Design patterns for Component-based Software Development

A dissertation submitted to the University of Manchester for the degree of Master of Science in the Faculty of Engineering and Physical Sciences

2012

Shilan Hajimohammadikasyan

School of Computer Science

Abstract

Regarding recurring and common problems in the field of object oriented software design for a specific context, the general solution opted for would be design patterns. As already tested and proven to be workable in successful designs, these patterns play a vital role in enhancing the quality of software flexibility, reusability and maintainability by solving the common design problems. From the said, it can be inferred that irrespective to the software implementation reusability, the design patterns can be reused with respect to software design. Hybrid composition connectors which are used in Component Based Software Development are in turn used as composition operators. One of the functions of the said connectors is to encapsulate the control structures through initialization of the control and handling the accompanied result. Due to their hierarchical nature, they have the property of being reusable, generic and compositional. This means that whether they are composite composition connectors or composition connectors, they both behave similar to behavioral design pattern which are design patterns only. The positive point about the said design patterns are they can be stored in a repository for the developer and be applicable and reused whenever required by the developer. The above mentioned patterns as involve high composition connectors and components are considered as strong and powerful composition operators. For any large scale software system, the said patterns reduce the number of connectors' level resulting in improvement of composition processing.

In the dissertation presented, in terms of software component model, my aim would be to implement and introduce certain suitable design patterns as composition operators. Furthermore, in terms of implementation reuse, I would aim to explain how the said patterns would enhance reusability and be stored in a repository to be retrieved for later usage by system developer.