ABSTRACT

There is an increasing demand for software that should cope with variability due to the diversity in customer needs, computing and operational environments. The software that can adapt to the changes in customer requirements or/and in environments is referred to as an adaptable software system. The software adaptation is classified as static, which adapts to the changes known during the design time and dynamic, which is the ability of handling changes at run-time. The dynamic adaptation is a challenging task. The existing adaptation techniques are cost and time consuming processes and their features are inadequate to support dynamic adaptation. Hence, the present work has focused towards developing an adaptable system that will possess the features of (i) Adapting run-time changes in the user requirements without reconfiguration (ii) Dynamic discovery of components based on changes in the requirements and (iii) Changing the services or components without affecting the existing structure. The dynamic aspect weaving adaptation approach and adaptable middleware model have been proposed in the present work for developing an adaptable system with the above mentioned features. Also, the metrics for evaluating the adaptability efficiency of a system have been devised using the principles of coupling.

The dynamic aspect weaving adaptation approach includes evolution of dynamic change aspect, dynamic structural introspection model and
dynamic function invocation techniques. The separation of concern principle stated in aspect-oriented approach has been redefined as representation of changes in the requirements in a separate module known as Dynamic Change Aspect, which is used to achieve dynamic adaptability through modularization. Dynamic Structural Introspection technique has been introduced to separate the components that are used for meta-information maintenance from the components that are associated with the changes by applying introspection for defining meta-components at run-time. The technique for establishing the compatibility between the implementation of the dynamic change and the incorporation of the changes in the component functions has been achieved through creating a wrapper for the function associated with the changes using the adapter pattern, which is referred to as ‘Dynamic Function Invocation’.

The adaptable middleware model provides the solution for the adaptability at the architectural level by realizing the techniques stated in the proposed dynamic aspect weaving adaptation approach using the components such as Aspect Adaptation Manager, Change Request Parser, Aspect Generator, Aspect Weaver, Component Function Wrapper and Component Meta Model.

The adaptability metrics for specifying the adaptability efficiency of the system have been proposed using the principles of coupling, which has been considered as the primary property that influences the maintenance task.
of a software system. The sum of the coupling of each method in a class with other classes is referred to as Conceptual Binding between Classes (CBC). This CBC is used to measure the adaptability efficiency of an object-oriented system and the coupling of a method is defined as the total number of the usage and re-usage or sharing of a method by other classes. The adaptability of a software system that is integrated with the proposed adaptable middleware is measured as the number of classes that are associated with the dynamic changes in the requirements, which is referred to as the Conceptual Binding between Aspect and Classes (CBAC). The CBC and CBAC metrics are used to measure the adaptability of a system at design level. In order to measure the adaptability at execution level, Functional Change Adaptation Time (FCAT) metric has been proposed to measure the adaptability of a system that uses compositional approach or the proposed adaptable middleware for adapting to the dynamic changes in the user requirements.

The proposed dynamic adaptability techniques and adaptability metrics have been demonstrated using banking transaction system, which is implemented in Java, AspectJ and Android platform. The results show that the efficiency of adapting the changes by a system that is integrated with the proposed adaptable middleware is high, if more number of functions in a system is associated with the changes; otherwise the adaptability efficiency is low.