論文の内容の要旨

A Study on a Single Construct for Events, Aspects, and Behaviors

（イベント、アスペクト、ビヘイビアのための単一言語機構に関する研究）

氏名　　荘　永裕

Programming paradigms are so important that a lot of research activities are devoted to the support for them. How to support the implementation of paradigms can be classified into three types of approaches: by design patterns, by dedicated constructs, and by generic constructs. However, none of them are sufficient. Design patterns can be used to implement most paradigms but not all. Furthermore, without language support the code tends to scatter and tangle. Dedicated constructs greatly improve the modularity of code, but also increase the number of constructs in a language supporting multiple paradigms; a large number of constructs complicates the language design. Generic constructs can be considered as a potentially good approach, but the number of supported paradigms in current research is quite limited; existing generic constructs are not flexible enough to support more paradigms.

To overcome the problem that existing generic constructs are not flexible enough, this thesis proposes a new generic construct, *method slots*, based on our observation of the common ground among the implementations of three important paradigms in the real world: OOP, the event-handler paradigm (event-driven programming), the aspect paradigm (aspect-oriented programming). The common ground has never been noticed before this thesis since the dedicated constructs for these paradigms were individually developed from the beginning. The observation on the similarities motivates us to extend the methods in JavaScript to *method slots*, which can be

used as methods, events, and advices. To demonstrate how *method slots* can be used in practice, a Java-based language named *DominoJ* is proposed with a compiler implementation. We then evaluate *DominoJ* by comparing with existing languages, running benchmarks for it, and rewriting programs as case studies.

The concept of *method slots* is very simple and easy to extend. To support this argument, we demonstrate how to extend *method slots* by taking the example of the reactive paradigm (functional-reactive programming). We first compare the reactive paradigm with the most similar one in the paradigms supported by *method slots*: the event-handler paradigm. We find that the major difference between them is whether the event composition is automatic or not. Then we discuss the definitions for event composition in existing event mechanisms, and get the conclusion that existing event mechanisms lack an *inference-based definition* to automatically select events for a higher-level event. This thesis proposes such an *inference-based definition* by adding only one more operator for *method slots*. How the operator can be used for the reactive paradigm is presented with a feasible implementation and discussed in detail to clarify the limitations.