Chapter 4
A Software Component Generation Model for Pervasive Computing Environment

Ratneshwer
Banaras Hindu University, India

ABSTRACT

*In order to develop software components that are reusable across the pervasive computing applications it would be required to consider the variations and properties (***mobility***, adaptability, composability, context awareness etc.) that may be required for different pervasive computing applications (application types). It should go without saying that various requirements and variations may not always be known a priori and hence developing all the multiple variants may not always be possible or feasible. It is quite unlikely that all the pervasive computing applications would be able to reuse a component 'as-is' always. One idea is to use lightweight components such that the overheads (those that are not required in a particular pervasive computing application) do not get transported with the body of the component. Based on this idea, a model of "***Generic Component***" with 'Component Generator' has been proposed that will generate components according to the requirements of a specific pervasive computing application. This work starts a discussion and calls for more extensive research oriented studies by professionals and academicians for perfection of the model.*

INTRODUCTION

The world of software development and the contexts in which software is being used are changing in significant ways. One of the emerging trends that promise to have a major impact on software development is that of pervasive computing. Pervasive computing comprises a computing universe populated by a rich variety of heterogeneous computing devices [Garlan & Schmerl, 2001]. Apart from the device heterogeneity, the hardware and software resources, devices and services, available to an application are highly dynamic, due to factors like user mobility, fluctuating network connectivity or

DOI: 10.4018/978-1-61520-741-1.ch004

changing physical context [Becker et. al., 2004]. Developing and executing applications in pervasive computing environments is a non-trivial task. Pervasive computing demands applications that are capable of operating in highly environments and of placing minimal demands on user attention [Henricksen & Indulska, 2004]. Pervasive computing is maturing from its origins as an academic research area to a commercial reality. This transition has not been a smooth one and the term itself, pervasive computing, still means different things to different people [Becker et. al., 2004]. Currently there is no well established approach to design, implement, deploy, execute and manage pervasive computing [Endres & Butz, 2005]. An approach towards pervasive application development preferably should have a modular structure as it is feasible to make small context aware modules such that a module can be easily modified, replaced and enhanced without affecting the other modules. What is required at this stage is that to move from traditional software development to reuse based development. **Component-Based Development** (**CBD**), an approach to develop a software system with help of reusable **software components**, may help to reduce development time and cost and will increase reliability and maintainability of such systems. Component development is considered as a separate activity in **CBD**. In **CBD**, components developed once, can be reused many times in various applications. **Software components** for a pervasive computing environment can be developed and stored in a **repository** and later they can be used in a pervasive computing software development (using component based approach). The idea is to design pervasive computing related **software components** with attributes and functionalities that will always be essential in its any deployment with some scope for addition of, or modification in, functionalities as per requirements of specific pervasive computing applications.

The normal practice in **CBD** is to develop components once and use them 'as-is' in various applications. It should be noted that the component technology has come into being on the line of usage of components in the hardware and consequently promotes *as-is* reuse of non-modifiable units knows as components. Such an assumption makes it necessary to keep multiple variants of a component so as to be able to provide the most suitable components whenever there is a request for such a component. It should go without saying that various requirements of a pervasive application may not always be known a priori and hence developing all the multiple variants may not always be possible or feasible. It is quite unlikely that all the pervasive applications would be able to reuse a component 'as-is' always. In order to make components that are reusable across the pervasive applications it would be required to consider the variations that may be required for different pervasive applications (application types). It may not always be possible to visualize all possible variations during development of a component. Further newer types of characteristics that may not have been considered when the component was developed may emerge at later times. For example, if a component is developed for desktop computer environment and after some time a need arise to implement the same functionality in mobile computing environment. It is the question that how a component that meant for a desktop application can work efficiently with mobile computing environment. Such a situation may demand possibly redesigning (or may be re-engineering) the component for the newer requirements. A component developer has to develop different versions of a component for different pervasive applications (that may have slight variations in their functionalities). In order to make components composable in various pervasive applications, one goes for incorporation of properties that form a superset of requirements of all possible pervasive application types and consequently such a component becomes heavy as, when deployed in an pervasive application, it carries with it attributes and operations that may