Abstract

Aspect-oriented software development (AOSD) consists of a number of technologies that promise a better level of modularization of concerns that cannot be separated in individual modules by using conventional techniques. Aspect-oriented programming (AOP) is one of these technologies. It allows the modularization at the level of software application code. It provides programmers with means to quantify over specific points in the base application code, called join points, at which the crosscutting concern code must be triggered. The quantification is achieved by special selection constructs called pointcuts, while the triggered code that is responsible for adapting the selected join point is provided by special construct called advice.

The selection and adaptation mechanisms in aspect-oriented programming depend heavily on the distinguishing properties of the join points. These properties can either be derived from the local execution context at the join point or they are considered to be non-local to the join point. Aspect-oriented systems provide a plenty of pointcut constructs that support accessing the local join point properties, while they rarely support the non-local properties.

A large research effort has been achieved to extend current aspect-oriented systems in order to solve the problem of non-locality. However, none of these proposals support the non-local object relationships. There are many situations where a good abstraction over non-local object information is needed, otherwise, the developers will be obliged to provide complex and error-prone workarounds inside advice body that conceptually do not reflect the semantics of join point selection and mix it with the semantics of join point adaptation. Such recurrent situations occur when trying to modularize the object persistence concern.

Object persistence, the process of storing and retrieving objects to and from the datastore, is a classical example of crosscutting concern. Orthogonal object persistence meets the obliviousness property of AOP: The base code should not be prepared upfront for persistence.

This thesis addresses the shortcomings in current aspect-oriented persistence systems. It shows that the reason for such shortcomings is due to the lack of supporting non-local object information by the used aspect-oriented languages.

To overcome this problem, this thesis proposes a new extension to the current pointcut languages called *path expression pointcuts* that operate on object graphs and make relevant object information available to the aspects. As an explicit and complete construct, a formal semantics and type system have provided. Moreover, an implementation of path expression pointcuts is discussed in the thesis along with its usage to show how the aforementioned problems are resolved.