Experiences in Using CC/PP in Context-Aware Systems*

Jadwiga Indulska, Ricky Robinson, Andry Rakotonirainy, Karen Henricksen

School of Information Technology and Electrical Engineering
The University of Queensland
St Lucia QLD 4072 Australia
{jaga, ricky, andry, karen}@itee.uq.edu.au

Abstract. Future pervasive systems will be based on ubiquitous, often mobile, interconnected devices supporting mobile users in their computing tasks. These systems need to be context-aware in order to cope with highly dynamic environments. In this paper, we present a context model and a context management system able to support a pervasive system infrastructure. This context model is based on the CC/PP standard proposed to support content negotiation between Web browsers and servers. We have defined a set of CC/PP components and attributes that allow to express a variety of context information types and relationships between context descriptions. The paper discusses pros and cons of using CC/PP as a basis for a context model and a context management system.

1 Introduction

The emergence of new types of mobile and embedded computing devices and advances in wireless networking extended the domain of computing from the workplace and home office to other facets of everyday life. This trend is leading towards future computing systems, termed pervasive systems, in which cheap, interconnected computing devices are ubiquitous and capable of supporting users in a wide range of tasks. Pervasive systems need to be able to deal with mobility of users and their devices and also, in the future, with users who change their computing device while running some applications. Pervasive computing environments will differ from current computing environments in many respects, but particularly in scale and complexity. As a result, the success of pervasive computing technologies will rely on a radical design shift. It will require computing applications to become more adaptive, in order to cope with highly dynamic environments and changing user requirements, as well as less demanding on user attention. Accordingly, applications will need to become increasingly sensitive to context. By context, we refer to the circumstances or situation in which a computing task takes place.

* The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science and Tourism of the Commonwealth Government of Australia.

As the programming of context-aware applications is complex and error-prone, there has been a recent emphasis on the creation of infrastructure that assists with tasks such as context acquisition from sensors [1, 2] and context modelling and dissemination [3–7]. However, the success of context-aware applications also demands appropriate standards for applications to express and exchange context information. Such a standard is currently being developed by the W3C in the form of CC/PP [8, 9]. CC/PP offers an XML-based format for exchanging context descriptions, as well as vocabularies for expressing context information related to device capabilities and user preferences. As CC/PP was designed to support content negotiation between Web browsers and servers, the types of context information it accommodates are limited. In this paper, we present types of context information that are needed in pervasive computing, describe a set of CC/PP components and attributes which we defined to express a variety of context information types and analyse some of the shortcomings of CC/PP. We also describe the architecture of our context management system and evaluate this architecture with respect to scalability.

The structure of the paper is as follows. Section 2 presents a brief overview of CC/PP. Section 3 characterises the types of context information that are important in pervasive computing, while section 4 evaluates the ability of CC/PP to support the types of context identified in section 3, and briefly presents some extensions to the CC/PP vocabulary. Section 5 presents an analysis of the architecture of our context management system and it is followed by section 6 which outlines the lessons learned through our work on CC/PP based context descriptions. Section 7 describes related work on context description and finally, section 8 concludes the paper.

2 Overview of Composite Capabilities/Preference Profiles (CC/PP)

Composite Capabilities/Preference Profiles (CC/PP) is a W3C proposed standard for describing device capabilities and preferences with a focus on wireless devices such PDAs and mobile phones. The intended use of such a profile is to allow an origin server to deliver content customised to a device's capabilities and preferences. When a device makes a request using a protocol such as HTTP, it sends its CC/PP profile in the request. The origin server can then filter, translate and adapt the content to meet the requirements of the requesting device.

CC/PP is based upon the Resource Description Framework (RDF) [10, 11]. RDF is a way of representing statements, each of which contain a subject, predicate and object. RDF can be serialized in many different ways, but CC/PP uses the XML serialization. A CC/PP profile is essentially a two-level tree containing components and attributes of those components. In this model, statements are made about the components (the subject) that have named attributes (the predicate) and values for those attributes (the object). For instance, a profile may specify that the screen component has a resolution of 640 pixels by 480 pixels. Each profile refers to one or more schemata that govern the types of components

that may appear in the profile, and the attributes that belong to each component. It often occurs that the profiles of many devices are the same or similar. For this reason, CC/PP includes the notion of component *defaults*.

These default values may be provided by a hardware or software vendor and stored in a place easily accessible by origin servers. If the profile received from a client device contains its own values for any of the default component attributes then they override the default values. In this way, a client need only specify those components and attributes which vary from the defaults, thereby saving space on the client device and minimizing the use of bandwidth between the device and the origin server.

3 Context information and its management

The goal of pervasive/ubiquitous systems is that computers and computing tasks fade into the background, so that users are less concerned with devices and specific applications. This goal includes "anywhere, anytime" computing which allows users to access the same computing applications and information on their workstation in the office, on their computing devices at home or on a variety of mobile devices when the users are on the move. The pervasive systems need to be able to deal with mobility of users and their devices and also, in the future, with users who change their computing device while running some computing applications. Pervasive computing infrastructure should allow users to move their computational tasks easily from one computing environment to another and should allow them to take advantage of the capabilities and resources of their current environment.

One of the great challenges of pervasive computing is to capture the requirements and current status of computing environments to allow both evaluation of changes in this environment and decisions about neccessary adaptation methods which can be applied to adapt the whole system to these changes. Such adaptations can be carried out at the various layers of the system (e.g. by adapting user applications or applications' communication streams, adapting the distributed computing infrastructure or the behaviour of the underlying operating system or communication protocols).

Therefore the research on context information models is concerned with creating models which describe (i) the features of the whole computing environment (user computing devices, computer network, operating system, distributed computing platform, etc.), (ii) user's computing applications, user requirements and preferences, and (iii) computing application's requirements (e.g, Quality of Service requirements for its communication streams, computational requirements, security) and information about how these requirements are currently met by the computing environment. In addition, some constraints and relationships which exist in the computing environment should also be captured in a context model as they should affect decisions about possible adaptations (these include relationships like device ownership, software licencing [12], but also dependencies between context information, like dependencies between bandwidth and power

in a mobile device [13]). The research on context can go beyond supporting "anywhere, anytime" computing in order to provide a better support for user everyday tasks [14]. In the latter case, which is not addressed in this paper, the role of context information is to capture a variety of user needs (tasks) and to provide intelligent computing support for these tasks with minimal user interactions.

The goal of our work was to extend the CC/PP vocabulary which focusses on mobile devices to capture basic classes of context information needed for the infrastructure of pervasive systems. The CC/PP approach was extended to include additional vocabulary for user context information (user profile describing, among others, user applications, user devices, user current location), device capabilities (device profile which includes hardware and software capabilities, device location, networks to which the device is and can be attached, potential and current network QoS on the device's network interfaces), and application requirements and their current status (application profile). The following section shows some examples of these extensions and also shows how relationships and context dependencies were captured in our approach.

Context management in pervasive systems must contain at least the following three separate but complementary functionalities:

- Context sensing (gathering context information) from a variety of hardware and software sensors.
- Interpretation of partial context information gathered from sensors (may involve complex processing and resolution of conflicting context information) to produce context information and represent it in a common format.
- Management of context information represented in common format.

The third functionality needs to provide a persistent storage of context information and allow a variety of clients to access the information or receive notification about context changes. Pervasive systems may be geographically large, have a heterogeneous computing and networking infrastructure, involve a large variety of context sensors and have a large number of clients of context information. Even one type of context information, like location information, may be gathered from a variety of hardware sensors (e.g. GPS coordinates, phone cells) and software sensors (e.g. operating system agent observing user activity on the keyboard, or device's IP address). The location information needs to be processed and conflicts need to be resolved. As users can be mobile such location information needs to be gathered in various domains. Due to these requirements the architecture chosen for a context management system has to be scalable to cope with a large variety of context types, a large variety and number of objects for which context has to be gathered, and a potential large number of clients of context information. In section 5 we discuss the architecture of our context management system and the design decisions which made the the proposed architecture scalable.

4 CC/PP based context information

CC/PP provides a model and a core vocabulary for describing device capabilities and user preferences. The goal of our project was to evaluate whether CC/PP can be used to model more general types of context information. Specifically, whether CC/PP is capable of modelling those types of context information described in Section 3.

The core CC/PP schema defines necessary items such as Components, Attributes and the enclosing Profile element. A small example attribute vocabulary is also provided. Third parties, such as device manufacturers and application developers, may introduce their own attribute vocabularies. Furthermore, they may introduce new schemas that contain specialized subtypes of the CC/PP Component class, and define Attributes that apply to these specialized components. For example, the UAProf [15] specification defines HardwarePlaform, SoftwarePlatform and BrowserUA to be subclasses of the generic CC/PP Component class. Then the UAProf specification defines attributes for each of these specialized classes.

Although CC/PP is designed to describe information about device hardware and software, it can describe a much wider variety of context information, as long as that context information can be described in terms of CC/PP Components and Attributes (or subtypes of them). For instance, location information can be modelled in CC/PP by defining a Location component, and perhaps some subclasses such as PhysicalLocation and LogicalLocation. Relevant attributes can then be defined for each of these Location subclasses.

Our extended vocabulary defines many new CC/PP components in order to describe a wide range of context information. Such components include Network Interfaces of devices, Quality of Service, Location, DisconnectionStatus, and Application Requirements. Such context information is used in our pervasive infrastructure to support a wide range of adptation techniques applied to applications when their computational context changes. For example, location information is used to allow usage of nearby devices, provide services which are relevant in a given location, recognise that a user is moving out of network coverage, etc. Quality of Service information describes QoS requirements of applications and is used to make decisions about adapting the application's communication stream or seamlessly moving the communication stream to a different overlaying network. The latter also requires information about network interfaces available on a given device - such information is needed when a decision is made to move the communication stream (seamless handover) to a different network. In this kind of adaptation the QoS requirements are used to map the stream to an appropriate QoS class in a new network, e.g. a QoS class in a GPRS network.

There is also a large number of complex relationships and constraints that need to be captured by the context model, and this is where it becomes difficult and unintuitive to use CC/PP as the underlying context model. The next subsection provides some examples of the Component and Attribute vocabulary we have created to describe more varied kinds of context than those described

by UAProf and the basic CC/PP vocabulary. The profile component-attribute trees are shown in a compact notation.

4.1 Examples of CC/PP extensions

Location There are several components that relate to Location information. These are PhysicalLocation, LogicalLocation, GeodeticLocation, Orientation and Modifications. The last component relates to measurement errors. The following diagram shows the structure of a Location profile.

```
[LocationProfile
    [PhysicalLocation [Country, State, City, Suburb]]
    [LogicalLocation [IPAddress]]
    [GeodeticLocation [Longitude, Latitude, Altitude]]
    [Orientation [Heading, Pitch]]
    [Modifications [VerticalError, HorizontalError, HeadingError, PitchError]]
]
```

NetworkCharacteristics NetworkInterfaces are modelled as a complex attribute of the NetworkCharacteristics component. NetworkInterface also has a complex attribute called QoS, which describes the quality of service offered by the network interface. DisconnectionStatus is also a complex attribute of the NetworkCharacteristics component.

```
[DeviceProfile
    [NetworkCharacteristics
        [DisconnectionStatus
            [ConnectionStatus, DisconnectionType,
             DisconnectionPeriod, ExpectedDisconnectionPeriod]]
        [NetworkInterface
            [QoS [Bandwidth, Jitter, Latency, PacketLoss, Throughput],
             InterfaceType, IPAddress]]
    ]
]
```

There may be many NetworkInterfaces contained in an RDF bag or sequence.

ApplicationRequirements The ApplicationRequirements models the requirements of a particular application installed on a device. The requirements may be related to hardware, software and communication Quality of Service. Each component is of a type defined in the UAProf specification, or of a type already defined above. Therefore, instead of listing the attributes of these types, the type of the component is given. The reader is referred to the UAProf specification [15] for further information.

```
[ApplicationRequirements
    [HardwareRequirements [type=HardwarePlatform]]
    [SoftwareRequirements [type=SoftwarePlatform]]
    [BrowserRequirments [type=BrowserUA]]
    [WapRequirements [type=WapCharacteristics]]
    [PushRequirements [type=PushCharacteristics]]
    [QoSRequirements [type=QoS]]
]
```

CurrentSession The CurrentSession profile captures the description of a user's current computing activities. For instance, the CurrentSession profile describes which device the user is currently using, the application the user is currently using, which network interface is currently being used and so on.

```
[CurrentSessionProfile
    [CurrentUser [CurrentSessionID, URI]]
    [CurrentDevice [URI]]
    [CurrentApplication [URI]]
    [CurrentNetworkCharacteristics [URI type=NetworkCharacteristics]]
    [CurrentPhysicalLocation [URI type=PhysicalLocation]]
]
```

The URI attributes point to some resource. Where the type is not constrained to an explicit CC/PP component type, the URI may point to any resource that uniquely describes the component. For example, the URI for the CurrentUser may link to the user's homepage, or it may reference an RDF resource. Where the type is constrained, the URI must point to a CC/PP resource of that type.

Relationships and Dependencies We found it convenient to model some relationships by grouping related components together in their own profile. For example, to state that a particular user is permitted to use only certain devices we can create a profile as follows.

```
[UserDeviceProfile
    [User [URI]]
    [PermittedDevices [Bag of URI]]
]
```

This profile states that the user identified by the resource at the given URI is permitted to use any of the devices in the RDF bag, each of which is identified by a URI. A different approach is to define another attribute for the User component called permittedDevices whose value is an RDF bag of device URIs.

Finally, there are certain dependencies we wish to model. For example, using more bandwidth requires more power. Therefore if the pervasive system infrastructure decides to increase bandwidth of a current application, the battery power has to be checked first. This is modelled at the schema level by introducing a "depends" RDF property between two attributes. First of all, a schema

is needed to define the "depends" property. Then, the schema that defines the related attributes (such as bandwidth and batteryPower) asserts the dependency between the two attributes. In our example, the "depends" relation will be defined as a property of the bandwidth CC/PP attribute, whose value will be the batteryPower attribute. This is shown in the XML fragment below.

```
<ccpp:Attribute rdf:ID="bandwidth">
  <rdf:range rdf:resource='#QoS'/>
  <constraints:depends rdf:resource='#batteryPower'/>
</ccpp:Attribute>
```

The above XML assumes that the QoS component type and the bandwidth attribute are defined in the same schema as the batteryPower attribute. It also assumes the existence of a constraints schema that defines the "depends" property.

4.2 A note on CC/PP structure

It should be noted that some of the described components have attributes which themselves are components. For example, the NetworkInterface component is modelled as an attribute of the standard UAProf NetworkCharacteristics component. The NetworkInterface component has an attribute called QoS, which itself is a structured attribute. To overcome the two-level tree constraint in CC/PP (as outlined in Section 3 of [8]), the value of each structured attribute is a URI that points to the complete description of the complex attribute. Usually the URI will be a fragment identifier that points to another CC/PP component within the same profile. This is achieved by using the RDF resource property as follows:

```
<prfx:networkInterface rdf:resource='#myNetworkInterface'/>
```

where the myNetworkInterface fragment describes the attribute in its entirity.

5 Context Management Architecture

The architecture of our context management systems consists of a set of interacting components (or modules) organised into three layers as shown in Figure 1. Each layer uses a message based notification service called Elvin [16] to interact with other layers. Elvin uses a client-server architecture for delivering notifications. Clients establish sessions with an Elvin server process and are then able to send notifications for delivery, or to register to receive notifications sent by other components. The Elvin subscription language allows to form complex regular expression. The architecture can be distributed over several locations due to the distribution transparency provided by Elvin.

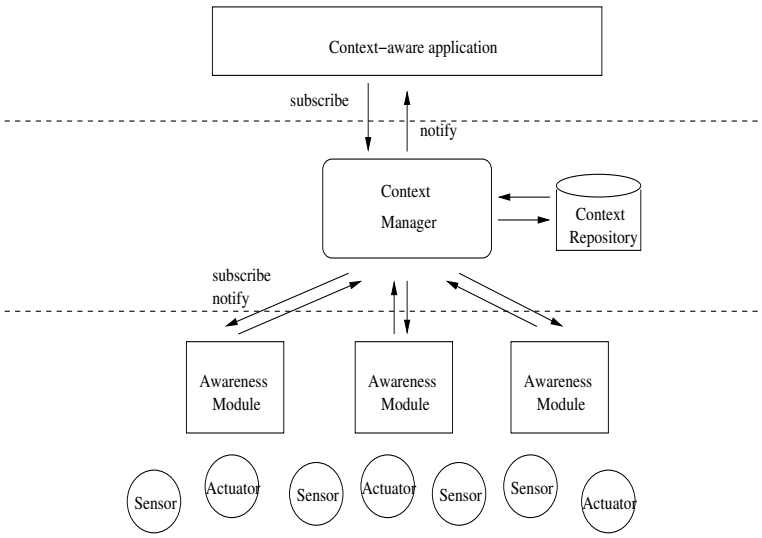

Fig. 1. Context management architecture

5.1 Three layer architecture

The ability to dynamically reconfigure the architecture in order to adapt it to new requirements or a changing environment was the main design principle. The architecture is structured into three autonomous layers:

- Context-aware Application layer: In order to be aware of the current context, the adaptation engine of the pervasive system and/or the context-aware application can subscribe to a set of events of interest (e.g. context changes) using Elvin's subscription language. These clients are then notified by the Context Manager upon occurrence of the events of interest.
- Context Manager layer: The Context Manager manages the persistent repository (database) of context information, receives context updates from the Awareness Module layer, detects context changes and provides notifications about the changes to all interested clients. The Context Manager manages events to/from upper and lower layers. The context manager can be seen as a layer that (i) co-relates events from different sensors (ii) coordinates events to be sent to actuators and (iii) cre ates relevant events to be sent to the applications.
- Awareness Module layer: These modules process context information from sensors. They monitor sensors and actuators. An awareness module is a specialised module that provides a particular functionality. For example, it can serve a GPS sensor that provides location coordinates or an actuator that locks a door. Awareness modules transform information from sensors into a format that can be understood by the context manager. They also transform events from the Context Manager to a format understandable by

actuators. The awareness modules use Elvin to send/receive events to/from the Context Manager. The communication between Sensors/Actuators and the awareness modules are often proprietary (e.g a specific protocol that uses RS-232).

5.2 Context repository

A context management systems needs to provide a persistent storage for context information which was gathered from sensors, processed and converted to a common format. There are efforts underway to define a mapping from RDF to a relational database [17]. The approach taken is to map the RDF model directly to one or more relational database tables. Since CC/PP is an application of RDF, it follows that CC/PP profiles can be stored to a database using one of these mappings. We used this approach to map our CC/PP context descriptions into a relational database. The advantage of such an approach is that even if new vocabularies are added to our CC/PP context model in the future, the mapping does not have to change. Profiles can be reserialised into XML from the database. Although the resulting XML may look different from the original profile (for instance, the mapping application may choose different identifiers for namespaces), the resulting profile will be equivalent to the original.

While there is ongoing work on querying XML files, including the work of the W3C XML Query working group, we used mappings of context information into a database. There are two main reasons for this choice (i) the potentialy large volume of context information stored in a repository, (ii) the work on XML queries is still a work in progress. Performance should be gained by storing profiles in a relational database rather than storing them as XML documents in a repository. The schemas can also be placed into the database. This increases the speed with which a profile can be validated against the schemas it references.

5.3 Awareness Modules

Several awareness modules have been developed for our context management system. The most complex of them is a location manager which receives location information from a variety of physical (GPS, badge system, movement sensors) and logical sensors (IP address, user actions observed by an operating system agent). The module provides mapping of a variety of sensed location information into a physical location. The complexity of the location manager stems from two requirements: (i) the manager needs to resolve conflicting location information (e.g. a reading from user's mobile phone left at home and information about user's current activities, e.g. keyboard typing, in the office) and (ii) privacy of location information has to be protected. We address the first issue by assigning weights to various location readings. To address the privacy issue the system provides authentication of clients requesting location information and we are currently analysing whether the emerging P3P standard can be modified to allow specification of privacy rules for location information.

5.4 Scalability

A context management system can potentially have a large volume of context information gathered from numerous distributed sensors and may need to support many clients requesting context information. To achieve a scalable context management system it is necessary to provide scalable solutions for the system's components. The solutions in the following areas will have high impact on the whole system's scalability:

- reception of context information from the Awareness Module layer and interaction with clients;
- updates of context information in persistent repositories;
- distribution of context information.

While the clients of the Context Manager may query the context database, it was assumed for scalability reasons that the main type of communication will be based on notifications. The same notification system is used to provide interactions between clients and the Context Manager and between the Context Manager and the awareness modules. Elvin has a sophisticated mechanism for the registration of interest in events for which notifications are to be delivered, and therefore it allows clients to specify the granularity of context information notifications, reducing the amount of communication required.

As context information is stored in a persistent repository, a scalable method for updating this repository needs to be developed for sensors which sense context information very frequently (e.g., a sensor providing GPS coordinates of a fast moving car). This is because the organisation of the repository (e.g., a database) is not usually designed for very frequent updates. There are many existing approaches (e.g. spatial indexing techniques) to this problem which need to be employed in the Awareness Module layer in order to limit the number of notifications sent to the Context Manager. As Elvin allows registration for a specified granularity of notifications, we achieved a scalable approach in the following way: the Context Manager registers with the appropriate awareness modules for notifications about changes in particular context parameters (e.g., current location, speed, direction) and the required granularity of this information (e.g., 'notify if the difference between the current location and the previous notification is greater than 100m). The clients of the Context Manager may need different granularities of context information. Therefore, to meet the needs of all the clients, the value of the granularity with which a given Awareness Module sends context information to the Context Manager could be a highest common divisor of the granularities of context notifications requested by the clients.

The proposed architecture, which is very modular and notification based, allows distribution of context managers. Wide-area coverage can be achieved by deploying a number of context managers which gather context information from geographically close awareness modules. On the other hand, if an entity for which context is gathered and evaluated moves a large distance from its home context management system, provision can be made for the creation of a visitor context profile in the context manager at the new destination. In addition, for

resource poor and disconnection prone mobile devices, it is possible to retrieve, from the context manager, and locally cache a needed subset of context information. This context information will be available during disconnections and will be updated immediately after the mobile device reconnects. The granularity of context updates may be selected to suit the level of resources in a mobile device.

6 Lessons learned

We have built a scalable, CC/PP based, context management system which includes basic context types needed to support infrastructures of pervasive systems, provides persistent repository of context information and provides notification about context changes to clients interested in context information.

There are however some limitations in CC/PP which make this model not very suitable as a context model for future pervasive systems. We introduced a variety of context types but future, full fledged pervasive systems will require much more sophisticated context models in order to support seemless adaptation to changes in the computational environment. The context models will need to specify a range of characteristics of context information, including temporal characteristics (freshness and histories), accuracy, resolution (granularity), confidence in correctness of context information, as well as a variety of information types (including various types of dependencies) [7]. However, we have already shown in Section 4 that it was difficult and unintuitive to capture complex relationships and constraints in CC/PP and that the Component-Attribute model becomes unwieldy if there are many layers of attributes.

In addition, there are other limitations which make specification of context information and automatic interpretation of a CC/PP specification difficult:

- CC/PP defines no constraints pertaining to the method of updating an attribute-value in a profile (Should the new value be appended to the old value? Should the new value override the old value? Or no update is allowed?) UAProf embeds these resolution constraints in a comments section of the schema.
- CC/PP defines no relational constraints pertaining to the existence or absence of several attributes within a profile component. For example, we may wish to enforce the constraint that if there is an attribute called "screensize" in a device's hardware profile, then there must also be an attribute called "colour" or "renderingCapability".
- RDF still has some missing functionality like being able to constrain the elements in a container (Bag, Sequence etc) to a specific type. Cardinality constraints on containers are also missing.

We see our project on a CC/PP based context management system as a valuable experience which produced a good prototype context management system which is used in our prototype pervasive system infrastructure to support a variety of adaptations to context changes. We are concurrently working on richer context models required for pervasive systems as described in [7].

7 Related Work

Current research on context-awareness is mostly concerned with providing frameworks that support the processing of context information from sensors and high-level models of context information which can support context-aware applications.

The context toolkit [1] and the sensor architecture of Schmidt et al. [2] address the acquisition of context data from sensors, and the processing of this raw data to obtain high-level context information. The first one is a programming toolkit that provides the developers of context-aware applications with abstract components (e.g. interpreters and aggregators) that can be connected together to gather and process context information from sensors. The second provides a layered model of context processing in which sensor output is transformed to form an abstract context description.

The research carried out by Schilit et al. [18] focuses on modelling context information and delivering it to applications. It proposes the use of dynamic environment servers to manage and disseminate context information for an environment (where an environment might represent a person, place or community). The model of context used is very simple, with context information being modelled as a set of environment variables. The Cooltown project [3] proposes a Web-based model of context in which each entity (person, place or thing) has a corresponding description that can be retrieved via a URL. Entity descriptions may be unstructured and are intended for human (rather than application) interpretation. The context modelling approach proposed by the Sentient Computing project [4] is based on an object-modelling paradigm. A conceptual model of context is constructed using a language based on the Entity-Relationship model, and context information is managed by a relational database.

The Owl context service which is in the early stages of development aims to gather, maintain and supply context information to clients [6]. It aims to address access rights, historical context, quality, extensibility and scalability.

The model under development by Henricksen at el. [7] aims to define a variety of types of context descriptions needed in pervasive systems, relationships and dependencies between context descriptions, and quality of context description (for example, freshness, accuracy, confidence, and so on). In this paper, the first two issues (context types and relationships) are used to examine the applicability of the CC/PP standard to context descriptions in pervasive systems. The quality of context description was not addressed as even the two addressed issues posed some difficulty in the CC/PP vocabulary extension.

8 Conclusions

In this paper, we described our experience in using CC/PP to define several types of context information needed in pervasive systems and building a scalable context management system. The extended CC/PP vocabulary includes, among others, network interfaces of devices and Quality of Service provided by these interfaces, location, disconnection status of devices, application Quality of Service

requirements, user preferences, and a variety of relationships and dependencies between context attibutes which are needed to provide various adaptations for applications running in a changing context. Context information is managed by the described context management system which provides persistent storage of context information, context information retrieval and notification about context changes delivered to interested users of the context management system. The users may be context-aware applications and other parts of the pervasive computing infrastucture such as resource discovery protocols or adaptation decision engines.

Our experience with CC/PP revealed that although it is possible to use CC/PP to define the required context information including relationships and dependencies, the limitations of CC/PP shown in this paper do not make CC/PP very suitable for future complex context models required for pervasive systems. The goal of this exercise was to show the gap between the currently proposed standard for context description and the needs of context models for pervasive systems. Many research teams address the issue of context modelling but their research will need to be followed by standardisation of context models to allow some cooperation between heterogeneous domains in pervasive systems.

References

1. Dey, A., Salber, D., Abowd, G.: A context-based infrastructure for smart environments. In: 1st International Workshop on Managing Interactions in Smart Environments (MANSE'99). (1999)
2. Schmidt, A., et al.: Advanced interaction in context. In: 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99), Karlsruhe (1999)
3. Kindberg, T., et al.: People, places, things: Web presence for the real world. Technical Report HPL-2000-16, Hewlett-Packard Labs (2000)
4. Harter, A., Hopper, A., Steggles, P., Ward, A., Webster, P.: The anatomy of a context-aware application. In: Mobile Computing and Networking. (1999) 59–68
5. Gray, P., Salber, D.: Modelling and using sensed context in the design of interactive applications. In: 8th IFIP Conference on Engineering for Human-Computer Interaction, Toronto (2001)
6. Ebling, M., Hunt, G.D.H., Lei, H.: Issues for context services for pervasive computing. In: Middleware 2001 Workshop on Middleware for Mobile Computing, Heidelberg (2001)
7. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Proceedings of The First International Conference on Pervasive Computing, Pervasive 2002. Volume 2414 of Lecture Notes in Computer Science., Zurich, Switzerland, Springer (2002) 169–180
8. Reynolds, F., Woodrow, C., Ohto, H., Klyne, G.: Composite Capability/Preference Profiles (CC/PP): Structure and vocabularies. W3C Working Draft (2001)
9. Saryanarayana, L., Hjelm, J.: Cc/pp for context negotiation and contextualisation. In: Second International Conference on Mobile Data Management, Hong Kong, Springer (2001) 239–245
10. Brickley, D., (eds), R.G.: Resource description framework (RDF) schema specification 1.0. W3C Candidate Recommendation (2000)

11. Lassila, O., (eds), R.R.S.: Resource description framework (RDF) model and syntax specification. W3C Recommendation (2000)
12. Bond, A., Gallagher, M., Indulska, J.: An information model for nomadic environments. In: Proceedings of the Ninth International Workshop on Database and Expert Systems Applications, Vienna, IEEE (1998) 400–405
13. Efstratiou, C., Cheverst, K., Davies, N., Friday, A.: An architecture for the effective support of adaptive context-aware applications. In: Mobile Data Management - MDM, Hong Kong, China, Springer (2001) 15–26
14. Sousa, J., Garlan, D.: Aura: An architectural framework for user mobility in ubiquitous computing environments. In: Proceedings of 3rd IEEE/IFIP Conference on Software Architecture, Montreal (2002)
15. WAPForum: User agent profile specification. WAP Specification Information Note (2000)
16. Segall, B., Arnold, D.: Elvin has Left the Building: a Publish/subscribe Notification Service with Quenching . In: Proceedings of AUUG97, Brisbane, Australia (1997) Available at <http://www.dstc.edu.au/Elvin/doc/papers/auug97/AUUG97.html>.
17. Melnik, S.: The RDF API Homepage. http://www-db.stanford.edu/ melnik/rdf/db.html, last accessed June 2002 (2001)
18. Schilit, B., Theimer, M., Welch, B.: Customising mobile applications. In: USENIX Symposium on Mobile and Location-Independent Computing. (1993)