International Journal of Engineering & Technology, 7 (4.31) (2018) 160-167

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Domain Ontology for Eliciting Usability Features

Chian Wen Too[1], Sa'adah Hassan[2], Abdul Azim Abdul Ghani[3], Jamilah Din[4]

[1]*Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, Malaysia*
[2]*Lee Kong Chian Faculty of Engineering & Science, Universiti Tunku Abdul Rahman, Malaysia*
**Corresponding author E-mail: cwtoo@yahoo.com*

Abstract

One of the crucial factors that can influence user preferences of software is usability. It assesses the extent of which software able to facilitate user and use the software easily and effectively. Typically, usability requirements are specified at the design stage of software development due to its characteristic that is subjective in nature and hard to be elicited at the early stage. As a result, the lack identification and improper treatment of usability features always caused the failure of a software product. Consequently, it increased the cost and effort of rework. Essentially, it suggested that usability need to be specified at the requirements engineering stage to complement the software functional requirements. This paper presents a preliminary study on current efforts to support the activities in identifying usability attributes in the software functional requirements. The potential of adopting pattern and ontology for identifying usability features are also presented. The development of domain ontology is proposed to fill the gap of the current efforts where there is lack of usability driven semantic knowledge model to support the usability elicitation tasks. The designed domain ontology is expected to overcome the problems resulted from software developer that lack of sufficient knowledge or expertise in eliciting usability features at requirements engineering level. The main contribution is to provide a guideline to aid the requirements engineer to elicit and specify usability requirements start from the early stage of development phases.

Keywords: *Ontology; Pattern; Requirement Engineering; Usability.*

1. Introduction

The design of the software application is depend largely on the requirements gathered during requirements engineering stage. The requirements can be categorized into two, functional and non-functional requirements. Functional requirements basically concern on the services or features that the system should provide, whereas, non-functional requirements more focus on constraints and desired quality attributes. Usability is one of the non-functional requirements, where users can use a product to their satisfaction in order to effectively and efficiently achieve specific goals in a specific context of use (ISO 9241-11). Specifying the functional requirements is basically more straightforward. Whereas, usability requirements are seems more difficult to be specified. Besides, specifying usability requirements is likely to be beyond the knowledge of the users. Usually, usability requirements are produced based on developers and experts' views as well as referring to usability guidelines.

Human Computer Interaction (HCI) community has defined usability guidelines for non-experts in usability. For example, Shneiderman[1] and Nielsen[2] proposed usability design guidelines that are widely accepted and used as tools to measure usability. However, these guidelines are usually described in such an abstract way that they are difficult to apply (directly) in software development. There are various usability guidelines have been introduced to complement the current technologies and communication devices with different platforms. For example, usability requirements elicitation using graphical notations ([3];[4]) as well as textually ([5];[6];[7]). However, developers need to have knowledge to select the appropriate guideline for the software that they want to develop.

The potential benefits of ontology approach in improving usability elicitation have been discussed in our early work [8]. Domain ontology is possible to overcome the mentioned problems. Besides that, the problems of understanding resulted from a variety of stakeholders expressed their needs in different ways can be reduced by using a common terminology represented in the ontology to promote a uniform communication standards during the usability requirement elicitation. In fact, the construction of domain ontology is to fill the gap in the current approaches or methods where there is lack of usability driven semantic knowledge model to support the usability requirements elicitation tasks.

This paper is divided into the following sections: Section 2 presents the related concepts and research efforts on usability requirements. Section 3 presents the development of domain ontology, followed by its evaluation in section 4. Finally, Section 5 describes the conclusions as well as recommendations towards to the new approach.

2. Related Work

This section highlights related efforts and studies on usability. Critical analyses on the existing approaches are also discussed.

2.1. Functional Usability Features (FUF) and Patterns

Over the past decade, researchers have developed a variety of heuristics, guidelines or approaches to enhance and improve usability in software system. One of the significant work from Juristo *et al.* [7] was generated a set of usability elicitation patterns from HCI literature recommendations. They have treated and integrated those usability features with major implications for software func-

tionality as a kind of functional requirements and named them using the term, functional usability features (FUF). According to the authors, recommendations provided in HCI literature to improve usability of a system can be categorized into three groups of impact depending on their effect on software development including impact on the user interface (UI), impact on the development process, and impact on design [9]. Impact of usability on the UI only affected the system presentation through slightly modifications on the UI components but not on the system core. Usability recommendations with impact on the development process are affected through modifying the techniques, activities or artifacts used during development. While, for impact on the design, it is referring to incorporating certain functionalities that should be provided to user into the software.

Usability has major implications with the functional requirements, thus, some studies suggested that usability should be considered together with functional requirements ([4];[7];[10];[11]). However, it is also mentioned that by using the usability attributes or usability goals to determine usability features is not easy and might lead to ambiguous or incomplete requirement as there are lack of detail descriptions in elicitation and specification [10].

Most of the HCI hueristics or guidelines are too general and need more detail information to properly incorporate the complete usability feature into a software system. As a result, a list of usability features grounded on solid HCI principles has been proposed for identifying usability features together with system functionality ([12];[13];[14]) and was termed as functional usability features (FUF). Based on the selected FUFs, they have further defined and specialized each usability feature provided by different HCI authors into more detailed goals or subtypes and termed as usability mechanisms [7]. These mechanisms are the usability aspects to be considered in software architecture or design and dealing them from the early stages of the development process [15]. Each usability mechanism is defined further with elicitation and specification guidelines respectively. Each generated usability mechanisms is packaged and named as a Usability Elicitation Pattern for knowledge reusable purpose.

Laura Carvajal [16] has proposed a usability oriented software development process which focused on providing software developers with guidance for including FUF into their software applications but their contribution is mainly in the software design phase. They suggested few artifacts for usability elicitation and analysis in their proposed guidelines. Two of the main artifacts were Usability Elicitation Guideline (UEG) and Usability Elicitation Cluster (UEC), used to help analysts in eliciting usability requirements related to each FUF in more structured and tangible way. The output generated from these two artifacts is a set of requirement with usability. The UEG, presented in pattern based format is extended and modified from the original Usability Elicitation Patterns [7]. Their extended UEG is added with additional features, like an *Intent* field, to provide a clearer description of the FUF; an *Interrelationship* field, to explain the impact of inclusion of a particular feature would have on other features; an *Elaboration* column, elaborated the high level HCI recommendations from a software engineering point of view and so on. From the list of issue to be discussed with stakeholders in UEG, a UEC is used to group the related discussion items to a single topic that must be achieved by the system being developed in a sequence ordering [16]. This set of topics represents the core goals and general responsibilities that need to be achieved or fulfilled by the system being developed.

2.2. Usability Requirements Elicitation and Specification

In the past, some efforts have been made regarding elicitation and specification of usability by researchers in software engineering field. The community believed that considering usability started from early stage of software development is very crucial and needed to avoid any rework in the later stage. Therefore, few methods and techniques have been proposed on how to elicit, analyze and specify usability requirements [17]. This section presents the recent state-of-the-art approaches proposed for the inclusion of usability requirement at the requirement engineering process.

The idea of using a reusable knowledge based catalogue to elicit and specify usability requirement was originally came from [4]. According to them, usability requirements should be treated as requirement that capture usability goals and associated measures for a system under development. They adopted i*framework [18] and based on personal experiences to model usability as goals to be achieved through different views of possible alternatives. The general usability goals has been categorized into three sub-goals or attributes used to achieve usability requirements and will be kept refining until an implementable mechanisms that can meet the usability goals achieved. However, the technique used was context-specific which is applicable only to a health care domain.

Rafla *et al*. [19] proposed a usability-driven adaptation of the quality attribute workshop (UQAW) to help the software developer in discovering and documenting usability requirements. They used the usability scenarios proposed by Bass and John [13] in conjunction with Folmer's usability property [20] to facilitate the elicitation of usability requirements process at requirements definition stage by incorporating the workshop as part of the activity of RE process. They mentioned that Folmer's work did not specify in details the way to capture and organize the usability requirements presented. Although empirical study was conducted to evaluate the benefits of their proposed method but all the works need to be done manually, thus, required huge effort from the developers. Similar to Cysneiros *et al*. [4], Roder [11] proposed to use a pattern-based approach in eliciting and specifying functional usability features. The author pointed that usability is not limited to the system user interface only but also should be properly considered in various development activities. Roder [11] has used a catalog of patterns to support usability features selection and an extended use case to specify the selected usability features during the requirement engineering activities. Even though the proposed method is supported by a semiformal specification template, but the use case notation did not help to form a structure to standardize the requirements with consistent syntax during requirement specification. Moreover, not all the 20 functional usability features in their pattern catalogue were proved to give major impact on the system functionality.

While, Ormeno *et al*. [10] proposed to capture usability requirement by organizing the information stored in different guidelines in a tree structure. The analyst will navigate through this structure to ask questions from end users and gather all the answers in order to produce a set of usability requirements. Unfortunately, their approach only focused on the Model-driven Development (MDD) paradigm. In the same year, Rivero *et al*. [21] proposed and developed a set of techniques and a tool to support software engineer to produce descriptive requirement specifications for web based application. Nevertheless, their proposal did not mention the usability features or model to support the elicitation task. Furthermore, Hassan, S., *et al*., [22] proposed an analysis framework to facilitate developers in identifying usability requirements. However, further research need to be done for capturing usability based on user needs.

2.3. Application of Ontology in Requirements Engineering

Ontology is firstly defined as "an explicit specification of conceptualization" [23]. This definition became the most quoted in the ontology community [24]. Later on, some authors have modified slightly Gruber's definition became "a formal explicit specification of a shared conceptualization" [25]. The term formal is referring to the fact that ontology should be machine-readable. The term

explicit means the type of concepts used and the constraints on their use are explicitly defined. While, the term share is reflecting the notion that ontology captures consensual knowledge that accepted commonly and the term conceptualization is referring to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon [26]. Besides, another definition stated that ontology is a representational artifact to specify the meaning or semantics about the knowledge or information in a particular domain in structured form [27]. Another word to say, ontology can link human and computer understanding by using formal and real world examples.

Information is unstructured and only understandable by human being. For instance, there is no way to tell the computer that this document is describing about a person unless it explicitly contains the word person. Therefore, in order to give some kind of intelligence, the computer must understand the meaning or the structure of the document and this involves semantics. However, only the structure is still not enough to reflect the relationship among the entities in a given domain. The content of the document or known as the domain model should be able to reflect the real world too. A domain model can be achieved through conceptualization by simplifying view of the world. In general, ontology can be used to represent a shared, agreed upon conceptualization [28].

Basically, ontology consists of four main components: classes or concepts, relations, axioms and instances [29]. Classes or concepts represents an entity in the domain and provide a way of describing part of the world. For instance, in a school, concepts are Teacher, Classroom, Student, Subject and Timetable. The classes represent taxonomies, which inherit the mechanism that can be applied. The relations represent the association between classes. Ontology usually has a binary relation to define domain and range. For instance, to express a Teacher *hasTeach* Mathematics, the binary relation is the *hasTeach*, the domain is Teacher and the range is Subject.

An axiom is used to specify the constraints, properties and definitions of derived concepts in ontology. A formal axiom represents the semantic of the terms used and infers knowledge of ontology to enable the verification of consistency. For instance, the axiom in a school domain is that it is impossible to schedule two subjects to be taught at the same time in the same classroom. An instance represents the individual or elements in the ontology. They are related to each other through property, a collection of relationship between individual. For examples, the instance of *Teacher* is *Peter John*, a *33 years old male* with *staff id:M101*.

Nowadays, with the emergence of semantic web, there is a growing interest of ontology-driven approaches usage in different domains. In software engineering field, study has shown that there is an increasing amount of research devoted to utilizing ontologies especially in requirements engineering [30]. According to the systematic literature review conducted by them, empirical evidences showing using ontologies in requirements engineering activities is useful in reducing the ambiguity, inconsistency and incompleteness of requirements. Commonly, the main problems happened in requirements engineering processes are due to some factors that contributed to the incomplete, incorrect or inconsistent functionalities defined. For instance, different interpretations of the same requirement by different stakeholders may cause ambiguity in requirements; poor requirements understanding may cause incomplete or incorrect of requirements definition; insufficient specifications due to absence of key requirements [28].

To reduce the problems caused by these factors, the used of ontology in requirements engineering will seem brings benefits. Ontology can be logically reasoned and shared within a specific domain [31]. It is a standard form to represent the knowledge of the application domain. Thus, it can be used in requirements engineering to explicitly model the requirements to enable a consistent way of requirements structuring. Application of ontology especially in the requirements elicitation is helpful in providing some guidance to elicitation realization [30]. Generally, ontology can minimize and resolve the problems in ambiguity between stakeholders by promoting a shared vocabulary and fostering a common understanding of the domain, ensuring the correctness and consistency of the concepts and relationships usage in the domain through validation and support the domain knowledge reusability.

3. Design and Development of Domain Ontology

Domain ontology is a kind of high-level models of knowledge underlying all things and concepts of a given domain [28]. Usually, ontology is formed by concepts (C), properties (R), axioms (X) and individuals (I) and the OWL Ontology is defined as:

$$O = (C, R, X, I) \quad (1)$$

Where:
C = the set of concepts
R is the finite set of properties which consists of datatype Property, Dt_p and object Property, Obj_p. Therefore ontology properties can be further described as:

$$R = (Dt_p, Obj_p) \quad (2)$$

Where:
X = a set of axioms which is expressed in an appropriate logical language.
I = a set of instances of a concept, also called individuals.

In this work, the domain ontology contains the facts about the domain context is developed to provide semantic guidance for eliciting usability features. This domain ontology design and development process consists of four steps (as illustrated in Figure 1) that are adopted from the existing ontology construction methodologies by Noy and McGuinness [32], METHONTOLOGY [33] and SABiO [34]. For illustration, the university online venue booking system is used as the domain. The domain ontology is resulted from the merging of usability ontology and functional ontology. Usability ontology encoades the knowledge about the selected usability features and their relevant elicitation guidelines that have been proven impacting the software functionality. Meanwhile, the functional ontology is designed to representing the core functional activities of the domain to be applied.

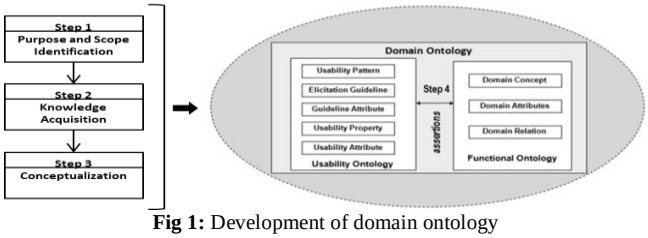

Fig 1: Development of domain ontology

3.1. Identification of Purpose and Scope

The first step of the development of domain ontology is to identify the purpose and the scope. The main purpose is to generate a knowledge model that contains information about a functional domain and their relevant usability features. This study focused on the venue booking or reservation domain. While, for scoping activity, a middle-out approach is used to enumerate a set of important terms that should be included in the ontology. This approach helps to identify the primary concept of the ontology ([33]; [32]).

3.2 Knowledge Acquisition

Knowledge acquisition activity is important to support the ontology conceptualization. In this work, the acquisition of the usability features related knowledge started from the consolidated biblio-

graphic materials such as HCI literature ([2];[7];[16];[20];[35]), reference models , and international standard models like ISO 9126, ISO 9241-11, ISO 25010 (International Organization For Standardization ISO, 2001, 2011; ISO, 1998)[36][37]. Domain experts are also the main sources of the knowledge acquisition. Interviews and brainstorming with the experts from the related domain have been applied. Besides that, the resources such as textbooks, and manuals of previously developed applications were also referred. During the knowledge acquisition process with domain experts, the glossaries enumerated earlier are refined through specialization and generalization depends on the necessity of the terms. This is to ensure that the terms used are relevant, consistent, and complete without duplication. The outcomes from the knowledge acquisition are presented and discussed in the conceptualization phase.

3.3. Conceptualization

Conceptualization is the most important task in the whole ontology development process. The goal of this step is to capture the domain conceptualization and structure them in a conceptual model. All the main concepts and relationships among them should be identified and organized in taxonomies properly. The terms used to refer the concepts and relations in the ontology should be chosen carefully. The main concepts and relations in the domain are defined based on the outcomes from the knowledge acquisition. The following sub-steps show how to capture and structure the domain Ontology.

3.3.1 Define the Concept and the Hierarchy

As discussed in the purpose identification, the knowledge captured for the proposed domain ontology is represented by two core ontologies: Usability Ontology and Functional Ontology. Figure 1 illustrated the elements of the Usability Ontology and Functional Ontology. The Usability Ontology representing knowledge about the functional usability features, their patterns, metrics, properties, elicitation guidelines and guideline attributes. The Usability Ontology is defined based on the OWL ontology and adopted the syntax definition from the Description Logic Handbook [38]. The main concepts for Usability Ontology (UO) are defined as follows:

$$UO_C = (UPAT, EG, GA, UPROP, UM) \quad (3)$$

Where:
UPAT = Usability Pattern
EG = Elicitation Guideline
GA = Guideline Attribute
UPROP = Usability Property
UM = Usability Metric
The Action *ACT*, Entity *ET*, Time_Unit *UT*, Format *FMT*, Method *MTD*, Symbol *SYM*, Quantity *QT*, Operational_Condition *OPCD* is categorized as subclasses in the Guideline Attribute *GA*. Thus, the concept classification is:

$$ACT, ET, UT, FMT, MTD, SYM, QT, OPCD \subseteq GA \quad (4)$$

The descriptions for each concept and sub concepts defined in Usability Ontology are shown in Table 1 and Table 2 respectively.

Table 1: The Concept in Usability Ontology

Concepts	Descriptions
Usability Pattern	representing a set of functional Usability features with major implications on software design [7], are the usability characteristics whose effects go beyond the user interface, used as the source of information for usability elicitation in the proposed Usability Ontology. Providing high level response to a need specify by a usability property.
Elicitation Guideline	adopted and extended from Usability Elicitation Guideline ([7]; [16]). It is formed by a list of issues to be discussed with stakeholders aimed to help requirement engineer in eliciting all the aspects related to a particular functional usability feature. Therefore, each usability pattern has their related usability elicitation guidelines to be specified in detail.
Guideline Attribute	Formed by all the discussions items used in the usability elicitation guidelines to provide guidance to fill in the boilerplates attribute values based on the given boilerplate templates.
Usability Property	Representing the general heuristic and design principles that have a direct influence on usability. Refined from the usability metrics to create a direct relationship to link the usability pattern to their metrics [20].
Usability Metric	Representing the measurable component of usability, high level goals used to achieve usability ([20]; [37]).

Table 2: The Sub Concept in Usability Ontology

Sub Concepts of Guideline_Attribute	Descriptions
Action	A behaviour that is expected to be fulfilled by the system. Formed by combining a verb + noun.
Entity	A separate entity in the domain, different from <Unit>, <Symbol>, <Method> or <Format >. Formed by noun.
Quantity	A numeric value denoting a value.
Time_Unit	The time measurement standard used.
Symbol	The representation of the <Entity>.
Method	The accessing method of the <Entity>.
Format	The presentation format of the <Entity>.
Operational Condition	A condition or event that occurs during system operation.

On the other hand, the functional ontology is used to represent the conceptual structure and core business activities of the application domain. It defines the domain concepts, attributes and relationship among the concepts. In this work, booking or reservation application domain is applied to validate the applicability of the proposed framework. According to Kaiya and Saeki [39], an ontology used in requirement analysis need to be interpreted in the same way by any stakeholder in a specific application domain.
Therefore, in the Functional Ontology, the concepts defined are represented using the terms such as Booking System, any application domain context related to booking or reservation; Actor, users who interact with the system; Function, the functional requirements or business activities of the domain; Object, the entity of the domain different from other concepts mentioned earlier, for instance booking profile, user profile and etc. Similar to Usability Ontology, the Functional Ontology is defined based on the OWL ontology and adopted the syntax definition from the Description Logic Handbook [38].
The main concept for Functional Ontology (FO) is defined as follows:

$$FO_C = BOOK \quad (5)$$

Where:
BOOK = application domain context related to booking or reservation.
The Actor *ATR*, Function *FUNC*, Object *OBJ* are categorized as subclasses in the Booking or Reservation domain *BOOK*.
Thus, the concept classification is:

$$ATR, FUNC, OBJ \subseteq BOOK \quad (6)$$

Where:
ATR = Actor

FUNC = Function
OBJ = Object
The descriptions for each Concept and sub Concepts defined in Functional Ontology are shown in Table 3.

Table 3: The Concept and Sub Concept in Functional Ontology

Concepts/Sub Concepts	Descriptions
Booking/ Reservation Domain	The application domain context related to booking or reservation.
Actor	A person or user who interacts with the system.
Function	The main functional requirements or business activities of the domain.
Object	The entity in the domain, different from Domain and Actor.

3.3.2. Define the Properties of Concepts

Once the classes of concept have been defined, the next step is to define the property of classes. Property is used to define the relationships between the concepts and individuals by describing the internal structure of concepts. In general, there are three types of properties: Object Property, Data Property and Annotation Property. In the domain ontology, the three types of properties are defined but focused more on the object property definition.

a. Object Property

The purpose of defining object property is to describe the relationship between two individual instances. Based on the suggestion from Ontology Engineering, it is recommended to use the prefix: 'has' or 'is' for properties naming in order to improve ontology readability and the completeness of property restriction. Most of the time, the 'is' will be the inverse property of 'has'. For instance, Usability Property Error Management ***isPropertyOf*** Usability Pattern Undo then Usability Pattern Undo ***hasProperty*** Usability Property Error Management. The object property defined in the Usability Ontology and Functional Ontology are given in Table 4 and Table 5 with the descriptions provided.

Table 4: Object Properties Definition for Usability Ontology

Object Property, Obj_r	Descriptions
hasMetric	Usability Property has Usability Metric
hasInterrelationships	Usability Pattern has interrelationships with other Usability Pattern
hasGuideline	Usability Pattern has Elicitation Guideline
hasAction	Elicitation Guideline has Action
hasEntity	Elicitation Guideline has Entity
hasQuantity	Elicitation Guideline has Quantity
hasSymbol	Elicitation Guideline has Symbol
hasOpCond	Elicitation Guideline has Operational Condition
hasMethod	Elicitation Guideline has Method
hasFormat	Elicitation Guideline has Format
hasTimeUnit	Elicitation Guideline has Time Unit
hasEffectToAction	Operational Condition has effect to Action
hasEffectToFormat	Operational Condition has effect to Format
SubClassOf	Action, Entity, Format, Method, Operational Condition, Quantity, Symbol and Time Unit are SubclassOf Guideline Attribute.

Table 5: Object Properties Definition for Usability Ontology

Object Property, Obj_r	Descriptions
perform	Actor perform Function
applyTo	Function applyTo Object

b. Data Property

Data Property is used to describe the relationship between an individual and its data value, or define restrictions on the values of attributes. For instance in the Functional Ontology, the datatype properties are used to describe the individual concepts and their relationship with the literal value. Table 6 shows part of the data properties defined for Functional Ontology.

Table 6: Data Properties for Functional Ontology

Data Property, Dt_r	Descriptions
hasName	An actor has name
hasRegID	An actor has registration identity
hasDate	Booking Profile has booking date information

c. Annotation Property

In the ontology design, annotation property is used to add a human readable label or information for the classes, instances, object and data properties. For Usability Ontology, annotation property is used to provide the additional descriptions for the defined Concepts in Usability Pattern and Elicitation Guideline. For instance in the concept Usability Pattern, the *Intent* annotation property was used to describe the main goal, the *Problem* annotation property described the problem being addressed and the *Context* annotation property illustrated the context applicable of this concept. On the other hand, for concept Elicitation Guideline, the *displayOrder* annotation property is used to display the sequence for each instance of Elicitation Guideline during the specification task.

3.3.3. Define the Facets of the Properties

Facets refer to the role restrictions used to define the object properties and data properties. An object property can have a variety of facets to describe their domains, ranges and cardinality. In the ontology designed process, the domain and ranges are defined. Domain is referring to the set of classes or concepts where the property is attached to. Allowed classes for properties of type instance are called range of properties. Therefore, domains and ranges are used by object property to link the individuals or instance of the concepts. The equations (adopted from Antoniou *et al*., [40])) used to represent the domain and range of object Property in the domain ontology is as followings:

$$\text{Domain } (Obj_{R,} C) = \forall ?\, x, ?y\, (R\, (\, ?\, x, ?\, y) \rightarrow C\, (?\, x)) \quad (7)$$

where $x, y \in C$

$$\text{Range } (Obj_{R,} C) = \forall ?\, x, ?y\, (R\, (\, ?\, x, ?\, y) \rightarrow C_r\, (?\, y)) \quad (8)$$

where $x, y \in C$
$Obj_R = \{hasC_{r1}\}$
Domain = (C)
Range = (C_1)
For instance in Usability Ontology, the object property *hasProperty* links the instance of concept Usability_Pattern, UPAT to instance belongs to concept Usability_Property, UPROP. It means that the domain of object property *hasProperty* is Usability_Pattern and the range is Usability_Property. In other words, the domain and range of an object property are referring to the concept and their related concept.
Obj_R = {hasProperty}
Domain = {UPAT}
Range = {UPROP}
Besides object property, domain and range for data property in the ontologies are also defined. The range values of data property are defined using the data types like string, integer, char and so on. Table 7 shows partially of the data properties defined in Functional Ontology with their domain and range.

Table 7: Data Properties and related Domain Range for Functional Ontology

Data Property, Dt_r	Domain	Range
hasName	ATR	String

hasGender	ATR	Char
hasDate	OBJ	String

3.3.4. Create Individual Instances

After defining the concept, relations and axioms, individual instances for concept defined in the hierarchy need to be created. The individual instance is referring to the ABox data. According to Noy and McGuinness [32], defining an individual instance for a class required three steps as follows:

i. Choosing a class or concept
ii. Creating an individual instance of that class
iii. Filling in the property values

The steps are used in creating the instances for the Usability and Functional Ontology. For instance in Usability Ontology, a concept *Usability_Property* is chosen and identify an instance named *Guidance*, a kind of usability property or requirement from HCI literature [20]. Thus, *Guidance* is created as an individual associated to Usability_Property *UPROP* concept. Next, the object property was filled in for the individual instance *Guidance*. It has an object property *has_UsaMetric* and described using the syntax below:

$$\forall \text{ has\_UsaMetric} . \text{Reliability\_In\_Use} \wedge \text{Learnability} \quad (9)$$

Based on the suggestion from SABIO [34], the use of a graphical model is important and needed to represent the ontology and ease the communication with the domain experts. In this work, UML class diagram is used to model the defined concepts and relations for Usability Ontology and Functional Ontology as shown in Figure 2 and 3. Figure 2 shows the Usability ontology model representing the concepts and relations about functional usability features. On the other hand, the meta-model in Figure 3 representing a domain according to two main components: concepts to which a domain refers and relations between these concepts. It is applicable to all applications related to booking or reservation domain that share the common concepts represented by this model based on high level view.

3.4. Cross Ontology Assertions

In the process of merging the ontologies, constraints are defined to represent the relevant Usability Features that need to be incorporated into the domain and their functional requirements. The concept *Usability_Pattern* of Usability Ontology were explicitly specified and associated with the concept *Function* of Functional Ontology by using two assertions as followings:

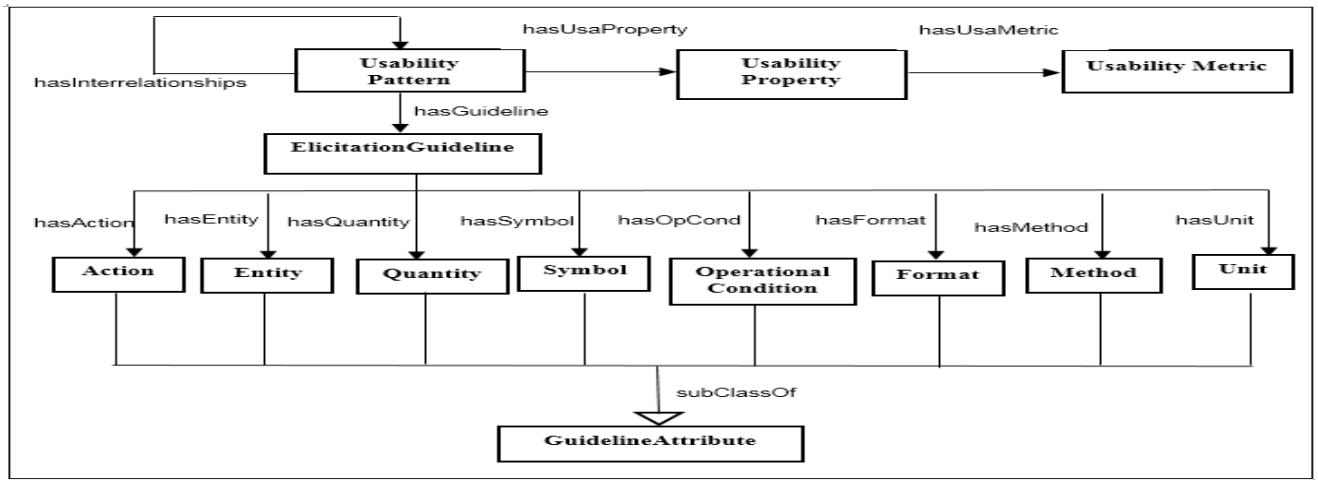

Fig 2: Usability ontology model

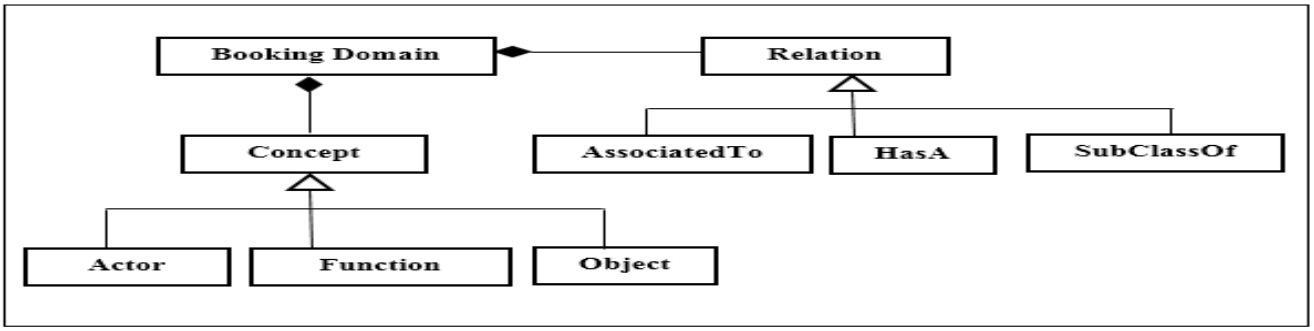

Fig 3: Functional ontology meta-model (high level view)

Assertion 1: Identify the entire related Usability Ontology concept, *Usability_Pattern* that should be integrated to a particular domain by defining an object property, *has_UsaPattern.* For example in a Booking Domain, the following Usability Patterns should be included: Undo, Progress Feedback, Abort, Warning, Favourites.etc.
has_UsaPattern Undo
has_UsaPattern Progress_Feedback
has_UsaPattern Abort
has_UsaPattern Warning
has_UsaPattern Favourites

Assertion 2: Map the above identified *Usability_Pattern* (from Assertion 1) to particular domain concept using object property, *has_UsaPattern.* For example in a Booking Domain, the *Usability_Pattern* that need to be incorporated to the domain concept *Function* (its individual instance, *Request_Booking* activity*)* are:
has_UsaPattern Undo
has_UsaPattern Progress_Feedback
has_UsaPattern Abort

The *Usability_Pattern* are reusable and applicable to other related domain concept Function such as *Cancel_Booking*, *User_Registration*, *Logout* and so on. The domain and range of object property *has_UsaPattern* used to merge the Usability and Functional ontology have been defined as shown in Table 8.

Table 8: Object Properties and related Domain Range for Domain Ontology

Object Property, Obj_r	Domain	Range
has_UsaPattern	$\forall_{?FUNC,?UPAT}$ (P(?FUNC, ?UPAT) → C (?FUNC))	$\forall_{?FUNC,?UPAT}$ (P(?FUNC, ?UPAT) → D (?UPAT))

4. Evaluation

The purpose of the evaluation is to ensure the correctness of an ontology that been developed and the output artifacts meet the specifications imposed (De Almeida Falbo, 2014; Fernández-López et al., 1997). It is important to verify that the class, properties between classes and axioms definitions are defined consistently and classified correctly during the development. In this evaluation process, Hermit Reasoner and ontology taxonomy evaluation techniques were used. The results are discussed in the following sections.

4.1. Verification using Hermit Reasoner

Verification was carried out using Hermit Reasoner along the process of ontology development by using Protégé editor tool. The verification task is performed by choosing the menu from Protégé to launch the Hermit Reasoner engine. Whenever there is any inconsistency in the taxonomy or axioms of the ontology, the erroneous result will be displayed and highlighted. The designed ontology was always checked by executing the reasoning functions and makes the necessary modifications or corrections when required. This verification step was repeated manually when there are any changes to the ontology to ensure the class consistency and infer subsumptions relationships.

4.2. Ontology Taxonomy Evaluation

The Ontology taxonomy evaluation is conducted manually with the domain experts from HCI and SE expertise. The purpose is to find out any inconsistency, incomplete and redundancy errors occur in the structure or taxonomy of the domain ontology. The results of the taxonomy evaluation as presented in Table 9.

Table 9: Taxonomy Evaluation Results

Errors	Sub Errors	Descriptions
Inconsistency	Circulatory Errors	No Errors.
	Partition Errors	No Errors.
	Semantic Errors	Firstly, the concept *Elicitation_Guideline* was defined as subclass under the superclass *Usability_Pattern*. After review, the HCI domain expert suggested to define it as disjoint class with Usability_Pattern.
Incompleteness	Incomplete Concept Classification	No Errors. All the related knowledge sources about Functional Usability Features and domain functionality were included into the designed ontology.
	Partition errors	Firstly, the concept *Action, Entity, Method, Format* and etc were defined as disjoint classes. After review, the domain experts suggested to define a relation between these classes. These classes were later defined under the superclass, *Guideline_Attribute*.
Redundancy	Grammatical Redundancy	No Errors.
	Identical formal definition of some classes, properties and instances (with a different name)	No Errors.

5. Conclusions

General speaking, usability is very subjective and abstract in nature. Based on the literature reviews, it is found that usability should not be treated as the quality attribute that only relevant with interface details but should also deals with the functionality that have implications on system design. However, efforts that deal with usability elicitation and specification at the requirements stage are limited. Based on the feature analysis conducted on the existing approaches, there is a need to propose a semantic knowledge representation to model and reason the usability features and their related information.

Ontology can be specially developed to represent knowledge in the FUF and support the elicitation process. The reasoning capabilities of ontology enable the assertions of usability features associated with the domain functionality. The ontology can also be used to encode domain knowledge to support the formulation of competency questions regards to the usability requirements relevant to the domain applied in order to facilitate the elicitation of a complete set of usability requirements. Currently, we continue this work by developing a semi-automated tool which applying the proposed domain ontology to provide support for usability features elicitation and specification activities.

Acknowledgement

We wish to acknowledge support from Universiti Putra Malaysia and Ministry of Education Malaysia through the Fundamental Research Grant Scheme (FRGS).

References

[1] Shneiderman B (1986), *Designing the User Interface: Strategies for Effective Human–Computer Interaction*. Addison-Wesley, Reading, MA.

[2] Nielsen J (1993), *Usability Engineering*. Morgan Kaufmann, 1994.

[3] Cysneiros LM & Leite JCSDP (2004), Nonfunctional requirements: from elicitation to conceptual models. *IEEE Transactions on Software Engineering, 30*(5), 328–350.

[4] Cysneiros LM, Werneck VM & Kushniruk A (2005), Reusable knowledge for satisficing usability requirements. *13th IEEE International Conference on Requirements Engineering (RE'05)*.

[5] Jokela T, Seffah A, Gulliksen J & Desmarais MC (2005), Eight Guiding Designers to the World of Usability: Determining Usability Requirements through Teamwork. *Springer Netherlands*, Vol.8, 127-145.

[6] Cronholm S & Bruno V (2008), Do you Need General Principles or Concrete Heuristics?: A Model for Categorizing Usability Criteria. In the *20th Australasian Conference on Computer-Human Interaction: Designing for Habitus and Habitat,ACM, Cairns,Australia*.

[7] Juristo N, Moreno AM & Sanchez M (2007), Guidelines for Eliciting Usability Functionalities. *IEEE Transactions on Software Engineering*, 33(11): 744-758.

[8] Too CW, Hassan S, Ghani AAA, Din J (2017), Towards Improving Usability Requirements Elicitation and Specification Using Ontology-Driven Approach. *Advanced Science Letters*, Vol.23, No.5, May 2017, pp:4077-4081(5).

[9] Juristo N (2009), Impact of usability on software requirements and design. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* Vol.5413, pp:55–77.

[10] Ormeño YI & Panach JI (2013), Mapping study about usability requirements elicitation. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* Vol.7908 LNCS, pp:672–687.

[11] Roder H (2012), Specifying Usability Features with Patterns and Templates. *UsARE 2012, Zurich, Switzerland*, pp:6-11.
[12] Bass L & John BE (2000), Achieving usability through software architectural styles. *CHI '00 Extended Abstracts on Human Factors in Computing Systems*, pp:171.
[13] Bass L & John B (2003), Linking Usability to Software Architecture Patterns through General Scenarios. *The J. Systems and Software*, vol.66, no.3, pp:187-197.
[14] Juristo N, Lopez M, Moreno AM & Sánchez MI (2003), Improving software usability through architectural patterns. *International Conference on Software Engineering (ICSE)*, pp:12-19.
[15] Ferre X, Juristo N, Windl H & Constantine L (2001), Usability Basics for Software Developers. *IEEE Software*, Vol.18, No.1, pp:22-29.
[16] Laura Carvajal (2012), *Usabilidad-Oriented Software Development Process*, *1*(2), 272. Retrieved from http://oa.upm.es/10599/1/LauraElena_Carvajal_Garcia.pdf
[17] Trienekens JJM & Kusters RJ (2012), A framework for characterizing usability requirement elicitation and analysis methodologies *(UREAM). IARA*, pp:308–313.
[18] Yu E (1997), Towards Modelling and Reasoning Support for Early Phase Requirements Engineering. In *Proc. Of the 3rd IEEE Int. Symp. On Requirements Engineering*, pp:226-235.
[19] Rafla T, Robillard PN & Desmarais M (2007), A method to elicit architecturally sensitive usability requirements: Its integration into a software development process. *Software Quality Journal*, Vol.15, No.2, pp:117–133.
[20] Folmer E, Van GJ & Bosch J (2003), A framework for capturing the relationship between usability and software architecture. *Software Process Improvement and Practice*, Vol.*8*,No.2, pp:67–87. http://doi.org/10.1002/spip.171
[21] Rivero L, Barreto R &Conte T (2013), Characterizing Usability Inspection Methods through the Analysis of a Systematic Mapping Study Extension. *Latin-American Center for Informatics Studies Electronic Journal*, Vol.16, No.1.
[22] Hassan S, Too CW & Kesava PR. (2013), An Analysis Framework for Identifying Usability Requirement in Mobile Application Development. *Journal of Next Generation Information Technology (JNIT)* Vol.4, No.4, June 2013.
[23] Gruber TR (1993), Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *Proc. Int'l Workshop on Formal Ontology 1992*.
[24] Arpírez JC, Corcho O, Fernández-López M & Gómez-Pérez A (2003), *WebODE in a nutshell*. AI Magazine, Vol.24, No.3, pp:37-47.
[25] Borst W (1997), *Construction of Engineering Ontologies*. PhD thesis, Institute for Telematica and Information Technology, University of Twente, Enschede, The Netherlands.
[26] Studer R, Benjamins VR & Fensel D *(*1998), Knowledge Engineering Principles and Methods. *Data and Knowledge Engineering*, 25, 61-197.
[27] Allemang D, Hendler JA (2008), *Semantic Web for the Working Ontologist: Modeling in RDF, RDFS and OWL*. Elsevier, Amsterdam.
[28] Castaneda V, Ballejos L, Caliusco ML & Galli MR(2010), The use of Ontologies in Requirements Engineering. *Global Journal of Researches in Engineering*, Vol.10, No.6.
[29] Ricardo de AF, Crediné Silva de M, Ana RR (1998), A Systematic Approach for Building Ontologies. *IBERAMIA* 1998: pp349-360.
[30] Dermeval D, Vilela J, Bittencourt II, Castro J, Isotani S & Brito P (2015), Applications of ontologies in requirements engineering: A systematic review of the literature. *Requirements Engineering*, pp:1–33.
[31] Fu Gh & Cohn A (2008), Utility Ontology Development with Formal Concept Analysis. Conference: Formal Ontology in Information Systems. *Proceedings of the Fifth International Conference, FOIS 2008*, Saarbrücken, Germany, October 31st - November 3rd, 2008, pp:297-310.
[32] Noy NF & McGuinness DL (2001), *Ontology Development 101: A Guide to Creating Your First Ontology*. Stanford Knowledge Systems Laboratory, 25.
[33] Fernández LM., Gómez P & Juristo N (1997), METHONTOLOGY: From Ontological Art Towards Ontological Engineering. *AAAI-97 Spring Symposium Series*, SS-97-06, pp:33–40. http://doi.org/10.1109/AXMEDIS.2007.19.
[34] De Almeida FR (2014), SABiO: Systematic approach for building ontologies. *CEUR Workshop Proceedings*, 1301.
[35] Shneiderman B & Plaisant C (2010), *Designing the User Interface:Strategies for Effective Human-Computer Interaction*. Fifth Edition. Pearson Addison-Wesley.
[36] International Organization For Standardization ISO (2001), ISO/IEC 9126-1. Software Process: Improvement and Practice. http://doi.org/10.1002/(SICI)1099-1670(199603)2:1<35::AID-SPIP29>3.0.CO;2-3
[37] International Organization For Standardization ISO (2011), *ISO/IEC 25010: 2011. Software Process: Improvement and Practice* Vol.2. Retrieved from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733
[38] Baader F, Calvanese D, McGuinness DL, Nardi D & Patel-Schneider PF (2010), *The Description Logic Handbook: Theory, Implementation and Applications*. Kybernetes, 32(9/10), 624. https://doi.org/10.1108/k.2003.06732iae.006
[39] Kaiya H & Saeki M (2005), Ontology based requirements analysis: lightweight semantic processing approach. In *Quality Software, 2005 (QSIC 2005). Fifth International Conference on* IEEE, pp: 223-230.
[40] Antoniou G, Franconi E & Harmelen FV (2005), *Introduction to Semantic Web Ontology Languages*, Norbert Eisinger Jan Małuszy´nski (Eds.), LNCS: Reasoning Web, First International Summer School, July 25-29, 2005, pp:1-21. Springer.