Software Production Technologies that Support Large-scale System Development

Hiroshi Tomiyasu

Abstract

Information systems that support social and industrial activities cannot be constructed without software on a large scale. Production technologies for developing such software, however, have made little significant progress over the past few decades, and what currently happens is that ever-increasing demands are met using labor-intensive methods. These Feature Articles describe innovative software production technologies that can overcome these issues with software development.

Keywords: software production technologies, software development automation, legacy systems

1. Introduction

In the 21st century, it is fair to say that society is growing in tandem with computer systems (hereinafter referred to as *systems*). All elements of society, including global financial networks, transportation infrastructure, telecommunications infrastructure, and corporate activities, in addition to social areas involving individual consumption activities and friendship-related social activities are supported by systems.

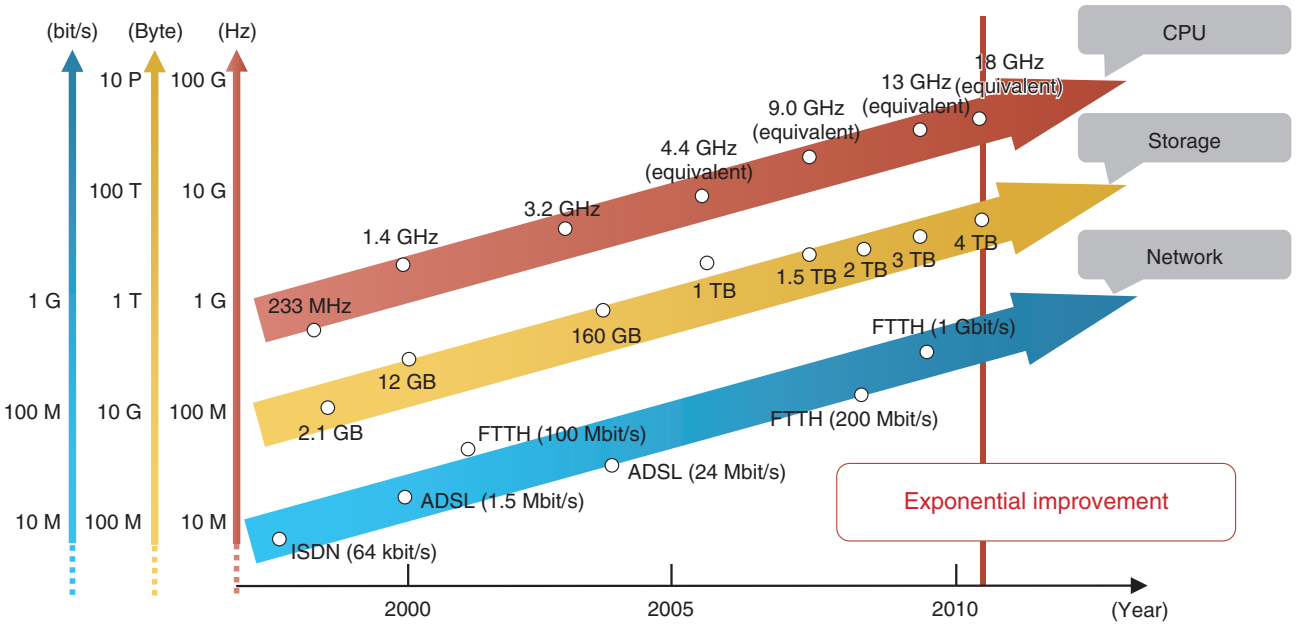
It was not very long ago, however, that the dramatic increase in the demand for systems occurred. The years 1995 and 2000 were respectively known as the *first year of the Internet* and the *year 2000 problem*. Before then, systems were used only in public offices and major companies in the financial, telecommunications, and manufacturing industries, and the presence of systems was not recognized or discussed by the general public. In other words, during the last decade or two, the existence value of systems has risen steeply, and systems have become an essential social infrastructure. This change was brought about namely by the increase in computer power achieved through the progressive technical development of hardware.

The capabilities of the three major element technologies: the CPU (central processor unit), hard disk (for storage), and network, have increased by ten thousand times in the last decade and a hundred thousand times in the last two decades, as shown in **Fig. 1**. In parallel with this, their prices have declined, another major factor that has enabled individuals to easily use computers and systems.

2. Role and progress of software technology

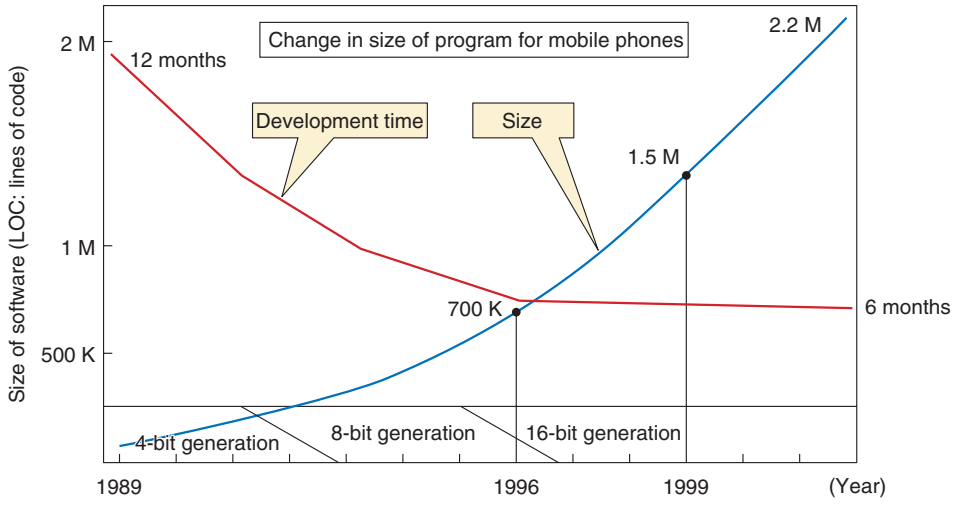
However, software is another element in addition to hardware that makes up systems. The development scale of software has been increasing in the last two decades so that systems can meet the various demands for them in society, while the development time has simultaneously been shortened. The changes in the sizes of software development programs and in the development time for mobile phones are shown in **Fig. 2**. In addition, a wide variety of software has been developed, including software for financial systems run on general-purpose computers, E-commerce software run on widely used operating systems, for example, UNIX and Windows on open servers, and also game software for smartphones. Today, software is an essential component to achieve or support such social activities.

However, the development of software technology



ADSL: asymmetric digital subscriber line
ISDN: integrated services digital network
FTTH: fiber to the home

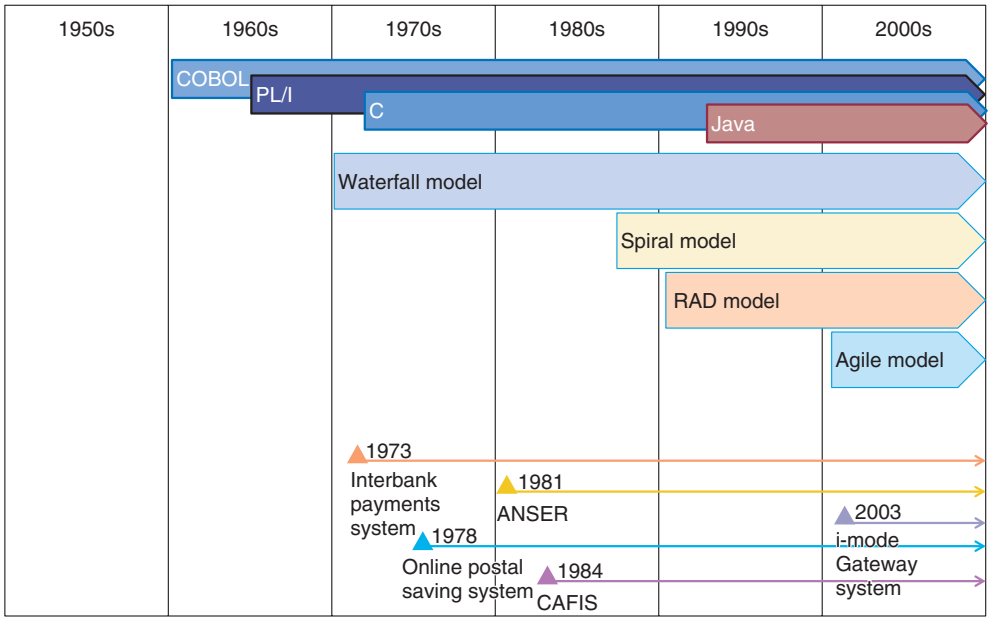
Fig. 1. Progress of three major element technologies.



Source: ET2002 TB-6 "Measures for improvement in quality in embedded system development" (Corporate Research & Development Center, Toshiba)

Fig. 2. Changes in size of software program and development time for mobile phones.

over the last two decades has progressed more slowly than that of hardware. The progress achieved in software technology is shown in **Fig. 3**. This progress includes the evolution of programming languages in the beginning of the development period. In the early computer age, software was developed using languages that ran directly on computers such as machine language and assembly language. These languages



ANSER: Automatic answer Network System for Electronic Request
CAFIS: Credit and Finance Information System
COBOL: common business-oriented language
PL/I: Programming Language One
RAD: rapid application development

Fig. 3. Technical progress of software development.

were too complicated for humans to understand, so high-level languages such as COBOL (common business-oriented language), C, and Java were developed later. These languages can be written with a writing system such as English that can be easily understood by humans, and as a result, they became widely used and mainstream programming languages. At the same time, development processes have progressed. Large-scale software development requiring many person-hours must proceed without gaps in awareness occurring between development personnel. This is achieved by defining the procedures and formats of necessary documents for design, programming, and testing. Over the years, a great number of development processes have been accumulated, which are now consolidated into the widely used *waterfall* and *agile* development processes.

As shown in Fig. 3, this progress in technology has enabled large-scale system development since the 1970s. When we look at software development only, however, and if we consider one benchmark for development productivity to be the amount of software that can be developed by one person, we find that these technologies have actually contributed very little to improving productivity in the last 20 years. As mentioned earlier, the hardware capability has been increased a hundred thousand times in the past two decades. Unfortunately, the progress made in software technology has not improved software development productivity.

The reason for this is that these technologies have been developed on the assumption that all of the work related to software development is performed by humans. In other words, we have been developing technologies to create an environment comfortable for humans that is easy to understand or that helps humans understand the flow of a complex development process more easily. However hard one person works, though, there is a limit to the number of documents and the amount of software that can be created or programmed by him/her only.

3. Software development technology to support large-scale software production

The social importance of software is increasing, and as described earlier, the amount of development is also increasing year by year. In addition, in Japan,

large-scale social infrastructure development is planned for the coming years, which will greatly increase the demand for software development. This is referred to as the *year 2015 problem*, because there are concerns that meeting the demand for such large-scale development will be difficult.

One solution to meet the greatly increasing demand is to increase the number of developers. Previously in offshore development, we outsourced development to countries endowed with abundant labor and with lower labor costs such as China and India, and we imported software created in such countries. However, the recent depreciation of the Yen offers little cost advantage in continuing with this approach.

Now, it is necessary to drastically improve the productivity of software development. To do this, we need to use computer power to create a process in which *no human is involved* or *nothing is produced* (explained below). There are differences of opinion as to the feasibility of this, but a look at the automobile industry shows that the same movement occurred in the 1990s and the 2000s, which led to success. Automobile design is now digitized, and the movement and performance of automobiles are evaluated by computer simulation to reduce the number of trial vehicles that are produced, which shortens the period to start a mass production. Needless to say, mass production is automated and performed by robots in the plants. Implementing the same sort of automation process in software development would drastically improve productivity.

NTT DATA is working to substantially improve productivity by carrying out activities to innovate software production technology. These activities target particular areas; these areas are outlined below and described in detail in the other articles in this issue.

(1) Promotion of automated software development [1]
(2) Research and development (R&D) of simulators [2]
(3) R&D for reuse of software [3]
(4) Legacy modernization [4]

With regard to (1), we aim to automate each process in conventional software development by reducing human involvement. Software development broadly consists of three processes: design, production (programming), and testing, and these processes are implemented using automation tools that can minimize the involvement of humans. The key point is that conventionally implemented workloads can be reduced with no involvement of humans, which leads to improved productivity. R&D of these automation tools was done previously, and many similar types of products exist around the world. However, they have not come into wide use. NTT DATA has been promoting this software development style since 2010, and the key point is how we expand the use of our style and make it the de-facto standard.

Regarding item (2), the objective is not to automate the processes that have been previously implemented by humans but to simulate the operation of software using digitized design information to determine whether the operation is normal before initiating production. This activity is modeled after the automobile industry mentioned earlier. This way of carrying out the process differently from that in conventional software development can achieve a significant improvement in productivity.

The approach for item (3) is to *produce nothing*. NTT DATA possesses a great amount of software and related documents including design specifications. We have tried reusing these assets many times and have failed every time. This is not an experience that pertains to NTT DATA alone but a phenomenon present in the software industry in general. Thus, we started R&D not only to clarify what prevents us from reusing them but also to derive conditions for successful reuse. We are also studying not only direct reuse of these software and design specifications but also how to use them indirectly to support other activities using information obtained from them.

Finally, item (4) concerns how we can revitalize the existing systems. Although the purpose is different from those described above, this may be a major issue in Japan in the future. In Japan, as represented by the word *mottainai*, we have a culture of using tools carefully for long periods of time. Systems and software are no exception. Hardware deteriorates with age, while software does not need to be changed unless any major changes are made to operations and services. Therefore, software can be used in society continuously for generations. Every time the configuration of hardware changes significantly, however, the software is affected by such changes.

There is currently a major trend towards migrating systems from hardware known as general-purpose computers or mainframes to open hardware or hardware with UNIX operating systems that include a Linux or Windows operating system. This trend means that we have to not only migrate software but also to make major unnecessary changes and modifications to it. Systems constructed on general-purpose

computers are often called *legacy* systems; therefore, this approach is called *legacy modernization*. Many extremely large-scale systems are nearing the end of their use due to the general aging and deterioration of general-purpose computers. Because many of these systems have been subjected to repeated revisions/changes over several decades, their internal structures are unknown, and there are no skilled programmers able to work with them. An improvement of this situation cannot be achieved with human labor alone; it requires analysis by means of computer power and automation of software development for revitalization, which we are now studying.

4. Conclusion

In the era of mass consumption of software along with the use of larger-scale and more diversified systems, it is clear that we have nearly reached the limit of software development by humans. Software development, however, is associated with risks, and we may face psychological barriers when adopting a new software development method. More time is needed to address this situation. NTT DATA will proceed with plans to challenge the common practices of software development by conducting R&D on tools to achieve a drastic improvement in productivity while steadily emphasizing the need to expand the use of such tools.

References

[1] T. Azuma, "Automation Technology for Software Development at NTT DATA," NTT Technical Review, Vol. 12, No. 12, 2014. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201412fa2.html

[2] T. Kaneko, "From Labor Intensive to Knowledge Intensive—Realizing True Shift to Upper Phases of Development with TERASOLUNA Simulator," NTT Technical Review, Vol. 12, No. 12, 2014. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201412fa3.html

[3] E. Yoshida, "Efforts to Reuse Software Assets," NTT Technical Review, Vol. 12, No. 12, 2014. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201412fa4.html

[4] H. Tanino, "Legacy Modernization for Continuous Use of Information Systems," NTT Technical Review, Vol. 12, No. 12, 2014. https://www.ntt-review.jp/archive/ntttechnical.php?contents=ntr201412fa5.html

Hiroshi Tomiyasu

Head of Center for Applied Software Engineering, Research and Development Headquarters, NTT DATA Corporation.

He received the B.Eng. in engineering sciences from Tsukuba University, Ibaraki, in 1990. He joined NTT DATA in 1990 and studied image recognition systems from 1990 to 2003. He then moved to the financial system division and developed financial systems for several years. He moved back to the R&D division in 2006, and until recently was leading research on software engineering, particularly techniques and tools for automating the design, implementation, and testing of large software systems. He is currently working on expanding the use of automation tools within the NTT Group. He is a member of the Information Processing Society of Japan.