How to cite this article:

Norki, F. A., Mohamad, R., & Ibrahim, N. (2020). Context ontology in mobile applications. *Journal of Information and Communication Technology,* 19*(1)*, 21-44.

CONTEXT ONTOLOGY IN MOBILE APPLICATIONS

Farhanah Atiqah Norki, Radziah Mohamad & Noraini Ibrahim
School of Computing, Universiti Teknologi Malaysia, Malaysia

farhanahatiqah@ymail.com; radziahm@utm.my; noraini_ib@utm.my

ABSTRACT

Mobile applications are expected to receive context input such as location, speech, and network from different context providers. Since context can be considered as knowledge, a formal method is needed to capture this knowledge. There is less work on ontology model that could be reused to model a new context ontology for Android mobile application. Therefore, this study proposed an ontology specifically for Android mobile application, COCCC, to formalize context knowledge present within it. METHONTOLOGY method was used to create COCCC ontology as it offers intermediate representation in the form of concepts. The concepts from the context ontology were extracted from various resources, sorted and categorized based on types and functions for standardization purposes. Survey was given to five domain experts for evaluation of COCCC ontology in terms of its usability. Data from these experts were analyzed and the results have confirmed that the proposed context ontology is usable to Android mobile application developers.

Keywords: Context ontology, knowledge representation, mobile application, ontology.

INTRODUCTION

Understanding and catching up on the latest mobile application technology are exhausting to some mobile application developers. The complexity

Received: 10/09/2018 **Revised:** 11/2/2019 **Accepted:** 26/03/2019 **Published:** 23/12/2019

of context-aware mobile application has led researchers to design several ontologies to capture the knowledge, as well as organize information within the mobile application and its users. The development of context-aware mobile application needs to be supported by sufficient context modeling and reasoning techniques. The context involved in mobile application development can be further categorized in order to better assist mobile application developers during the development phase.

Ontology can be referred as ideas abstraction that represents concepts inside some body of knowledge (Mohd-Hamka & Mohamad, 2014). The categorization of context in a specific body of knowledge provides a formal model of conceptualization (Gruber, 1993). A formal ontology consists of a specific vocabulary expressed in a representation language (Sulaiman, Nordin, & Jamil, 2017). Ontology is made up of hierarchical definitions of principal concepts in a domain (Zeshan & Mohamad, 2012). The relationships between these different hierarchies are depicted as is-a, part-of, and so on, depending on the taxonomy (Subhashini & Akilandeswari, 2011). The structure of knowledge in a particular domain can be shared, reused, and merged with other ontology models in different domains (Lee, Lee, & Kwan, 2017).

One of the famous mobile application ontologies is mIO! Ontology Network. mIO! Ontology has managed to cover many subdomains in mobile applications; device, environment, interface, location, network, provider, service, time, and user (Poveda Villalon et al., 2010). However, mIO! Ontology cannot be reused to fit Android mobile applications at the source code level as the ontology lacks the application program interface (API) present in Android mobile apps.

RELATED WORK

Context ontology allows representation of multiple context knowledge and also provides a formal language and logic to context knowledge, which assist in integration and sharing of information between contexts.

There is a variety of context ontologies with different subdomains such as location, environment, device, network provider, role, service, source, and user. Table 1 shows a comparison of different existing context ontologies with regard to different context subdomains.

COBRA-ONT describes the regular relationships related to people and activities which enable the sharing of knowledge and ontology reasoning within the CoBra (Context Broker Architecture) infrastructure (Chen, Finin, & Joshi, 2003). To solve some challenges in Ambient Intelligent computing infrastructures, the CoDAMos ontology provides intelligent services to the

user which includes application adaptation, automatic code generation and code mobility, and user interfaces (Preuveneers et al., 2004). The ontology consists of four fundamental entities: user, service, environment, and platform.

Table 1

Comparison of Existing Context Ontologies

Ontology Subdomain	COBRA-ONT	CoDAMoS	CONON	SOUPA	Delivery Context Ontology	mIO! Ontology
Environment	√	√	√		√	√
Location	√	√	√	√	√	√
Device	√	√			√	√
Network					√	√
Role	√	√				√
Service		√				√
User	√	√	√	√		√

CONtext ONtology (CONON) provides an extensible context modeling for pervasive computing environments (Wang et al., 2004). CONON presents common concepts such as location, user, activity, and computational entity in order to capture information on executing situations. These entities can be extended hierarchically by adding domain specific concepts. The context model is divided into two; upper ontology (high-level ontology) and specific ontology. The upper ontology expresses common characteristics of fundamental contexts whereas the specific ontology describes details of common concepts and their characteristics in their subdomains. Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) addresses general issues via upper ontologies method (Chen et al., 2004). SOUPA is divided into two main blocks; SOUPA Core and SOUPA Extensions. SOUPA Core consists of concept vocabularies that associate with person, agent, belief-desire-intention (BDI), action, policy, time, space, and event, while SOUPA Extensions supports specific concepts in certain domains, for example, management (Chen et al., 2004). The limitation of SOUPA ontology is the small number of features that describes a mobile device in mobile computing environments.

Delivery context ontology gives a formal model of environment characteristics in which different devices communicate with physical services (Fonseca & Lewis, 2009). The ontology includes the attributes of the device, the software for accessing the service, and the network which provides the connection (Fonseca & Lewis, 2009). Principal entities present in this ontology are environment, hardware, software, location, and measure. The most prominent work in the ontology mobile computing environment is mIO! Ontology, which incorporates nine sets of contextual subdomains (Poveda Villalon et al., 2010). The contextual information in mIO! Ontology is categorized into 11 different ontologies: device, environment, interface, location, network provider, role, service, source, time, and user. The goal of mIO! Ontology is to represent the user's contextual knowledge which may later influence his or her interactions with mobile devices. The user will be able to interact with services provided by companies through his or her mobile device. mIO! Ontology reuses knowledge from other context ontologies such as CODAMOS, SOUPA, and Delivery context ontology.

Based on the existing context ontologies, there is no specific model that can be reused for modeling a new context ontology that caters specifically for Android mobile applications. Speech context, which is widely popular due to its speech recognition ability to guide users, has not been covered in these existing ontologies. The proposed context ontology however, focuses more on the environment surrounding mobile devices especially on location and speech. The new ontology specifies how hardware like sensor is connected to specific activities such as gravity detection, and more.

CONTEXT ONTOLOGY MODELING

Modeling context ontology is possible since context can be considered as a specific kind of knowledge (Poveda Villalon et al., 2010). Concepts in ontology construction can be modeled as classes or sub-classes, depending on the hierarchy. Since ontology describes the sharing of comprehension of specific domains, it can be used as a basic structure to solve problems in knowledge sharing (Uschold & Gruninger, 1996). Besides, ontology also helps in improving communication between humans or computers. METHONTOLOGY helps in developing ontology from scratch (Fernández-López, Gómez-Pérez, & Juristo, 1997). For an inexperienced ontology builder, METHONTOLOGY offers intermediate representation which is easy to understand (Pinto & Martins, 2004). Furthermore, METHONTOLOGY provides guidance in creating ontology through specification, conceptualization, formalization,

implementation, and maintenance (Corcho et al., 2005). Figure 1 shows five development activities in METHONTOLOGY.

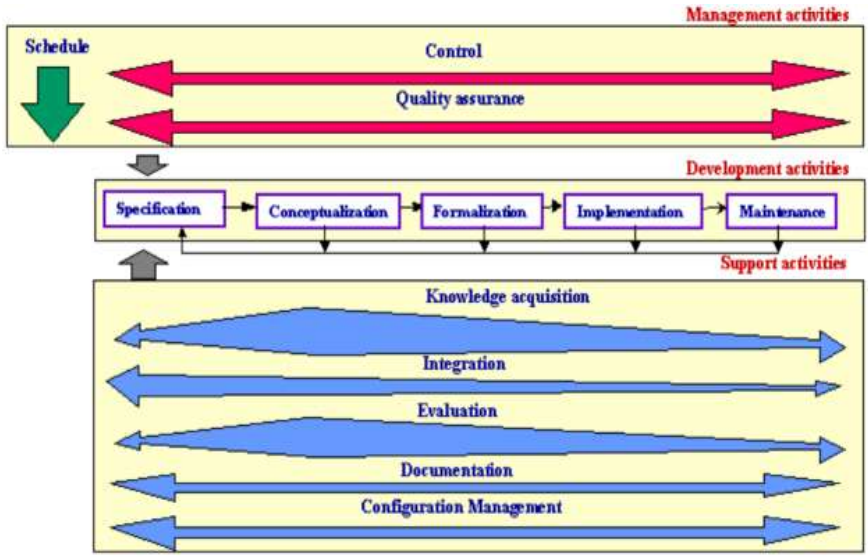

Figure 1. Activities in ontology development proposed by METHONTOLOGY (Corcho et al., 2005).

These activities can be further described as follows:

- Specification - During this stage, we had to specify the reason why the ontology is being built, the use of the ontology, and its target user.
- Conceptualization - In this stage, we tried to organize the domain collected by tabulating them in a way that can be easily understood by ontology experts and developers. At the end of this stage, we came up with a list of ontology concepts that are involved in context.
- Formalization - During this activity, we transformed the conceptual model from conceptualization into a more formal model.
- Implementation - We represented the knowledge involved in context, grouped them together, and established a relationship between them by using Web Ontology Language (OWL).
- Maintenance - Any adjustment, update, or correction on the proposed ontology was made during maintenance activity.

The evolving nature of mobile computing complicates the formalization of all context information in its entirety. However, this research has found that the concepts in mobile environments can be categorized into five classes:

mobile device, context, application manifest, application, and activity. Apart from application manifest and mobile device, all information for the other three classes are specifically grouped together, to show the link between each class to promote understanding of mobile applications to developers. These five classes are the most basic concepts in mobile environments. They form the backbone of mobile environments, constituting upper-level entities which contribute versatile extensibility that enables the addition of more concepts later. Table 2 defines main classes and their corresponding descriptions.

Table 2

Main Classes

Main Classes	
Android Application	An android application can be interpreted as an application that is run on mobile devices with Android operating system.
Activity	Activity here is referred as detection and recognition performed by the application which manipulates input from context.
Mobile Device	A mobile device refers to a portable computing device such as a smartphone or tablet computer.
Context	Context refers to any information that can be used to characterize the situation of an entity which includes lighting, level of noise, network connectivity, location, speech, bandwidth, et cetera.
Application Manifest	Provides necessary information of the application to the Android system, which must be possessed by the system before it can run any of the application's codes.

Figure 2 shows the relationship design between Android application, application manifest, context, mobile applications, and mobile device. The Application Manifest comes from the Android Manifest.

To build a functional context-aware application, a platform, which is the mobile device itself, is needed. Sub-categories from the mobile device include hardware, software, and battery. The hardware can be called into use by declaring them inside the application manifest. The declaration from the application manifest enables the application to use hardware such as sensors. Depending on which hardware is activated, the Activity class will start to detect or recognize any potential Context available within the user's surroundings. Later, the class sends back the information on context to the running Android application.

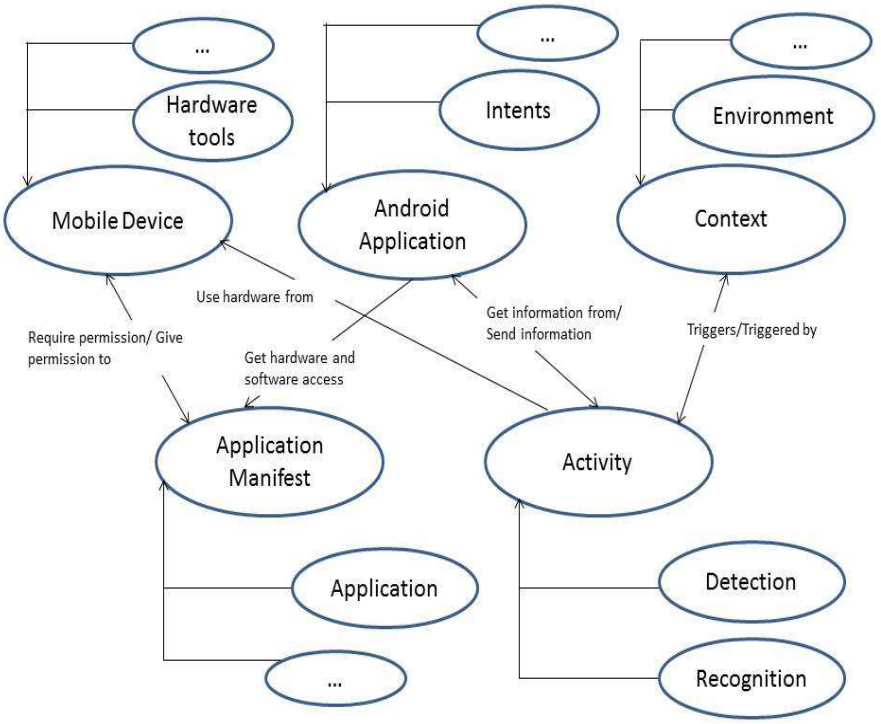

Figure 2. COCCC Ontology.

Concepts

METHONTOLOGY suggests the conceptualization of ontologies with a set of tabular and graphical Intermediate Representation (IR). These IRs enable the modeling of the components explained in this section. Concept can be perceived differently depending on the main classes. For example, concepts in mobile environments include activity, light detection, sensor, etc. These concepts are generally organized in taxonomies where inheritance is usually involved. Specific types of annotations and metadata are added later to the document. Before developing ontology, all concepts related to mobile environments need to be identified. The identification of concepts prevents redundancy in the ontology, as well as gives a clear idea on which class the concept belongs to. Table 3 lists all the concepts involved during context ontology development and Table 4 lists the glossary of lower level concepts and their corresponding descriptions.

Table 3

Ontology Concepts

Ontology Concepts			
Activity	Context	Mobile Device	Detection
Recognition	Environment	Foundation	User
Battery	Hardware Tools	Sensors	Gravity detection
Humidity detection	Light detection	Location detection	Location detection (without GPS)
Motion detection	Nearby object detection	Noise detection	Temperature detection
Touch detection	Image recognition	Speech recognition	Location
Surroundings	Gravity	Humidity	Light
Motion	Nearby object	Noise	Temperature
Touch	Vision	Connectivity	Memory
Power	Input	Social Environment	Interface
Speech	Sound	Image	Video
GPS	SD Card	Touchscreen	Accelerometer Sensor
Ambient Temperature Sensor	Gyroscope Sensor	Light Sensor	Magnetic Field Sensor
Pressure Sensor	Proximity Sensor	Relative Humidity Sensor	Android Application Project
Manifest File	application	instrumentation	Permission
supports-gl-texture	supports-screen	uses-configuration	uses-feature
uses-permission	uses-sdk	activity	Provider
receiver	service	uses-library	Android Application
Notifications	Widgets	Broadcast Receivers	Intents
Content Providers	Services	Activities	Wi-Fi
USB	Telephony	NFC	Microphone
Camera	Bluetooth		

The concepts for the ontology are obtained from Android Developer website (Android Developer, 2009) and various existing ontologies as discussed earlier. As COCCC is a developing ontology, new concepts can be added from time to time, forming a more detailed knowledge tree.

After the extraction of entities, the next step is the formation of taxonomy. All concepts are arranged in a taxonomic hierarchy. Forming taxonomy helps people understand the ontology better and serves as reference for integration and reuse of other ontologies. Some of the concepts from Table

3 are represented as classes or subclasses and the relationship between them are represented as is-a. For example, in Figure 3, Social Environment is a child of the User context.

Table 4

Glossary of Concepts

	Glossary of Concepts
Activity	An activity is the entry point for any interaction with the user.
Content Provider	A content provider handles a shared set of application data that can be stored in the file system, in a SQLite database, on the web, or on any other non-volatile storage location that can be accessed by the application.
Services	Services have a range of potential use, acting as the entry point for retaining an application running in the background for all forms of reasons.
Broadcast Receiver	A broadcast receiver is a building block that enables the system to carry events to the app outside of a normal user flow, allowing the app to react to system-wide broadcast announcements.
Foundation	A context that influences existing resources in mobile devices.
Environment	A context influenced by surroundings, nearby conditions and circumstances.
User	Any context that requires interference or interaction from the user.
Sensor	A device capable of sensing and detecting change in the environment.
Ambient Temperature Sensor	A sensor that is used to detect only surrounding temperature.
Relative Humidity Sensor	A sensor that is used to detect humidity.
Light Sensor	A sensor that is used to detect light.
Gyroscope Sensor	A sensor that is used to detect motion.
Accelerometer Sensor	A sensor that is used to detect gravity.
Proximity Sensor	A sensor that is used to detect nearby objects.
Pressure Sensor	A sensor that is used to detect location.
GPS	A system that is used to detect location.
uses-feature	Nodes that can be used to specify which hardware features are required.

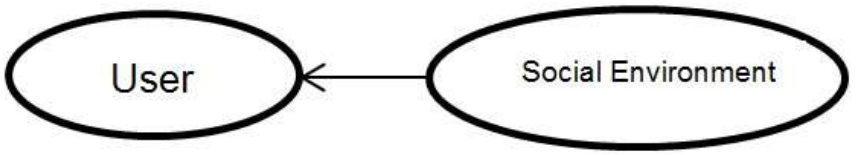

Figure 3. Example of class and subclass.

The concepts are grouped as a collection of subclasses for different main classes. Figure 4 shows the grouping of the main classes with their subclasses. Except for the application manifest and android application which are marked in a box, all the information for the other three classes are specifically grouped together in a certain way to show the link between each class in order to promote understanding of mobile applications to the developer. These five classes are the most basic concepts in mobile environments. Every application comes with an AndroidManifest.xml file in its root directory. The manifest file supplies important information regarding the application to the Android operating system, which is required for running the application. The references used for designing the Application Manifest's branch in COCCC ontology comes directly from the Android Developer website.

The context class branches out into different subclasses that share common features but also differs significantly in detailed characteristics. Figure 5 describes the subclasses of contexts.

Context is divided into four categories: environment, user, location, and foundation. Environment context is defined as a context influenced by surroundings, nearby conditions and circumstances. Other subclasses of surroundings include humidity, gravity, noise, touch, motion, vision, nearby object, position, light, and temperature. User context is defined as any context that requires interference or interaction from the user. For example, input from the user, the user's interaction with application interface, and also how information uploaded by the user (e.g. video, image, and sound) is presented to other users. Location context describes the current location of the mobile device in which the context is influenced by latitude and longitude. Foundation context is defined as context that influences the existing resources in mobile devices. Foundation is divided into three: power, memory, and connectivity. These resources are available in all mobile devices. Figure 6 shows the Application Manifest class.

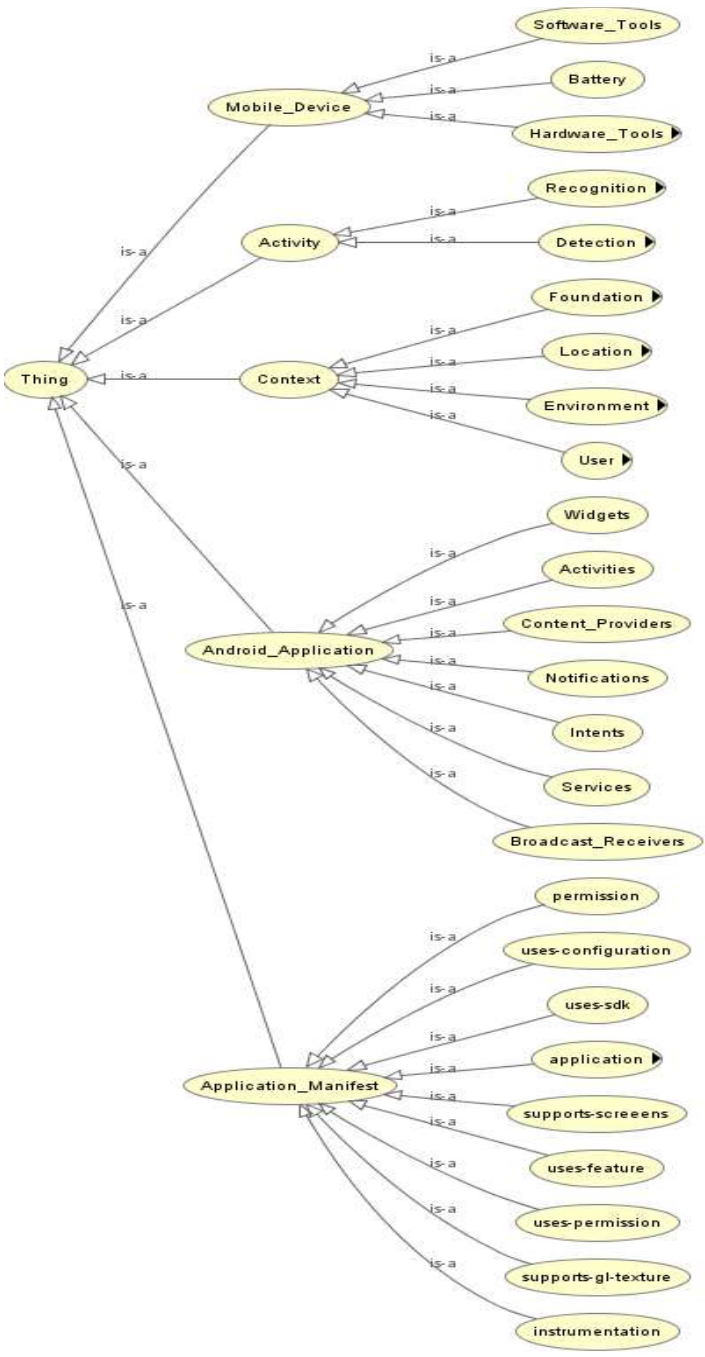

Figure 4. Main classes and their subclasses.

Figure 5. Subclasses of contexts.

The application manifest file offers necessary information about the application to the Android operating system. The information declared in the manifest is important in order to determine which components will be used to run the application. The manifest describes the Android application class components. At the same time, the manifest publishes the application's capabilities, for example, the Intent messages they can handle. Besides, the manifest declares permission to use hardware components, by allowing the hardware to be integrated with available APIs. To sum up, the application manifest acts as an intermediary between certain concepts which enables interaction between them, as permitted by the manifest.

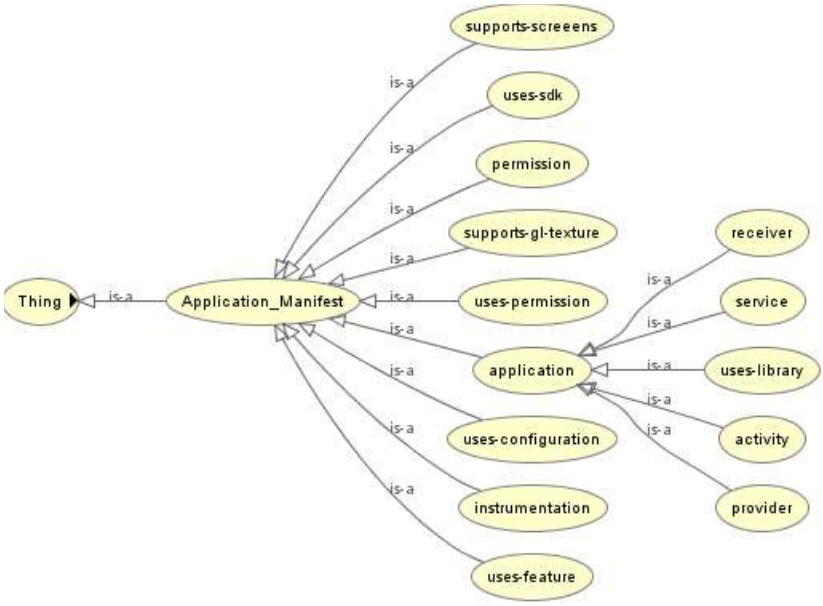

Figure 6. Components inside the application manifest.

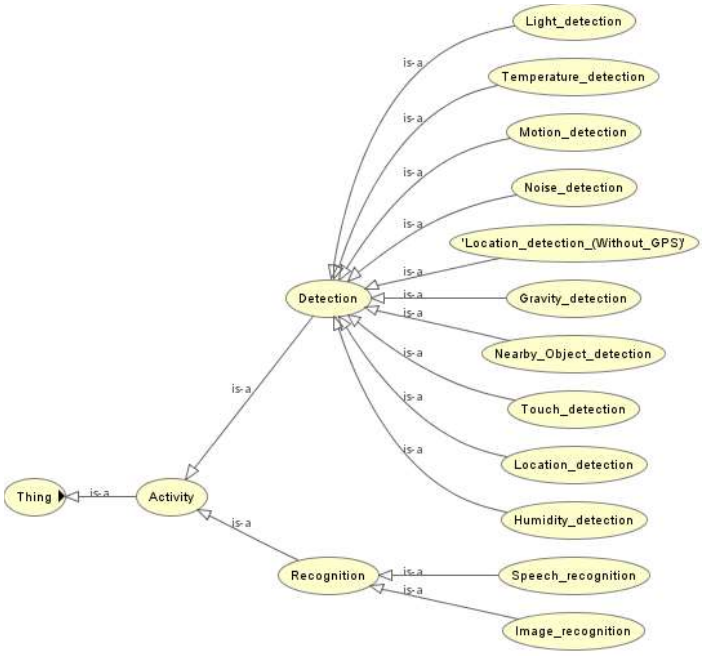

Figure 7. Detection and recognition activity.

Figure 7 shows the Activity class. The class is divided into two: detection and recognition. The detection subclass refers to any action or process of identifying the presence of something in the environment and location which includes light, temperature, motion, noise, gravity, nearby object, touch, location, and humidity. Recognition is the ability of an application to recognize language or object(s). Recognition class includes speech and image recognition.

Figure 8. Hardware tools present in mobile devices.

Figure 8 shows the subclasses of hardware which falls under the mobile device class. Hardware tools consist of many subclasses such as sensors, camera, touchscreen, audio, microphone, USB, and others. The sensors can be further categorized into three types: motion, environment, and position. The motion sensor enables the user to monitor the motion of the device. The position sensor allows the user to determine the position of a device. The environment sensor lets the user keep track of different environmental properties.

Relationship

Semantic relationship is a direct relationship that exists in ontology while semantic association is an indirect relationship in ontology. Relationship depicts a type of connection between predetermined concepts. For example, *GPS* and *Location Detection* are linked by isUsedFor relation as shown in Figure 9. *GPS* is used for location detection. The inverse of this relation is isDetectedBy. Table 5 shows the relation and inverse relation of concepts.

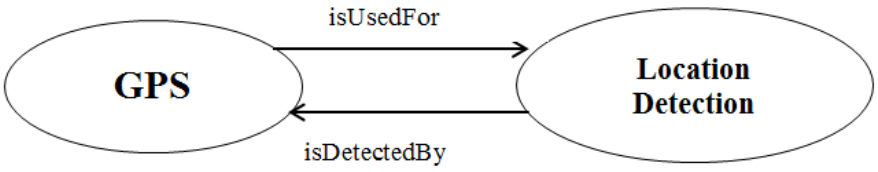

Figure 9. Relationships between GPS and Location Detection.

Table 5

Relationship Table

Relation name	Source Concept	Target concept	Inverse relation
isUsedFor	Ambient Temperature Sensor	Temperature Detection	isDetectedBy
isUsedFor	Relative Humidity Sensor	Humidity Detection	isDetectedBy
isUsedFor	Light Sensor	Light Detection	isDetectedBy
isUsedFor	Gyroscope Sensor	Motion Detection	isDetectedBy
isUsedFor	Light Sensor	Noise Detection	isDetectedBy
isUsedFor	Accelerometer Sensor	Gravity Detection	isDetectedBy
isUsedFor	Proximity Sensor	Nearby Object Detection	isDetectedBy
isUsedFor	Pressure Sensor	Location Detection (no GPS)	isDetectedBy
isUsedFor	GPS	Location Detection	isDetectedBy
canDetect	Ambient Temperature Sensor	Temperature	canBeDetectedBy
canDetect	Relative Humidity Sensor	Humidity	canBeDetectedBy
canDetect	Light Sensor	Light	canBeDetectedBy
canDetect	Gyroscope Sensor	Motion	canBeDetectedBy
canDetect	Accelerometer Sensor	Gravity	canBeDetectedBy
canDetect	Proximity Sensor	Nearby object	canBeDetectedBy

(continued)

Relation name	Source Concept	Target concept	Inverse relation
canDetect	Pressure Sensor	Location	canBeDetectedBy
canDetect	GPS	Location	canBeDetectedBy
canRecognize	Camera	Image	canBeRecognizedBy
canRecognize	Speaker	Speech	canBeRecognizedBy
giveDeclare	uses-feature	Hardware Tools	declaredBy
givePermission	uses-permission	Hardware Tools	isGivenPermissionBy

Axiom

Axiom is assertion in a logical form. Ontology has its own line of axioms and these axioms are made up of statements which are assumed as true. For example, "sensor is an element of mobile devices and is used for detection", would have satisfied the axiom. Another example of axiom is: Ambient Temperature Sensor is used for detecting only surrounding temperature and cannot be used to detect light or other contexts. The axioms for the context ontology are presented in Table 6.

Table 6

Logical Table

Concept name	Axiom description	Logical expression
Sensors	A device capable of sensing and detecting change in the environment	• Sensor $\subseteq$ Hardware Tools $\cap \exists$ isUsedFor. Detection
Ambient Temperature Sensor	A sensor that is used to detect only surrounding temperature	• AmbientTemperatureSensor $\subseteq$ Sensor $\cap \forall$ is UsedFor.TemperatureDetection • AmbientTemperatureSensor $\subseteq$ Sensor $\cap \forall$ canDetect.Temperature
Relative Humidity Sensor	A sensor that is used to detect only humidity	• RelativeHumiditySensor $\subseteq$ Sensor $\cap \forall$ isUsedFor.HumidityDetection • RelativeHumiditySensor $\subseteq$ Sensor $\cap \forall$ canDetect.Humidity
Light Sensor	A sensor that is used to detect only light	• LightSensor $\subseteq$ Sensor $\cap \forall$ isUsedFor. LightDetection • LightSensor $\subseteq$ Sensor $\cap \forall$ canDetect.Light
Gyroscope Sensor	A sensor that is used to detect only motion	• GyroscopeSensor $\subseteq$ Sensor $\cap \forall$ isUsedFor. MotionDetection • GyroscopeSensor $\subseteq$ Sensor $\cap \forall$ canDetect. Motion

(continued)

Concept name	Axiom description	Logical expression
Accelerometer Sensor	A sensor that is used to detect only gravity	• AccelerometerSensor ⊆ Sensor ∩ ∀ isUsedFor.GravityDetection • AccelerometerSensor ⊆ Sensor ∩ ∀ canDetect.Gravity
Proximity Sensor	A sensor that is used to detect only nearby object(s)	• ProximitySensor ⊆ Sensor ∩ ∀ isUsedFor. NearbyObjectDetection • ProximitySensor ⊆ Sensor ∩ ∀ canDetect. NearbyObject
Pressure Sensor	A sensor that is used to detect location	• PressureSensor ⊆ Sensor ∩ ∀ isUsedFor. LocationDetection • PressureSensor ⊆ Sensor ∩ ∀ canDetect. Location
GPS	A system that is used to detect location	• GPS ⊆ HardwareTools ∩ ∀ isUsedFor. LocationDetection • GPS ⊆ HardwareTools ∩ ∀ canDetect. Location
uses-feature	Nodes that can be used to specify which hardware features are required	• uses-feature ⊆ ApplicationManifest ∩ ∀ giveDeclare.HardwareTools

To ask for some information from the ontology, simply type the expressions on DL Query. Figure 10 and 11 show examples of class query that can be performed on the context ontology. First, it can be useful to see which things are classified "under" this expression.

Q: "Which hardware tools can detect certain surroundings?"

To query what kind of sensor is used for certain conditions, the following question is asked:

Q: "What type of sensor is used to detect nearby object(s)?"

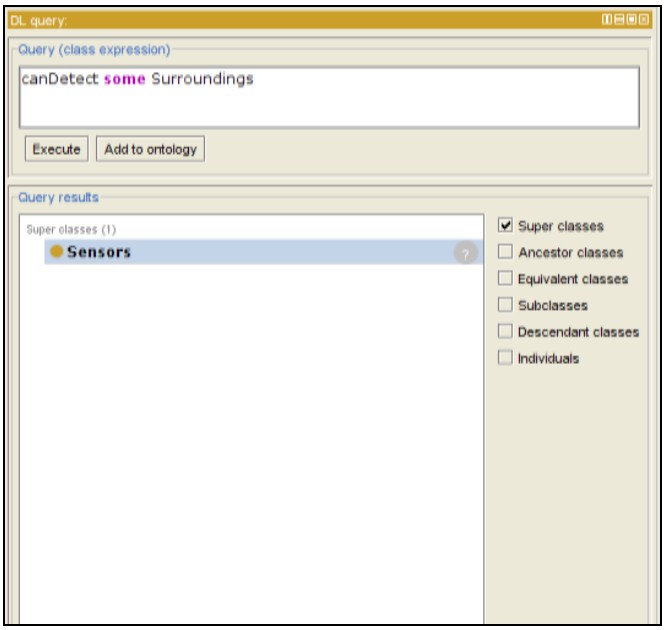

Figure 10. Example 1: DL Query in Protégé.

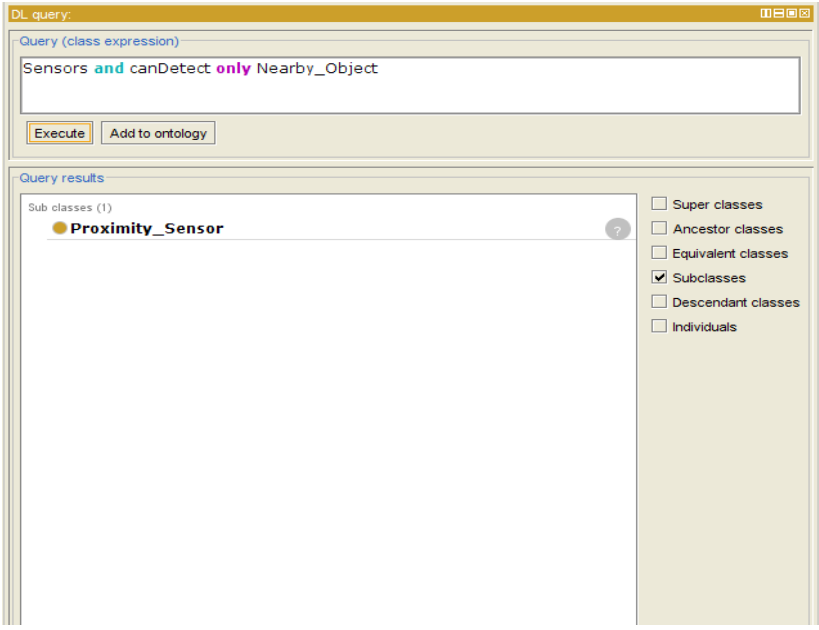

Figure 11. Example 2: DL Query in Protégé.

Consistency Checking

Redundancy in instances can lead to an inconsistent ontology which could decrease the practicality of the ontology itself. Pellet reasoner was used to check for any inconsistencies since it can handle growing OWL ontologies. Any inconsistencies in concepts, relationships, and labeling were removed prior to the use of the Pellet reasoner. Figure 12 shows the results of validation of context ontology using Pellet.

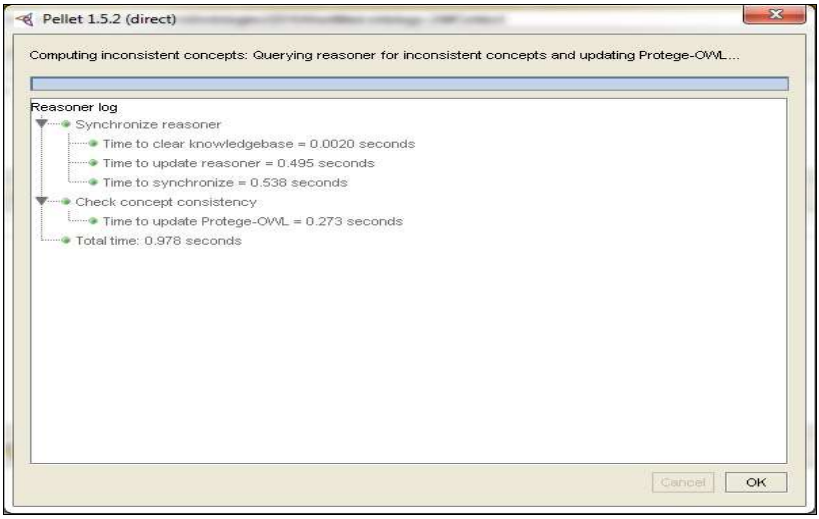

Figure 12. Pellet consistency checking in Protégé.

If the query detects inconsistent relationship, it will send out a red error message (Noy, Fergason, & Musen, 2000). Therefore, it is assumed that the context ontology is free from redundancy since no error is shown in the message results.

SURVEY ON ACCEPTANCE

The COCCC ontology is evaluated through survey. The survey sets a benchmark for COCCC in terms of usability from the domain expert's view. The demography or experience data of the respondents were measured using demography questionnaires which covered the respondents' work background in mobile application development. Respondents consisted of three males and

two females, with varying work backgrounds in mobile apps development or testing. The variation in age range was small since the first Android operating system was launched in 2008, about 10 years ago. Two respondents were involved in less than five Android projects, two respondents were involved in 5–10 projects, and the other respondent was involved in more than 10 projects. Table 7 includes detailed demographic data of the respondents.

Table 7

Demographic Data

ID	Work background in mobile apps development/ testing	Industry background (research or industrial training)	Projects involved	Determine suitable source codes related to location and speech context for Android	Determine type of context awareness embedded in mobile application
1	Less than 3 years	Less than 3 years	Less than 5 projects	Average	Average
2	Less than 3 years	3–6 years	5–10 projects	Substantial	Average
3	Less than 3 years	3–6 years	Less than 5 projects	Minimal	Minimal
4	Less than 3 years	3–6 years	5–10 projects	Average	Average
5	3–6 years	Less than 3 years	More than 10 projects	Substantial	Professional

After undergoing the COCCC ontology for 30 minutes, technical and usability questionnaires were administered to the respondents. The technical part required them to answer questions related to Android mobile application with regard to COCCC ontology whereas the usability questions measured their opinion on COCCC ontology in terms of usefulness and ease of use. Descriptive analytics was used to better understand data obtained from the questionnaires. Descriptive analytics is a process of using exploratory analysis consisting of statistical techniques such as mean, measures of dispersion (standard deviation), charts, graphs and frequency distribution to help in understanding and visualizing big datasets (Yusuf-Asaju et al., 2018). Table 8 shows the results of each question in which ‘m’ represents the mean while ‘s’ represents the standard deviation. The answers are reflected using a scale of 1 to 5, with 5 indicating strongly agree; 1 indicating strongly disagree.

Table 8

Usability Questionnaire Results

Usefulness	**1**	**2**	**3**	**4**	**5**	**m**	**s**
Using the proposed context ontology, the relationship between mobile applications, devices, contexts, Android Manifests, and activity elements are clear and understandable.				2	3	4.6	0.55
Using the proposed context ontology, it is easier for beginner mobile applications developers to identify contexts, sensors, and the relationship between context and sensor in mobile applications.				3	2	4.4	0.55
Using the proposed context ontology, it helps to develop mobile applications more easily.				4	1	4.2	0.45
The proposed context ontology is clear and easy to follow.				3	2	4.4	0.55
The proposed context ontology is comprehensive and easy to understand.				3	2	4.4	0.55
Assuming the context ontology is realized in a sophisticated tool, I would prefer to use it as it helps me in developing and testing context-aware mobile applications.				3	2	4.4	0.55
Assuming the context ontology in a sophisticated tool is available; I would recommend its usage as the ontology is accurate, complete, and comprehensively documented.				3	2	4.4	0.55

Based on Table 8, it has been found that most respondents considered COCCC ontology useful in helping them identify elements needed to develop a mobile application, as well as assisting them in identifying location and speech related context in a mobile application. Most respondents agreed that the proposed context ontology was clear and easy to follow and understand.

CONCLUSION

Context-aware mobile applications are able to sense their surroundings and adapt themselves to changes. These changes include connectivity and graphical changes according to the user's screen orientation (Püschel et al., 2012).

mIO! Ontology Network has managed to cover all subdomains: device, environment, interface, location, network, provider, service, time, and user (Poveda Villalon et al., 2010). Delivery context focuses on device, environment, location, and network while SOUPA discusses the ontology of location, time, and user (Rodríguez, 2014).

The COCCC ontology focused on the environment and location. Although these subdomains have been covered in mIO! Ontology Network, it introduced context in a general way, failing to go into the details of each subdomain. On the other hand, the COCCC goes in depth into the function of each sensor as well as the API to manipulate context in the Android. In terms of consistency, there is no contradictory knowledge found in the work so far since the ontology construction follows the METHONTOLOGY method step-by-step. The COCCC defined every axiom used; corrected hierarchies levels and was able to relate one concept with another accurately. Future work could consider the extension of more domain knowledge such as touch and visuals in the context ontology. The work could also be extended to iOS mobile applications. This can be achieved by adding iOS API in the context ontology. Also, other possible methods of developing the ontology could be considered as the METHONTOLOGY method has already been used to lay the foundation for creating COCCC ontology.

ACKNOWLEDGMENT

We would like to thank the Ministry of Higher Education Malaysia (MOHE) for funding the research through the FRGS (vote no. 4F857) and Universiti Teknologi Malaysia for providing the facilities and support.

REFERENCES

Android Developer. (2009). Android Developer. Retrieved from https://developer.android.com/index.html

Chen, H., Finin, T., & Joshi, A. (2003). An ontology for context-aware pervasive computing environments. The Knowledge Engineering Review, 18(3), 97–207.

Chen, H., Perich, F., Finin, T., & Joshi, A. (2004). Soupa: Standard ontology for ubiquitous and pervasive applications. The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 258–267.

Corcho, O., Fernández-López, M., Gómez-Pérez, A., & López-Cima, A. (2005). Building legal ontologies with METHONTOLOGY and WebODE. Law and the semantic web. Springer Berlin Heidelberg. 142–157.

Fernández-López, M., Gómez-Pérez, A., & Juristo, N. (1997). Methontology: From ontological art towards ontological engineering. Proceedings of the Ontological Engineering AAAI-97 Spring Symposium Series, 33–40.

Fonseca, J. M. C., & Lewis, R. (2009). Delivery context ontology. W3C Working Draft. (Work in progress.).

Gruber, T. R. (1993). A translation approach to portable ontology specifications. Knowledge Acquisition, 5(2), 199–220.

Mohd-Hamka, N., & Mohamad, R. (2014). OntoUji–Ontology to evaluate domain ontology for semantic web services description. Jurnal Teknologi, 69(6), 21–26.

Lee, K., Lee, J., & Kwan, M. P. (2017). Location-based service using ontology-based semantic queries: A study with a focus on indoor activities in a university context. Computers, Environment and Urban Systems, 62, 41–52.

Noy, N., Fergason, W. R., & Musen, A. M. (2000). The knowledge model of protege. Proceedings of the 2nd International Conference on Knowledge Engineering and Knowledge Management, 17–32.

Pinto, H. S., & Martins, J. P. (2004). Ontologies: How can they be built? Knowledge and information systems, 6(4), 441–464.

Poveda Villalon, M., Suárez-Figueroa, M.C., García-Castro, R., & Gómez-Pérez, A. (2010). A Context Ontology for Mobile Environments. Workshop on Context, Information and Ontologies - CIAO 2010 Co-located with EKAW 2010, ISSN: 1613–0073.

Preuveneers, D., Van den Bergh, J., Wagelaar, D., Georges, A., Rigole, P., Clerckx, T., Berbers, Y., Coninx, K., Jonckers, V., & De Bosschere, K. (2004). Towards an extensible context ontology for ambient intelligence. EUSAI, 148–159.

Püschel, G., Seiger, R., & Schlegel, T. (2012). Test Modeling for Context-aware Ubiquitous Applications with Feature Petri Nets. Proc. Workshop Model-based Interactive Ubiquitous Systems (MODIQUITOUS), 37–40.

Rodríguez, N.D., Cuéllar, M.P., Lilius, J., & Calvo-Flores, M.D. (2014). A survey on ontologies for human behavior recognition. ACM Computing Surveys (CSUR), 46(4), 43.

Subhashini, R., & Akilandeswari, J. (2011). A survey on ontology construction methodologies. International Journal of Enterprise Computing and Business Systems, 1(1), 60–72.

Sulaiman, M. S., Nordin, S., & Jamil, N. (2017). An object properties filter for multi-modality ontology semantic image retrieval. Journal of Information and Communication Technology, 16(1), 1–19.

Uschold, M., & Gruninger, M. (1996). Ontologies: Principles, methods and applications. Knowledge engineering review, 11(2), 93–136.

Wang, X. H., Zhang, D. Q., Gu, T., & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL. Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference, 18–22.

Yusuf-Asaju, A. W., Zulkhairi Md Dahalin, & Azman Ta'a. (2018). Framework for modelling mobile network quality of experience through big data analytics approach. Journal of Information and Communication Technology (JICT), 17(1), 79–113.

Zeshan, F., & Mohamad, R. (2012). Medical ontology in the dynamic healthcare environment. Procedia Computer Science, 10, 340–348.