Context Modeling for Context-Aware Systems

Dennis Lupiana
Institute of Finance Management, Dar es Salaam, Tanzania

Abstract

Modeling context is a key factor for enabling context-aware systems to be responsive to users' needs and consequently for making computing "invisible". A context model, which is a product of context modeling, provides a systematic way of identifying and representing knowledge about users' environment in context-aware architecture. Through this knowledge, and its sensing and reasoning capabilities, a context-aware architecture is able to determine what is going on in the environment and share this knowledge with context-aware systems. This article presents a research work for modeling context. In particular, the article explains development of a Knowledge-intensive Context Model (KiCM). KiCM extends theory-based context models by including knowledge of users' computers and their computer-related activities.

1. Introduction

A context model provides a simplified representation of the real world in which users and devices interact. Hence, a context model provides a systematic way of identifying and representing knowledge about relevant entities, and relationships among them, required to represent a context. Thus, a context model forms a basis for representing and reasoning knowledge about contexts in a context-aware architecture. Thus, a context model plays a key role on developing a context-aware architecture and subsequently context-aware systems.

This work extends the work in the theory-based context models to develop a model of context. This work adopts Actor-Network Theory (ANT) and Semantic to develop a Knowledge-intensive Context Model (K*i*CM). ANT provides a systematic approach for identifying and representing potential entities and the relationships among them. ANT also treats entities equally. Therefore, K*i*CM is developed based on a comprehensive list of entities and unbiased relationships. Represented by Semantic Network, KiCM is simple and consistently represented.

The rest of the article is organized as follows. A background on context-awareness is provided in section 2. A background on ANT and how is relevant in this work is provided in section. Section 4 describes how SN is adopted to develop K*i*CM. Section 5 describes the steps required to use K*i*CM. An example of how a developer can use K*i*CM is provided in section 6. Section 7 provides an illustration on how knowledge about context modeled by K*i*CM can be represented by different knowledge representation languages. Section 8 provides an analysis of the state-of-the-art while section 9 provides limitations of K*i*CM. A summary and a conclusion are provided in section 10.

2. Background on context-awareness

The philosophy behind context-awareness was introduced by Weiser [1]. He asserted that information within a device's proximity can play an important role into making the use of devices intuitive. He argued that if a device knows its location and surroundings it can adapt its behavior in significant ways. According to him, location is a physical environment rather than merely a piece of information such as a name or geographical coordinates. This is further elaborated when Weiser [2] argued that understanding of people's surroundings is a key into making devices *invisible*.

Conforming to Weiser, Schilit et al. [3] envisaged systems that examine and adapt to user's dynamic contexts. They refer to such systems as *context aware systems*. They argue that as users and their devices move, their contexts change. Hence, these systems are meant to seamlessly adjust their behaviors to provide or gather information in response to their environments and the users' computing needs. Schilit et al. [3] argue that *"the interesting part of the world around us is what we can see, hear and touch"*. They argue that information about location, people you are with, nearby devices, environment conditions and time is crucial in realizing context-awareness.

Context-aware systems have a significant impact on how users perform their daily activities. Imagine walking to a room where a meeting is in progress and your mobile phone automatically switches to a silent mode and back to its normal settings immediately after the meeting is finished. Imagine walking into a lecture room and a projector in the room automatically switches ON and displays your presentation slides. These scenarios and many more, exemplify how context-aware systems can be useful.

2.1. Categories of context-aware systems

Schilit et al. [3] categorize context-aware systems based on their ability to provide information or execute computing services. Similarly, Dey and Abowd [4] categorize these systems by their ability

to present information and services, execute services or facilitate acquisition and management of information for later retrieval. Likewise, this research classifies the existing context-aware systems based on their functionalities.

Context-aware systems that are considered in this article are those that exhibit some kind of autonomy. This article also takes into account only contexts that trigger systems' responses. If a context-aware reminder, for instance, takes identity of the reminder creator, time when the reminder was created and locations where the reminder is required to be triggered, then only location is regarded as a context. This is because the reminder will only be triggered when a user reaches the specified locations. The information about the creator and time the reminder was created only provides details of the reminder.

2.1.1. Service triggering systems. As discussed by Schilit et al. [3], and Dey and Abowd [4], context-aware systems in this category can automatically execute computing services. In such systems, a context is used as an input to trigger execution of a certain computing service. In the Active Badge location system [5], for instance, information about location and identity of a user is used to determine user's location and subsequently to forward calls to the recipient. From a Computer Science perspective, delivering or gathering of content is subject to execution of certain computing services. In this category, however, the definition of a computing service is limited to the services that affect behaviors of devices. Therefore, context-aware systems that can assist users by gathering or delivering some kind of content are separately categorized.

2.1.2. Content gathering systems. In this category, context-aware systems support users by automatically gathering information/context from a certain source. Examples of these systems include classroom multimedia notes generating [6] and experience capturing systems [7]. Like in the previous category, context in these systems is used as an input and hence the systems directly respond to context parameters.

2.1.3. Content delivery systems. Context-aware systems in this category automatically provide content to the users. A system that provides map information to the tourists or a system that reminds users are both providing the users with content. The tour guide system may provide text, image and audio whereas the reminder system may provide text or audio. Examples of these systems include reminder systems [8], tour guide systems [9] and recommendation systems [10].

As is evident, context is regarded as discrete information and thus location, identity and time information each is a context. Schilit et al. [3], however, argue that context is a much more powerful concept and hence researchers should focus on its broader view. In this article, therefore, context is defined as a social setting (such as a meeting) where users involved aim to achieve some goals. Thus an input to a context-aware system is referred to as a *context parameter* and categorized them as *primary* and *secondary*. Primary context parameters are those used to derive other context parameters (i.e. secondary context parameters).

Dey [11] argues that researchers should address difficult issues of knowledge modeling and representation in order to come up with interesting and more useful context-aware systems. Hence this article adopts the definition of a model from Gregory [12] who defines a model as a simplified representation of a reality. The reality that this article is interested with is users' context. Thus, in this article a context model is an abstract and a simplified representation of meaningful relationships between relevant entities required to describe users' context.

3. Theoretical background of K*i*CM

K*i*CM is developed based on the principles of ANT. ANT is a socio-technical theory that focuses on explaining the influence of technology in a society. K*i*CM adopts the theoretical network representation of ANT to specify relationships between relevant entities required to describe a context. These entities, as explained in section 3.2, include the users, venue, time, computing devices and computing services.

3.1. Actor Network Theory

In ANT, network is defined as a point of interaction of actors that forms a society. In this theory the definition of an actor is not limited to a human being, it includes non-human actors. Since a context involves interactions between many entities, it can also be represented as a network. To identify relevant entities, the two principles of ANT are used. Below is the explanation of these principles.

3.1.1. Framing. In ANT, a network is a point of convergence of actors. According to Latour [13], relationships between actors are dynamically formed as actors interact. The authors argue that the relationships between actors should not be stagnant and thus introduced the principle of *translation*. Later, Callon [14] refers to this principle as *framing*. He defines framing as a process of identifying distinct and relevant actors required to accomplish a certain task. Latour [13] refers to this process as a *summing-up* process.

3.1.2. Disentanglement. This principle signifies the importance of actors' flexibility to enter and exit a network. It is through this freedom that networks cannot be caught in a loop [15]. If actors are limited to a particular network, then there will be a fixed set of actors in each network and hence few networks will exist. This principle is also important for defining attributes, intentions and actions of existing actors. Latour [13] argues that what matters most in framing actors is their influence on other actors. Law [16] argues that actors acquire their attributes when interacting with other actors.

The framing principle emphasizes using relevant actors when identifying actors required to define a society. More importantly, the disentanglement principle emphasizes actors that affect other actors and the flexibility of actors to join and quit a network. Since a context involves different entities interacting and affecting each other, these principles are used as a guideline for identifying potential entities required to model a context. Consequently, the theoretical network representation of ANT is used to model a context. Section 3.2 identifies and describes the potential entities of K*i*CM.

3.2. Potential entities

In principle, context occurs in a venue at a particular time whereby at least one user is involved. The venue and/or the user may have one or more devices. Any changes in the venue may imply a change of context. A change can be caused by (i) introducing a new user in the venue, (ii) a change of physical properties such as sound, light and temperature or (iii) a change in the state of devices. Hence, the users, venue, time and devices are potential entities for modeling context. As will be discussed in this section, computing service is also important and hence is adopted as a potential entity.

3.2.1. Users. "*People are a major part of the dynamics of work environments"* [17]. People belong to a community and thus their activities are influenced by each other. As the majority of devices become mobile, users' computing needs may also change due to, for instance, social relationships, sensitiveness of information, and users' emotions. Activity of a research student, for instance, may change as the student's supervisor enters the student's research room. This, subsequently, may change computing services the student and the supervisor may need. Thus the knowledge about users should also be taken into account when modeling a context.

3.2.2. Venue. Peoples exist in a physical world. Hence knowledge about different venues which are accessible to users in their daily routines is also important for modeling a context. Apart from location identity, which can be a name or a number, other physical properties such as temperature, sound and light are also important. Any change in a venue, including any environmental changes such as temperature, sound and light, can imply a change in a context. In an office, for instance, if the users were quiet and then immediately start talking it may imply that a context has changed from the users being 'busy working' to being in a 'meeting'.

3.2.3. Computing devices. Knowledge about devices that users have or available in venues is also important for modeling a context. If a meeting, for instance, is strictly known to be using a projector, then absence of a projector in any room means that the meeting cannot occur in that room. Devices can also be used to determine users' computer-related activities, which are important for recognizing users' ongoing context. Devices within a particular venue can also be a source of sound, temperature and light, which are essential for modeling context.

3.2.4. Time. Users' activities occur within a period of time and therefore knowledge about time is also important when modeling context. Similar entities converge at different timestamps within a day to describe different contexts. To differentiate and keep records of these interactions and the contexts they describe, time is also a crucial entity when modeling context. Additionally, most of the activities are structured and hence they have deadlines which may imply a change in a context and therefore time is required to differentiate between contexts.

3.2.5. Computing services. Relationships between users, venues, computing devices and time provide no useful information about users' computer-related activities and hence it becomes difficult to recognize ongoing context. Analogous to human life, if only relationships between users and their physical environments are taken into account, it will be difficult to infer users' intentions. When a user wants to print a document, for instance, he/she interacts with a computer that through computing services interacts with a printer. Thus, knowledge about computing services is also crucial.

4. Conceptual representation of K*i*CM

The Knowledge-intensive Context Model (K*i*CM) is developed when links between users, venue, computing devices, computing services and time are established. Since Semantic Network (SN) is renowned for knowledge representation, this research adopts it for conceptual representation of the model. SN is a graphical notation for representing knowledge. As noted by Woods [18], the unique feature of SN is the notion of a link which connects individual facts into a total structure.

In SN, there are two types of links; *property links* and *relation links* [19]. A property link specifies a connection between a node and an attribute or set of attributes while a relation link specifies a connection between two nodes. According to Woods [18], if a connection specified by a relation link is between two different nodes, then that link is an *assertional link*. If a connection specified by a relation link is between nodes of similar type, then that link is a *structural link*. The assertional links specify associations between two nodes while the structural links provide more details about the same node, such as *is-a* relationships in ontology.

Fig. 1 provides a conceptual representation of K*i*CM. The entities and relationships are represented as nodes and arcs respectively. To differentiate the nodes and the properties, the circle and oval shapes are used respectively. To differentiate the property links and the relation links, dotted and solid lines are used respectively. The model does not specify any classification of entities and therefore only assertional links are used. To differentiate primary and secondary context parameters, the dotted and solid oval shapes are used respectively. As noted by Henricksen [20], a context model should cater for uncertainties. Hence each entity is assigned with a certainty level, denoted by <*p*>.

According to Woods [18], each property of an entity should be separately specified by a property link. If an entity has five properties, for instance, then five separate property links should be specified. This rule is useful when a finite property list of an entity can be well defined prior to developing a model. In a case where the property list is ill defined and changes depending on where the model is applied or on what sensing technology is available, like in Context-Awareness Computing, this rule is inadequate. The links can be meshed up if the entities specified in a model have many properties. Subsequently, this can make the model too complex and hence difficult to read and interpret.

Thus, to specify the relationships between an entity and its context parameters, we used one property link. Since the visibility of an entity is determined by its primary context parameter, its relation is separately specified. Hence, if an entity has two categories of secondary context parameters, then three property links are used (one to specify relationship between the primary context parameter and the entity, and the rest to specify the relationships between each category of secondary context parameters and the entity). As shown in Fig. 1, for instance, the identity of a user is a primary context parameter and hence its relationship to the entity is specified separately from other relationships. Thus, there are at least two specified property links for each entity.

In K*i*CM, the primary context parameter of each of the entities is strictly defined as the minimum required knowledge of an entity. As for the time entity, all elements of a timestamp should be used. Through this specification, developers use all of the specified entities and their relationships but only a subset of context parameters. Thus, instead of instantiating all context parameters of an entity even if only a few are used, as suggested by Kaenampornpan [21], only a subset can be instantiated. This enables the model to be reused in different problem domains and to be adjusted accordingly. As is illustrated in sections 5 and 6, K*i*CM can be used to extensively model and represent knowledge about contexts.

5. Steps required to use K*i*CM

To use K*i*CM one needs to follow six steps; 1.) naming of contexts; 2.) identifying instances of the entities; 3.) specifying relationships between the instances; 4.) identifying relevant context parameters; 5.) specifying relationships between the instances and their context parameters and 6.) specifying certainty levels of the instances. The rest of this section describes each of these steps.

5.1. Naming of contexts

In this step a developer assigns a unique label that will be used to identify a specific context. Depending on the nature of the phenomenon, a label can be a 'meeting', 'busy on computer or 'busy at desk'. In case there are more than one context, different labels should be assigned to different contexts. This is required in order to differentiate two or more contexts that are within the same problem domain. Logically, the first step should be to identify contexts. In most cases, however, a developer knows the contexts that need to be supported by context-aware systems. Hence including context identification as one of the steps is trivial.

5.2. Identifying instances of the entities

After labeling the required contexts, a developer should identify relevant entities, and their instances, required to represent each context. Since K*i*CM specifies the entities, the developer uses it to identify instances of the entities required to model each context. In this paper an instance is defined as an occurrence of an entity. To identity instances of the entities, relevant nouns from context description should be used. In case there are no relevant nouns or instances to match with any entity from K*i*CM, verbs from the description of the context and domain knowledge should be used to deduce relevant instances. This is further illustrated in section 6.

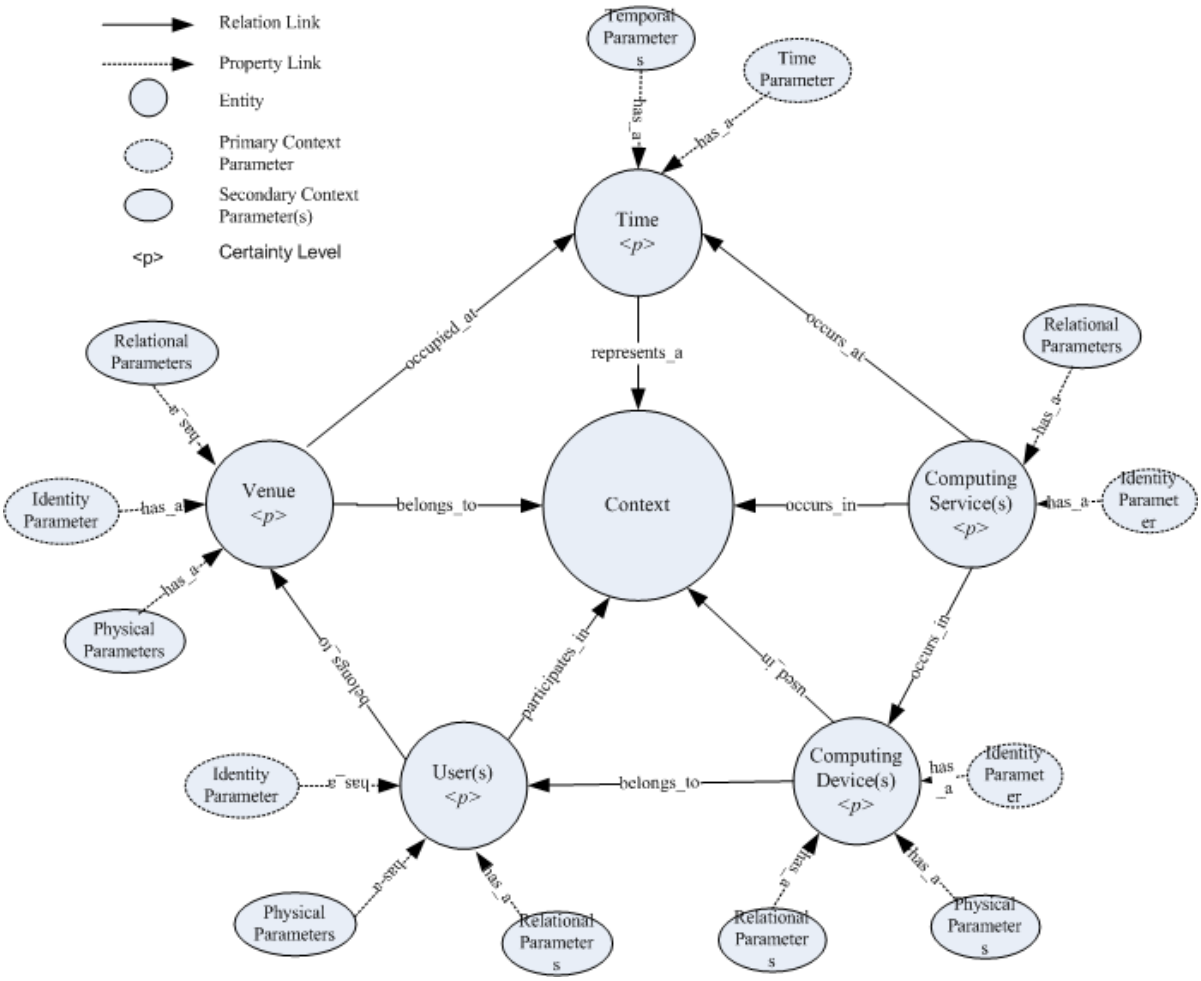

Figure 1. Knowledge-intensive Context Model

5.3. Specifying relationships between the instances

Having car parts, without assembling and connecting them, is not the same as having a car. One must assemble and connect these parts for a car to exist. Likewise, meaningful connections between the instances identified in step 2 must be established for a simplified form of the real world context to exist. In this step, a developer specifies the meaningful relationships between the instances of each of the required contexts as indicated on K*i*CM.

5.4. Identifying relevant context parameters

In this step a developer identifies relevant context parameters for each of the instances identified in step 2. These parameters might be acquired directly through sensors or by deriving from existing context parameters. If the identified context parameters are to be acquired from interpretation of data gathered from sensors, the developer should make sure that appropriate sensing technologies are in place. If a developer identifies sound, for instance, as an important context parameter then a technology to monitor sound level should be available.

5.5. Specifying relationships between the instances and their context parameters

At this stage, a developer will have a skeleton model of a context where by only instances of the entities and their relationships are specified. The developer will also have a set of relevant context parameters for each of the instances. To enrich the model and hence to sufficiently represent real world contexts, each instance of the entities should be connected to its relevant context parameters. In this step, therefore, a developer specifies the relationships between the instances and their context parameters as indicated on K*i*CM.

5.6. Specifying certainty levels of the instances

Lastly, a developer specifies a certainty level of each of the context parameters. This paper defines a certainty level as a value that indicates the degree of trustworthy of a sensor data. As noted by McKeever [22], this value can be obtained from training data, domain expert or manufacturer specifications while taking into account users' actions.

6. Example of using KiCM

To illustrate how K*i*CM can be used, a worked example of a context whereby a research student writes to or reads from his/her computer at his/her desk in his/her research room is used.

6.1. Naming of contexts

In this example, we used **'busy on computer'** label to uniquely identify the specified context.

6.2. Identifying instances of the entities

In the description of the context, there are three nouns of interest; **research student**, **computer** and **research room**. These nouns represent instances of the user, the device and the venue entities of K*i*CM. To deduce instances of the computing services entity, the verbs **writes** and **reads** from the description are used. Since the student writes to and reads from the computer, there should be at least two processes to monitor these activities. These processes are denoted as P_1 and P_2 respectively. Since the student belongs to an organization, domain knowledge is used to deduce working hours and subsequently instances of the time entity.

6.3. Specifying relationships between the instances

Here the connections between the entities are established as specified on K*i*CM, see Fig 1.

6.4. Identifying relevant context parameters

In this example, **name**, **role**, **officename** and attendance **status** are important parameters of the instances of the user entity. The room **identity** and its **category** are important parameters of the instances of the venue entity. The **identity** of the computer, its **owner**, the **room** it is located and its **status** (whether it is ON or OFF) are important parameters of the instances of the computing device entity. The **name** of the processes, their **host** computer, their **status** (whether are active or inactive) and **timestamps** are important parameters of the instances of the computing services entity. **Timestamps** and their time **categories** (such as 'working hours' or 'out of work hours') are important parameters of the instances of the time entity.

Identity and name are primary context parameters since they identify the computer and the room, and the user and the computer services, respectively. Timestamp is also a primary context parameter since it differentiates two points of time. The rest are secondary context parameters. Office name and host of the computing services are relational context parameters since they associate the student and the processes with the room and the student's computer respectively. The owner and the room name of the computing device entity are also relational context parameters because they associate the computer with the student and the room respectively. Table 1 provides a summary of the context parameters.

6.5. Specifying relationships between the instances and their context parameters

Here the connections between the entities and their context parameters are established as specified on K*i*CM.

6.6. Specifying certainty levels of the instances

In this paper, we assume that identity of the user and the venue, status of the computer and the two processes (P_1 and P_2), and timestamps are acquired by interpreting data from sensors. Hence, five sensors will be required to monitor and gather relevant data for the specified context parameters to be acquired. In this example I assume that each of these sensors has a certainty level of 100% i.e. 1.0.

Table 1. Identified Relevant Context Parameters

Devices	Computing Services	Users	Venue	Time
Identity	Name	Name	Identity	Timestamp
Status	Status	Role	Category	Category
Owner	Host	Office name		
Room name	Timestamp	Status		

7. Representing knowledge using K*i*CM

Knowledge about contexts needs to be represented, or *encoded*, in a context-aware architecture as inference rules in order to be processed. To illustrate how K*i*CM simplifies this process, the production rules and Bayesian network are used. These two are used because are common in Context-Awareness Computing and there are many inference mechanisms to support them. In these examples, P_1 and P_2 are assumed to be implemented by a mouse activity monitor and a keyboard activity monitor respectively.

7.1. Production rule

Production rule is a knowledge representation language in production systems where knowledge about a context is represented as an IF THEN rule. A production system is a program that provides pseudo intelligence by emulating cognitive ability of a human [23]. The context parameters are represented as patterns of conditions at the left hand side of the rule while maintaining their relationships by logical operators. The right hand side of the rule specifies actions to be invoked when the conditions are satisfied. In this example, the rule displays a message. In practice, however, the rule can invoke a context-aware system or an application manager.

```
rule "1.0 Busy on Computer"
 when
    $roomResidents: List()
    $userStatus: List()
    $userIn: User()
    $deviceList: List()
    $time: Time(timeFor != "AfterWorkingHours")
    Room($venue: roomname, roomcategory == "Research Room")
    $user: ArrayList(size == 1) from collect (User(userrole == "Student",
    officename == $venue) from $roomResidents)
    not (User(userrole == "Supervisor" || userrole == "Adviser")
    from $roomResidents)
    Device(username == $user.username, room == $venue,
    type == "Computer", status == "ON") from $deviceList
    UserStatus(username == $user.username,
    pname[0].processname == "keyboard", pname[0].status == "true",
    pname[1].processname == "mouse", pname[1].status in ("false", null))
    from $userStatus

  then
    String situation = drools.getRule().getName().substring(4);
    AppManager applicationManager = new AppManager(situation,
    $roomResidents);
    System.out.println("Time: " + $time.time + ", Venue: " + $venue +
    ", Social Context: " + situation);
end
```

Figure 2: Rule Representation of a 'busy on computer' Context

The resulting model can be represented as a rule, as shown in Fig. 2, where the parameters outlined in table 1 are used. Primary context parameters are not shown in the rule because have no direct impact on the occurrence of this context. The parameters are indirectly used to determine the secondary context parameters. Hence, the number of parameters, as outlined in table 1, is reduced to 12. Since a context takes different values of context parameters, one context model may result into more than one rule.

7.2. Bayesian Network

A Bayesian Network (BN) is a directed acyclic graph where nodes represent random variables from a problem domain and directed arcs represent causal relationships between the variables [24]. A node that causes effects is called a *parent* while affected node is called a *child*. When building a BN, one should start by identifying variables of interest, establish relationships between the variables and quantify the relationships [25]. To "quantify relationships" means to specify a conditional probability distribution for each node. The focus of this section is to illustrate how knowledge about context modeled by K*i*CM can be represented as a BN and therefore the quantification of relationships is not illustrated.

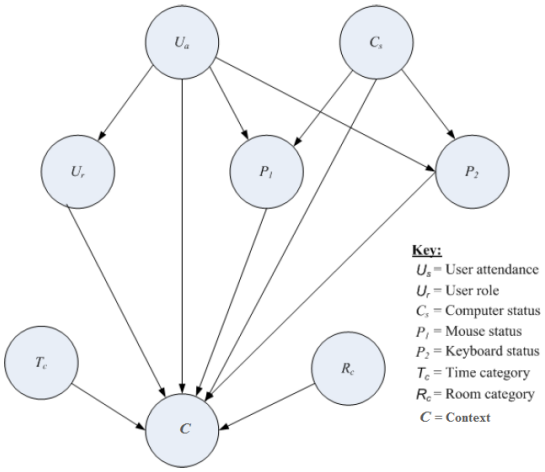

Figure 3: Bayesian Network Representation of a 'busy on computer' Context

With K*i*CM, the first two steps of building a BN are simplified since the variables and the relationships between them are specified in a model of a context. After identifying which variable affects which variable, the resultant BN is shown in Fig. 3.

This BN implies that (i) the likelihood of a mouse or a keyboard to be active depends on the user's presence in the room and the status of her computer and (ii) the probability of the context to occur depends on the presence of the user in the room, her role, category of the room, time and the status of the computer, mouse and keyboard. The primary context parameters are excluded from this BN as they do not have a direct impact on the occurrence of the context. The relationships between the computer, the student, and the room are implicit and hence are not shown in the BN.

8. Related Work

To date, there are many context models. Initial context-aware architectures were designed to directly

respond to sensor-derived context parameters and hence context models were simply a representation of an entity and its attributes. Emphasis was on representation of information related to the data and consequently interpretation of this data. Therefore, much effort was on implementation of appropriate data structures to facilitate interpretation of data captured from sensors into context parameters. As a result, the models are implemented as objects of an implementation language. In this article these models are referred as *attribute-based* because their structure is based on the notion of attribute and value.

Strang and Linnhoff-Popien [26] and Baldauf *et al.* [27] indicate that Ontology is a promising approach for context modeling. The majority of researchers adopt Ontology to abstract domain concepts from implementation languages. A comprehensive list of the existing ontology-based context models is provided by Ye *et al.* [28]. Ontology-based context models organize domain concepts based on individual entities but provide more details about the entities. By using Ontology, for instance, relationships between entities of similar types can be specified by *is-a* relationships. Ontology also specifies different meaning of terminologies used to describe entities and constraints for using these terminologies.

The attribute-based models, however, do not take into account the impact of other nearby entities and domain concepts are integral part of architectures. Hence, these models are based on limited knowledge and incapable of supporting knowledge reasoning. In contrast, ontology-based models are richer and through their *is-a* relationships, can facilitate knowledge reasoning. Through these relationships, for instance, knowledge about the role of a user can be derived from the information about the user's identity. Nonetheless, ontology-based models are still insufficient for developing a meaningful model of context. The models only take into account relationships between entities of similar types and hence fail to capture relationships of other entities, which are fundamental for modeling a context.

To remedy this limitation, Dey [3] and Henricksen [20] proposed a situation abstraction model. This model has become as a *de facto* model in context-aware systems. In this model, context parameters required for a task to be accomplished are used to specify task-specific inference rules in context-aware systems. This enables the systems to be responsive to more than one context parameters simultaneously. Nevertheless, these parameters are limited to a specific task and hence the systems are unable to respond to social dynamics that occur in today's computing environments. Since inference rules are specified in context-aware systems, this model introduces a burden to developers as are required to familiarize with knowledge representation languages.

Recently, researchers have started to adopt socio-technical theories in an effort to develop meaningful models of context. Kofod-Petersen [29] and Kaenampornpan [21], for instance, adopt Cultural Historical Activity Theory (CHAT) to model a context. CHAT is an extension of Activity Theory that provides a theoretical framework for analyzing different aspects of human activities in social settings while emphasizing on a community [30]. CHAT explores relationships between subject, object, artifact, division of labor, rules and community. Kofod-Petersen [29] extends a context model from AmbieSense project by adding social related context parameters as specified by the theory. Kaenampornpan [21] extends CHAT by including time as part of her model. Since the theory implicitly includes a physical environment, she also adopts location as part of her model.

The existing theory-based models, however, are developed to represent different social and physical aspects required to accomplish a user's objective. Hence, these models are limited to relationships between venue, people and time. In addition, both of the existing models are based on Activity Theory, which treats entities differently and hence the relationships among the entities are biased. The theory treats a subject as a ``super entity''. As a result, knowledge about whether users are present or not is used as an input to provide user-tailored computing needs. This implies that knowledge of computer-related activities of nearby users have no impact on recognizing ongoing contexts. This is in the contrary to the real world environments. The status of devices, the users' computer-related activities and social relationships between the users have a significant impact on ongoing contexts.

9. Limitations of K*i*CM

K*i*CM provides a simple way of identifying and representing knowledge about contexts in a context-aware architecture. A context-aware architecture, through its reasoning capability, monitors a real world environment and uses the knowledge it possesses to determine what is going on (or ongoing context). K*i*CM, like any other model, is an approximation of the real world and hence it does not provide a mirror image of real world contexts. K*i*CM only combines relevant knowledge about relevant entities to represent the world in which the users and devices interact. This implies that the knowledge about contexts that a context-aware architecture shall possess shall be intensive but always incomplete.

K*i*CM influences development of context-aware architectures by implying (i) minimum sensing capabilities required by architectures (ii) what and how knowledge about entities within a physical environment should be represented in architectures and (iii) how knowledge about contexts should be

represented and reasoned by architectures. K*i*CM does not support learning, which is an essential attribute of a context-aware architecture that supports context-aware systems in a dynamic environment. With K*i*CM, knowledge represented when architecture is developed remains unchanged and learning capability of a context-aware architecture depends on an inference engine. Therefore, K*i*CM does not provide room for learning capability.

10. Conclusions

This article presented a work that extends theory-based context models to develop a Knowledge-intensive Context Model (K*i*CM). K*i*CM improves the existing context models by including knowledge about more entities that are essential for describing an occurrence of a context. In particular, K*i*CM improves the existing context models by including knowledge of users' computers and their computer-related activities in modeling context. To develop K*i*CM, this work has adopted the Actor-Network Theory (ANT) and Semantic Network (SN).

ANT provides a systematic approach to identify and represent potential entities and relationships among them thus providing no limitation of entities to be used. SN's notations are used to conceptually represent K*i*CM. The notations provide a clear distinction between nodes, attributes and the type of links required to establish the relationships between nodes, and between a node and its attribute or attribute list. This consistently represents complicated relationships between the entities, and entities and their context parameters in K*i*CM thus simplifying the model. This representation also enables few context parameters to be used to model a context. This feature also adds flexibility to identify and use parameters of their choice.

K*i*CM is used to represent knowledge about contexts in a context-aware architecture. Using its reasoning mechanism and sensing capability, the architecture uses the knowledge to determine users' context based on the information it senses from an environment. K*i*CM, however, cannot be directly used by any of the existing context-aware models. A study should therefore be conducted to review existing context-aware architectures and determine their suitability and, if need arise, to develop a suitable context-aware architecture.

K*i*CM requires one to manually represent or encode knowledge represented by K*i*CM to a knowledge representation language that can be understood by a context-aware architecture. This is tedious, especially when developing a context-aware architecture to support context-aware systems in a dynamic environment. A study should therefore be conducted to design and develop a tool to imitate K*i*CM and automatically transform models into knowledge representation languages used by context-aware architectures. The tool, for instance, can be used to create a model of a context using K*i*CM and transform the model into production rules.

11. References

[1] Weiser, M. (1991). The computer for the 21st century. *Scientific American*

[2] Weiser, M. (1994). The world is not a desktop. *interactions*, 1, 7–8

[3] Schilit, B., Adams, N. & Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, 85 –90

[4] Dey, A. & Abowd, G. (2000a). Towards a better understanding of context and context-awareness. In CHI 2000 workshop on the what, who, where, when, and how of context-awareness, 304–307

[5] Want, R., Hopper, A., Falc˜ao, V. & Gibbons, J. (1992). The active badge location system. ACM Trans. Inf. Syst., 10, 91–102

[6] Abowd, G. (1999). Classroom 2000: An experiment with the instrumentation of a living educational environment. IBM Systems Journal, 38, 508–530

[7] Kukkonen, J., Lagerspetz, E., Nurmi, P. & Andersson, M. (2009). Betelgeuse: A platform for gathering and processing situational data. Pervasive Computing, IEEE, 8, 49–56

[8] Ludford, P.J., Frankowski, D., Reily, K., Wilms, K. & Terveen, L. (2006). Because i carry my cell phone anyway: functional location-based reminder systems. In Proceedings of the SIGCHI conference on Human Factors in computing systems, 889–898, ACM

[9] Cheverst, K., Davies, N., Mitchell, K., Friday, A. & Efstratiou, C. (2000). Developing a context-aware electronic tourist guide: some issues and experiences. In Proceedings of the SIGCHI conference on Human factors in computing systems, 17–24, ACM.

[10] H. Liu (2010). Biosignal controlled recommendation in entertainment system. Technische Universiteit Eindhoven, Eindhoven, 1133.

[11] Dey, A.K. (2000). *Providing Architectural Support for Building Context-Aware Applications*. Ph.D. thesis, Georgia Institute of Technology

[12] F. Gregory (1993). Cause, effect, efficiency and soft systems models. Journal of the Operational Research Society, 333-344.

[13] M. Callon (1991). Techno-economic networks and irreversibility. A sociology of monsters: Essays on power, technology and domination, 38, 132-161.

[14] B. Latour (1999). On recalling ANT. The Sociological Review, 47, 15-25.

[15] M. Callon (1999). Actor-network theory - the market test. The Sociological Review, 47, 181-195.

[16] J. John. *After ANT: complexity, naming and topology*. The Sociological Review, 1999, 47, 1-14

[17] B. Schilit (1995). *A System Architecture for Context-Aware Mobile Computing*. Ph.D. Thesis, Graduate School of Arts and Sciences, Columbia University

[18] W. Woods (1975). Whats in a link: Foundations for semantic networks. Tech. rep., DTIC Document

[19] J. Sowa (1991). Principles of semantic networks

[20] K. Henricksen (2003). A Framework for Context-Aware Pervasive Computing Applications. Ph.D. thesis, School of Information Technology and Electrical Engineering, The University of Queensland

[21] M. Kaenampornpan (2009). A Context Model, Design Tool and Architecture for Context-Aware Systems Design. Ph.D. thesis, Department of Computer Science, University of Bath

[22] S. McKeever (2011). Recognising Situations Using Extended DempsterShafer Theory. Ph.D. thesis, School of Computer Science and Informatics, National University of Ireland

[23] A. Newell (1973). Production systems: Models of control structures. Tech. rep., Defence Technical Information Center

[24] F.V. Jensen (1996). An introduction to Bayesian networks, vol. 210. UCL press London

[25] K.B. Korb and A.E. Nicholson (2003). Bayesian artificial intelligence. cRc Press

[26] T. Strang and C. Linnhoff-Popien (2004). A context modeling survey. In : Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England

[27] M. Baldauf, S. Dustdar and F. Rosenberg (2007). A survey on context aware systems. International Journal of Ad Hoc Ubiiquitous Computing, 2, 263-277.

[28] J. Ye, L. Coyle, S. Dobson and P. Nixon (2007). Using situation lattices to model and reason about context. In Modeling and Reasoning in Context (MRC) with Special Session on the Role of Contextualization in Human Tasks (CHUT) which is held in conjunction with CONTEXT, 112

[29] A. Kofod-Petersen(2007).A Case-Based Approach to Realising Ambient Intelligence among Agents. Ph.D. thesis, Department of Computer and Information Science, Norwegian University of Science and Technology

[30] F. Igira and J. Gregory (2009). Cultural historical activity theory. Handbook of research on contemporary theoretical models in information systems, 434-454

12. Acknowledgements

This work has been fully funded by the Institute of Finance Management, URL: www.ifm.ac.tz. I am therefore sincerely thankful for the support from the Rector and Management team of the Institute. I am also very thankful for professional guidance from Dr. Fredrick Mtenzi, Mr. Ciaran O'Driscoll and Prof. Brendan O'Shea of the Dublin Institute of Technology, Ireland.

