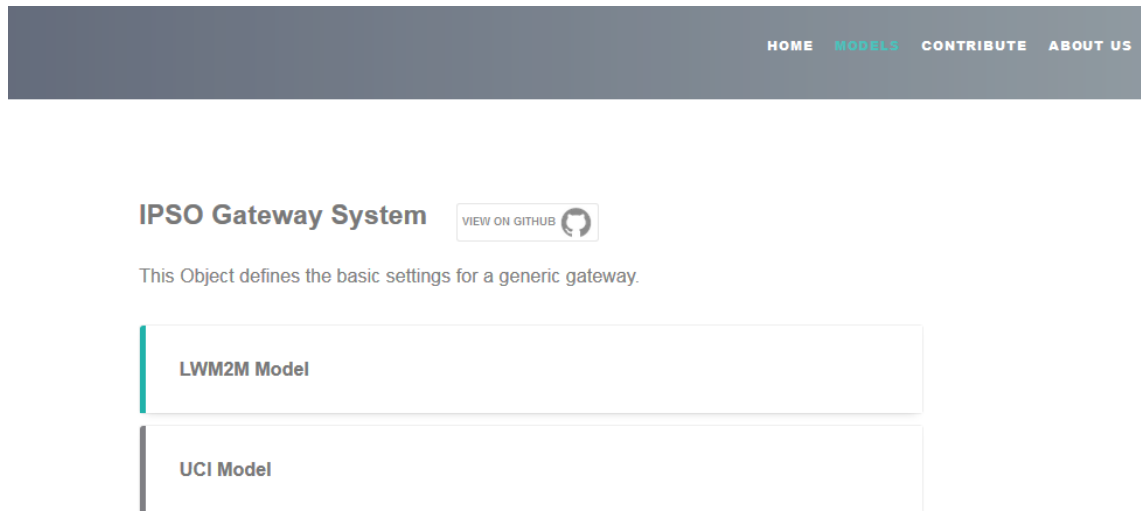
**Figure 5-11: The design of homepage**

In total, there are four main pages in the repository, homepage, model page, contribute page and about us page. More precisely, model page contains all objects, contribute page shows the information of contribute work while about us page includes basic background information. Moreover, there is a button named “View GitHub” in every page, which connects the webpage and GitHub repository. Another supplement is in model page, by considering of more data models will come in future, the pagination is added as shown in Figure 5-12.



***Figure 5-12: The design of model list with the pagination***

Upon clicking one object in the model list, the link goes to the page with multiple data models, which displayed in Figure 5-13. More specification and description are available on this page for the object.

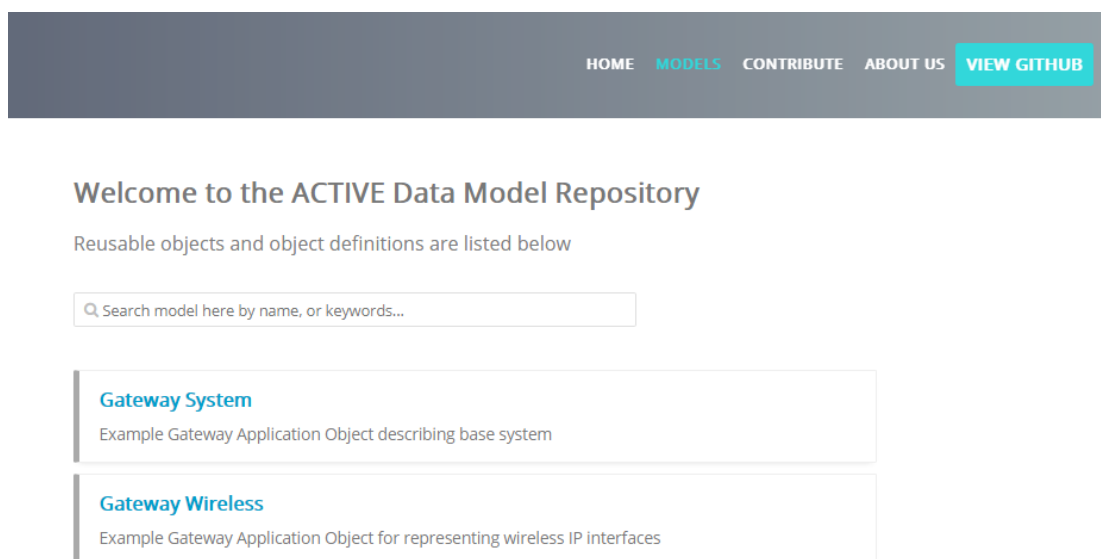


**Figure 5-13: The page of multiples data models for one object**

For the whole repository, the design style is simplified and elegant. All design elements try to be straightforward to make the repository easy to use. More importantly, the simplicity of user interfaces can help users focus on their goals without any interruption. Additionally, the selection of color for different elements is according to their priority, for instance, the entrance of model list on homepage; it is an obvious clickable button with bright color, for it is call to action button and expected to attract user's attention at first.

### 5.5.2. New features

According to the proposals from user tests, two new features were implemented. The first and the most significant addition to the repository is the search function. Figure 5-14 displays the design of the search bar.

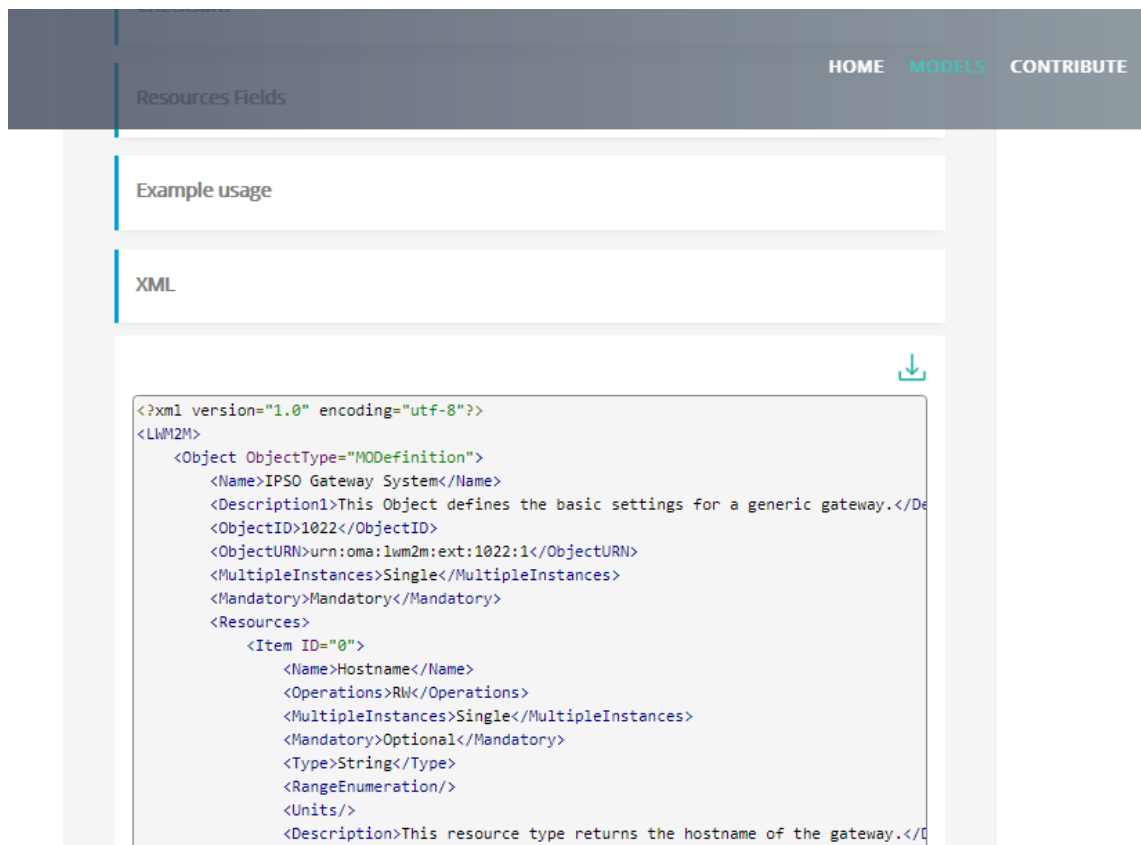


**Figure 5-14: The design of search bar, which above the model list**



With search function, users can find the data models by either name or any keywords of models. It definitely facilitates the browsing work when developers go through in plenty of data models, especially when they do not have any idea about the name of data model. In that case, they can try by keywords. Moreover, the function turned out worked quite well in the user tests. Through typing the word, the result will show up without clicking any search button or pressing Enter button on keyboard.

Another feature is download of files. Based on the user tests, there is a need for downloading files among developers. When it comes to the LWM2M data model, the XML file or JSON file is used mostly in participants' work, so it would more efficient if they could download it. Figure 5-15 shows the design of the download function, and another download button could be found in JSON session in same place.



**Figure 5-15: The design of download function of xml file**


In addition, more information about contribution work is provided on both web page and GitHub repository. More specifically, it is the contribute page which is presented in Figure 5-16. Since the contribution is mainly executed on GitHub platform, the provided information includes the introduction of pull request, and a process is shown to clarify the contribution work. Moreover, there is information about the enterprise user, and a guide document of maintenance work is provided as hyperlink.

[HOME](#) [MODELS](#) [CONTRIBUTE](#) [ABOUT US](#) [VIEW GITHUB](#)


## Let's contribute

Currently, since the Meta Model Repository is hosted on GitHub, contributions can be done by pull requests. In this way, contributors can submit object definitions, examples and implementation code to the Meta Model Repository.


### Why pull request



Universal platform for software development, no extra account needed




Quality guarantee for data models by merging request with checking



Facilitate in communication among contributors and repository maintainers

### How to contribute

Once contributors have the domain-specific data model, they could follow steps as below to contribute:



- 1  
login with your GitHub account
- 2  
find our repository and *fork* to your own
- 3  
create a new folder of data model under Model\_List folder, add files e.g. XML, JSON, code and send *pull request*
- 4  
wait for merging

! The minimum requirement is XML file when adding LWM2M data models

Once a pull request is approved, the contributor should be able to see the models on the web page of the repository soon. Let's [get started](#).

### For Enterprises use

Additionally, enterprises or originations could fork the repository for their own use. After forking, you can customize repository purposely. Since the limitation of GitHub page, for example, no database could be attached, at this moment you need to maintain the new models which from pull request by creating html page manually, more instruction and template of html code can be found [here](#).

**Figure 5-16: The page of contribute**

The final designed repository was evaluated in the second round of user tests. All new features worked quite well and received positive feedback. Indeed, they facilitate the users' work not only in terms of browsing the data models, but also help in contributing work. Therefore, the repository becomes more usable and valuable. Meanwhile, it aims to provide appealing user experience for different groups of users.

## 5.6. Summary

The final repository was implemented based on the results from two rounds of user tests. Based on the first user test, some usability problems had been discovered and analyzed. These problems mainly addressed in the homepage, which full of text and the entrance of data models was hide in the text. For such an informative website, users tend to want a more explicit navigation and they prefer more functions related to the data models, for example, search and download of file. By observing user perform tasks, it turned out most users missed the UCI data model because of bad design. They also commented the resource field of LWM2M data model is hard to read. Therefore, the major problems of the UI of mockups can be categorized into three groups as follows:

- Poor design in terms of navigation, display of data model, unobvious entrance and confusing information.
- Lack of detailed contribute information.
- Need of related functions.

Moreover, other ideas based on personal preference were mentioned. However, the overall appearance of repository seemed simple and nice for users and they all thought it is valuable. Another iteration design was guided by these results and feedbacks. Clearly, the UI plays an important role in helping a user accomplish tasks and ease the use of system. Thus, the new repository was trying to solve the existing problems with a new design. First, it started with an update of navigation and the overall visual design was made in clean and elegant style. In addition, the required functions and information were provided.

According to the second user tests, the repository worked successfully in terms of navigation and additional features. Users also like the design and agreed it is easy to use. They also pointed out more features such as grouping of data models, label the name of contributors, they will be concerned inn future since they do not affect the current use of system. More important, the mobile version also worked well and received positive feedback. Another compliment is from search function, some users thought it work nice since they can get the results immediately without clicking of button. That reflects how detail can affect the use of system and users do care about the design.

Overall, the repository is more usable and satisfied after improved with the help of users. Through observing and discussing with real users, more understanding could be obtained. Hence, the product can be shaped by the real users instead of designers' imagination. However, different users may have distinct or even opposite opinions based on their previous experience. It is significant to figure out the priorities and communicate more with users to learn. Therefore, a final product can be made a success by keeping users in mind and achieving users' potential needs.

## 6. DISCUSSION

In this section, a discussion of the overall research work is presented. It mainly consists of reflections of the research questions and related limitations during research work. Additionally, this chapter points out the direction for future work.

### 6.1. Reflections of the research questions

The thesis work took nearly six months, with the design process of the repository lasting for three months. The target of this thesis was to propose a solution of sharing and distributing standard and non-standard data models for IoT devices. In this thesis work, a meta model repository was designed and implemented to achieve this goal. Since the repository can serve for different user groups - developers, contributors and enterprise users, designing a usable user interface for these three user groups was another objective of the thesis. In total, two research questions were resolved in this thesis work as follows.

#### 1. *How to share and distribute standard and non-standard data models for IoT devices?*

This question was considered by starting with an understanding of the concept of data model and relevant terms in IoT device management. Then based on the exploration of various data models, diverse tools for data model management are shown. Therefore, a meta model repository was purposed for a unified data model management platform. To help a better data model management, the repository is supposed to be a collaboration platform, which can facilitate the work of different user groups. Through analyzing and comparing existing collaboration platforms, GitHub turned out to be the most suitable one for the repository. It allows the repository to be developed efficiently and effortlessly in a short period. Thus, the repository was developed as GitHub pages with a Bootstrap UI framework. It can provide a collaboration platform for developers browsing data models while they can contribute data models for sharing. Meanwhile, enterprise users can benefit from the repository by easily having their own platform for presenting data models for IoT devices.

#### 2. *How to design a usable user interface for data model repository for developers, contributors and enterprise users?*

Since the repository was designed as webpages hosted on GitHub, a usable user interface was also expected to be provided. Through applying a UCD approach, especially with two rounds of usability testing and semi-structured interviews, the repository was totally built on the needs of users in terms of UI design and functionality. More concretely, the first user test aimed to know the user better by evaluating the existing mockups. Based on the results and feedbacks, the new version of repository was made as the high-fidelity

prototypes. Moreover, it was designed by taking principles and guidelines of UI into account. However, due to the challenges in reaching real users, the repository was implemented without testing of prototypes. Therefore, it was the functional repository tested in the second round of user tests. Through observing users perform tasks and interview session, the repository seemed easy to use and helps users to complete tasks efficiently. Moreover, the general UI design of the repository is engaging and promising according to users' feedback. Furthermore, the final repository was designed iteratively based on some suggestions and comments from users. By constantly involving real users in the design work, a UCD method significantly helped in creating a usable product with satisfying user interface for specific user groups.

Overall, the outcome of this thesis work generally succeeded in achieving two goals according to research questions. From the perspective of the repository, it will contribute to a better IoT device management through improving the way of managing data models. By means of encouraging contribution of data models, it will come to be more valuable and significant. Besides, it does present data models more attractively for developer to browse. However, the data model is still displayed in the form of plain text. In future, there could be more user interaction designed for developers to manipulate the IoT devices in a more innovative way.

## **6.2. Limitations**

During the research work, some limitations existed in both exploration phase and design process. The first limitation came from the literature review. There is limited research work has been done in data model management in IoT development. Instead, most literature introduced the different platforms and organizations, respectively. Therefore, there is not so much inspiration that this work could be built on.

Afterwards, in user centered design process, at the beginning, the target user groups are defined into three types, developers, contributor and enterprise users. More specifically, the first group – developers, which means those who work with multiple data models, and need to search them for implementation. Meanwhile, they could be contributors when they want to share data models. Since the target user is not restricted in one special group, they could be anyone who works in IoT development, particularly working with multiple data models. That means the real user can be worldwide and vary widely. Thus, it is hard to reach end users and no user research was conducted at the early stage. Moreover, the repository was developed based on an untested prototype. It was built as GitHub pages and it is easy to modify and update.. Nevertheless, it is still not recommendable to start implementation before performing testing with users by prototypes, especially for these products with more complicate information architectures. It is always wise to contact user as soon as possible, and test with prototypes before putting the effort and cost to develop the system.

Another limitation derived from the selection of participants. All participants in user study are male users, which may generate the different suggestions, since gender can have some effects in design work. Additionally, the user tests only aimed for two user groups. Due to the forking feature of GitHub, there is no test of third user group involved, which may lead to some misunderstanding of the usage of enterprise users. Altogether there was a small sample size of the user tests, which included 6 and 7 participants respectively in two rounds. It seems inadequate to reflect the integrated concept model of the repository from such a small user groups. However, all these user tests did provide valuable feedback and help in discovering usability problems. They also indicated the expectation of repository of real users. More importantly, talking with end users helps learn what they could benefit from the repository and illuminate the work of further development.

Apart from the limited number of users, two inexperienced users were involved in the test. Both of them only worked with LWM2M data models. Therefore, they saw no need to search other models for IoT devices. That led to some misunderstanding and confusion during user tests. Since they have difficulties to understand the value of the repository, it definitely caused the results of deviation. Indeed, choosing correct users undoubtedly ensures the valuable feedback of the test target. Thus, verifying if users are end users when inviting them to a user test is always significant. In future design work, the personas could be applied during the process, which is one of UX technique to help keep the end user in mind. After all, the core theory behind the UCD method is to focus more on the real users and their actual needs throughout all design process.

The last limitation is about the repository itself. Due to the selection of pull request, the new data models need to be configure in the HTML file manually instead of updating on webpage immediately. Hereby, there are still work of configuration for the operator of repository. Based on all these limitations, the design of the repository concentrated more on creating better general user experience. However, most of limitations are expected to be addressed in the further research work.

### **6.3. Future research**

In general, the elementary implementation of the meta model repository has been completed, especially the overall structure was established. Therefore, the part of future work will concern to extend the repository in terms of fulfilling more content. Currently, the repository only supports for LWM2M and UCI data models. More examples of data model from other standards will be introduced soon. For example, BLE data model, which has a similar structure as LWM2M but different forms of resources. In BLE, the definition of data model is described with profile and services; one profile consists of multiple services while several characteristics are contained in one service.

Despite the variety of data formats, it still can be presented in the meta model repository through a consistent user interface. With more data models coming, the design of repository may need to be updated according to the condition of various data models. More concretely, a variety of data models could be categorized and labelled if required. The layout of the model list could be modified based on the suggestions from the user tests. Two participants suggested changing the view to the pattern of photo gallery so that there will be more data models shown on one page compared to current design. Additionally, the search function may have more choices and filters to help users explore the data model more efficiently.

Aside from the design work, the contribution work in the repository still needs to be improved. As in the latest version, all the data models can be contributed through pull requests; the contributor submits the definition and files of data model in a pull request and waits for the submission to be merged. Nevertheless, since there is one customized HTML file for each object. For the operator of the repository, it is laborious work when they have to configure multiple HTML files of new object models manually. To tackle this problem, one possibility is to build some tool for generating the HTML code; it is not unrealistic but the work quite time-consuming since all the HTML files are customized. Another approach is using markdown, since there are some online tools available to convert the markdown to customized html page. Therefore, the contributor can be required to submit the data model in specific form of markdown, which could facilities the work of others. However, some work still need to be done in terms of contribution if the meta model repository aims to serve a more practical use.

Another valuable purpose of the meta model repository is Machine to Machine usage. In IoT development, the repository could serve as a platform for server to inquire the unknown data models. Technically speaking, when application or server search for the data model, it can send the request to the meta model repository. The definition and specification can be retrieved effectively. In this case, most work is addressed in the utilization of API of GitHub and programming. Moreover, the usage of the mobile phone version of repository is enlarged, since the phone itself occasionally can work as a server and through downloading the file on the phone directly, more work could be completed efficiently and straightforwardly.

To sum up, more than provide a solution of data model management of IoT devices, the meta model repository also demonstrates another interactive way for displaying data models. In future, there might be more appealing interactions with data models in IoT development with emerging technologies.

## 7. CONCLUSION

This thesis work primarily contributed to a better solution of data model management in IoT development. It started with defining the fundamental terms and concepts in IoT device management. Then it explored the statement of the IoT device management by presenting diverse standards and organizations of data models. Through studying existing various data models, a repository was anticipated to serve as a collaboration platform for data model management.

Moreover, this thesis introduced and compared several existing collaboration platforms before selecting GitHub. To design a usable repository for three target users, a user centred design method was being applied in the research process. Meanwhile, related methods such as usability testing, high fidelity prototyping were indicated in this thesis. More specifically, two rounds of user tests and results were presented explicitly. Based on all these UX techniques, a repository with both appealing visual effects and usable user interface has been implemented efficiently.

Furthermore, the upcoming use of repository has been pointed out. It intends to support more types of data models and provide more valuable usage in Machine to Machine side. More importantly, this thesis is not merely focusing on developing a repository for managing the data models, it also aims to motivate and prompt a superior device management by demonstrating other possibilities of visualization of data models. Indeed, there could be even more stunning way to present and manipulate data models with the advent of new technologies in future.



## REFERENCES

- James, A., Cooper, J., Jeffery, K., & Saake, G. Research Directions in Database Architectures for the Internet of Things: A Communication of the First International Workshop on Database Architectures for the Internet of Things (DAIT 2009).
- Abras, C., Maloney-Krichmar, D., & Preece, J. (2004). User-Centered Design. In Bainbridge, W. Encyclopedia of Human-Computer Interaction. Thousand Oaks: Sage Publications.
- Abidi, F. (2013). Cloud Servers vs. Dedicated Servers – A Survey. IEEE.
- Anindhita, V., & Lestari, D.P. (2016). Designing Interaction for Deaf Youths by Using User-centered Design Approach. (Case Study: Educational Media for Learning English as Foreign Language). Published in: Advanced Informatics: Concepts, Theory And Application (ICAICTA), 2016 International Conference On. IEEE.
- Bandyopadhyay, S. & Bhattacharyya, A. 2013, Jan, "Lightweight Internet protocols for web enablement of sensors using constrained gateway devices," 2013 Int. Conf. Comput. Netw. Commun., pp. 334–340.
- Barnaghi, P., 2014. Semantic Technologies for the Internet of Things. Institute for Communication Systems (ICS). University of Surrey, Guildford, United Kingdom.
- Bjorklund, M. (2010). YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). Request for Comments: 6020.
- Burzagli, L., Billi, M., Gabbanini, F., & Graziani, P. (2004). The Use of Current Content Management Systems for Accessibility. Springer.
- Bähr, B., & Möller, S. (2016). Blended Prototyping (pp. 129-160). Springer International Publishing Switzerland.
- Blischak, J.D., Davenport, E.R., & Wilson, G. (2016). A Quick Introduction to Version Control with Git and GitHub.
- Castro, M., Jara, A.J., & Skarmeta, A.F.G. (2013). Smart Lighting solutions for Smart Cities. 27th International Conference on Advanced Information Networking and Applications Workshops.
- Dumas, J. S., & Redish, J. C. (1993). A Practical guide to usability testing. Norwood, NJ: Ablex.

Elgazzar, M.H. (2015) Perspectives on M2M protocols - A comparative study between different M2M protocols. 2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS'15).

Galitz, W. (2007). *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*.

Giner, p., Fons, J., & Pelechano, Y.V. (2008). October. "A Domain Specific Language for the Internet of Things", 13th The Conference on Software Engineering and Databases, vol. 2, no. 3, pp. 7-10.

Gomez, C., Oller, J., & Paradells, J. (2012). Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. *Sensors* 2012, 12(9), 11734-11753.

Gong, J., & Tarasewich, P. (2004). Guidelines for handheld mobile device interface design. In *Proceedings of the 2004 DSI Annual Meeting*.

Goroshko, N. (2014). *Content management system (CMS)*.

Gousios, G., Pinzger, M., & Deursen, A. (2014). An exploratory study of the pull-based software development model. Published in: *Proceeding: ICSE 2014 Proceedings of the 36th International Conference on Software Engineering*. Pages 345-355. New York. ACM.

Hacker, W. (2013). *Mobile Prototyping with Axure 7*. Published by Packt Publishing Ltd.

Hassenzahl, M. (2008). User experience (UX): towards an experiential perspective on product quality. *Proceedings of the 20th Conference on Interaction Homme-Machine*. Pages 11-15. ACM.

Hassenzahl, M., & Tractinsky, N. (2006). User experience – a research agenda. *Behaviour & Information Technology*, Vol. 25, No. 2, March-April. Pages 91 – 97.

Howison, J., & Crowston, K. (2004). The perils and pitfalls of mining SourceForge.

Hu, Y., Nanda, A., & Yang, Q. (1999). Measurement, analysis and performance improvement of the Apache Web server. *Performance, Computing and Communications Conference, 1999 IEEE International*.

Jimenez, J., Koster, M., & Tschofenig, H. (2016). *IPSO Smart Objects*. Position paper for the IOT semantic interoperability workshop, March 2016. US.

Jiang, J., Yang, Y., He, J., Blanc, X., & Zhang, L. (2017). Who should comment on this pull request? Analyzing attributes for more accurate commenter recommendation in pull-based development. *Information and Software Technology*. Volume 84, April 2017, Pages 48-62.

Lanubile, F., Ebert, C., Prikladnicki, R., & Vizcaíno, A. (2010). *Collaboration Tools for Global Software Engineering*. IEEE Society.

McDonald, N., & Goggins, S. (2013). Performance and participation in open source software on GitHub. Published in *Proceeding: CHI '13 Extended Abstracts on Human Factors in Computing Systems*. Pages 139-144 ACM.

Miles & Gilbert, (2005). Fylan F. Chapter 6: Semi-structured interviewing. In J. Miles & P. Gilbert (Eds.), *A Handbook of Research Methods for Clinical and Health Psychology* New York, USA: Oxford University Press; 2005.

Nielsen, J. (1994). Heuristic evaluation. In Nielsen, J., and Mack, R. L. (Eds.), *Usability Inspection Methods*, John Wiley & Sons, New York, 25-64.

Nielsen, J. (1995). 10 Usability Heuristics for User Interface Design. [Online] Available at: <http://web.archive.org/web/20180125065812/https://www.nngroup.com/articles/ten-usability-heuristics/> [Accessed 25 01 2018].

Norman, D. (1988). *The design of everyday things*. New York: Doubleday.

Otte, S. (2009). *Version Control Systems*. Computer Systems and Telematics, 2009.

Pras, A., & Schoenwaelder, J. (2003). On the Difference between Information Models and Data Models. Network Working Group. Request for Comments: 3444.

Preece, J., Rogers, Y., & Sharp, H. (2002). *Interaction design: Beyond human-computer interaction*. New York: John Wiley & Sons, Inc.

Patel, S.K., Rathod, V.R., & Parikh, S. (2011). Joomla. Drupal and WordPress - A Statistical Comparison of Open Source CMS. IEEE.

Paz, F., Paz, F.A., Villanueva, D., & Pow-Sang, J.A. (2015). Heuristic Evaluation as a Complement to Usability Testing: A Case Study in Web Domain. 12th International Conference on Information Technology.

Perez-Riverol, Y., Gatto, L., Wang, R., Sachsenberg, T., Uszkoreit, J., Leprevost, F.V., Fufezan, C., Ternent, T., Eglén, S.J., Katz, S.J., Daniel S., Pollard, T.J., Kononov, A., Flight, R.M., Blin, K., KAI & Vizcaíno, J.A. (2016). Ten Simple

Rules for Taking Advantage of Git and GitHub. <https://doi.org/10.1371/journal.pcbi.1004947>.

Rajeshkumar, S., Omar, R., & Mahmud, M. (2013). Taxonomies of User Experience (UX) Evaluation Methods. 3rd International Conference on Research and Innovation in Information Systems – 2013 (ICRIIS'13). IEEE.

Rao, S., Chendanda, D., Deshpande, C., & Lakkundi, V. (2015). Implementing LWM2M in Constrained IoT Devices. IEEE Conference on Wireless Sensors.

Robles & Jokela, (2015). Design of a Performance Measurements Platform in Lightweight M2M for Internet of Things.

Roto, V., Obrist, M., & Väänänen-Vainio-Mattila, K. (2009). User Experience Evaluation Methods in Academic and Industrial Contexts.

Rudd, J., Stern, K., & Isensee, S. (1996). Low vs. high-fidelity prototyping debate. *Magazine Interactions*. Volume 3 Issue 1, Jan. 1996. Pages 76-85. ACM New York, NY, USA.

Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). User interface design and evaluation. Pages 3-7. (The Morgan Kaufmann Series in Interactive Technologies). Morgan Kaufmann.

Staff, E. <http://www.wpbeginner.com/opinion/wordpress-vs-blogger-which-one-is-better-pros-and-cons/>. 2018.

Tarasewich, P. (2003). Designing mobile commerce applications. *Communications of the ACM - Mobile computing opportunities and challenges*. Volume 46 Issue 12, December. Pages 57-60. ACM.

Tracey, D., & Sreenan, C. (2017). OMA LWM2M in a Holistic Architecture for the Internet of Things. IEEE.

Thung, F., Bissyande, T.F., Lo, D., & Jiang, L. (2013). Network Structure of Social Coding in GitHub. 17th European Conference on Software Maintenance and Reengineering.

Tayur, V.M., & R, S. (2017). Review of Interoperability approaches in Application Layer of Internet of Things. International Conference on Innovative Mechanisms for Industry Applications. (ICIMIA 2017).

Väänänen-Vainio-Mattila, K., Roto, V., & Hassenzahl, M. (2008). Towards practical user experience evaluation methods. In E. L-C. Law (Ed.), *Proceedings of the*

International Workshop on Meaningful Measures: Valid Useful User Experience Measurement (VUUM), June 18th, 2008, Reykjavik, Iceland (pp. 19-22).

Vredenburg, K., Mao, J., Smith, P.W., & Carey, T. (2002). A Survey of User-Centered Design Practice. April 20-25, 2002, Minneapolis, Minnesota, USA. ACM.

Wilson, C. (2014). Interview Techniques for UX Practitioners: A User-Centered Design Method. Chapter 2. Semi-Structured Interviews. Pages 24-25.

Yu, B., Xu, L., & Li, Y. (2012). Bluetooth Low Energy (BLE) based mobile electrocardiogram monitoring system. Published in: Information and Automation (ICIA), 2012 International Conference on. IEEE.



## APPENDIX A: CONSENT FORM

### CONSENT TO RECORD A USABILITY EVALUATION

We are kindly asking you to participate in a usability testing for a repository evaluation while is part of project work in HII ACTIVE Task 4 Semantic Meta Model Repository. Meanwhile, this is also part of my thesis work to explore the usability of repository and GitHub platform, which is mainly supervised by Prof. Kaisa Väänänen in Pervasive Computing department in Tampere University of Technology. By participating in the evaluation part, you will help me to figure out exist usability problems, through which the repository could be improved in better way and users could benefit from good user experience in the future using.

You will be asked to complete several tasks about functionalities of repository during testing. Later there is an interview include some questions for you as well. The whole part will be recorded and all recordings will be destroyed after the thesis work has been done. Moreover, a summary of the main results will be reported and analyzed anonymously in the thesis, however, other personal details will not be revealed.

You can stop participating in the evaluation at any point. I am happy to answer, if you have any questions during testing. Here is my contact information and I really appreciate your participation.

- Hanning Zhao

- hanning.zhao@tut.fi

**By signing this form, you will accept the above terms.**

Date and place: \_\_\_\_\_

Signature: \_\_\_\_\_

Name clarification: \_\_\_\_\_





**APPENDIX B: USER BACKGROUND QUESTIONNAIRE****BACKGROUND QUESTIONNAIRE****Background Information**

**Gender:**     Male     Female

**Age:**

Under 20

20 to 30

30 to 40

40 to 50

More than 50

**Occupation**

Student

Employee

Self-employed

Unemployed or on leave

Else:

**Education:**

High school

College

Bachelor degree

Master degree

Ph.D.

Else:

**Working experience**

Less than 1 year

1 to 5 years

6 to 10 years

More than 10 years

---

## APPENDIX C: USER SATISFACTION QUESTIONNAIRE

### USER SATISFACTION QUESTIONNAIRE

Below are some statement related to the repository you tested. Please check the option that best matches you level of disagreement of agreement with the statement.

Evaluate the following statements	Strongly disagree	Disagree	I don't know	Agree	Strongly Agree
The repository was easy to use.	[ ]	[ ]	[ ]	[ ]	[ ]
It was hard to perform the given tasks.	[ ]	[ ]	[ ]	[ ]	[ ]
The appearance of the repository was pleasant.	[ ]	[ ]	[ ]	[ ]	[ ]
I was able to find what I needed quickly.	[ ]	[ ]	[ ]	[ ]	[ ]
The repository included unfamiliar terms.	[ ]	[ ]	[ ]	[ ]	[ ]
It was difficult to navigate within the repository.	[ ]	[ ]	[ ]	[ ]	[ ]
The information provided by the repository is valuable.	[ ]	[ ]	[ ]	[ ]	[ ]
I would like to use the repository also later.	[ ]	[ ]	[ ]	[ ]	[ ]

**Which overall grade would you give to the service (on a scale from 1=poor to 5=very good)?**

\_\_\_\_\_

## **APPENDIX D: TASKS IN FIRST USER TESTS**

Task 1: Choose and open a web browser. Go to website

<https://t4active.github.io/>

Task 2: Find the information about work of contributors

Task 3: Find the data model list

Task 4: Find the LWM2M data model for object “Gateway System”

Task 5: Find out the URI of the UCI data model for object “Gateway Wireless”

Task 6: Find out the Resource Filed of LWM2M data model for object

“Gateway Wireless”

Task 7: Find the GitHub page of this repository



## APPENDIX E: TASKS IN SECOND USER TESTS

Task 1: Choose and open a web browser. Go to website:

[https://hanningz.github.io/SMMRepository./](https://hanningz.github.io/SMMRepository/)

Task 2: Find the information about how to contribute

Task 3: Find out the data model list

Task 4: Locate the LWM2M data model of object “Gateway System”

Task 5: Search for the data model of the object “Device Binding”

Task 6: Find the URL in UCI data model of the object “Gateway Wireless”

Task 7: Search the LWM2M data model for the object with ID: 25010

Task 8: Download the XML file in LWM2M model of “Gateway System”

Task 9: Contribute new data model according to the instruction for contributor, just use the XML file you have downloaded as example file

Task 10: Find the GitHub page of this repository



## APPENDIX F: QUESTIONS IN INTERVIEWS

1. Was there anything you were expecting to see which was absent
2. Is there anything you dislike about the repository? Why
3. How did you feel about the appearance of the repository?
4. Were you able to find what you were looking for?
5. How do you feel about the navigation?
6. Would you like to use this repository again? Why?
7. Is there any function you would like to see in this repository, why?
8. Any other suggestions or comments on it? Is there anything you are thinking now, or any comments on test, about this kind system?