Design Technique for Enhancing Software Quality and Development Suitability

Kunihiro Yamada¹, Kouji Yoshida², Masanori Kojima³, Nobuhiro Kataoka¹, and Tadanori Mizuno⁴

¹Professional graduated School of Embedded Tokai University
   yamadaku@tokai.ac.jp
²Shonansaka Institute of Technology
³Osaka Institute of Technology
⁴Aichi Institute of Technology

Abstract. Taking software designing as a technical procedure will greatly contribute to the enhancement of software quality and development efficiency. To take it as a technical procedure means to get some specific processes from specifications by means of design, which also means that it is inevitable in the present technology. When dealing with software, the method of realizing a specific process covers a fairly wide scope compared with hardware, mechanisms and systems, which weakens the relationship between specifications and processes, and it eventually deteriorates the quality and aggravates the development efficiency. Hence, we present a software design technique as a single set of three items for a method of taking it as a technical procedure. This paper covers concise functional partitioning, an assessment-minded software structure and selection of the best technique, which is the most important, and it also describes the reasons for the said items.

1 Software Design Technique

Different types of studies and activities that deal with technical enhancement, of software are found in many fields. We see structured programming techniques for software structure [1, 2], method of quantifying software quality assessment [3], test of programming in parallel processes [4] and collaborative design test, which requires simultaneous assessment of hardware and software [5], for example. They need parallel processes of software and hardware-collaborative assessment. Problems in software do not remain within the software itself, but they greatly give effects on the process of developing hardware, mechanisms and systems. For software design and development, a number of techniques are found [6], and a variety of operations are seen as well. However, such measures have not achieved any complete solution so far.

In place of the above, our proposed software design technique offers a set of three items including concise functional partitioning, assessment-minded software structure and the best technique selected, together with the reasons for the selection. Concise functional partitioning intends to obtain a structured construction while structured designing involves an important idea in software development. Execution of optimal structuralization firstly requires a proper process of functional partitioning, and the
more proper and adequate is the functional partitioning, the simpler and more streamlined the partitioning will be. Therefore, it is named "Concise Functional Partitioning", here. It is essential in the first place that the assessment-minded software structure is subjected to proper functional partitioning by means of concise functional partitioning, and a further problem is whether we can reduce the items for assessment. For the selection of the best technique and the description of its reasons, the best item is selected out of the specifications and a number of method of realization so that we can be ready to answer later confirmation.

2 Concise Functional Partitioning

The major task of the concise functional partitioning is how to partition and layer the functions to be achieved by the software. It aims at fewer errors made and man-hours consumed during the development, and further, it also focuses on their high extensibility and maintainability. The most focused point is that the software must have clear algorithms and applicable routines for other software to be developed later. Concise functional partitioning means that functions are classified as shown in Figure 2, supposing the volume of whole tasks to be "1" as shown in Figure 1. We refer to this functional partitioning as a "Method of Functional Partitioning Based on Conditions".

![Fig. 1. All functions (tasks) of software](image)

![Fig. 2. Concise functional partitioning of software](image)

![Fig. 3. Inadequate functional partitioning of software](image)