Programming Environment for the Evolutionary Prototyping Technique with Abstract Interpretation

Shingo Ban (110106)

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

February 14, 2003

Keywords: Abstract Interpretation, Java, Programming Environment, Prototyping, Stepwise Refinement.

1 Background

In recent years, we need high technology to develop large and complex software, which requires to control complex hardware systems and many computers within a large network. For example, a mobile phone system, a network application for Internet and so on.

Stepwise refinement is one of an efficient technique for developing large and complex software. In the refinement, we initially develop a simple and critical part of software, then we refine the software on demand repeatedly until the software is complete. The problem of the refinement is that we cannot execute the program under development. Hence, we cannot detect bugs in the early stage of the development.

The evolutionary prototyping technique solves this problem by programming using abstracted data. In this technique, we develop a primitive system with abstracted data first, then decide the system details as reifing them. The technique allows us to execute the system as a whole with abstract interpretation, even though some parts of the system are abstracted or partially implemented. Hence, we can detect bugs of the system in the early stage of the development.

However, we cannot evaluate the effectivity of the technique since the execution and development environments, to which we can apply the technique, does not still exist. Moreover it is not still clear how to realize the environments, because the technique is new.

Applying abstract interpretation to software developments is not new. ISDR(Incremental Software development method based on Data Reification) introduced by Yoshioka deals with function refinement in ML. The drawback of this work is that they cannot cope with side-effects, flow-controls and so on. In constract, our technique is based on Java which is Object-Oriented Language with side-effects.

2 Objective

In this research, we propose a support environment for the evolutionary prototyping technique. We also evaluate the technique by experimental developments on the environment. Our environment, which we build, consists of two parts as follows;

- Interpreter for abstract interpretation in Java
- GUI applications, which visualize evolution relations

First, we develop the interpreter for abstract interpretation. The evolutionary prototyping technique is characterized by allowing us to execute a whole system including objects in several evolution levels. In order to realize the execution, we clarify the structure of the interpreter and algorithms for execution with abstract interpretation.

Secondly, we develop useful GUI applications on software development. In prototyping, the documents are not often written. But, in our technique, we need many programs, since software is made in stepwise development. Since their evolution relations are complex, we cannot understand correctly the relations without documents, which are written about the relations.

We propose two GUI applications as follows:

- Evolution Relation Editor.

- Visualizer for execution with abstract interpretation.

The evolution relation editor express static evolution relations of programs and edit the relations. The visualizer express behavior of the programs with abstract interpretation.

3 Approach

In execution with abstract interpretation, in order to realize method invocation between objects in different evolution levels, it is essential that an object can refer to some objects in different evolution levels. There is a possibility that each of the objects has different type. But Java is a strongly typed object-oriented language. Therefore, standard Java programs can not meet above requirement.

To solve this problem, we propose Proxy Object for abstract interpretation. The proxy object combines objects in different evolution levels. The proxy object is realized by the Java Reflection API and XML. The Reflection allows an executing Java program to examine itself, and manipulate internal properties of the program. The proxy object can cope with many objects in different evolution levels by this technology. We express evolution relations as XML Document. XML is suited to express the relations, since they construct tree structures.

We built the GUI applications with Swing. Swing is a graphical user interface(GUI) component kit for Java. We visualize static evolution relations of programs with JTree, which is a tree structure component of Swing. We realize the visualizer for execution with abstract interpretation by using sequence diagram, which is one of UML diagrams. The diagram shows object interaction, which focus on temporal relationship. The diagram makes easy for us to understand when an abstraction occurs in execution.

4 Experiment

We conducted two experiments on our environment. The experimental applications are BLACKJACK game and Inventory Management System.

In the former experiment, first, we abstract trump cards to a value and develop the most abstracted program with the value. Secondly, we develop concrete programs stepwisely, as we gradually reify the value to real trump cards. In this experiment, we showed that execution with abstract interpretation is effective with two examples, if abstracted data is well-designed. As one example, execution with abstract interpretation give more correct result and it costs lower than execution using stub. As another example, testing with abstract interpretation can detect bugs in concrete programs.

In the latter experiment, we show that it is impossible that we relevant abstracted data. As an example, we cannot design abstracted data of amount of inventory well in the development of Inventory Management System.

We may not define calculations on abstract data domains, which is not well-defined. Hence, it may be impossible to execute the system with the domain. To solve this problem, we propose the mechanism, which developer decide the calculation in runtime. This mechanism make the system executable.

5 Conclusion

In this research, we built a support environment for the evolutionary prototyping technique. The environment consists of the interpreter for abstract interpretation and GUI applications, which are the evolution relation editor and the visualizer for execution with abstract interpretation. We conducted two experimental developments on the environment. And we improved the environment to define calculations on abstract data domains in runtime.

In conclusion, we obtain following results:

- Execution with abstract interpretation is effective, if we design abstracted data well.
- Execution with abstract interpretation is possible, even if we cannot design abstracted data well because of properties of target applications.