EXTENDING OBJECT ORIENTED PROGRAMMING WITH ROLE SUPPORT

SUMMARY

Real world systems can be classified either as static or dynamic. In static systems, entities to be modeled can be divided into distinct classes and they do not change their classes. Such systems can be modeled efficiently by using the object oriented programming (OOP) paradigm as the relationship between an object and its class is persistent, static and exclusive in OOP. On the contrary, dynamic systems contain entities which constantly change and evolve. Modeling such entities with the class structure of OOP is hard because objects of dynamic nature are required to be modeled by classes having static nature. Although OOP has features to add some scale of dynamism to the static nature of its classes, it cannot fully eliminate the obstacles it faces while modeling dynamic systems.

When modeling dynamically evolving entities, using instance level specialization will eliminate the obstacles faced by the class level specialization. In that case, a real world entity is represented by multiple role objects, each representing one of the responsibilities of the real world entity. A role of an entity can be defined as the set of properties which are important for an object to be able to behave in a certain way expected by a set of other objects. The programming paradigm which is based on using roles is called role based programming (RBP), while a role model is a software which specifies a style of designing and implementing roles. Role models provide a mechanism for object level inheritance while preserving the fact that multiple objects are used to model a real world entity.

Role models receive much attention with their usefulness among the proposed ways of modeling dynamic systems as they match with the OOP paradigm well and represent a direct way for solving the problem at hand. Role models which are based on object level specialization naturally extends the OOP which is based on class level specialization. Therefore, the Java programming language is extended with role support in order to solve the problem of modeling dynamic systems better than the ways available with the pure OOP paradigm, the problem which is targeted by this dissertation. For that purpose, a role model named JAWIRO (Java with Roles) is implemented.

Different researchers define the features expected from the roles in slightly different ways. According to the results of the research and evaluations done during this thesis work, the features of roles which are expected to be provided by a role model are divided into two groups as the basic and extended features in this dissertation. The basic features of roles can be listed as below:

- Viewpoint: Each role played by a real world entity provides a viewpoint to that entity. The access scope of an object is restricted by the currently played role. However, access to members of a different role can be provided via role switching.
- Role hierarchies: A role should be able to play another role. This will lead to the ability to arrange role objects in various hierarchical relationships.

- Gaining and loosing roles: Roles can be gained and lost independently from each other.
- Representation with multiple objects: The fact that a real world entity is modeled with multiple objects should be preserved. Each role object should be aware of its owner, its subroles and the root of its hierarchy for this purpose.
- Role switching: An entity should be able to switch between its roles whenever required.
- Member access via the owner: A role of an entity should be able to reach the members of the other roles of this entity by using either of the previous two features.
- Conflict prevention: Different roles should be able to contain member fields and methods with the same name.
- Use of the object level and class level inheritance together: Object level inheritance does not replace class level inheritance. On the contrary, object level inheritance completes class level inheritance. As these two types of inheritance relationships can be found together in the real world, they should be used together in role models as well.
- Run time role checking: Entities can be queried whether they are playing a particular role type or not.
- Aggregate roles: An entity should be able to play multiple instances of the same role type. Such roles are called aggregate roles and they are distinguished from each other by an identifier.

The advanced features of roles are listed below as well. The last six of these features represent the unique contributions of this dissertation:

- Preventing abnormal role relationships: It's usually inevitable that the coders and users of a piece of software are different people. This will increase the probability of users to commence operations which contrast the nature of the modeled system. The necessity to prevent abnormal role relationships stems from this fact.
- Persistency: A persistent role model lets users to save an entire role hierarchy to disk for later use.
- Role transfer: A role can be transferred to another owner without dropping its subroles.
- Object level multiple inheritance: A role type can be played by instances of different types.
- Member access by name: A desired member field or method of a participant of a role hierarchy can be accessed from any participant of that role hierarchy. This is achieved by solely mentioning the name of the desired member and without any direct reference to the owner of that desired member. In case of multiple members with the same name, the member of the most evolved participant is returned.
- Dominant roles: The previous rule can be overridden by determining some participants of the role hierarchy as dominant.
- Suspending and resuming roles: Roles can be suspended when they become temporary unnecessary and resumed when they are needed again.

- Support for delegation and consultation mechanisms: These two mechanisms are distinguished from each other by whether the original recipient of a message is preserved or not. As both mechanisms exist in the real world together, they should coexist in the role models as well.

Role models are not the sole solution for efficient modeling of dynamic systems. Other ways for overcoming the difficulties faced by OOP while modeling dynamic systems are proposed as well. These alternatives are examined in this dissertation, too. However, the comparisons made in this dissertation show that role models are more valuable than the other proposals for modeling dynamic systems. It is also shown that role models increase the internal quality attributes of software and reduce the costs of producing software. Meanwhile, the role model JAWIRO implemented as a part of this thesis work is surfaced as an important tool for modeling dynamic systems, thanks to its unique contributions to features of roles and its neglible overhead on runtime performance.