2016

KiCM: A Knowledge-Intensive Context Model

Fredrick Mtenzi
Technological University Dublin, Fredrick.Mtenzi@tudublin.ie

Denis Lupiana
Institute of Finance Management, denis.lupiana@ifm.ac.tz

Recommended Citation

Mtenzi, F. & Lupiana, Denis. (2016). KiCM: A Knowledge-Intensive Context Model. *The 11th International Conference for Internet Technology and Secured Transations 2016.* doi: 10.1109/ICITST.2016.7856701

KiCM: A knowledge-intensive context model

2
Author(s)

Dennis Lupiana ; Fredrick Mtenzi

View All Authors

Abstract:
A context model plays a significant role in developing context-aware architectures and consequently on realizing context-awareness, which is important in today's dynamic computing environments. These architectures monitor and analyse their environments to enable context-aware applications to effortlessly and appropriately respond to users' computing needs. These applications make the use of computing devices intuitive and less intrusive. A context model is an abstract and simplified representation of the real world, where the users and their computing devices interact. It is through a context model that knowledge about the real world can be represented in and reasoned by a context-aware architecture. This paper presents a Knowledge-intensive Context Model (KiCM). KiCM improves the existing context models by including knowledge about more entities that are essential for describing an occurrence of users' real context such as a meeting.

Published in: Internet Technology and Secured Transactions (ICITST), 2016 11th International Conference for

Date of Conference: 5-7 Dec. 2016

Date Added to IEEE *Xplore*: 16 February 2017

ISBN Information:

INSPEC Accession Number: 16674908

DOI: 10.1109/ICITST.2016.7856701

Publisher: IEEE

Conference Location: Barcelona, Spain

Contents

Download PDF

Download Citations

View References

Email

Print

Request Permissions

Export to Collabratec

Alerts

SECTION I.
Introduction

Schilit *et al.* [1] argue that context is a much more powerful concept and hence researchers should focus on its broader view. In this paper, therefore, context is defined as a social setting, such as a meeting, where users involved aim at achieving a goal. This definition emphasises meaningful interactions between relevant entities required to describe the real world that is of interest to the users and their devices [2]. In this regard, an input to a context-aware application is referred to as a *context parameter*. Since context parameters play different roles in Context-Awareness Computing, we have categorized them into *primary* and *seconday*. Primary context parameters are those that are used to derive other context parameters (i.e. secondary context parameters). This categorization is also used by Dey [3], and Chen and Kotz [4].

To provide interesting and more useful context-aware applications, Dey [3] argues that researchers should address difficult issues of knowledge modeling and representation. Hence this paper adopts the definition of a model from Gregory [5] who defines a model as a simplified representation of a certain reality. The reality that we are interested with is the users' context. Thus, in this paper a context model is an abstract and simplified representation of meaningful relationships between relevant entities required to describe a users context.

A context model that we refer differs from those for access controls such as in [6] and [7]. A context model that we refer forms a basis for representing and reasoning knowledge about context and hence plays a key role on developing a context-aware architecture. This paper presents a Knowledge-intensive Context Model (*KiCM*). *KiCM* is a model of the real world in which the users and devices interact. To develop *KiCM*, we used the Actor-Network Theory (ANT) and Semantic Network (SN). ANT were used to systematically identify and represent potential entities and relationships among them whereas SN's notations to conceptually represent *KiCM*.

The background of ANT and how is used in developing *KiCM* is provided in section II. The section also provides a description of the potential entities for developing *KiCM*. Section III describes how SN is adopted to develop *KiCM*. Section IV describes the steps that a developer needs to follow to use *KiCM*. An example of how a developer can use *KiCM* is provided in section V. Section VI provides an illustration on how knowledge about context modeled by *KiCM* can be easily represented by different knowledge representation languages. Section VII provides an analysis of the state-of-the-art while section VIII provides a summary and a conclusion.

SECTION II.
Theoretical Background OFKICM

Full Text

Abstract

Authors

Figures

References

Citations

Keywords

Back to Top

KiCM 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. *KiCM* adopts the theoretical network representation of ANT to specify relationships between relevant entities required to describe a context. These entities, as explained in section II-B, include the users, venue, time, computing devices and computing services.

A. 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.

1) Framing

In ANT, a network is a point of convergence of actors. According to Callon [8] and Latour [9], 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 [10] 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 [9] refers to this process as a *summing-up* process.

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 [8]. 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 [9] argues that what matters most in framing actors is their influence on other actors. Law [11] 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 II-B identifies and describes the potential entities of *KiCM*.

B. 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.

1) Users

"*People are a major part of the dynamics of work environments*" [12]. People belong to a certain community and thus their activities are highly influenced by each other. As the majority of devices become mobile, the 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 supervisors may need. Thus the knowledge about the users should also be taken into account when modeling a context.

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) Computing Devices

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

4) Time

Users' activities occur within a period of time and therefore knowledge about time is also important when modelling context. Similar entities converge in different timestamps in the course of 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 modelling 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.

5) Computing Services

It is ease to forget computing services and focus on relationships between users, venues, computing devices and time. These relationships, however, provide no useful information about users' computer-related activities and hence it becomes difficult to recognise 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 which subsequently interacts with a printer. Thus, knowledge about computing services is also crucial.

SECTION III.
Conceptual Representation OFKICM

The Knowledge-intensive Context Model (*KiCM*) is developed when links between users, venue, computing devices, computing services and time are established. Since Semantic Network is renowned for knowledge representation, this research adopts it for conceptual representation of the model. A Semantic Network is a graphical notation for representing knowledge. As noted by Woods [13], the unique feature of Semantic Network is the notion of a link which connects individual facts into a total structure. In this paper, this notion is exploited to conceptually represent the model.

In a Semantic Network, there are two types of links; *property links* and *relation links* [14]. 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 [13], 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.

Figure 1:
Knowledge-intensive context model

View All

Fig. 1 provides a conceptual representation of *KiCM*. The entities are represented as nodes while the relationships are represented as arcs. 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 [15], a context model should cater for uncertainties and hence each entity is associated with a certainty level, shown as <*p*>.

According to Woods [13], 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 *KiCM*, 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 [16], 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 section V and VI, *KiCM* can be used to extensively model and represent knowledge about contexts.

SECTION IV.
Steps Required to Use KICM

To use *KiCM* to model a context, a developer 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.

A. 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 contexts, 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 applications. Hence including context identification as one of the steps is trivial.

B. 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 KiCM 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 KiCM, verbs from the description of the context and domain knowledge should be used to deduce relevant instances. This is further illustrated in section V.

C. 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 KiCM.

D. Identifying Relevant Context Parameters

In this step a developer identifies relevant context parameters of each of the distinct instances identified in step 2. These parameters might be acquired from the interpretation of data gathered from 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.

E. 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 KiCM.

F. 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 [17], this value can be obtained from training data, domain expert or manufacturer specifications while taking into account users' actions.

SECTION V.
Example of Using KICM

To illustrate how KiCM can be used by a developer, 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.

A. Naming of Contexts

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

B. 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 KiCM. 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 organisation, domain knowledge is used to deduce working hours and subsequently instances of the time entity.

C. Specifying Relationships Between the Instances

Here the connections between the entities are established as specified on KiCM.

D. Identifying Relevant Context Parameters

In this example, **name, role, officename** and attendance **status** are important context parameters of the instances of the user entity. The room **identity** and its **category** are important context 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 context parameters of the instances of the compputing device entity. The **name** of the processes, their **host** computer, their **status** (whether are active or inactive) and **timestamps** are important context 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 context parameters of the instances of the time entity.

Table I: 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		

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 I provides a summary of the context parameters.

E. Specifying Relationships Between the Instances and Their Context Parameters

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

F. Specifying Certainty Levels of the Instances

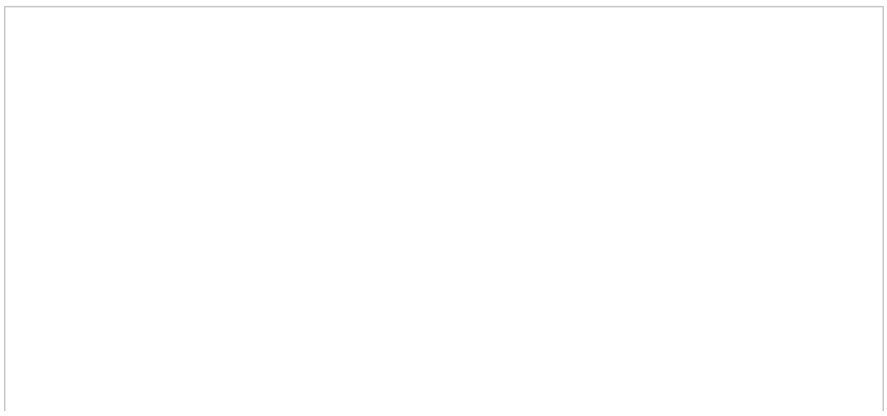
In this paper, we assume that identity of the user and the venue, status of the computer and the two processes (P_1 and P_2), and timestamps are acquired by interpreting data from sensors. Hence, five sensors will be required to monitor and gather relevant data for the specified context parameters to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

SECTION VI.
Knowledge Representation

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 KiCM simplifies this process, the production rules and Bayesian network are used. We used production rules and Bayesian network as are common in Context-Awareness Computing and there are many inference mechanisms to support them. In these examples, we assumed P_1 and P_2 are implemented by a mouse activity monitor and a keyboard activity monitor respectively.

A. 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 [18]–[19]. 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 an application 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

View All

The resulting model of context from section V can be represented as a rule, as shown in Fig. 2, where the parameters outlined in table I 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 I, is reduced to 12. Since a context takes different values of context parameters, one context model may result into more than one rule.

B. 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 [20]–[21]. 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 these variables and thereafter quantify the relationships [22]. 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 modelled by KiCM can be represented as a BN and therefore the quantification of relationships is not illustrated.

With KiCM, 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.

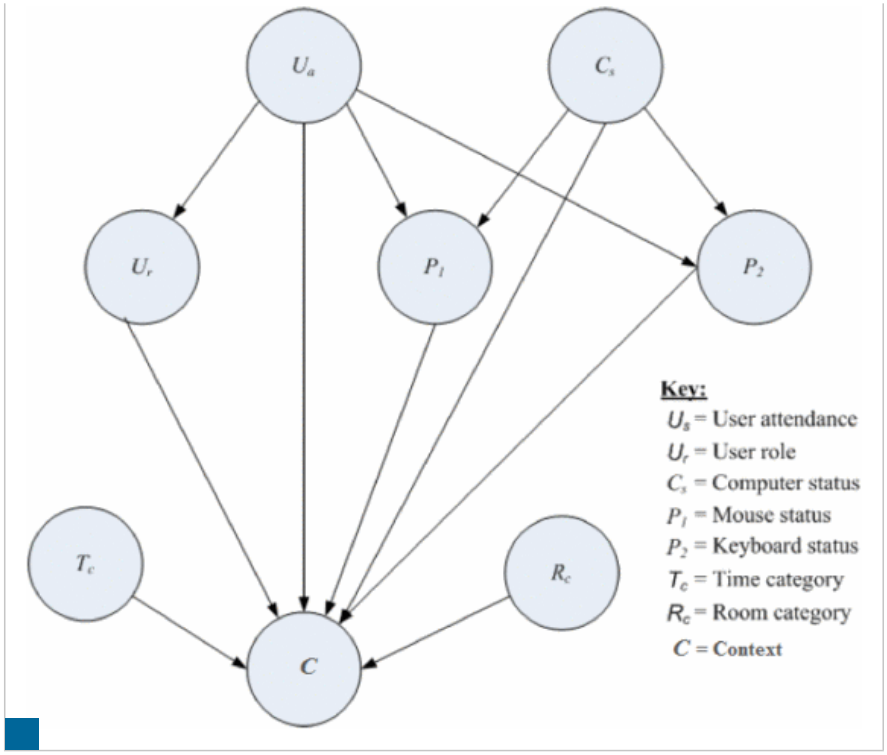


Figure 3:
Bayesian network representation of a 'busy on computer' context

View All

SECTION VII.
Related Work

To date, a lot of effort has been invested in developing context models and therefore there are many context models. The initial context-aware architectures are designed to directly respond to sensor-derived context parameters and hence their context models are simply a representation of an entity and its attributes. Thus, emphasis is on representing this information into machine-interpretable languages and implementing 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 [23]. Strang and Linnhoff-Popien [24] refer to these models as *key-value*. Lupiana [2] and Henricksen [15] refer to these models as *attribute-based* because their structure is based on the notion of attribute and value.

Strang and Linnhoff-Popien [24], and later Baldauf *et al.* [25], indicate that Ontology is a promising approach for context modeling. Consequently, the majority of the researchers [23], [26]–[27] adopted 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]. Like attribute-based, ontology-based context models organise 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 semantically poor and incapable of supporting knowledge reasoning. In contrast, ontology-based models are

semantically rich. Also through their *is-a* relationships, these models facilitate some kind of 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. These 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 [15] proposed a situation abstraction model. This model has become as a de facto model in context-aware applications [29]–[30][31]. In this model, context parameters required for a task to be accomplished are used to specify task-specific inference rules in context-aware applications. This enables the applications to be responsive to more than one context parameters simultaneously. Nevertheless, these parameters are limited to a specific task and hence the applications are unable to respond to social dynamics that occur in today's computing environments. Since inference rules are specified in context-aware applications, this model introduces a burden to developers as are required to familiarise with knowledge representation languages.

Recently, researchers have started to adopt socio-technical theories in an effort to develop meaningful models of context. Kofod-Petersen [32] and Kaenampornpan [16], 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 [33]–[34]. CHAT explores relationships between subject, object, artifact, division of labour, rules and community. Kofod-Petersen [32] extends a context model from AmbieSense project by adding social related context parameters as specified by the theory. Kaenampornpan [16] 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 nearby user(s) 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.

SECTION VIII.
Conclusion

This paper presented a Knowledge-intensive Context Model (*KiCM). KiCM*) is developed by using Actor-Network Theory (ANT) and Semantic Network. ANT provides a systematic approach for identifying and representing potential entities and relationships among them. This feature enables *KiCM* to include computing devices and computing services. This is an important feature as it enables *KiCM* to take into account computer-related activities of nearby users. This makes *KiCM* knowledge intensive and hence more realistic to model situations where the users and their devices continuously interact.

Semantic Network's notations are used to conceptually represent *KiCM*. 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 simplifies

and consistently represents the complicated relationships between the entities, and entities and their context parameters in *KiCM*. This representation enables few context parameters to be used to model a context. This feature gives developers the flexibility to identify and use parameters of their choice.

Unlike many predective models of machine learning, to evaluate the performance of *KiCM* there is no need for a dataset. Only a context-aware architecture is required for providing sensing and reasoning mechanisms. *KiCM* is used to represent knowledge about contexts in a context-aware architecture. Using its reasoning mechanism, the architecture uses the knowledge to determine the users' context based on the information it senses from an environment. *KiCM*, however, cannot be directly used by any of the existing context-aware models. A study should therefore be done to review existing context-aware architectures and determine their suitability and, if need arise, to develop a suitable context-aware architecture.

Keywords

IEEE Keywords

Context, Context modeling, Computational modeling, Semantics, Internet, Computer architecture, Temperature

INSPEC: Controlled Indexing

ubiquitous computing, knowledge representation

INSPEC: Non-Controlled Indexing

computing devices, knowledge-intensive context model, KiCM, context-aware architectures, context-aware applications, user computing needs, dynamic computing environments

Author Keywords

Actor Network Theory, Context-Awareness, Context Model, Context Modelling, Ontology

Authors

Dennis Lupiana
Faculty of Computing, Information Systems and Mathematics, Institute of Finance Management, Dar es Salaam, Tanzania

Fredrick Mtenzi
School of Computing, Dublin Institute of Technology, Ireland

Related Articles

Conducting Situated Learning in a Context-Aware Ubiquitous Learning Environment
Ting-Ting Wu; Tzu-Chi Yang; Gwo-Jen Hwang; Hui-Chun Chu

Living with Internet of Things: The Emergence of Embedded Intelligence
Bin Guo; Daqing Zhang; Zhu Wang

Modeling of sensor data and context for the Real World Internet
Claudia Villalonga; Martin Bauer; Vincent Huang; Jesus Bernat; Payam Barnaghi

A unified semantic knowledge base for IoT
S. N. Akshay Uttama Nambi; Chayan Sarkar; R. Venkatesha Prasad; Abdur Rahim

Smart Spaces and Smart Objects Interoperability Architecture (S3OiA)
Mario Vega-Barbas; Diego Casado-Mansilla; Miguel A. Valero; Diego Lopez-de-Ipina; Jose Bravo; Francisco Florez

Dynamic Service Model Based on Context Resources in the Internet of Things
Jianhua Liu; Weiqin Tong

Composition of Self Descriptive Protocols for Future Network Architectures
Dennis Schwerdel; Abbas Siddiqui; Bernd Reuther; Paul Müller

Enabling Runtime Evolution of Context-Aware Adaptive Services
Mahmoud Hussein; Jun Han; Jian Yu; Alan Colman

Context-aware user model for personalized services
Aekyung Moon; Young-il Choi; Byung-sun Lee

Internet-based teaching evolution in Computer Architecture
Eugenio Lopez; Elio Sancristobal; Sergio Martin; Gabriel Diaz; Manuel Castro; Juan Peire; Jose M. Gomez; Paloma Lopez