Development of numerical library software in Java

H. Okazawa[1] *and T. Sasaki*[2]

[1] International and cultural studies, Shizuoka Seika College, Japan
[2] KEK, High Energy Accelerator Research Organization, Japan

Abstract

In the analysis of HEP experiment, the software components tend to be large and long-lived. So it is efficient to adopt OO technique in the development of software components in HEP analysis. It becomes easier to integrate or understanding the software system if OO technology is adopted. Java, a simple OO language, began to affect scientific computing world and a lot of numerical software are developing. But those are not sufficiently analyzed or designed in a viewpoint of OO technology. In this paper, the development of numerical libraries in Java using OOA/OOD is reported.

Keywords: Java, OOA/OOD

1 Introduction

In the analysis of HEP experiment, the software components tend to be large and it will be used for a long time more than 10 years or so. So it is efficient to adopt Object Oriented (OO) technique[1] in the development of software components in HEP analysis. Because it is very easy to,

- Reuse software component,
- Change the specification of the software,
- and Understand an overall of the system.

Now a lot of numerical analysis software in Java is developing and many of those source codes are opened to the public. But those are not sufficiently analyzed or designed in a viewpoint of OO technology. And many people don't think that we need to adopt OO technology for development of numerical analysis. However, suppose different people develop different kind of numerical software with their own manner, users always need to consult with each different manuals to use them. Also when one wants to try another algorithm or implementation for a function, modification on their source code are not avoidable. This situation is very inconvenient. From this point of view, common interface for numerical libraries are demanded. So we need to adopt OO technology for the merit as mentioned above. Well designed software requires minimum efforts on changing user's code when one changes the function to the other.

In this report, our attempt on OO Analysis and OO Development (OOA/OOD) to the general numerical library is reported. As an example of implementation of a functionality, the development of data fitting program with OOA/OOD has done. And as a first step of development of numerical analysis libraries in Java, we present the detail of the design.

2 Object Oriented Programming

Development of software in OO technique takes on round-trip engineering, which consists of 3 phases, analysis, design and implementation phase[2]. It increases reliability and production ca-

pability of software by iterating these processes. C++ OO language illustrates high-level performance and is useful because it has various run-time libraries, which inherited libraries, used in C language[3]. Although C++ is very useful language, it takes long time in understanding or developing software for physicists, no expert of C++ language programming. On the other hand, Java[4], a simple OO language, is

- Easily moved among computing platforms,
- Very safe and reliable under networking condition,
- Easy to understand and integrate the software,
- And easy to import visualization program to many platforms.

And more detailed comparison is listed in Table I[5]. Thus, it is very effective to develop large software in Java OO language such as HEP analysis program. To organize and visualize the structure and components of software intensive systems, the modeling of the system is very effective. Using models, you capture requirements, identify and specify systems, and visualize logical and physical elements.

3 Analysis and Design of Numerical library

In OOA/OOD process, we consumed a lot of time to identify objects and define classes in the numerical library. In C++ case, functions in a numerical library are implemented as template functions. This makes maintenance and re-usability of functions very efficient. However, in Java, there is no template functionary and everything must be in class. We investigated two scenario on it.

- **First scenario**
  The object is identified as an existence to manage the numerical library. This is a super class and all of other functionary is realized to inherit it. Only interfaces are defined in the super class.
- **Second scenario**
  Objects are identified as variable or arrays to be used for mathematical manipulations. In this scenario, we need to prepare define many classes according to basic data types in Java. The functions are required to be implemented in each classes tautologically.

After considering merit and demerit on these scenarios very carefully, we choose the first scenario. First scenario looks a bit far from naive OO concepts, however, it has advantage in simplicity of implementation and also maintainability.

We analyzed the general requirement on an existence to manage the numerical library. These requirements leads very simple usecase diagram and class diagram as shown in Fig. 1. An usecase diagram describes the relation between system and an object outside a system.

In this OOA/OOD by visual modeling, we adopted Unified Modeling Language (UML)[6]. The UML is a language, which defines, visualizes and composes a system generated by OO technique. The UML provides a uniform vocabulary, graphical and textual and reading well-formed models. To visualize UML, the Rational Rose/Java CASE tool[7] was used in this development.

From the result of OOA/OOD, we have defined the super class for the numerical library.

4 Development of fitting program in OOA/OOD

As an example of usage for the super class which we have defined in previous section, we implemented a class for fitting. It is very important to define a class structure and its name, members of the class, and class connection in OO programming.

Requirements for the fitting program are listed below.

- read data to fit

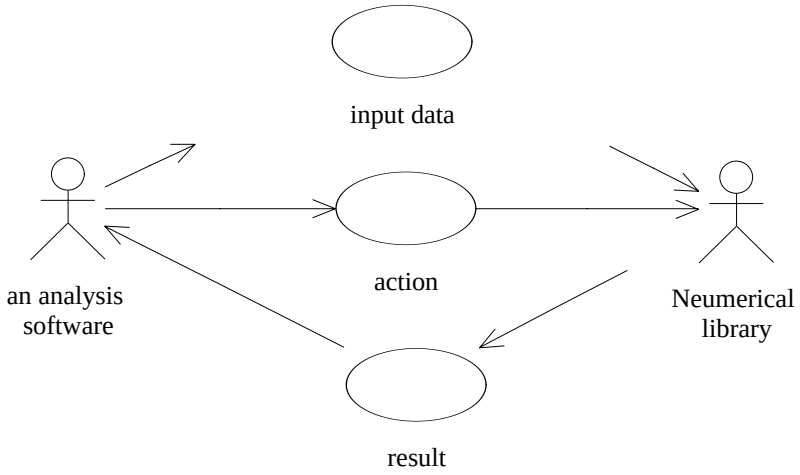

Figure 1: Usecase Diagram for fitting software

- select a function to fit
- calculate fitting parameters, errors
- returns fit result

From the requirements above, a class diagram was designed as shown in Fig. 2.

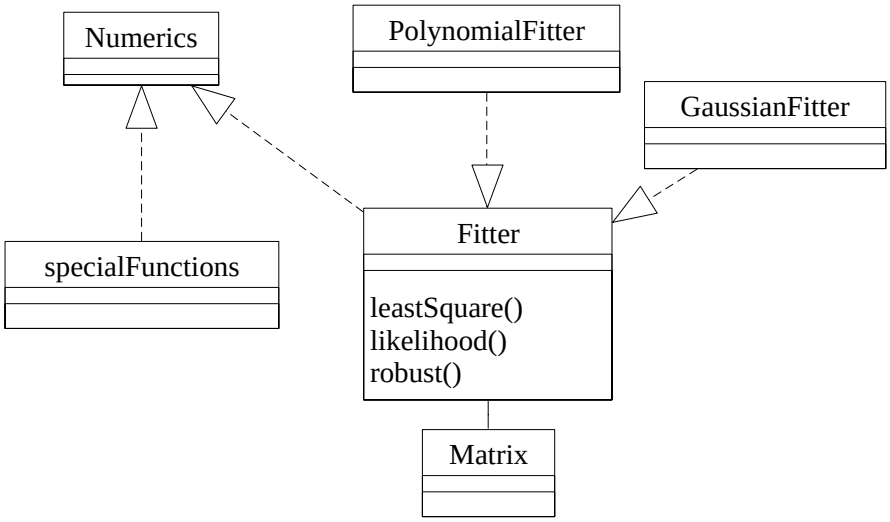

Figure 2: Class Diagram for fitting software

A class diagram describes a class structure. One box in this class diagram corresponds to one class. Each class box contains its name, members and operation. Lines or arrows indicate a relationship between two classes. Rational Rose/Java generates Java source codes based on this class diagram. In this development, the Java fitting software was based on this class diagram. We can notify the system far easier than looking into the Java source code.

Although this class library is very simple now, further iteration of analysis, design and implementation makes more elegant, widely usable software package. A nd we can easily integrate

the system using this class diagram. For example, if we try to integrate this package into more suitable algorism, we don't have to take it into account user program.

5 Summary

We have attempt OOA/OOD on the numerical library. And simple Java fitting software by OOA/OOD was developed on it. By adopting OOA/OOD, we can reuse the package like "Matrix", "Polynomial" and so on. Further iterating of software analysis, design and implementation, we can easily construct large analysis software as required in HEP analysis.

For more complex case, several steps to prepare before calculation might be necessary. However, we did not take into account this. For the next step , we will try to investigate much more numerical functions and do a requirement analysis again. This will be feedbacked to the current OOA/OOD results. And this software is used in the package[8].

Table I: Comparison between Java and C++.A:Excellent, B:Good, C:Available, D:Not supported

	Java	**C++**
Memory management	B	C
Exception handling	B	C
Reference to class	A	B
Inheritance	B	A
Safety	D	C
Template library	D	B
GUI	A	B
Developping speed	A	B
Multi platform	B	C
Performance	C	B

References

1 B. Meyer, "Object-Oriented software construction", ISBN 0-13-629155-4.

2 P. Kruchten, "A Rational Development Process", http://www.rational.com/sitewide/support/whitepapers/index.jtmpl

3 B. Stroustrup, "C++ Programming Language", ISBN 0-201-88954-4.

4 C. S. Horstmann, G. Cornell, "Core JAVA 1.2", ISBN 0-13-081933-6

5 H. Kumagaya, "Comparison between Java vs. C++", Java World, pp.48-58 , December 1999.

6 I. Jacobson, G. Booch, J. Rumbaugh, "The Unified Software Development Process", ISBN 0-201-57169-2.

7 http://www.rational.com/

8 N. Takashimizu, "A Histgraming Package in Java", submitted to Computing in High Energy Physics 2000, Padova, Italy, February, 2000