Int J Software Informatics, Volume 5, Issue 3 (2011), pp. 535–546
International Journal of Software and Informatics, ISSN 1673-7288

E-mail: ijsi@iscas.ac.cn
http://www.ijsi.org
Tel: +86-10-62661040

Polyphonic Aspects of Software Process: Looking for Philosophical Foundation of Software Engineering

Kouichi Kishida

(Technical Director, SRA, Inc., Japan)

Abstract This retrospective essay summarizes 50 years experience of a first-generation programmer in Japan focusing on his thoughts on software process issues. It started from strong concern of the structure of program execution processes inside computer hardware, then went out of digital box to think about software development and maintenance processes, namely how to deal with issues between people and software. Finally his main concern turned to the social issues of software evolution processes in the Internet society. The concept of "immaterial labor" seems to be important to think about today's process issues outside of digital boxes.

Key words: software process; software development; software evolution; technology transfer; philosophical foundation; process model; immaterial labor

Kishida K. Polyphonic aspects of software process: Looking for philosophical foundation of software engineering. *Int J Software Informatics*, **Vol.5, No.3 (2011): 535–546. http://www.ijsi.org/1673-7288/5/i97.htm**

1 My First Encounter with "Process"

Before coming into the world of software, I have been self-training myself as an abstract painter. My virtual teacher was Paul Klee, I have studied fundamental artistic technique through his book "Das bildnerische Denken (The Artist's Thinking)"[1]. This 500 pages book was the collection of his essays and the complete record of his lecture note at the Bauhaus.

November 14 1921, in the opening session of a series of workshop lectures at the Bauhaus, Klee gave a short speech titled as "Analyse als Begriff (Analysis as Concept)". In this talk, Klee pointed out that analysis in art is totally different from analysis in science.

If some poisonous object is brought into laboratory, the chemist will break it down to components and perform chemical analysis to find out which component is the source of poison. In the world of art, such approach is not useful unless you want to make a fake copy of target masterpiece. Klee stated as follows:

"The motivation of analysis in our job is of course different form science. We want to know the way other artists walked in the creation of their work to find out

Corresponding author: Kouichi Kishida, Email: k2@sra.co.jp
Paper received on 2011-02-13; Revised paper received on 2011-03-28; Paper accepted on 2011-03-31; Published online 2011-04-03.

the road for ourselves to walk. Using such an approach, we can avoid wrong attitude of understanding art works as fixed and invariant things. The main purpose of this special style of analysis is to study the PROCESS of art works creation."

This process-sensitive way of thinking and behavior became the primary behavioral principle for me as an artist, then later as a computer programmer.

2 First Step into the World of Software

1960 was my last year as university student. My major was astronomy, but I've already lost my interest in astrophysics. One day, I got a part-time job at a computer seminar company. The English material given to me to translate into Japanese was a textbook used at Columbia University's computer programming course[2].

It was the era of machine language. The author (Joachim Jeenel of IBM) explained how to make machine language code in unique systematic way. I was fascinated his method of presenting program structure using a set of hierarchical flowcharts. They look like a kind of abstract art drawings, and reminded me a famous quote of Paul Klee:

"Art does not reproducing the visible, rather, it makes visible".

This statement implies the essence of abstract art. And I realized that the job of computer programmer is also to make invisible program execution process inside the machine visible.

After dropping out from the university, I started my programming career as a freelancer at first, then got a job in a computer sales company. Dealing with a variety of application programs (mathematical calculations, linear programming business data handling, system programming, etc), I always kept my process-sensitive attitude. All of application requirement specifications were treated just as examples to seek beautiful abstract process structure. "Make invisibles visible" was my slogan as a programmer.

3 Structured Program Theorem

Year 1967 was a turning point of my programming life. Together with some colleagues, I started a small company SRA. It was one of the first generation independent software houses in Japan. It was rather dangerous venture project, because at that time software was treated as a kind of free supplement for expensive computer hardware. My first job, besides managing a number of contract-based projects, was to establish the importance of software to public and make our company name visible to business community as a whole. I wrote many articles for business and technical magazines, translated various computer-related English books, and also published original books presenting our programming methods.

Among those efforts, two books were successful in establishing our company's unique technical strength. These works were the result of elaboration of our program design method inspired by the structured program theory paper published in CACM magazine in 1966[3].

Our first book was a case study of structured design of business data processing application. We used hierarchical flowchart as tool of program design and succeeded to unify the structure of data processing application into one simple structure. We

made a program generator tool based upon this design method. It was adopted some number of computer users including a large commercial bank, a steel manufatcturer, etc.

The other book was yet another application of the same method to systems programming. We explained the mythical structure of assembler and compiler as a kind of file processing. This book got a large number of young readers because there was no such a book on systems programming in Japan before.

Around same time, Edgar Dijkstra, Michael Jackson, and Jacque Wrier proposed their own version of structured programming idea. It was the worldwide trend of software technology. Chief Programmer Team paper written by F.T.Baker and Harlan Mills in IBM System Journal accelerated the trend[4]. Our approach was just synchronized to the trend.

4 Software Engineering was Born

1970s was a rather busy decade. Japanese government (MITI) kicked of a new funding agency called IPA (Information Technology Promotion Agency), and started a series of national projects using government funds. In response, 20 software houses including SRA established SIA (Software Industry Association), and I was assigned as a member of Software Technology Committee of the association (SIA/STC).

In the summer of 1970, I made my first visit to USA, and attended WESCON'70 Conference in Los Angels. I listened to Winston Royce's presentation about "the waterfall model" of software development there[5]. It makes me notice that there are much more important process issues in-between people and software outside of computer hardware. Nevertheless, my major concern was staying inside of the digital box. In the first national project exploring software modularization technology, SRA team prototyped a new process management structure for future operating system. It was an experiment of naïve object oriented approach using message-passing technique for process structure of operating system.

Internationally, the concept of software engineering was proposed as the result of two workshops (1968 in Germany, 1969 in Italy), and International Conference on Software Engineering (ICSEs) started in 1975. My first attendance to ICSE was the second one held in San Francisco 1976. Among various interesting sessions, I was deeply impressed in "programmers' workbench" presentations by ATT Bell Lab people. It made me realized that the concept of programming environment of integrated tools is very important to support software development process.

December 1976, Barry Boehm's paper "Software Engineering" was published in IEEE Transaction on Computing[6]. This paper gave a strong impact to Japanese software community including myself. It made me deepen my thought on socio-technical aspects of software development process. SIA/STC decided to invite Barry to Toyo and held a public seminar to promote the concept of software engineering.

Early 1979, I had a chance to know M.M.Lehman and invited him to have a seminar on his theory of Software Evolution Dynamics. It was also impressive and I was very much interested in his S- and E-type categorization of software. That became my motivation to organize a volunteer organization called Software Maintenance Study Group later in 1990.

5 Social Process of Technology Transfer

Members of SIA/STC gradually recognized the importance of mutual discussion and information exchange to strengthen technological power of software houses. So, we started domestic software engineering conference December 1980, this event has been continued until today as a major technical conference on software in Japan. We invited keynote speakers from abroad every year (for example, M.M.Lehman in 1982, Lee Osterweil 1987, etc.) to promote international technology transfer into domestic industry.

In August 1980, SRA introduced Berkeley Version of Unix on VAX11-780 computer. It was not a research tool, but we wanted to construct our own environment for application software development upon Unix. The first target application was a process-control system for a steel-mill factory in Shanghai, China. We developed a special test environment on Unix for the FORTRAN application program and succeeded in achieving surprising productivity and quality. It was almost the same as Barry Boehm's team at TRW achieved the next year[7].

We reported our experience in a number of technical meetings. Then almost all people in Japanese software industry (software houses, computer manufacturers, and user companies) visited SRA to see what the Unix is and how to use it. Also, our government (MITI) asked me to take the position of technical manager of next generation national joint project. I accepted the request and organized an experimental Unix-based tool integration project for the industry. Group of programmers from a number of software houses enjoyed to learn new style tool integration techniques in the project. Lee Osterweil's Toolpack project provided a guideline for us[8]. It was a kind of experiment of industry-wide technology transfer process. I called it as "Amusement Park in the Industry". That was the subject of my keynote presentation at ICSE-10 in Singapore 1988. I still remember beautiful follow-up talk given by Barry Boehm.

When the project approaching to the final stage, MITI asked us the idea for the next project. We prepared a proposal to do prototyping future software development office using PCs and workstations connected by local area network. MITI told us: "Let's use wide-area network rather than local area". So, the planning of the project called SIGMA started.

Our original idea was a grass root style project organization to discuss everything via e-mail and bbs over the nation-wide computer network. But the government people did not like such a bottom-up project structure. They wanted to set up a centralized network structure as a tool to control software industry.

After a series of strong debates, I gave up on the government finally, and walked out from SIGMA. It was around the end of 1985. Fortunately, almost same time, technical staff of Sony Corporation came to me asking help to produce new workstation. Our cooperative action was very quick. The output from first product line of Sony-NEWS workstations appeared in the market in the summer of 1986. Having enthusiastic support from young researchers/engineers around us who were interested in new software environment and mutual communication activities via network, we made a big commercial success and won the war against SIGMA. It was another evidence of the importance of social process in software technology development.

November 1986, on the way back from some conferences, I had an interview by

Unix Review Magazine in San Francisco and told my version of story about SIGMA[9]. This article was published in February 1987 Issue of the magazine. In April, when I was staying New York after ICSE-9, I got an international telephone call from Seoul Korea. It was Richard Stoleman calling. He said: "Your idea of grass-root movement is same as my philosophy. Let's talk about cooperation." We met in Tokyo a week later, I decided to make a small amount of financial donation to GNU project and send 2 programmers to Boston to help development activity. It was my first step into technology development and transfer process for Free Software Movement.

6 The Year 1984 and After

Beside George Owell's famous novel, 1984 was a special year to me.

Our SIA was ordered by MITI merging into larger industry association called JISA (Japan Information Service Association). Activity of our committee SIA/STC lost some degree of freedom. We did not like the situation, and finally decided to go out and establish an independent organization called SEA (Software Enjoiners Association) in December 1985.

In February 1984, M.M.Lehman organized the first ISPW (International Software Process Workshop) in London. I was invited as an observer to the event. It was the gateway to the world of software process for me. I enjoyed listening to exiting presentations and discussions. Among them, Barry Boehm's presentation of Spiral Model was interest to me. It looked like a good modification of Waterfall Model to software development process, but I was somehow frustrated maybe because it is still based upon product-oriented paradigm.

In the second ISPW in Los Angels, the program committee planed to set up a hierarchical structure of concept. It has a pyramid like structure: Process, Process Model, Methodology, Supporting Environment, and Tools. Everybody was requested when speaks to locate which level his/her remarks are. After listening this orientating explanation, my friend Bob Balzer raised "Objection!" He said: "I'm a tool developer. If my tool is strong enough, the whole process will be changed. So, this hierarchy is nonsense". I was surprised at first, then realized his remark is right. Tools are not support elements for process. Real tool should have a power to destroy existing process structure.

The third ISPW held with ICSE-9 PC meting up at a Rocky mountain ski resort. We had two interesting presentations: The idea of "Process Programming" by Lee Osterweil and "CMM" by Watts Humphrey. As program co-chairs, Balzer and I selected Osterweil as the keynoter of next year's ICSE-9 with M.M.Lehman as the counter speaker. It was an exciting opening session, and "Process" became the most important keyword for software community worldwide in 1990s.

October 1990, I hosted IWPW-6 in Hakodate, Japan. It was a memorable event for me. The result of workshop discussions was published as "Software Process Modeling Example Problem". Also, before the workshop, I organized a small meeting in Tokyo to discuss process issues in Japanese software industry inviting Watts Humphrey and some key members of SEA. Japan.

Just after ICSE-9, I organized an international industry-academia joint project called as "SDA (Software Designer's Associates)" with a great help by Bill Riddle. Some number of Japanese software-related companies including SRA became sup-

porters of the project, Using funds donated from these supporting companies, we held a series of working group discussions about software design process and the architecture of support environment quarterly in Japan, Hawaii, Australia, Canada, and Mainland USA. The project lasted about 6 years. The intermediate result of the discussion was presented at ICSE-10 in Singapore[10]. Also some number of prototype software was developed. But the real purpose of the project was to establish strong industry-academia human network over the Pacific Ocean. We have succeeded to get involved a number of researchers in Japanese computer science community into the field of software engineering.

7 Technology Transfer Activity in Asia

At ICSE-8 in London in 1985, I met Xichang Zhong from Beijing. Next August, he invited me to an international symposium held in Beijing. It was my first visit to China. On the way back home, I visited Xi'an and Shanghai, became acquainted with some key persons in Chinese software community. It was the starting point for me to organize a series of technical meetings in China.

From 1987 to 1990, SEA Japan organized China-Japan Software Symposium 3 times: 1987 in Shanghai, 1988 in Hangzhou, and 1990 in Osaka (Japan). Year 1998, we postponed the event because of Tiananmen accident. Instead, SEA invited a group of Chinese professors as guests of our domestic software symposium. Prof. Chi song Tang gave keynote about his XYZ system. I still remember it was a great talk.

From 1991 to 1995, we changed the contents of meeting by focusing on software tools and environment, and calling for participation form other Asian countries. Five International CASE Symposiums were held in Beijing (1991), Taian(1992), Wulumqi(1993), Kunming(1994), and Changsha(1995).

Around that time, general facilities for such meetings were rather poor in China. We had to bring OHP projectors, transparency sheets, and marker pens from Japan. But the situation was improved rapidly, and also technology trend of software engineering was changed somehow confronting the Internet.

So, we remodeled the scope of meeting to include more wide variety of discussion topics. ISFST (The International Symposium on Future Software Technology) was started from 1996 in cooperation with UNU/IIST (Software Institute of United Nations University) directed by Dines Bjørner. The event traveled Xian (1996), Xiamen (1997), Hangzhou (1998), Nanjing (1999), Guizhou (2000), Zhengzhou (2001), Wuhan (2002), and Xian again (2004). We found new friend at every place and enjoyed fresh discussions. Late Xichang Zhong helped us as a local coordinator. He passed way because of cancer a few years ago. I am very sorry. It was a big loss for us.

Beside these meetings in China, SEA organized a number of technology transfer workshops in Korea, Hong Kong, Seattle, and Kyoto as co-located events with other international symposium like ICSE or APSEC.

In 2005, we changed the style of meeting in China from, paper-presentation symposium to discussion-oriented workshops. The first IWFST was held in Shanghai as a pre-event for ICSE 2006, the workshops continued to Beijing (2006), Hangzhou (2007), Macau (2008), and in 2009 one moved to Munbai India.

We are now planning next IWFST in Kolkata, India. It seems better now to focus on concrete idea for future technology development needed in Asia. The next IWFST

will call for various ideas about future technology development projects. What is important for us to strengthen Asian software community as a whole? That is the issue to be discussed there.

8 Looking for Philosophical Foundation

Through discussions in ISPWs and SDA workshops, I had been rather frustrated in listening people's talks based upon software engineering discipline. They were talking process improvement towards "should-be" process models in their mind, or discussing ideal architecture of support environment to achieve higher productivity and better quality of products. Those talks were sitting upon factory paradigm of software production.

I prefer atelier-style software development model rather than factory model; because I was an artist before coming into programming world (I had been continuing artistic activity as a moonlight painter until now).

Once I prepared a process model, which has a flower-like image. A kind of knowledge base is located in the center of the model and various development activities are like flower pedals around it occurring in random sequence. I proposed that model to ISPW-2 but it was rejected. Program committee said it is little chaotic and too much context-free. Later, I knew metaphor of "Rhizome" through an interesting philosophical book written by Felix Guatari and Gilles Deleuze[11].

In botany, a rhizome is a horizontal stem of a plant that is usually found underground. A typical example of a network of bamboo root. Guatari/Deleuze used it as a metaphor of the image of human thoughts. They proposed that the concept of "Rhizome" could be used as a model of knowledge and a model of society.

As a model for culture, the rhizome resists the organizational structure of the root-tree system which charts causality along chronological lines and looks for the original source of "things" and looks towards the pinnacle or conclusion of those "things". A rhizome, on the other hand, ceaselessly established connections between semiotic chains, organizations of power, and circumstances relative to the arts, sciences, and social struggles. The rhizome presents history and culture as a map or wide array of attractions and influences with no specific origin or genesis, because a rhizome has no beginning or end; it is always in the middle, between things, inter-being, and intermezzo.

According to Guatari/Deleuze, the principle of "Rhizome Model" is as follows:

- Principles of connection and heterogeneity: any point of a rhizome can be Connected to anything other, and must be.

- Principle of multiplicity: only when the multiple is effectively treated as a substantive, "multiplicity" that it ceases to have any relation to the One

- Principle of signifying rupture: a rhizome may be broken, but it will start up again on one of its old lines, or on new lines

- Principle of cartography and decalcomania: a rhizome is not amenable to any structural or generative model; it is a "map and not a tracing"

I feel that these principles are very much suitable to construct a new model for software process. It will be better than Flower Model.

9 Confucianism and Software Engineering

At ICSE-10 (Singapore, 1988), Dewayne Perry and Gail Kaiser presented a paper about the models of software development environment[12]. They classified Software Development Environments into four sociological categories: Individual, Family, City, and State. It reminded me of one of the Confucian classics: "Great Learning":

"Those who wish to conquer the world would first bring order to their states. Those who wish to bring order to their states would first regulate their families. Those who wish to regulate their families would first cultivate their personal lives by rectifying their minds...."

We might feel that we are listening to a Watts Humphrey lecture about the hierarchy of PSP (Personal Software Process), TSP (Team Software Process), and CMM (Capability Maturity Model of organization).

In the Spring of 1990, I visited the headquarter of Eureka Software Factory project in Berlin, where I chanced to come across a report written by Professor Trygve Reenskaug of Norway, a description of his version of an object oriented method[13]. In the preface of that document, he indicated that, German philosopher Max Weber was the true "father" of Object Oriented methods.

Weber's concept of the "ideal" bureaucracy has key characteristics in common with O-O:

(1) Emphasis on form,
(2) Concept of hierarchy,
(3) Specialization of task,
(4) Specified sphere of competence, and
(5) Established norm of conduct.

This left my thoughts spinning. I made a mental time trip back in time two thousand years. A famous Confucian philosopher Hsun-tzu (BC298-238) was then writing a considered essay on rectification of names. He described therein a systematic hierarchy of concepts and their names (in OO terminology, relationship between meta-class, class, instance, etc.).

Confucius himself speculated about this issue a little differently:

"If names are not rectified, then language will not be in accord with truth. If language is not in accord with truth, then things cannot be accomplished. If things cannot be accomplished, then rites and music will not flourish. If rites and music do not flourish, then the punishment will not be just. If punishment is not just, then the people will not be able to know how to move their hand or foot. Therefore, the superior man will give only names that can be described in speech, and say only what can be carried out in practice."

What might this have to do with software project management? Well, apply a simple conversion table as follows:

Name	Concepts
Language	Process Model
Things	Projects
Rites	Methods
Music	Tools/Environment
Punishment	Management
People	Software Developers

Hand/Foot Development Activities

Certainly there are levels of concerns about team efforts in software development, and large-scale projects do begin to look like cities, as Perry and Kaiser indicated. But, what kind of cities are they?

In the tradition of our agricultural civilization, a city exists mainly for management. Typical examples are the walled city-states in ancient China and Europe. They all have concentric urban planning around the royal palace, reflecting a hierarchical structure of management.

Such cities look beautiful, but are easy to corrupt from inside. Sure enough, there are much historical evidences of political corruption of Confucian city-style (or bureaucratic) management. Further, these hierarchical social structures tend to lack creative power of their own. Innovations were always brought from outside, typically by wandering philosopher like Confucius himself.

Cities in today's industrial society (a successor to agricultural society and now being transformed into an information society), have functions above and beyond management. Walls surrounding them have almost disappeared (we can only find a few remaining immigration barriers, similar to firewalls on the Internet). Modern cities are essentially "free" places for exchanging information, trading materials, and mixing cultures.

Historically speaking, we can trace the origins of such "free" cities to the 7th and 8th century Arab world. Commercial civilization was the mother of the new cities. A few centuries later, Khbilai Khan constructed great empire covering almost all parts of Eurasia continent by connecting major cities with an "internet" of highways. Those free cities had facilities as network nodes for transportation of people, commodities and also information.

I believe such a "free" city is the appropriate symbolic image of a large-scale software project. Software project exist to produce some software system. But the project does not exist only for the sake of management of production. There are other important outputs other than software products. Large-scale software project is also a place for cultural and informational exchange for a variety of people coming and going. Through this process, people grow up and new concepts or technologies will be evolving out.

10 Immaterial Labor

A few years ago, one of my colleagues reported that he found an interesting article in a web site, which has a rich collection of philosophical literature.

The article was:

Maurizio Lazzarato: "Immaterial Labour" at http://www.generation-online.org/c/

This information was circulated in the circle of engineers around me, and everybody was strongly interested to this paper[14].

The author, Maurizio Lazzarato, is an Italian sociologist working in Paris, France. In this article he explores the hegemonic form of labor under today's post-Fordism capital society and sketches a project, which entails an analysis of the forms of subjectivity that inform and are informed by this immaterial labor.

Immaterial labor is composed of two different parts.

As regards the 'informational content' of the commodity, it refers directly to the

changes taking place in workers' labor processes in big companies in the industrial and tertiary sectors, where the skills involved in direct labor are increasingly skills involving cybernetics and computer control (and horizontal and vertical communication in the organization they belongs).

As regards the activity that produces the 'cultural content' of the commodity, immaterial labor involves a series of activities that are not normally recognized as work – in other words, the kinds of activities involved in defining and fixing cultural and artistic standards, fashions, tastes, consumer norms, and more strategically, public opinion.

Lazzarato extends the original sociological notion of the expansion of work to waged and unwaged spaces and the resulting "social factory" in which virtually every activity we engage in becomes productive.

According to the definition above, software is an immaterial product, and all our activities in development, maintenance, use, and evolution of software should be considered as immaterial labor. An important characteristic of the immaterial product is that it is not destroyed by consumption. Rather, a variety of new idea to improve or enhance functionality of the product will emerge through the process of use and maintenance. Some of them are called as deliberative development.

The barrier between developer and user of software disappears. It is not meaningful to think about development process only. We should consider the process of development, maintenance, and evolution of software product altogether. This is the real implication of M.M.Lehman's evolution dynamics theory.

History of our software development environment has been followed the line of society change analyzed by Lazzarato. Environments in early generation were discipline-oriented relying on waterfall-style process model. Then, communication-based new generation environments using CSCW technology have been came into the scene. But they are still control-oriented. We need to have more flexible development environment to support immaterial software process. It will be co-constructive to allow us on-the-fly modification of process and process model.

Barry Boehm's proposal of "Incremental Commitment Model (ICM)" is on the stream of this technological trend. He wrote in the introduction of his paper[15] as follows:

Many projects have difficulties in integrating their hardware, software, and human factors aspects. This is basically due to differences between these aspects with respect to their underlying economics, evolution patterns, product subsetability (ability to deliver usable partial initial operational capabilities), user tailorability, adaptability, underlying science, and testing considerations

The phenomenon described here can be seen all over the world, and it is because of immaterial nature of software systems which we are constructing, using, and maintaining. ICM is considered as one of the steps towards future innovation of our process technology.

11 Concluding Remark: Importance of Polyphony

In the last section of his essay, Lazaretto introduced the two names of philosophers: George Simmer and Mikhail Bah tin. He wrote as follows:

The works of Simmer and Yachting, conceived in a time when immaterial produc-

tion had just begun to become "productive", present us with two completely different ways of posing the relationship between immaterial labor and society.

The first, Simmer's, remained completely invested in the division between manual labor and intellectual labor and give us a theory of the creativity of intellectual labor. The second, Brahmin's, in refusing to accept the capitalist division of labor as a given, elaborate a theory of social creativity. Simmer, in effect, explains the function of "fashion" by means of the phenomenon of imitation or distinction as regulated and commanded by class relationships. .... Yachting, on the contrary, defines immaterial labor as the superseding of the division between "material labor and intellectual labor" and demonstrates how creativity is a social process.

Mikhail Yachting analyzed the work of Great Russian novelist Fyodor Dostoevsky and found the principle of "Polyphony and Unfinalizability". In Dostoevsky's novel, a number of characters make discussions about some profound "truth" of life. Dostoevsky treats voices of these characters complete equally. They keep going into endless dialogue and the discussion never comes to final conclusion even the story ends at some point[16].

Bakhtin's conception of unfinalizability respects the possibility that a person can change, and that a person is never fully revealed or fully known in the world. Bakhtin found in Dostoevsky's work a true representation of "polyphony" (many voices). Each character in Dostoevsky's novel represents a voice that speaks for an individual self, distinct from others. This idea of polyphony is related to the concepts of unfinalizability and self-and-others, since it is the unfinalizability of individuals that creates true polyphony.

Bakhtin criticized the assumption that, if two people disagree, at least one of them must be in error. For him, truth is not a statement, a sentence or a phrase. Instead, truth is a number of mutually addressed, albeit contradictory and logically inconsistent, statements. Truth needs a multitude of carrying voices. It cannot be held within a single mind, it also cannot be expressed by "a single mouth". The polyphonic truth requires many simultaneous voices. Bakhtin did not mean to say that many voices carry partial truths that complement each other. A number of different voices do not make the truth if simply "averaged" or "synthesized". It is the fact of mutual addressability, of engagement, and of commitment to the context of a real-life event, that distinguishes truth from untruth.

We, software engineers, have been discussing software process issues; process models, process improvement, process support environment, etc., etc. Our dialogue will continue forever. We should realize the unfinalizability of discussion and notice the importance of polyphony.

Within SEA, we had a series of working group discussions starting at IWFST in Hangzhou (China). Then at Software Symposium in Takamatsu (Japan) and other domestic discussion forums. The position statement presented at the SIGSOFT SoFER workshop last November is the unfinalized summary of our discussion so far[17].

References

[1] Klee P. Das bildnerische Denken. Benno Schwabe & Co. Verlag, 1990 (New Edition).
[2] Jeenel J. P and Co., 1958rogramming for Digital Computers. McGraw Hill, 1959.
[3] Bohm C, Jacopini G. Flow Diagrams, Turing Machines and Languages with only Two Formation Rules. CACM, 1966, 9(5).

[4] Baker FT, Mills H. "Chief Programmed Team" in "Classics in Software Engineering" Classic. Yourdon, 1979.
[5] Royce W. Managing the Development of Large Software Systems. Proc. of ICSE-9. 1987.
[6] Boehm B. Software Engineering. Trans. on Computers, December 1976.
[7] Boehm B, *et al*. TRW Software Productivity System. Proc. of 6th ICSE.
[8] Osterweil L. Toolpack: An Experimental Software Development environment Research Project. IEEE Trans. on Software Engineering, 1983, 9(6).
[9] Wright S. Interview with Kouichi Kishida. UNIX Review Magazine, February 1987.
[10] Kishida K, *et al*. SDA: A Novel Approach to Software Environment and Design. Proc. of ICSE 1988.
[11] Guatari F, Deleuze G. Thousand Plateaus. University of Minnesota Press, 1987.
[12] Perry DE, Kaiser Ge. Models of Software Development Environment. Proc. of ICSE 1988.
[13] Reenskaug T. The Design and Description of Complex, Object-Oriented Systems. Rapport/Senter for Indusyrrforskning, 89-272-1, ISBN 820411-0193.-7.
[14] Lazzarato M. Immaterial Labour. In: Radical Thought in Italy: A Potential Politics. Minneapolis: University Press. pp. 133–147.
[15] Boehm B, Lane JA. Using the Incremental Commitment Model to Integrate System Acquisition, System Engineering, and Software Engineering. http://sunset.usc.edu/csse/ TECHRPTS/2007/usc-csse-2007-715/usc-csse-2007-715.pdf
[16] Bakhtin M. Problems of Dostevsky's Poetics. Minneapolis: University Press, 1984.
[17] Ye Y, Kishida K, Nakakoji K, Yamamoto Y. Through the Looking Glass of Immaterial Labor. Position Paper for ACM-SIGSOFT Workshop on Future of Software Engineering Research, November 2010.