Formalization and Evolution of Collaboration-Based Object-Oriented Methodology

Nguyen Truong Thang (010203)

School of Information Science,
Japan Advanced Institute of Science and Technology

August 15, 2002

Keywords: evolutionary domain, mixin term, mixin layer, role-based design, static structure, dynamic behavior.

Collaboration-based design, also known as role-based design, is a method in which a system is partitioned into layers according to collaborations. The thesis claims that collaboration-based design is a special case of *aspect-oriented software development* (AOSD). As a result, such a design carries all useful features in AOSD tipped as the future software development paradigm for its highly maintainable and evolvable software development. In particular, this thesis concentrates on the evolution of role-based softwares in terms of roles and collaborations.

The evolution of collaboration-based systems is examined with respect to changes in system specification. Such changes are observed from the perspective of *evolutionary domain*. Evolutionary domain is a general framework of software evolution which defines the evolution relationship in an entity set and operators applicable on that set. In essence, this framework consists of three important components. Two evolutionary domains are assumed on the specification and program sets respectively. Between these domains, a mapping called *evolutionary development process* is used to correspond a specific specification fragment with its implementation in program domain. The thesis is about a clarification of this general framework in a specific case, namely role-based design evolution.

This research initially attempts to formalize the static structure specification of a collaboration-based system in terms of classes, collaborations, role mapping and collaboration dependency. Each role in this specification is corresponded by a *mixin term* which is treated equally as a regular object-oriented (OO) class. A tuple of mixin terms composes a collaboration which is later mapped to a *mixin layer* in implementation phase. In addition, a preliminary model of dynamic behavior is formalized by utilizing *Decomposed Petri Nets* (DPN). This mechanism is used due to its capability in handling concurrency and synchronization between role execution threads. The evolutionary specification domain is then formed from the combination of static structure and dynamic behavior specifications.

This thesis does not go into detail of forming the evolutionary program domain for some justified reasons. Intuitively, this domain is assumed to exist. Furthermore, its structure is very similar to that of specification domain. The rationale behind these arguments is that the more evolved a specification with respect to specification domain is, the more evolved its associated program should be with respect to program domain.

Rather than going into detail of formal evolutionary program domain, the thesis shows some development processes transforming specification to program written in two representative programming languages, namely C++ and Java, in a systematic way based on mixin layer. Those language selection is because their respective layer composition mechanisms are at two extremes. C++ directly supports mixin layer, while Java does not. As a consequence, Java relies on an AOSD technique to compose layers together. This software development framework is quite concise in showing how to map a collaboration-based design into C++ and Java codes, although it is not described in a formal way.

Finally, by utilizing the proposed evolutionary development process, programs implementing collaborative designs are examined for their strength and weakness. Even though there are still some programming issues within layer context, this approach clearly outplays traditional OO methods in terms of constructing evolvable, maintainable and adaptable software components. It is worthwhile to investigate further the formal AOSD model, especially a complete formal model for role-based designs.