

ULRR

Classification and modeling of the quality of contextual information

| | |
|---------------|---|
| Item Type | Article |
| Authors | Razzaque, Mohammed Abhur;Dobson, Simon;Nixon, Paddy |
| Citation | IEEE TCAAS Letters; |
| Publisher | IEEE Computer Society |
| Download date | 2026-06-06 13:43:26 |
| Item License | https://creativecommons.org/licenses/by-nc-sa/1.0/ |
| Link to Item | https://hdl.handle.net/10344/1524 |

Classification and Modeling of the Quality of Contextual Information

M.A. Razzaque, Simon Dobson, and Paddy Nixon

Abstract – Pervasive and autonomic environments make extensive use of contextual information to guide adaptations to changing external demands and circumstances. Context takes many forms, and some form of categorisation of information from different viewpoints can assist designers in deciding how information should affect behaviour, and inform the handling of uncertain or conflicting information. We discuss the various approaches to context modeling, and derive an initial methodology for including contextual information in adaptive applications.

Index Terms – Pervasive computing, autonomic computing, development methodologies

I. INTRODUCTION

Pervasive and autonomic computing envisage a world with users interacting naturally with device-rich environments to perform a variety of tasks [15]. These environments are dynamic and heterogeneous, and are required to be self-managing and autonomic, demanding minimal user guidance. In this heterogeneous environment, context-adaptation is a key concept to meet the varying requirements of different clients [2]. In order to enable context-aware adaptation, context information must be gathered and eventually presented to the application performing the adaptation. It is clear that some form of context classification is helpful given the wide range of heterogeneous context information. Two important classification viewpoints are:

- *Conceptual viewpoint* – who, where, what occurs, when, what can be used, what can be obtained
- *Measurement viewpoint* – what is the room

Manuscript received 24 March 2006. This work is supported by Science Foundation Ireland under the grant “Secure and Predictable Pervasive Computing”.

The authors are with the Systems Research Group, School of Computer Science and Informatics, UCD Dublin IE (email abdur.razzaque@ucd.ie).

temperature or network bandwidth or network latency

Systems can typically directly adopt the measurement viewpoint, but applications are typically designed using the conceptual viewpoint. To facilitate the programming of context-aware applications an infrastructure is necessary to gather, manage and disseminate context information to applications. There are number of existing approaches to context description, generally based on one of the following methods:

- Set theory
- Directed graphs
- First-order logic
- Preferences and user profiles

Most models fail to both represent dependency relations between the diverse context information and to utilise these dependency relations. Some support only a narrow classes of context and applied to limited types of application. Furthermore most do not consider the issue of *Quality of Context*, where the infrastructure takes account of the reliability, precision and other factors associated with the data being collected. This will be a critical issue for the next-generation pervasive computing: the quality of a given piece of contextual information will dramatically effect the decisions made by an autonomous application.

In this paper we explore several approaches context classification and modeling, with a view to determining the various factors that affect a system’s ability accurately to model its environment. We develop some quality notions for contextual information, and use these to develop an initial methodology for context modeling.

Section II briefly discusses context and context awareness. Section III presents context classification and analyses its benefits, which are then used in section IV to analyse various existing approaches. Section V presents

an initial methodology for incorporating contextual information into applications in a principled manner, while section VI concludes with some directions for the future.

II. CONTEXT AND CONTEXT AWARENESS

It is unlikely that a single definition of context would be accepted by all researchers: the definition varies from time to time, and from application to application. Historically, context has been adapted from linguistics, referring to the meaning that must be inferred from the adjacent text. In respect to computing world definitions of context varies with computing environment (available processors, devices accessible for user input and display, network capacity, connectivity, and costs of computing) user environment (location, collection of nearby people, and social situation) and physical environment (lighting, noise level etc). According to Dey, context is “any information that can be used to characterize the situation of entities (i.e. whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity and state of people, groups and computational and physical objects [3].” Although this definition encompasses the definitions given by previous authors, it is sometimes too broad. Winograd has given a more specific and role based definition: according to him context “is an operational term: something is context because of the way it is used in interpretation, not due to its inherent properties [22].” Most recently Coutaz *et al.* stated that context “is not simply the state of a predefined environment with a fixed set of interaction resources. It is part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed, and multiscale resources [2].”

Context awareness is a term from computer science, which is used for devices that have information about the circumstances under which they operate and can react accordingly. Context-aware computing involves application development that allows for collection of context and dynamic program behavior dictated by knowledge of this environment. Context-awareness is not unique to ubiquitous computing: explicit user models used to predict the level of user expertise or mechanisms to provide context-sensitive help are good examples used in many desktop systems. With increased user mobility and increased sensing and signal processing capabilities, however, there is a wider variety of context available to tailor program behavior. Through the use of context-

awareness, rapid personalisation of computing and communications services is increasingly possible.

However, current computer systems – even on mobile devices – remain unaware of the user's context. They do not discern what the user is doing, where the user is, who is nearby and other information related to the user's environment. Instead they take the explicit input from the user, process it, and output the result.

Pervasive computing will greatly change the way today's computers behave. The basic idea is to instrument the physical and digital worlds with various sensors, actuators, and tiny computers. A huge amount of information can then be collected and processed by computer systems, enabling computer systems to deduce the user's situation and act correspondingly with user's intervention [16].

III. CONTEXT CLASSIFICATION

Context classification refers to the recognition that “not all sources of information are created equal,” and forms a vital part of context-aware applications. The intention is to help application designers and developers to uncover the possible context and simplify context manipulation, without introducing “unfounded certainty” from ignoring inherent or generated errors. Classification of context information can be helpful in providing quality context information. For example, conflicts can be resolved by favouring the sources of context that are most reliable (static, profiled) over those that are more often subject to error (sensed, inferred, derived).

Different systems have typically taken different views on classifying their context sources, including:

- Internal context (the state of the user) *versus* external context (the state of the environment) [9]
- Material context (the location, device and available infrastructure) *versus* social context (social aspects and personal traits) [17]
- Primary context (location, time and activity) [3]
- Primary context (user environment, physical) *versus* secondary context (environment, computing environment) [19]

| Category | Semantics | Examples |
|-------------------------|-----------------------|---|
| <i>User context</i> | Who? | <i>User's profile</i> : identifications, relation with others, to do lists, etc |
| <i>Physical context</i> | Where? | <i>Physical Environment</i> : humidity, temperature, noise level, etc |
| <i>Network context</i> | Where? | <i>Network environment</i> : connectivity, bandwidth, protocol, etc |
| <i>Activity context</i> | What occurs, when? | <i>What occurs, at what time</i> : enter, go out, etc |
| <i>Device context</i> | What can be used? | <i>Profile and activities of devices</i> : identifications, location, battery lifetime, etc |
| <i>Service context</i> | What can be obtained? | <i>Information on functions which system can provide</i> : file format, display, etc |

Figure 1. Some possible conceptual contextual parameters

None of these models really captures the distinctions between the characteristics of different contextual sources. A broader classification of viewpoints are the *conceptual* or domain-level view, and the *measurement* or physical view. Systems will typically be able to access the latter directly, while the former is more useful to applications.

It is interesting to note that this distinction in viewpoint is often not explicitly recognised by designers. Many location-based systems, for example, are phrased in terms of an application's adaptation to a user's movements in space (a conceptual view of context). However, the underlying sensor infrastructure often has access only to observations of a user's PDA, RFID tag or other device (measurement view). The measured observations act as a *proxy* for the concept of user location, but are not *actually* that location: the user may have lent their PDA to someone else, forgotten their RFID badge, and so forth. This use of one form of context as a proxy for another is a significant source of potential errors.

A. Structure within viewpoints

The classification of context viewpoints provides a useful starting point, but even within a viewpoint certain additional structures can be identified.

1) Measurement viewpoint

Measurement involves abstracting a physical or (less frequently) digital phenomenon into a more tractable form. A number of different measurement styles may be identified

In *continuous context* the value of context changes continuously. The continuous context component is function of

- current value of the context component,
- lowest threshold value
- highest threshold value
- compare value
- the metric of the value and it uses function formula for the calculation.

In *enumerative context* the values of context are a set of discrete values and defined in a list or set. The set may be infinite, but more typically will consist of a finite collection of values from which one or more are chosen at any given time.

An important limited form of enumerative context is *state context*. This category consists of two opposite values and they toggle between them. This provides an important link to predicate calculus.

Finally, *descriptive context* is based on the description statement of the context and for this purpose it uses predicate calculus to combine other statements.

2) Temporal view

Within the above forms of context we may differentiate between elements of similar form but different temporal properties. *Static context* describes those aspects of a pervasive system that are invariant, such as a person date of birth, social security number etc. However, pervasive systems are typically characterized by frequent changes; the majority of information is dynamic. The persistence of *dynamic context* information can be highly variable; for example, relationships between colleagues typically last for months or years, while a person's location and activity often change from one minute to the next.

The relationship between dynamic context and truth is subtle. Using an old dynamic value can be a source of errors in applications. However, whether a *particular use* of old information will lead to an erroneous behaviour depends critically on the degree to which behaviour changes with values. An old value, if “close” to the “real” value, may result in an “acceptable” (even if technically “wrong”) behaviour. We have used quotation marks for good reason: these concepts are all semantic – that is to say, conceptual – rather than being strictly the domain of measured context. It is in general impossible to assign a measurement significance to the degree of “outdatedness” of a measurement, although such significance is often obvious when considered in the conceptual viewpoint.

3) *Outline conceptual viewpoint*

The *users’ context* refers to information that is directly or indirectly related to one or more users. It is the context of which the system should be aware is that of one or more humans. Predominately users’ context is driven by his/her goals and their corresponding application.

The *system’s context* is composed of a model of the user’s context plus a model of its own internal context. The system’s model of the user’s context provides the means to determine what to observe and how to interpret the observations. The system’s model of its own context provides a means to compose the federation of components that observe the user’s context.

A number of conceptual contexts are outlined in figure 1.

4) *Derivation of context*

The importance of the measurement/conceptual distinction is that the conceptual viewpoint derives indirectly from the measurement viewpoint. In general it is vital that we maintain the process by which this derivation is obtained, in order to understand (for example) which measurements give rise to which concepts. This linkage between concept and measurement may then be used to ensure that behavioural adaptation follows the structure of the external measured world [5].

Direct or primary context is directly derived from the sensors or information sources. Generally, this context maintains a one-to-one relation with the measurement underlying it. By contrast, *derived or secondary context* is derived from one/more primary context is known as the derived or secondary context. Sometimes due to

unavailability of appropriate sensors or context we need to use derived context. Again this context can be derived using the following ways:

In the *one-to-one* case single context is derived from a primary context. For example, a GPS co-ordinate of a location is a primary context and derived name space is the secondary context. In the *many-to-one* case more than one primary context will need to generate the secondary context. For example, for the derivation of the comfort-ability (derived context) of a room we need temperature, humidity, sound level, etc.

In the *one-to-many* case one primary context or a derived context will need to generate more than one secondary context. For example, if we know the temperature in a room is in a particular season (say summer) is higher than the usual one then we can assume or derived that the air conditioning is not working, room occupancy is high and so on. Even if we know the information the comfort (derived context) of a room, we can derive approximate temperature, humidity, sound level, and so on.

Conceptual and measurement viewpoint contexts could be again classified as static or dynamic contexts. The above classifications are not exhaustive: future pervasive computing where context information will exhibit more diverse characteristics but these could be very helpful for application designer and developer in pervasive computing to manipulate context information efficiently.

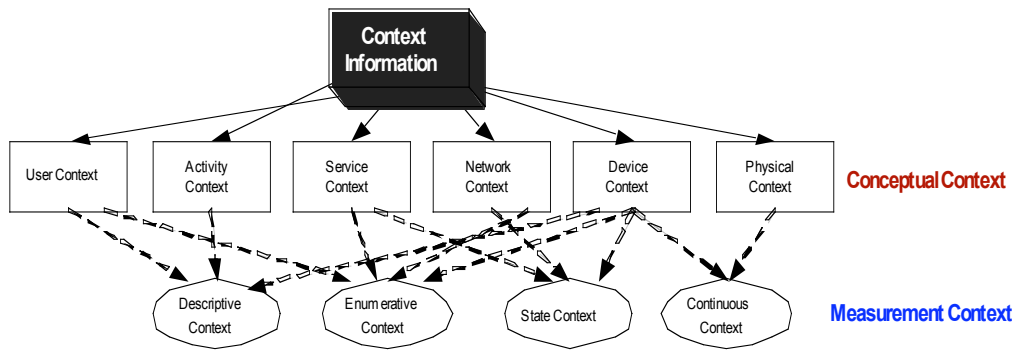


Figure 2: Hierarchical classification of context information

B. Benefits of Context Categorization

In near future use of context awareness in the computing and communication world will be used widely. That will require to deal with wide range of contexts. In that situation, some of the possible benefits we can get from the categorizations are followings:

- **Context manipulation.** Context classification can help application designer and developer to uncover the possible context and simplify the context manipulation.
- **Quality of context:** Classification of context information can be helpful in providing quality context information. For example, conflicts can be resolved by favoring the classes of context that are most reliable (static followed by profiled) over those that are more often subject to error (sensed and derived).
- **Selection of appropriate providers:** It is possible that a particular context may be gathered by different infrastructures or context providers. Now to select the best possible and appropriate context information for a particular application or services from more than one context providers categorization of context may help in selecting the best possible context source.
- **Context refinement:** It can be utilized for the context refinement process of a context-aware system. Main concern of context refinement is the derivation of high level context information form low level context information. This derivation is necessary due to unavailability of appropriate context sources. As mentioned earlier categorization could be help in obtaining a measure of quality of context, and the quality of the particular low level context information is

an significant indicator of whether or not the generation of high level context information makes sense at all, and , if so, how to determine the quality of the produced context information. Not only context refinement but also transformation between different formats of representation or techniques like detection, filtering, or inter- and extrapolation can be manipulated further.

IV. CONTEXT MODELING

To facilitate the programming of context-aware applications an infrastructure is necessary to gather, manage and disseminate context information to applications. And this infrastructure ultimately requires the modeling of contextual information. Context modeling is highly important to capture user requirements/profile, application requirements, device capabilities and relationships between contexts

Context information is gathered, stored, and interpreted at different parts of the system. A representation of the context information should be applicable throughout the whole process of gathering, transferring, storing, and interpreting of context information. Most of the existing context models are based on one of the following methods:

- Set theory
- Directed graphs
- First-order logic
- Preferences and user profiles

A. Set theory

Schmidt *et al.* used set theory for the context presentation. The context T is described by a set of two-dimensional vectors [20]. Each vector h consists of a symbolic value v describing the situations and a number p indicating the certainty that the user (or the device) is currently in this situation.

Yau *et al.* also used set theory for the context and a *context-tuple* is defined as a tuple $\langle a_i, a_j, a_k, \dots, a_n, t \rangle$ of size n , where n is the number of unique contextual-data sources present in the device. Each variable a_i in the tuple represents a value, which is valid for the corresponding type of context. The variable t represents the time of the tuple creation time [23].

Set theory describes context schematically and dependency relations are not embodied.

B. Directed graphs

Hendrickson *et al.* proposed an object-based context modeling in which context information is structured around a set of entities, each describing a physical or conceptual object such as person or communication channel [11]. It uses the form of a directed graph for the diagrammatic representation of context, in which entity and attribute types form the nodes, and associations are modeled as arcs connecting these nodes. This is a comprehensive model which includes QoC and dependency relations but fails to represent the dependency relation accurately.

C. First-order logic

Ranganathan *et al.* proposed a context model named ConChat and it is based on first-order predicate calculus and Boolean algebra [18]. It covers the wide variety of available contexts and supports various operations, such as conjunction and disjunction of contexts and quantifiers on contexts. It allows the creation of complex first-order expressions involving context, so it is possible to write various rules, prove theorems, and evaluate queries. This modeling is consists of the four elements:

- the type of context
- the person, place, or thing, with which the context is concerned
- a value associated with the subject
- a comparison operator, verb, or preposition

Examples:

context(people, Room 22, >=, 3)
context(application, PowerPoint, Is, Running)
context(RoomActivity, 22, Is, Presentation)

This is a well-defined modeling to specific field like electronic chat but in this model relation between continuous data cannot be described easily and even it is not dealing with QoC.

D. Preferences and user profiles

Composite Capability/Preference Profiles (CC/PP [13] is the W3C's proposal for a profile representation language and it is a framework based on the Resource Description Framework (RDF). CC/PP is intended to express both device capabilities and user preferences. Its specification defines a basic structure for profiles. A profile is basically constructed as a strict two-level-hierarchy: each profile having a number of components, and each component having a number of attributes (shown in figure 3). The particular components and attributes are not defined by the CC/PP specification. The definition of a specific vocabulary is up to other standardization bodies. Although CC/PP able to fulfill all the requirements except structural property of profile representation mentioned but vocabulary is not rich enough; it needs to be extended. Most importantly it can't represent the complex relationships and constraints. Even Component/Attribute model becomes clumsy if there are many layers.

Comprehensive Structured Context Profiles (CSCP) [10] is based on the Resource Description Framework (RDF) and overcomes the shortcomings of the Composite Capability/Preference Profiles language (CC/PP) regarding structuring. Furthermore it extends the mechanisms to express user preferences. It cannot represent the complex relationships and constraints. Component/Attribute model becomes clumsy if there are many layers.

E. Dependency relations

From the above study it is quite clear that existing context models are suffering at certain extent which makes them not very suitable as a context model for future pervasive systems. Future's full fledged pervasive systems will require much more sophisticated context models in order to support seamless adaptation to changes

in the computational environment. The context models will need to specify a range of characteristics/quality of context information including temporal characteristics (freshness and histories) accuracy resolution (granularity) confidence in correctness of context information, as well various types of dependencies among the different context information.

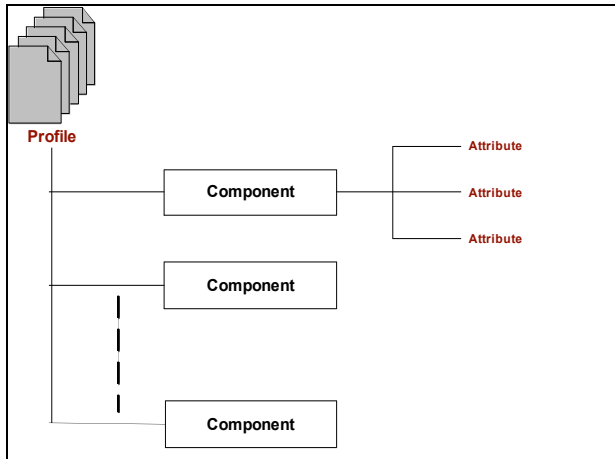


Figure 3. CC/PP

Future pervasive and context aware systems will need to deal with heterogeneous services and contexts. It is very likely that these context information will be somehow interrelated and dependent. According to Henrickson, “a dependency is a special type of relationship, common amongst context information, which exists not between entities and attributes, as in the case of associations, but between associations themselves [11].” Here associations are the unidirectional relationships between the entity and its attributes and a dependency shows the reliance of one association upon another. Efstratiou *et al.* showed the importance of capturing dependencies in context aware applications. Without knowledge of such dependencies, inappropriate decisions can be made by context-aware applications that lead to instability and unwanted results **Error! Reference source not found.** Moreover, knowledge of dependencies is important from a context management perspective, as it can assist in the detection of context information that has become out-of-date. Dependency relations will be critical in diverse context information and it can't be ignored most of the cases. Above analysis on the number of existing context models show that they don't include these dependency relations and suffer for this issue. Hence future context models should include these dependency relations more comprehensively.

Constraint Logic Programming Language is a style of

programming language, which allows the programmer simply to state relationships between objects and this, could be used for the description dependency relation [14]. Constraint languages provide powerful, high-level descriptions for rule-based systems modelling which can operate on different types of (primary and derived) data. Consider, for example, displaying information in a smart phone like Nokia 6630. Figure 4 shows a sample scenario of the dependency description related to display information in a smart phone where two main concerns are battery power and file format.

V. TOWARDS A METHODOLOGY FOR ADDING QUALITY OF CONTEXT TO A CONTEXT MODELING FRAMEWORK

In context aware systems, errors in context information may arise as a result of errors in *gathering* (sensing), *interpretation* and *presentation* level. As context information is relied upon by applications to make decisions on the user's behalf, it is indispensable that applications have some means by which to judge the reliability of the information. *Quality of context* (QoC) is a judgment parameter or criteria for the contextual information or data. Most of the existing context models do not consider the issue of quality. This will be a critical issue for the next generation pervasive computing; primarily because the quality of a given piece of contextual information will dramatically effect the decisions made by the autonomous application. Poor information or data quality can have severe impact on the overall effectiveness of the context aware system. Therefore inclusion of QoC in the future context model is highly necessary.

Pervasive and context aware systems will need to deal with heterogenous applications which will require diverse context information. Moreover these assorted applications will require various qualities of service. To provide these QoS we need various QoC to be incorporated in the context model.

Before analyzing or managing information or data quality, one must understand what information or data quality means. Information quality management requires understanding which dimensions of information quality are important to the user or application. According to Wang *et al.* we can define *QoC* in terms of information quality parameters and information quality indicators as below [21]:

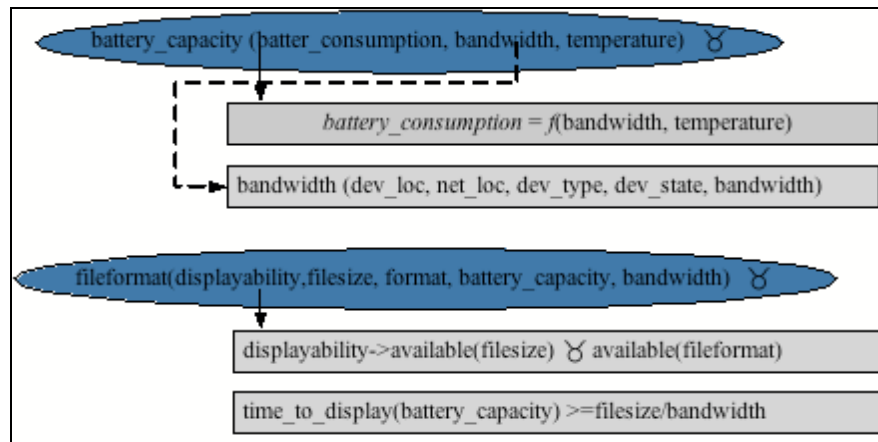


Figure 4. Dependency Description

- An information quality parameter is a qualitative or subjective dimension by which a user evaluates context information quality. Source credibility and timeliness are examples.
- An information quality indicator is a context information dimension that provides objective information about the context. Source, creation time, and collection method are examples.
- An information quality attribute is a collective term including both quality parameters and quality indicators.
- An information quality indicator value is a measured characteristic of the gathered and stored data. The information quality indicator source may have an indicator value like from a sensor or user.
- An information quality parameter value is the value determined for a quality parameter (directly or indirectly) based on underlying quality indicator values. Application-defined functions may be used to map quality indicator values to quality parameter values. For example, because the source is user himself for his date birth information, so credibility is high.
- Information quality requirements specify the indicators required to be tagged, or otherwise documented for the information related to an application or group of applications. If a context model includes this then it is possible to make the context aware system more efficient and effective.

The need for diverse quality of context information has been broadly recognized in number of research works, yet none of the existing works address the problem in an adequate or general way. Dey *et al.* suggest that ambiguity in information can be resolved by a mediation process involving the user [4]. But in case of potentially large quantities of context information involved in pervasive computing environments and the rapid rate at which context can change, this approach places an unreasonable burden on the user. Ebling *et al.* describe a context service that allows context information to be associated with quality metrics, such as freshness and confidence, but their model of context is incomplete and lacks formality [6]. Castro *et al.* define the notion of quality based on measures of accuracy and confidence, but their work limited to location information [1]. Schmidt *et al.* associate each of their context values with a certainty measure that captures the likelihood that the value accurately reflects reality [20]. They are concerned only with sensed context information, and moreover take a rather narrow view of context quality. Gray and Salber include information quality as a type of meta-information in their context model, and describe six quality attributes: coverage, resolution, accuracy, repeatability, frequency and timeliness [7]. Finally Henricksen *et al.* include QoC in their directed graph based context model but this could be limited to this sort of modeling [11]. Most of their quality models are not formally defined, as they are intended to support requirements analysis and the exploration of design issues, rather than to support the development of a context model that can be populated with data and queried by applications.

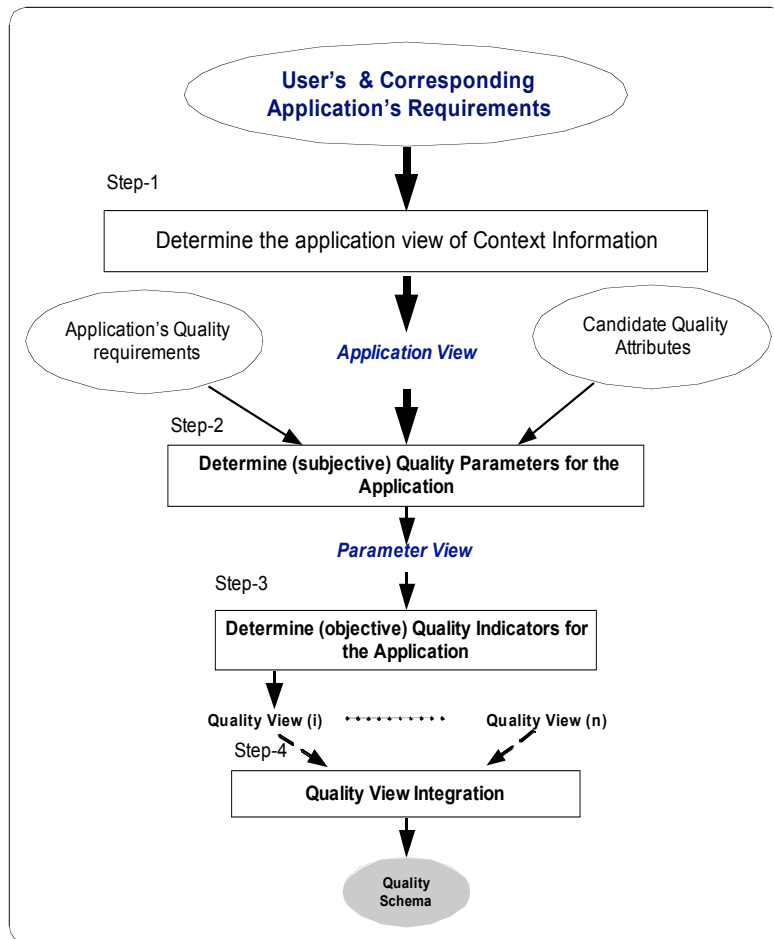


Figure 5. The process of quality contextual information modeling

| Step | Input | Output | Process |
|------|---|------------------|---|
| 1 | User's and Corresponding Application's requirements | Application view | It embodies traditional context information modelling and objective is to extract and document application requirements of context information. |
| 2 | Application view, application quality requirement, candidate quality attributes | Parameter view | It determines the quality parameters (like timeliness, reliability etc) to support information quality requirements. |
| 3 | Parameter view(application view included quality parameters) | Quality view | It converts the subjective quality parameters into measurable characteristics or quality indicators(like timeliness to date, etc) |
| 4 | Quality view/views | Quality schema | This involves the integration of quality indicators. |

Figure 6. Brief description of the methodology for quality contextual information modeling

Considering the above limitations in quality modelling, our effort is to provide a generic approach of quality context information modelling based on Wang *et al.* [21] Figure 5 shows the step by step methodology for quality contextual information modelling where initial input is *user's and corresponding application's* requirements and the final outcome of the modelling is the *quality schema*. Each step includes the *input, process* and *output*. Figure 5 provides a brief description of each step

VI. CONCLUSION

Next-generation context aware systems have to deal with diverse context information. Classification of this context information will be helpful for the context aware application designers and developers. To address this issue, this paper deals with categorizations and quality modeling in context information.

We have explored the various ways in which contextual information can be classified along a number of different axes. We have further illustrated the need to maintain linkages and dependencies between information at different levels, especially in derived context where the errors inherent in measured information can affect inferencing.

A more principled approach to context modeling must take account of these linkages and errors – the overall quality of contextual information. Such a model provides important information to any reasoning system layered on top of a context model, since the build-up of uncertainty cannot always be masked from applications. We believe that this approach to context-aware applications development – in which rich interconnections are leveraged in a number of ways to control adaptations – is central to building reliable and predictable adaptive systems.

REFERENCES

- [1] P. Castro, P. Chiu, T. Kremenek, R. Muntz “A probabilistic room location service for wireless networked environments,” Proceedings of UbiComp 2001. Atlanta GA, 2001
- [2] J. Coutaz, J. Crowley, S. Dobson, and D. Garlan. “Context is key.” Communications of the ACM **48**(3), March 2005
- [3] A.K. Dey, G.D. Abowd. “Towards a Better Understanding of Context and Context-Awareness,” CHI2000 Workshop, 2000
- [4] A. Dey, J. Manko, G. Abowd, “Distributed mediation of imperfectly sensed context in aware environments,” Technical Report GIT-GVU-00-14, Georgia Institute of Technology, 2000.
- [5] S. Dobson, P. Nixon, “More principled design of pervasive computing systems,” in Human computer interaction and interactive systems, LNCS 3425, Springer Verlag, 2005.
- [6] M. Ebling, G.D.H. Hunt, H. Lei, “Issues for context services for pervasive computing,” Middleware 2001 Workshop on middleware for Mobile Computing. Heidelberg DE, 2001
- [7] C. Efstratiou, K. Cheverst, N. Davies, A. Friday. “An architecture for the effective support of adaptive context aware applications,”. Proceedings of Mobile Data Management. Hong Kong, CN: Springer, 2001, pp.15-26
- [8] P. Gray, D. Salber, “Modelling and using sensed context in the design of interactive applications,” Proceedings of the 8th IFIP Conference on Engineering for Human-Computer Interaction. Toronto CA, 2001
- [9] J.Gwizdzka. “What’s in the Context?,” CHI2000 Workshop, 2000
- [10] A. Held,, S. Buchholz,, A. Schill, “Modeling of Context Information for Pervasive Computing Applications,” Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002). Orlando, FL, 2002
- [11] K. Henriksen, J. Indulska, A. Rakotonirainy. “Modeling Context Information in Pervasive Computing Systems,” Proceedings Pervasive 2002, 2002.
- [12] J. Indulska, R. Robinson, A. Rakotonirainy, K. Henriksen. “Experiences in Using CC/PP in Context-Aware Systems.” Proceeding of Mobile Data Management, 2003.
- [13] G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, “Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies,” W3C Working Draft, Mar 15, 2001
- [14] K. Marriott, P.J. Stuckey, “Programming with Constraints: An Introduction,” MIT Press, 1998.
- [15] N. Streitz and P. Nixon, “The Disappearing Computer”, Communications of the ACM **48**(3), March 2005
- [16] P. Nixon, F. Wang, S. Terzis and S. Dobson. “Engineering context aware systems,” Proceedings of the International Workshop on Engineering Context-Aware Object-Oriented Systems and Environments, 2002
- [17] D. Petrelli, E. Not, C. Strapparava, O. Stock, M. Zancanaro. “Modeling Context is Like Taking Pictures,” CHI2000 Workshop, 2000
- [18] A. Ranganathan, R.H. Campbell, A. Ravi, A Mahajan. “ConChat: A Context-Aware Chat Program,” IEEE Pervasive Computing **1**(3), July-Sept. 2002, pp.51–57
- [19] B. Schilit, N. Adams, R. Want. “Context-aware computing applications,” Proceedings of IEEE workshop on Mobile Computing Systems and Applications. 1994, pp.85-90
- [20] A. Schmidt, K. A. Aidoo, A. Takaluoma, U. Tuomela, et.al. “Advanced Interaction in Context,” Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing, 1999
- [21] R.Y. Wang, H.B. Kon,, “Data Quality Requirements Analysis and Modeling,” Proceedings. Ninth International Conference on Data Engineering, 1993, pp.670-677
- [22] T. Winograd, “Architecture for Context,” Human Computer Interaction **16**, 2001, pp.401-419
- [23] S.S. Yau, F. Karim. “Context-Sensitive Middleware for Real-time Software in Ubiquitous Computing Environments.” Proceedings. 4thIEEE International Symposium on Object-Oriented Real-Time Distributed Computing, 2001. Pp.163-170
- [24] Arkady Zaslavsky. “Adaptability and Interfaces: Key to Efficient Pervasive Computing.” NSF workshop series on Context-Aware Mobile Database Management, 2002

M.A. Razzaque Mohammad Abdur Razzaque is a PhD student in the School of Computer Science and Informatics at UCD Dublin. He holds bachelor’s and master’s degrees from the University of Dhaka in Bangladesh. His research focuses on the use of multi-level context to adapt the behaviour of communications networks.

Simon Dobson Simon Dobson is with UCD Dublin's School of Computer Science and Informatics, where his research centres on the semantics of adaptive systems and the programming languages and methodologies used to create them. He has held academic and research posts at Trinity College Dublin and the CLRC Rutherford Appleton Laboratory, and for two years was CEO of a research-based start-up company. He has published extensively in the fields of distributed and pervasive systems.

Paddy Nixon Paddy Nixon is Professor of Distributed Systems in the School of Computer Science and Informatics at UCD Dublin. His interests are in distributed and pervasive computing systems, supported by over 70 internationally-reviewed papers. Prior to moving to UCD he was a professor at the University of Strathclyde and academic director of the Kelvin Institute, and academic/industrial partnership funded by the Scottish government.