Efforts to Reuse Software Assets

Eiji Yoshida

Abstract

While the technique of software reuse has been successful to some extent, a lot of failures have also been witnessed due to the inherent difficulty of reusing software. However, recent trends in big software data may lead to a solution to this problem. This article describes NTT DATA's latest research and development activities involving software reuse techniques.

Keywords: software engineering innovation, software reuse technique, software repository mining

1. Introduction

Software reuse is an idea that goes back a long way in the software development process. At the first software engineering conference held in Garmish, Germany in 1968, the idea of software *components* was presented. Since then, the development of methods for software reuse has had a great influence on the software development process. Some reuse methods commonly used in the software development process are listed in **Table 1**.

Although some reuse methods have become common, there have also been a lot of failures. For example, the San Francisco Project that was initiated in the late 1990s aimed to share and reuse business software components, but the project ended unsuccessfully.

Successful methods of software reuse target components of the software infrastructure such as small common libraries and frameworks for routine processing. Thus, the benefit of reuse to productivity is limited. Although the use of packages is regarded as an effective reuse method, packages tend to be usage- and business-specific. Thus, the usage of packages is also limited. As a result, the progress made in implementing reuse techniques has stagnated.

2. Transforming reuse techniques with big data of software development assets

A notable change has been occurring in systems development that may transform the existing reuse techniques. This change is the digitization of all sorts of development assets. Extremely large volumes of development related assets including not only source codes but also documents, testing materials, and records of development projects are now digitized and distributed. The important point here is that development assets are in digitized form and available for computer processing.

For example, development documents are converted to the XML (Extensible Markup Language) format using Office tools or UML (Unified Modeling Language) editors. Bug information is consolidated in a database using a BTS (bug tracking system) project management tool.

NTT DATA creates and accumulates approximately 50 million steps of source programs annually and more than 1 million files of documents for the programs. In addition, digitized data of development assets have become widely available for different organizations over the web. Currently, roughly more than 100 million webpages of development information are available on the web.

Big data has become a major trend. Techniques and tools supporting this trend, and those used for capturing, storing, and analyzing large volumes of different datasets, are readily available. More specifically, sophisticated data processing techniques including text searching, non-structured data analysis, data mining, pattern matching, natural language processing, large-scale distributed processing, and machine learning can be used. Therefore, it is important to develop new methods for reusing as much as possible large volumes of digitized development assets in the software development process by applying advanced

Table 1. Common reuse methods.

Methods for software reuse	Description
Subroutine	Became widely used as structured programming evolved. Frequently used codes are packed into subroutines as program components. Subroutines are used in relatively limited environments or for specific types of software.
Class library, framework	Became popular as object-oriented language evolved and progressed. Software components are highly scalable and easily reusable. Many components are developed for reuse in multiple projects. Some components are used worldwide as open source software.
Package	A large set of software components for transactions is reused instead of using small program units separately. Packages are commonly used in ERP (enterprise resource planning) by, for example, SAP.
Software patterns	Standard formulas of software development. Typical examples are design patterns containing standard formulas for software design and architecture patterns containing standard formulas for software structure.

data processing techniques.

3. Why is reuse difficult?

The difficulty in reusing software comes down to cost, specifically, the cost related to context mismatching. That is, the cost of reuse increases due to mismatching of contexts (backgrounds and conditions). Each reusable asset has its own suitable context. If an asset is applied to even a slightly different context, the assent cannot be used as is, and it requires customization. As a result, it is often said that if you need to customize 20% (or more) of reusable assets, it is better to spend the money on developing a program from scratch.

If we go deeper into the issue of context mismatching, two difficulties of reuse emerge. One difficulty arises in creating reusable assets and the other in applying reusable assets.

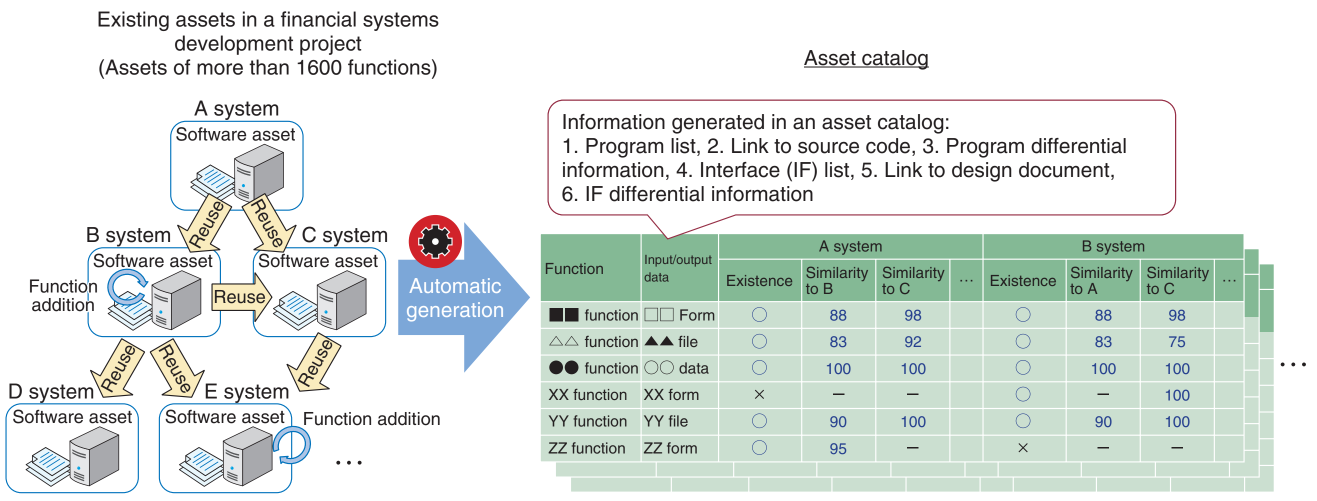
4. Systematic reuse and experience-based reuse

The difficulty of *creating reusable assets* means that it is difficult to systematically create assets with the intention of widely reusing them in different software programs in the future. In other words, it is difficult to develop reusable software components that are not needed for the current software functions but that may be widely used in the future. We often hear that creating a reusable component is three times as difficult as creating a module used in a single program. In fact, if developers know in advance that they will have to develop several similar types of software, a systematic reuse method might be effective. However, functions that are not expected at the beginning of development are often required as the development progresses. Thus, it is extremely difficult to picture a future need and to develop a highly reusable resource in a real development process. In addition, it is likely that functions implemented for expected future use may never be used.

If that is the case, is it possible to modify existing assets that have been developed in order to make reusable assets or to extract reusable assets from the existing assets by analyzing them? Some software development projects have included steps to identify common functions from multiple different development processes and to make reusable components for them. However, most of these activities are not systematically implemented. They are performed without a plan or, in a more favorable sense, are based on experience.

In addition, efforts to create reusable components are not extensively and widely implemented because such efforts depend on the developers' capability and foresight as well as their spare time in projects. Therefore, it is important to support activities carried out to make reusable components based on experience.

More specifically, even if a software program has been developed and reused by a simple copy-and-paste method without any systematic reuse planning, it is possible to extract and generate highly reusable designs and source codes by automatically analyzing source codes and design documents to identify common components and functional differences.



Information generated in an asset catalog:
1. Program list, 2. Link to source code, 3. Program differential information, 4. Interface (IF) list, 5. Link to design document, 6. IF differential information

Function	Input/output data	A system				B system			
		Existence	Similarity to B	Similarity to C	…	Existence	Similarity to A	Similarity to C	…
■■ function	□□ Form	○	88	98		○	88	98	
△△ function	▲▲ file	○	83	92		○	83	75	
●● function	○○ data	○	100	100		○	100	100	
XX function	XX form	×	–	–		○	–	100	
YY function	YY file	○	90	100		○	90	100	
ZZ function	ZZ form	○	95	–		×	–	–	

Fig. 1. Example of generation of asset catalog to help developers understand assets.

5. Difficulty of understanding reusable assets

The second difficulty of *applying reusable assets* relates to understanding the reusable assets. When reusing a component, we need to precisely understand what information is required for reuse such as the component's function, application, and customizable parts.

Understanding reusable components is a time-consuming process and is often very difficult. As mentioned earlier, because each reusable asset has a suitable context for application, it is difficult to understand both the asset and its context. As a result, reuse is often avoided. This issue may be solved by manually collecting information required for reuse, although it would incur a large cost. Therefore, we need a method for automatically extracting necessary information from existing assets to help understand reusable assets and expedite the application of reusable assets.

6. Achieving better understanding with asset catalogs

NTT DATA has been implementing a method in a financial systems development project to *automatically* sort existing assets and generate asset catalogs (**Fig. 1**). This helps developers to understand the reusable assets. In the target project, there are more than 1600 function assets in a total of 7 systems. When a new system is developed based on the existing assets, it takes a long time to examine the reusability of the assets and the reuse methods. We have automatically analyzed large volumes of existing source codes and documents to identify information about availability, similarities, and differences of functions and have created asset catalogs that make it easier to understand the reusable assets. We are using the asset catalogs for a new system development and have reduced the man-hours for reuse design by 84% (8% of the total project man-hours).

This achievement is due to the fact that we have targeted multiple functionally similar systems and because design documents are fully standardized for automatic information analysis. NTT DATA is currently studying asset analysis methods that can be applied to wider and more diverse assets.

7. Software repository mining

In terms of utilizing big data generated in development projects, we need to focus on direct use of assets as well as methods that can identify useful knowledge for development from large amounts of data on development assets.

Currently, not only source codes but various data including documents and bug information can be generated and accumulated in systems development projects. Studies on methods of extracting useful knowledge for development from large amounts of development data using data mining techniques are gaining momentum. In the academic field, these studies fall under mining software repositories. Researchers regularly exchange ideas on this topic at an

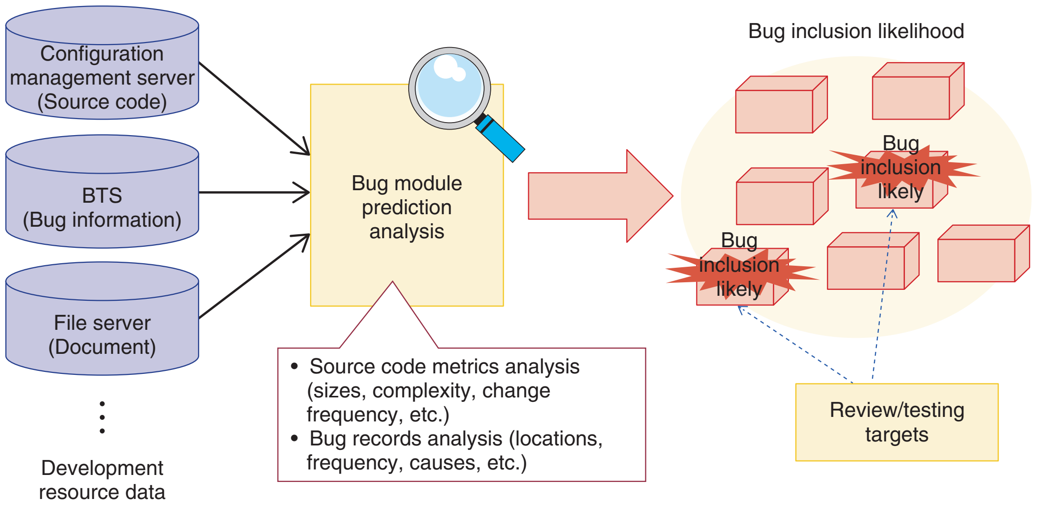

Configuration management server (Source code)
BTS (Bug information)
File server (Document)
⋮
Development resource data
Bug module prediction analysis
• Source code metrics analysis (sizes, complexity, change frequency, etc.)
• Bug records analysis (locations, frequency, causes, etc.)
Bug inclusion likelihood
Bug inclusion likely
Bug inclusion likely
Review/testing targets

Fig. 2. Example of software repository mining.

annual international conference [1, 2].

Software repository mining does not directly use existing development assets. Rather, it analyzes large volumes of development assets, identifies trends and patterns, and converts the identified trends and patterns into knowledge that is used for development. For example, in bug module prediction, bug records and source code conditions (e.g., size, complexity, degree of coupling, intensity) are analyzed, and modules having a high likelihood of bugs are identified (**Fig. 2**).

NTT DATA is developing methods for applying this bug module prediction technique to quality management in the system development process. If we can quantitatively predict modules having a high likelihood of bugs, we can develop a more effective and efficient testing strategy by focusing on testing of the particular modules and appropriately allocating manhours for testing. A similar approach involves predicting bug modules according to source code correction logs and using the predictions for review [3].

Possible applications of software mining repositories in addition to bug module prediction include identifying reusable knowledge and communicating the results of analyses within a development team. Recent advances in more mature data analysis techniques such as data mining and machine learning as well as development of high performance computing environments where we can analyze large volumes of development data in a realistic processing time will expedite the evolution of software repository mining.

8. Conclusion

Maximizing the use of development assets that are being accumulated in large volumes is a key to further developing reuse techniques. NTT DATA aims to shift from quantity to quality-based reuse approaches and will continue to develop new reuse methods that will drastically improve software development productivity.

References

[1] A. Monden, “Technical Trends and Application of Software Repository Mining,” Proc. of Software Japan 2013, Tokyo, February 2013.
[2] The 11th Working Conference on Mining Software Repositories, http://2014.msrconf.org/
[3] Bug Prediction at Google, http://google-engtools.blogspot.jp/2011/12/bug-prediction-at-google.html

Eiji Yoshida
Research Manager, Center for Applied Software Engineering, Research and Development Headquarters, NTT DATA Corporation.
He received the B.Eng. and M.Eng. in computer science from Tsukuba University, Ibaraki, in 1996 and 1998, respectively. He joined NTT DATA in 1998 and studied network technologies and middleware technologies such as CORBA (Common Object Request Broker Architecture) and web services during 1998–2006, and until recently studied software engineering, particularly techniques and tools for automating the design, implementation, and testing of large software systems. He is currently studying software analytics technologies that take full advantage of data generated in software development projects.