ATLANTIS PRESS
Advances in Computer Science Research (ACSR), volume 76
7th International Conference on Education, Management, Information and Mechanical Engineering (EMIM 2017)

Research on the Use of Java Annotation Metadata Mechanism in Basic Program

Hui Yuan [1, a] and Lei Zhao [2, b *]

[1]Xianning Vocational Technical College, China

[2]China Light Industry Design & Engineering Co., Ltd, China

[a]zhaoleies@163.com, [b]437209812@qq.com

*The Corresponding author

Keywords: Oriented software; Metadata; Software evolution; Basic program

Abstract. With changes in user requirements and changes in the software operating environment, the software system needs to evolve to accommodate this new need and change. Facing aspects of the underlying software in the structure of the information changes, will lead to unexpected connection points lost. The reason is that the definition of the connection point closely depends on the structure of the underlying program, this tight coupling seriously hinders the evolution of aspect-oriented software. Therefore, it is a research hotspot in the field of software engineering to propose how a software evolution method based on metadata and reflection is how to realize software evolution.

Introduction

The simple definition of metadata is data about the data, which is an important concept of a wide range of applications. Computer and information technology, the earliest clear its concept and importance, and now gradually developed into a complex system. The development of metadata can be divided into four stages:

Static Metadata Management Phase. Before the database management system appeared, the software manager developed for code reuse could be used to manage metadata. But it is used both to process the data and to process the metadata. Software manager development is still very successful and is still widely used, such as the standard C library, C ++ class library and MFC library [1]. Although this way of managing metadata and business data is not very complicated, these libraries are static from beginning to end. So once a software library is developed and released, it is quite difficult to change its structure and interface.

Dynamic Collaborative Metadata Management Phase. In this stage of metadata management, the development of database management system is more mature, so put forward the meta-data for more detailed management of the new research objectives. In order to achieve this goal, the previous static software library has been unable to meet the needs of the user requirements can be very easy to retrieve, add, delete and update the contents of the software library, to achieve the dynamic software library, you need to use the software library itself data. This technique requires the collection of metadata for the code to be combined with the actual library application. The main way is to store the metadata in the database, when the software library to upgrade the time to modify the corresponding place in the database [2]. There are already a lot of tools to use this technology, such as IBM's development of information management tools ReDiscovery.

The Link Metadata Management Phase. This stage marks the maturity of metadata and its management, and metadata becomes a link to a variety of business resources. The first attempt at this stage is to introduce a data dictionary that provides some information about the data stored in a data warehouse, such as format, meaning, relationship, source, and domain. In the late 1980s, IBM began to build integrated data warehousing, and provide a unified data interface, making the need for these data tools can be easily obtained from. In order to understand the composition of data warehousing, we look at the important features of data warehousing. An important feature of data warehousing is the type of metadata it manages. Including metadata such as database metadata, data model metadata, data movement metadata, business rule metadata, application component metadata, data access

Copyright © 2017, the Authors. Published by Atlantis Press.

This is an open access article under the CC BY-NC license (http://creativecommons.org/licenses/by-nc/4.0/).

metadata, and data warehouse [3]. Other features include information models describing warehouse core metadata types, formal description languages, language that supports interoperability with different products, and standard query languages. Microsoft Repository Service is one of many data warehousing products.

Metadata Integration Management Phase. This stage of the task is to manage a variety of metadata integration. Integration is generally divided into two forms, namely, tool integration and data integration. Different tools will produce their own specific metadata, metadata integration is to establish the interrelationship between the tools. Similarly, data formats from different departments may not be the same. In order to help with decision support, these data also need to be integrated. As long as the metadata follows a common data model, the data warehousing technology described in the third stage can support metadata integration. However, different tools come from different companies, with different specifications and different emphases, and therefore do not have a generic model. The Metadata Coalition (MDC) and Object Management Group (OMG) organizations began to develop metadata standards in the 1990s. The MDC developed the Open Information Model (OIM) standard in the late 1990s, and on the other hand, OMG developed the Common Data Warehouse Meta model (CWM) standard in the early 2000s. But now there is no strong tool to fully support the CWM model.

Oriented Aspect of the Program Design

Aspect-Oriented Programming (AOP) method to the software system functions and non-functional requirements platform features and many other different concerns are independent, to achieve a better modular. At the same time, it cross-cutting concerns Weaving to the base program (base program, to achieve the function of the system, does not include the aspect of the program) to achieve the mechanism for the evolution of software provides a new way.

AOP is a technology based on the separation of concerns, successfully solved the problem of code scattering and entanglement in object-oriented programming. The core idea is to realize the separation of concerns [4]. The focus is a particular need or consideration that must be addressed in order to meet the goals of the entire system. A complex software system can be seen as a result of the implementation of many concerns.

(1) Core concerns: The system to achieve the main functions and objectives. In general, the modules that implement these concerns are independent of each other, and they perform the main functions of the system separately, depending on the specific business requirements.

(2) Crosscutting concerns: The focus of cross-cutting concerns with the core concerns, distributed in the core concerns everywhere, that is, all sub-business may be involved in some public behavior.

AOP's idea is to encode these two cross-cutting concerns separately. Crosscutting concerns that some of the requirements that are scattered in the architecture are modularized; the core concerns are achieved by programming the core concerns using the traditional programming language, and then through the aspect weaving of the two categories of concerns Mix together to construct a new system.

The entire AOP development process shown in Fig. 1.

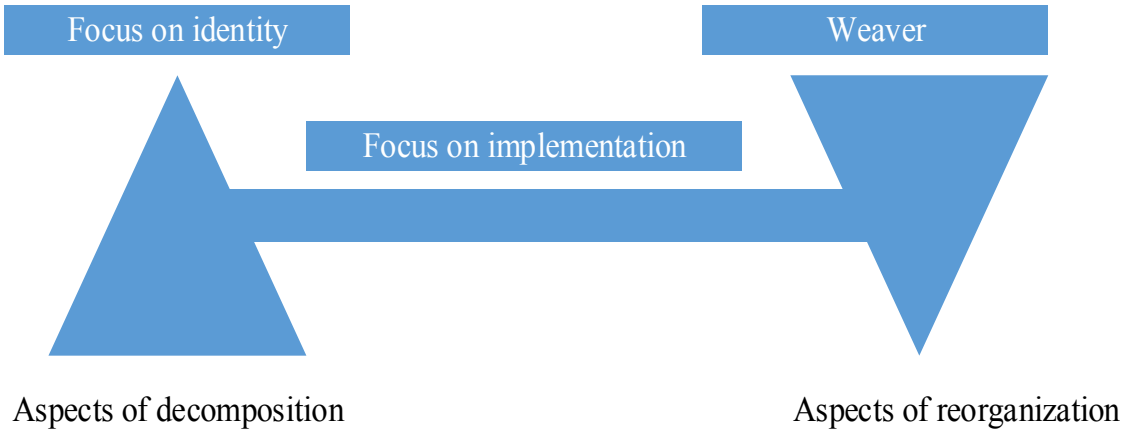

Figure 1. Finite AOP development process

Java Annotation Metadata AOP Evolution Method

Using Java Annotation metadata labeling mechanism, in the basic program using logic information on the function, the variable marked [5]. At the same time, the definition of the connection point is also described using logical information. The functions and variables that are marked with logical information are called reflection and reflection variables, respectively. The evolution of the method shown in Fig. 2.

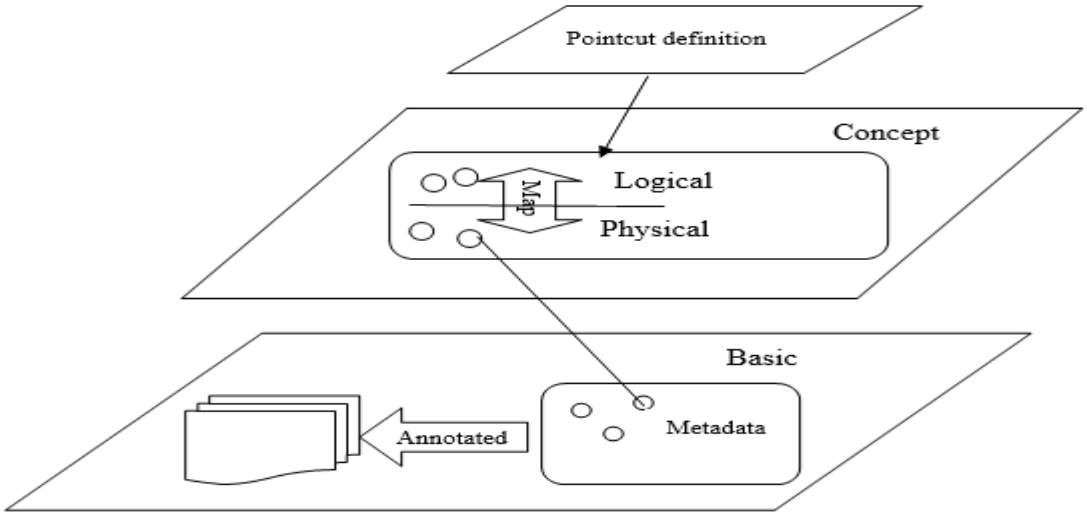

Figure 2. Finite Face-oriented software evolution based on metadata and reflection

In order to achieve the conversion of the underlying program and the automatic mapping between the physical structure information of the base program and the information represented by the conceptual model, we use the Java Annotation metadata annotation method to mark the underlying program. Table 1 is the label we define for describing the logical information.

Table 1 Describes the Java Annotation for Logical Information

The definition of annotations	Description	Be applicable
@interface LogicalConsInfo{ String modifier ( ); Table 1, cont.	Represents the logical information corresponding to the constructor. The attributes of the annotation represent the	
String name( ); String[ ] paramtype( ); }	logical modifier, the logical function name, and the parameter type of the reflection constructor, respectively.	Constructor
@interface LogicalMethodInfa{ String modifier ( ); String retumtype( ); String name( ); String[ ] paramtype( ); }	Represents the logical information corresponding to the function. The attributes of the label represent the logical modifier of the reflection function, the logical return value type, the logical function name, and the parameter type, respectively.	Ordinary function
interface GetMethodInfo{ String modifier ( ); String retumtype( ); String name( ); }	The get function that represents the variable. The annotated attributes represent the modifier of the get function, and return a value type, the function name.	variable
interface SetMethodInfo{ String modifier ( ); String name( ); String[ ] paramtype); }	The Set function information that represents the variable. The attributes of the label represent the modifier, function name, and parameter type of the Set function, respectively.	variable

Conclusions

With the software running environment and changes in user needs, the evolution of software systems become more frequent. Software evolution has become a new hot spot in modern software engineering research. A system can only evolve to meet the changing requirements, whether to provide uninterrupted service has become an important indicator of people to measure the software system. In view of the new natural characteristics of the software system and the development trend of the current software theory, technology and methods and the research hotspot, this paper uses the aspect-oriented technology to carry out the evolution theory of the software architecture. This paper introduces a method of aspect-oriented software evolution based on metadata and reflection. By adding the concept layer between the entry point and the base program, the connection point definition and the decoupling of the basic program structure are realized. According to the metadata marked in the basic program. Which utilizes the reflection mechanism to realize the automatic matching of the connection points. This method effectively solves the problem of accidental connection point loss caused by the evolution of the basic program in the aspect-oriented software, which is helpful to the evolution of aspect-oriented software.

References

[1] F.C. Yang and X.M. Long. Research on Nonfunctional Attribute of Software [J]. Journal of Beijing University of Posts and Telecom, 2004 (3): 1-12

[2] H. Masuhara and K. Kawauchi. Data flow pointcut in aspect-oriented programming .Asian Symposium on Programming Languages and Systems[C]:Springer-Verlag, 2003. 105-121.

[3] C.Y. Li, Y.J. He and Y.L. Li. Software dynamic evolution technology [M]. Beijing: Peking University Press. 2007: 71-82.

[4] Walter Cazzola, JeanMarc J'ez'equel and Awais Rashid. Semantic join point models: Motivations, notions and requirements. Proceedings of the Software Engineering Properties of Languages and Aspect Technologies Workshop[C].New York: ACM, 2006.

[5] T. Wu, S.G. Ju and T. Cai. Metadata Management Strategy Based on DBMS [J] .Application Research of Computers. 2010(4): 1297-1300.