An application of design patterns to specific domain

Hiromitsu Shimizu

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

Februaly 14, 1997

Keywords: object-oriented design, reuse, design patterns, frameworks.

The advantage of object-oriented design, which makes abstraction to real world using objects and designs software, is reusability. Though there is reusable, designing object-oriented software is hard, because you must find pertinent objects, factor them into classes at the right granularity, define class interfaces and inheritance hierarchies, and establish key relationships among them.

Design patterns are reusable and systematically explains reccurring design in object-oriented design. Each design pattern systematically names, explains and evaluates an important and recurring design in object-oriented systems. Design patterns make it easier to reuse successful designs and architectures. Design patterns can even improve the documentation and maintenance of existing systems by furnishing an explicit specification of class and object interactions and their underlying intent. It is a partially proved that each of design patterns for general purpose is useful. But, design patterns are more abstract, and how those apply specific domain isn't studied enough. So, this paper show a case study for building framework, which is a set of cooperating classes that make up a reusable design for a specific class of software are discussed. Applying design patterns, and guess problems with reusable designing. A framework is a set of cooperating classes that make up a reusable design for a specific class of software, and most reusable in object-oriented component.

We apply the approach to designing programming language processor. Programming language processor established in its design and apply various area. A target systems consist of the sub-systems including scannar (lexical analyser), parser, evaluator and viewer.

We discuss that the essential point in designing framework for programming language processor are as follows: (1) extendability of grammar: Grammar symbol in context-free

grammar are categorized to terminal symbol or nonterminal symbol. Each of them have some property. So grammar must be able to change. (2) flexibility in output: Output of parser are a parse tree and an abstract syntax tree. Node of them varys in language, but a way of building parse tree is fixed. And a way of building abstract syntax tree maybe changed that is not changing a kind of node handled in language. Therefore framework must be able to easily change a way of building abstract syntax tree. (3) Use semantic rule: To describe attribute grammar in used of static semantics which is difficult that prove in context-free grammar. Generally, attribute grammar may have a various type. So framework must be able to handle flexible type of attributes.

The result of analysing require for programming language processor, about structure of framework, design patterns is usable as follows: (1) Using Interpreter pattern, framework declared a property of each expression of terminal symbol and non-terminal symbol, and grammar symbol expression is inheriting them. So this pattern allows to change grammar, can output a parse tree after parsing and provide a way of semantic analysis. (2) Builder pattern separates abstract syntax tree builder from grammar symbol expression, so framework allows to change to a way of building abstract syntax tree. And making abstraction of a class required in semantic analysis, framework allows to change to type of attribute. (3) Iterator pattern take the responsibility for access and traversal out of an aggregate object which is list expressed attribute or parse tree or abstract syntax tree, make traverse without exposing its internal structure, and allows to change to a way of traversal.

After abstract classes structured framework proposed in this paper inherits and implements application of programming language processor, for testing framework that it's design is reasonable, we give an example of simple programming language processor, and application which applied framework designed and implemented. The result shows that framework make it easy to implement it. Especially, we show that the parser can be built by the grammar automatically. Therefore, we show that framework captures almost of the design decisions that required in building applications of programming language processor. If we use design patterns and building systems with them, design patterns which can solve the design problem in target systems. We must evaluate the result of using that pattern, whether it is applicable under condition of it. So in applying design patterns to specific domain, precise documentation is useful to build similar systems. And therefore, application of design patterns to problems which it gives no direct supports for is future work.