Current Trends of the Software Reliability in Japan and Needs of Future Related Activities

Koji TORII
Graduate School of Information Science
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, Japan 630-01
torii@is.aist-nara.ac.jp

1 Introduction

There are a variety of ways to assure reliability, one of the most important factors related to software quality. The most common one is testing. Another way is to review documents such as designs and code. However, it is hard to tell when either testing or review is completed. One reason is that when deadlines come close, there is heavy pressure to start and finish writing code even if the code is not very good and even if it is neither tested or reviewed. Another reason is that it is hard to fully understand any reliability assurance method without having a precise knowledge of software development; and acquiring such precise knowledge requires the gathering and analysis of precise data about software development.

Note that "quality assurance" is often quoted, while "reliability assurance" is note. In Japan, this is reflected, as has been widely observed, by the usual presence of an organization for quality assurance but the usual absence of an independent team from the software development group as part of this organization. Thus, although reliability looks important, it is not treated as so important as to be separated from development. If the quality assurance effort is focused and huge effort is needed, managers are afraid that the associated software development cost will be huge. This is a real dilemma in real companies. This article has two parts. First, several issues about the real ways in which Japan approaches software reliability are discussed. Second, some proposals about how to improve assurance based on focusing on the importance of raw data are presented.

2 What is being done in companies

No one will deny that, in order to assure reliability, a quantitative approach is indispensable. Because of this, data collection is being done actively in Japan, especially at the makers of computer mainframes. But not all companies can do this because of the cost and difficulties of getting agreements from the developers. Test data are collected based on the reports from the developments group and the quality (or, more precisely, the reliability) assurance group. On the other hand, data on software development phases are based on (post-mortem) reports, since development is usually done at subsidiary companies. It is rather hard job for mainframe makers to cooperate with developers in collecting data, since it is not easy to explain the importance of data collection to developers, especially at independent software houses. Therefore the reliability of collected data is itself highly unreliable.

TQC (Total Quality Control) is one of the famous managing methods in typical Japanese companies. The QFD (Quality Function Deployment) seems a famous and effective way to assure quality. Unfortunately, there are also reports that QFD cannot be applied easily in software development compared to other manufacturing engineering, construction industry, chemical industry, service industry, etc, because the software development process is so complex with so many influential factors.

3 What is being done at academic field

It is rare to do a cooperative project between universities and companies, so academic researchers do not have actual data. So they ask companies for data, though most of them are not the data that they really want; also, sometimes the data are old. Research results on various software reliability growth models are frequently published, and the data are used in the papers to validate these model. But the above problems mean that those data are sometimes incomplete,

0-8186-4010-3/93 $03.00 © 1993 IEEE

because some data on expected or necessary factors are lacking.

In our group, we have tried to collect exact data from class experiments by using an automatic data collecting system called GINGER. This system has been actively used for more than five years. One of the defects of this approach is that the project examples used in the classes are small; for example, the biggest project we have studied was the construction of a compiler, which students wrote in roughly 1000-3000 lines of code.

4 Domestic Conference Activities

Since there are several academic journals related to software engineering in Japan. There are also several domestic conferences, some of which are refereed and some of which are non-refereed. But specific conferences on software reliability are very rare; in fact, there is only one exception, which we have been running annually for more than ten years. This is called the "software reliability symposium", which primarily consists of informal reports on actual problems in software reliability. As easily imagined, the reports from academic fields are uncommon, and most of reports come from industry. The most familiar topic of this symposium is usually the results of specific quality control activities at the development scene. Thus, reliability modeling is still too far away to do in these cases.

5 Proposals for the future

As we have seen, software reliability is still in its early, immature stages. It is very ironic to understand that the reliability of the raw data on which product reliability now relies on is itself very unreliable. Based on the observations described above, I would like to propose the following two activities.

Proposal 1: Consistent proper data collections using the same tool.

One goal is allow us to share and compare the data collected at different sites, which requires the gathering of the same kind of data at different sites. The big advantage comes mostly from the fact that we can share all of data collected all over the world. (There is another advantage that students evaluation is done fairly in programming classes based on the process of software development and the final products.)

To achieve this goal requires a data gathering system that has many properties similar to our GINGER system. Our system allows us to collect the same kinds of raw data at different places at any time. Also, it collects data directly from the computer system with no human interaction, which reduces the problems of getting data from unreliable post-mortem reports. Such a system should also use automatic and real-time collection.

This approach is just like data collection in astronomy. Astronomers collect data for very small time slices during their life, but they get lots of data about different stages of a star's life, which allows them to understand—by combining those different kinds of data—the data about a full star's life. Our system GINGER, though it is still a prototype level, could be used in this way.

Proposal 2: More active data collection activities.

The above approach to data collection is based on the process of programming as is visible to and can be collected by the computer. But we also need to know more about different kinds of data, such as human behavior, especially during the specification and design phases. This requires discussions with colleagues in a group, consultation with different kinds of documents or data on actual experiences. Therefore, we have to observe the behavior using video cameras, etc. In this case, we need an experimental laboratory to do such a controlled experiment. Moreover if we can connect the laboratory to remote site by network, we can control the experiment from different points of views. We are now planning a cooperative joint project with companies at our institute to do such an experiment. I expect more people to join to do different kinds of data collection on which anyone can share the data to do future research.

6 Conclusions

Recently, software development is following hardware development in terms of features such as downsizing and short time-to-market. In the past, software development took place slowly in a centralized computer environment. On the other hand, recent software has the potential to be released in a rather short development time, because of the availability of distributed development environments and workstations. In this kind of short lifecycle, software reliability should be a central quality concern but not be overwhelmed. The trade-off among different factors of qualities becomes even more important than in the traditional big size development style. We have to evaluate the acquired quality in quantitative way, and in order to do this we need a suitable measure. To have such a measure, we need to have appropriate data we can analyze. As a result, we must have precise data.