Towards Requirements Analysis and Assessment of Pervasive Systems

Sajjad Mahmood, Ahmad M. Al-Barrak, and Muhammed S. Al-Mulhem

***Abstract*—Context-Aware Pervasive Systems (CAPS) are dynamic and heterogeneous software applications that focus on the collaborative use of computing devices available in the physical environment of users. In this paper, we present a CAPS specification model that is used to specify context features and their dynamic interactions in the physical environment of users. CAPS specification model presents the notation of 'context chunk' to specify system-to-be features and associated rules for dynamic configuration under changing environments.**

***Index Terms*—Context, Context-Aware Systems, Requirements Analysis**

I. Introduction

In recent years, context-aware computing [8] has become a reality as technology improves to make everyday appliances and devices context-aware. These developments have led software engineers to the idea of creating a software system, called a pervasive system, which supports the collective use of devices available in the environment of users. Pervasive systems need to be aware of the available resources (i.e. aware of their context) to detect changes in the environment (context changes); and adapt their functionality and behavior. However, because of the anytime/anywhere/any-media nature [8], pervasive system development activates are somewhat different. Hence, the CAPS specification requires consideration of both context information and different types of computing devices in an operating environment.

A context-aware pervasive system can be viewed as having three basic functionalities [8]: *sensing, thinking and acting*. System can vary in sophistication in each of these functionalities. Pervasive software development becomes more complex because the software must function in a setting that is increasingly open and dynamic. Applications will require behavior that is highly adaptive and very much dependents on the availability of various resources which may also be transient in nature.

The success of a CAPS development depends largely on specifying system-to-be features with reference to context changes and their impact on the interacting devices in an environment. Significant work has been done to address the challenges associated with context-aware pervasive system requirements analysis and assessment. For example, Bresciani et al. [1] have presented an agent-oriented methodology, namely Tropos, that allows modeling interactions between software and human agents; and the operating environment of the system.

In this paper, we present a CAPS specification model, which helps a system analyst to specify individual features and their rules of interactions with other devices in an environment. The CAPS specification model supports specifying context-aware system features and individual interaction rules; and their dynamic interactions in the physical environment of users. CAPS specification model represents both structural and operational relationships between context-aware system features and classifies features at four abstraction levels, namely, user layer, device layer, environment layer and implementation layer.

Sajjad Mahmood, Ahmad M. Al-Barrak and Muhammed S. Al-Mulhem are with the Information and Computer Science Department, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia. (e-mail: smahmood@kfupm.edu.sa).

II. Related Work

In this paper, we adopt Dey [3] context definition which is deemed suitable for pervasive computing by other researchers (e.g. [8]): *'Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves'*.

Finkelstein et al. [5] propose a framework for developing ubiquitous web applications which consists of nine attributes, namely, goal, service, environment, context, requirements, application, rule and a meta level representation of the application. Furthermore, Eila et al.[10] classifies requirements of ubiquitous computing systems into three categories: system, software and business and organization. Dan et al. [6] have proposed to use context as a basic of requirements elicitation for ubiquitous applications. They classify context into three categories, namely, computing context, user context and physical context; and use these categories to determine the appropriate interaction between the user and the application. Munoz et al. [9] presents a method to develop context-aware pervasive systems based on model driven architecture approach and proposes a set of models to specify contextual information at conceptual level and a strategy to automatically generate code from these models.

Recently, feature oriented modeling techniques [7] from software product lines domain has been applied to specify context-aware software systems. For example, Naoyasu [11] uses the feature oriented modeling and VDM based formal design with the notion of aspects to reduce design complexity of context-aware systems. Paula et al. [4] have also presented a modeling notion, called UbiFEX, for representing context information and defining context adaptive rules in a feature models. Carlos et al. [2] have used feature modeling techniques to to represent evolution of a pervasive system and transform theses models to an executable reconfiguration plan. However, these approaches lacks clear guidelines for handling interactions between contextual information which is important to further enhance the adoption of pervasive systems in the industry.

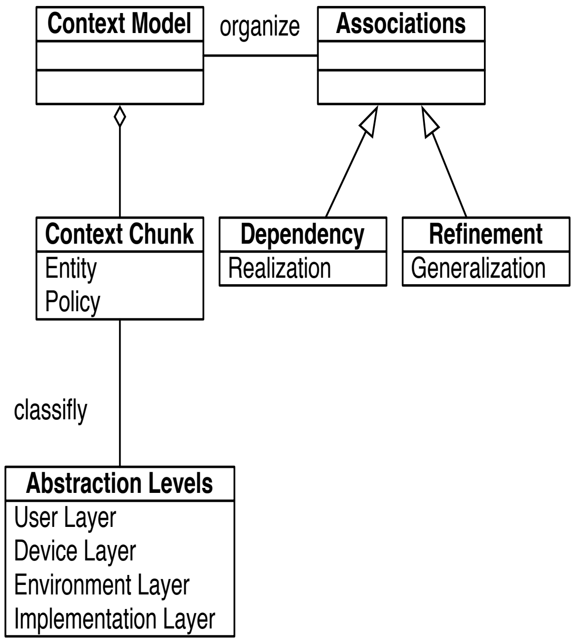

Fig. 1. CAPS Specification Meta Model

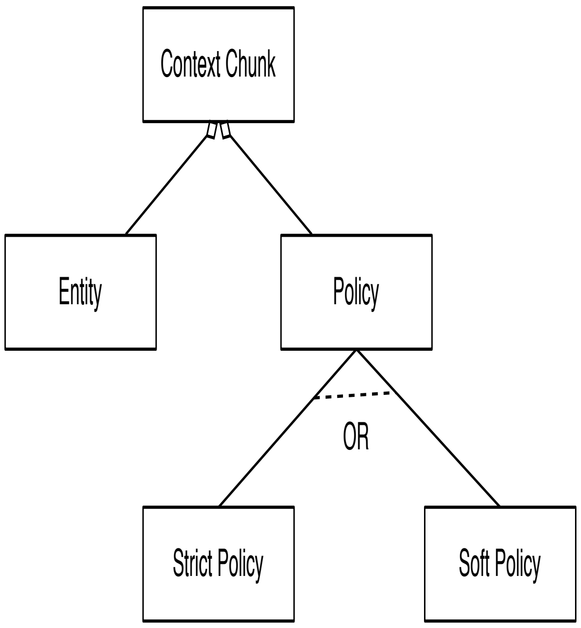

Fig. 2. Context Chunk Model

III. CAPS Specification Model

CAPS development process needs a specification technique with a focus to elicit and assess stakeholder needs and facilitate individual interests for a highly adaptive envirnoment. CAPS specification not only consists of individual system-to-be features but also depends on the available computing devices and the underlying services available in an environment. A CAPS specification model needs to understand and specify system-to-be features; and individual rules of dynamic configuration in a given environment. Figure 1 shows the meta-model of the CAPS specification model.

CAPS specification model provides a platform to analyze stakeholder requirements of dynamic contexts and their possible interactions with a range of devices in an environment. The method focus on the identification and elaboration of stakeholder requirements to specify user and computing devices information; and captures dynamic configurations of a pervasive system. CAPS specification model is based on the concept of software features [7] that are user-visible aspects of characteristics of a software system. CAPS specification model presents the notion of a 'Context Chunk' and is defined as a pair <E, P> where E is an entity and P is a policy. CAPS specification model represents the structural relationship between context chunks through generalization relationship. Furthermore, CAPS specification model also supports dependency relationships to supplement the structural relationship between context chunks by specifying implementation dependency between context chunks at different classification levels.

A. Context Chunk

A context chunk is defined as a pair of entities and associated policy, as shown in Figure 2; and acts as a unit of analysis in the CAPS specification model. An entity is an active system component which plays a specific role in satisfaction of stakeholder requirements. In this paper, we adopt the entity definition [3]: *An entity is a person, place or object that is considered relevant to the interaction between a user and an application, including the user and application themselves*. We define policy as a set of rules for a proposed specific use of an entity and governs the entity's interactions with other entities in a system. We classify a policy as either a strict policy or a soft policy. A strict policy prescribes intended entity behavior where a set of rules must always hold. On the other hand, a soft policy prescribes preferences among alternative behaviors where a set of rules are more fulfilled along some alternatives and less along others.

Context chunks are used to specify pervasive system requirements using entity-policy coupling as it allows to simultaneously specify static information regarding users and resources; and their dynamic configuration under changing situations. Context-chunk provides a systematic process for refining high-level requirements into concrete-level requirements.

B. Context Model

CAPS specification model organizes pervasive system requirements as a directed graph called a context model. Context model is a graphical hierarchy of context chunks that represents both structural and operational relationship between context chunks. Context chunks are identified and classified in terms of user, device, environment and implementation abstraction levels in a context model.

The profiles at user layer are stakeholder needs specified as distinct requirements. Devices are generic computing resources that are used to implement pervasive system services. The environment represents the way of implementing services or operations. Context chunks at the implementation abstraction level are context chunks that represent concrete level requirements

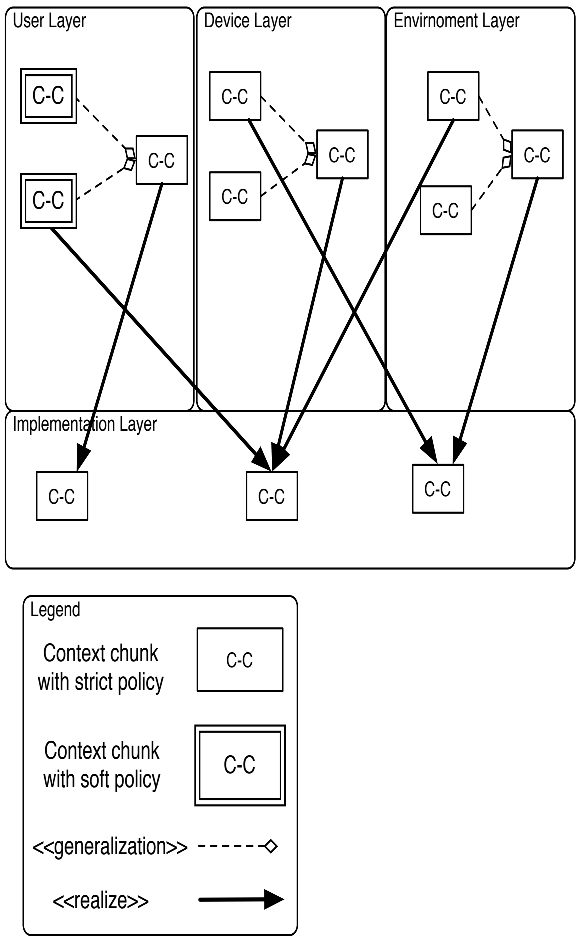

Fig. 3. Context Model

C. Context Chunk Relationships and Dependency

CAPS specification model organizes context chunks using structural generalization relationship and realization dependencies, as shown in Figure 3. The structural generalization relationship leads to the organization of the context chunk as a hierarchy of chunks at the same classification level of a context model. The generalized relationship is used to discover a new context chunk where two similar context chunks at the same classification level can be specialized into more specific ones with the help of additional rules for the context chunk policy. The generalization relationship is modeled with stereotype <<generalization>> and is represented as broken lines with a diamond, as shown in Figure 3.

We believe that in addition to the structural relationship, it is important to consider the operational dependencies among context chunks as it plays a significant implications during the development of a context-aware pervasive system. We define the realization dependency as a relationship between context chunks in different classification levels of a context model such that they collaborate with each other to perform a task in a pervasive system. The realization dependency relationship is modeled with stereotype <<realize>> and is represented as solid lines, as shown in Figure 3.

IV. Conclusion and Future Works

In this paper, we have present CAPS specification model that guides stakeholders through a process of requirements elicitation to analyze individual adaptive behaviors with reference to dynamically changing operating environments. We present the notion of context chunk to elicit stakeholder requirements and its acts as a unit of analysis in the CAPS method. Furthermore, context models and associated structural and operational dependency relationships are used to classify and organize stakeholder requirements for a CAPS.

In future work, we plan to extend CAPS specification method with formal rules for specifying policies associated with each context chunk and guidelines for analyzing realization relationships between context chunks at different classification levels. We also aim to study the impact of context dependent requirements analysis process on the reconfiguration of a pervasive system. Further, the ability to model pervasive system requirements analysis and assessment in a graphical context model also offers the potential for tool support.

Acknowledgement

The authors would like to thank King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia for its continuous support of research.

References

[1] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methdology. *Autonomous Agents and Multi-Agents Systems*, 8:203 – 236, 2004.

[2] Carlos Cetina, Joan Fons, and Vicente Pelechano. Applying software product lines to build autonomic pervasive systems. In *12th International Software Product Line Conference*, pages 117 – 126, 2008.

[3] A.K. Dey. Understanding and using context. *Personal and Ubiquitous Computing Journal*, 5(1):5 – 7, 2001.

[4] Paula Fernandes, Claudia Werner, and Leonardo Murta. Feature modeling for context-aware software product lines. In *20th International conference on software engineering and knowledge engineering*, pages 758 – 763, 2008.

[5] Anthony C.W. Finkelstein, Andrea Savigni, Gerti Kappel, Werner Restschitzegger, Eugen Kimmerstorfer, Wieland Schwinger, Thomas Hofer, Birgit Proll, and Christian Feichtner. Ubiquitous web application development - a framework for understanding. In *6th World Multiconference on Systematics, Cybernetics and Informatics*, 2002.

[6] Dan Hong, Dickson K.W. Chiu, and Vincent Y. Shen. Requirements elicitation for the design of context-aware applications in a ubiquitous envirnoment. In *7th International conference on electronic commerce*, pages 590 – 596. ACM Press, 2005.

[7] Kyo C. Kang, Jaejoon Lee, and Patrick Donohoe. Feature-oriented product line engineering. *IEEE Software*, 19(4):58 – 65, 2002.

[8] Seng Loke. *Context-Aware Pervasive Systems*. Auerbach Publications, 2006.

[9] Javier Munoz and Vicente Pelechano. Applying software factories to pervasive systems: A platform specific framework. In *Eight International Conference on Enterprise Information Systems Databases and Information System Integration*, pages 337 – 342, 2006.

[10] Eila Niemela and Juhani Latvakoski. Survey of requirements and solutions for ubiquitous software. In *3rd International Conference on Mobile and Ubiquitous Multimedia*, pages 71–78. ACM Press, 2004.

[11] Naoyasu Ubayashi and Shin Nakajima. Context-aware feature-oriented modeling with an aspect extension of vdm. In *The 2007 ACM symposium on applied computing*, pages 1269 – 1274, 2007.