Concurrent Development of Software Systems
A New Development Paradigm

Mikio Aoyama

Telecommunications Software Department
FUJITSU LIMITED
1015 Kamikodanaka, Nakahara-ku, Kawasaki 211, JAPAN
and
Software Engineering Laboratory
Department of Electrical Engineering and Computer Science
The University of Illinois at Chicago
Box 4348, Chicago, IL 60680

1. Introduction

Due to the rapid evolution [1] of software systems and to shortening the development period, a software development process must change its style, that is, from sequential to concurrent.
The term *concurrent development* means that a number of development groups performs a number of development activities, such as specification definition, implementation and testing, concurrently. In practice, these development activities are mutually dependent via some common software objects to be modified. In this situation, correct management of development activities is indispensable for productivity and products-quality. Moreover, this activity has some analogies to the control of concurrent processes of a distributed software system.
This article intends to discuss some concepts and properties of concurrent development. Some key issues on managing concurrent developments will be tossed out. Since an appropriate model is necessary to describe a concurrent development and to analyze it, *Extended Modified Petri Net (EMPN)* is proposed as a model of it. Finally, the requirements and concepts of the enviroment to support the management of concurrent development will be discussed.

2. Why Concurrent Development ?

As a formal representation, the *waterfall paradigm* [2] has been widely accepted in software industry. However, various *new paradigms* [3] have been proposed for last decade. The examples are *prototyping paradigm* [4], *operational paradigm* [5], and *transformational paradigm* [6]. The prototyping and operational paradigms focus on the specification process. The main concern is how to get a correct specification in the early stage of development. The transformational paradigm was born of various research on the automation of programming. From the view point of development management, Boehm proposed a new paradigm called *spiral model* or *risk-driven formalization* [7].
However, these development paradigms are basically sequential in which development activities arise sequentially. Nowadays, development activities must be performed concurrently due to the following reasons;
1) Explosion of requirements:
We have been applying various techniques to reduce the quantity of software to be developed. However, for a sophisticated software system, there are still large amount of requirements.
For example, a telephone switching system now provides many kinds of switching features which serves not only conventional voice switching but also digital data switching, and ISDN (Integrated Service Digital Network) features and so on [8]. The conventional voice switching feature is the kernel of the whole switching features, and is developed first. Then, many new features are developed concurrently.
2) Quick delivery of software:
With the rapid advancing of telecommunication technology, new switching features must be provided quickly. Therefore, the development period must be shortend drastically.
3) Rapid evolution:
To keep the pace with the dynamical changes of the information society, the software systems are evolving rapidly. Therefore, the maintenance activity must be faster and more productive.

From my experience, the evolution speed of a current switching software is about three times to that of the previous one.

3. Concurrent Development and Concurrent Processes

Note that we got a great computation power by the change of architecture from sequential instruction fetch to pipeline control, or from single processor to multiprocessors. Why we cannot apply these experiences to the software development ?
The software development activities consist of diverse human activities and then the behaviour is highly complicated. However, we can observe the following analogies between software development activities and computation actions. The human activity corresponds to the computation proccess and software objects to be developed correspond to the data objects which manipulated by the processes. The concept of process programming [9] is based on the notion of some analogies between software development and software.
If we regard software development activities as software, we can find various properties in terms of software processes. As an example, suppose two programmers update the same program module. It is quite similar to that which multiple modules access to single data object.

4. Management of Concurrent Development

With some analogies to concurrent processes, the following problems are essential to the successful management of concurrernt development;
1) Modeling of concurrent development,
2) Methodology of managing concurrernt development,
3) Support environment for concurrent development.
This article focuses on the modeling and management issues. The discussion on the support environment will be presented elsewhere.

4.1 Modeling of Concurrent Development

An appropriate model is necessary to describe a concurrent development and to analyze it. Note that many models are proposed for concurrent processes of distributed software. These models are classified into a number of groups [10]. One group consists of distributed languages. Another group consists of graphical models such as Petri nets [11], Data flow diagram [12], and so on.
Various graphical models have been applied to the management of software development. The typical examples are *PERT* (Program Evaluation and Review Technique) and *CPM* (Critical Path Method). However, they do not provide enough capability to describe the concurrent activities and interaction between activities and objects.
To describe a concurrent development process, the following capabilities will be neccesary to the model;
a) Levels of abstraction: to describe development activity in any levels of detail, from a personnel task to a group task, or from an individual program module to the whole system,
b) To describe interaction among concurrent development activities,
c) To describe changes of software objects,
d) To describe interdependencies among software objects,
e) To analyze development processes with some tools on a computer: it is necessary to analyze the development process to assess the development schedules, resources and so on,
f) To represent development activities comprehensively.
Based on the above requirements, the authors propose Extended Modified Petri Net [13, 14], abbreviated as EMPN, as a model of concurrent development. EMPN is an extension of Petri nets which can describe not only control of activities but also objects accessed by concurrent processes. Besides the various properties inherited from the original Petri nets [11], EMPN model has a number of advantages which meet the above requirements. They are briefly summarized in terms of software development.
a) The integrated representation for both development activities and software objects.
b) The capability for the timing specification which is essential to the managent of the development

progress.
c) The hierarchical representation capability.
d) Based on the graph theoretical analysis, there are various analytical tools for Petri nets [15].
e) The associated textual representation called *EMPN Description Language (EMPNDL)* is developed.

4.2 Examples

Fig.1 shows an EMPN representation of conventional sequential waterfall model. This figure is drawn with a Petri net package named *Great SPN* [16]. This package has a graphical editor for statistical Petri nets and various analysis capabilities.
A circle is called place and means a development process. Black box is called transition and means a proceeding of the development process. A white box is a software object developed, such as document and program.
Fig. 2 shows the extension of waterfall paradigm from sequential to concurrent. First, project A starts the development processes. After then, project B starts the enhancement of the system. However, suppose the project B catches up the project A at the coding stage. At this time, both projects may access the same object OBJ_A and this situation is quite similar to *mutual exclusion.*

5. Methodology of Managing Concurrent Development

The key techniques are as follows.
1) Scheduling of development processes:
The control of schedules must be rigorous, real-time, and flexible.
2) Load distribution of development:
Work load changes day by day. Dynamic load distribution is necessary.
3) Synchronization and communication between each development activities:
Together with the control of schedules and load distribution, it is indispensable to synchronize each activities, so that each designer can access any software objects without any interference.
4) Consistency of modification of software objects:
Consistency can be classified into two types, that is, space consistency and time consistency. Space consistency means consistency among objects which are mutually coupled directly and/or indirectly. Time consistency is the consistency of a number of concurrent modifications to one software object. Both types of consistencies have to be guaranteed.
5) Integrity of a system:
A system consists of a large number of software objects which may be of different versions or levels. Strict integrity control is necessary.

6. Support Environment for Concurrent Development

An integrated support environment is necessary to manage a concurrent development of a large scale software system. Such an environment is essential to a successful concurrent development. The enviroment has to support the techniques which are pointed out in the previous section. However, since the environment has to provide diverse facilities, it is under consideration and will be explored further.

7. Conclusions

This article proposes a new style of software development process which is called concurrent development. The concurrent development is viewed as a new paradigm to the current and future development processes.
A Petri net based model called EMPN is proposed for the modeling and management of concurrent development. Then, this article points out several basic properties of concurrent development which has some analogies to concurrent processes of a distributed software. To manage the concurrent development process, several basic requirements are discussed.
However, since this theme has wide relationships with a number of different areas, such as development organization, methodologies and environments, there are still many problems to be studied. One important issue is the support environment for the concurrent development which will be presented elsewhere.

Acknowledgement

I would like to thank Professor Carl K. Chang of the University of Illinois at Chicago for his encouragement and for his helpful discussion. Thanks are also due to Mr. Kiyoh Nakamura, Mr. Shin-ichi Tajima, Mr. Makoto Suzuki, and Mr. Tadamichi Suzuki for their support during my visit to the University of Illinois at Chicago. Many colleagues in FUJITSU LIMITED gave me encouragements, discussions and cooperation. They all have my gratitude.

References

[1] M. M. Lehman and L. A. Belady, *Program Evolution*, London:Academic Press, 1985.
[2] B. W. Boehm, "Software Engineering," *IEEE Transactions on Computers*, Vol. C-25, No. 12, December 1976, pp. 1226-1241.
[3] W. W. Agresti, *New Paradigms for Software Development*, IEEE Computer Society, 1986.
[4] T. Taylor and T. A. Standish, "Initial Throuth on Rapid Prototyping Techniques," *ACM SIGSOFT Software Engineering Notes*, Vol. 7, No. 5, December 1982, pp. 160-166.
[5] P. Zave, "The Operational Versus The Conventional Approach to Software Development," *Comm. ACM*, Vol. 27, No. 2, February 1984, pp. 104-118.
[6] H. Partsch and R. Steinbruggen, "Program Transformation Systems," *Computing Surveys*, Vol. 15, No. 3, September 1983, pp. 199-236.
[7] B. W. Boehm, "A Spiral Model of Software Development and Enhancement," *ACM SIGSOFT Software Engineering Notes*, Vol. 11, No. 4, August 1986, pp. 22-42.
[8] M. Aoyama, T. Suzuki, M. Suzuki, and H. Fujimoto, "Development of Telecommunications Software Based on Paradigms," *Proc. of 6th Int'l Conference on Software Eng. for Telecomm. Switching Systems*, April 1986, pp. 112-117.
[9] L. Osterweil, "Software Processes are Software Too," *Proc. of 9th ICSE*, March 1987, pp. 2-13.
[10] S. S. Yau and M. T. Caglayan, "Distributed Software System Design Representation Using Modified Petri Nets", *IEEE Transactions on Software Engineering*, Vol. SE-9, No.6, November 1983, pp. 733-745.
[11] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Englewood Cliffs: Prentice-Hall, 1981.
[12] R. E. Filman and D.P. Friedman, *Coordinated Computing: Tools and Techniques for Distributed Software*, New York: McGraw-Hill, 1984.
[13] C. K. Chang, M. Aoyama, and T. M. Jiang, "Design Methods for Distributed Software Systems", *Proc. of NCC '87*, June 1987, pp. 477-483.
[14] M. Aoyama, "Concurrent Development: A New Development Paradigm," presented at NCC '87, June 1987.
[15] F. Feldbrugge, "Petri Net Tools," *Advances in Petri Nets 1985*, G. Rozenberg ed., New York: Springer-Verlag, 1986, pp. 203-233.
[16] G. Chiola, "Great SPN User's Manual, Version 1.0," *Technical Report*, Departimento di Informatica, Universitata' di Torino, 1986.

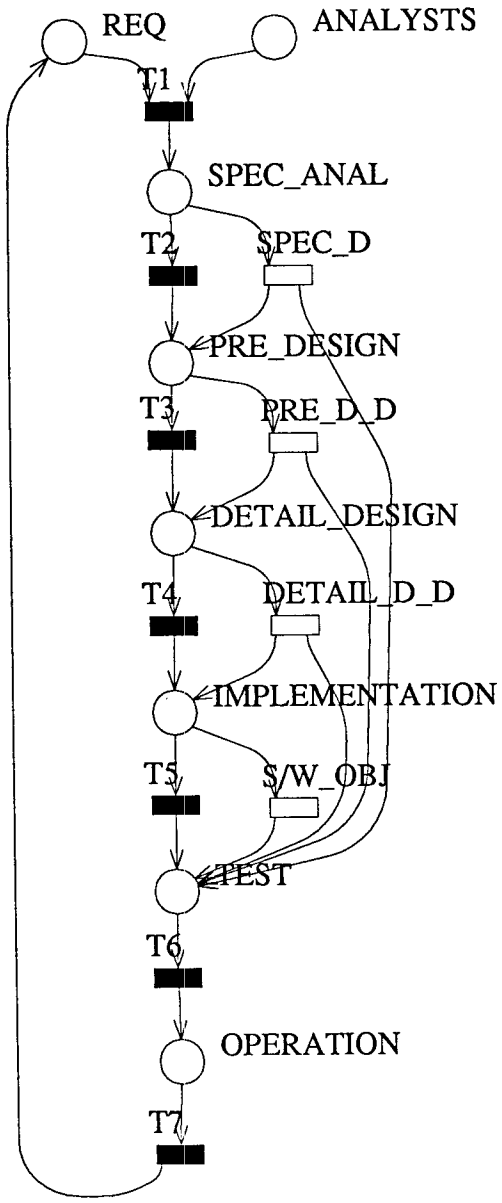

Fig. 1
Sequential Development

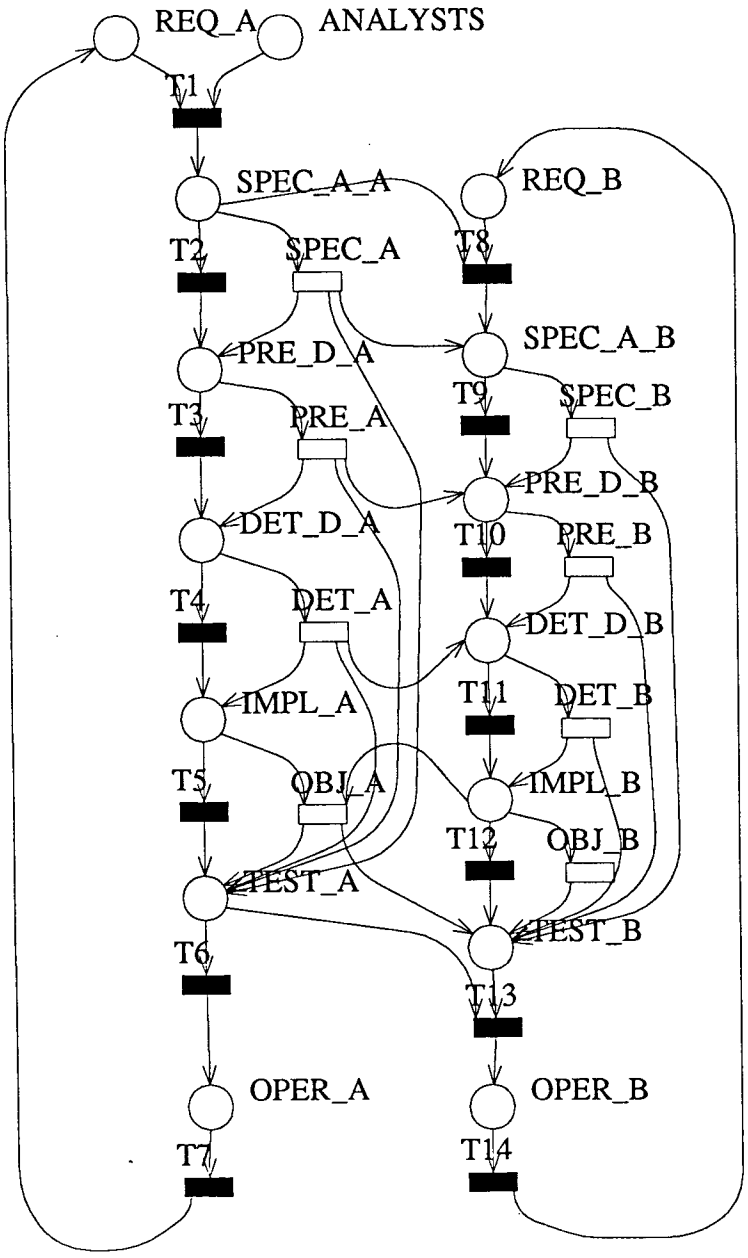

Fig. 2
Concurrent Development