

Lyee Internet Information

## **New Software Methodology in the 21<sup>st</sup> century**

Interview

with

Gregor v. Bochmann, professor at the University of Ottawa in Canada

and

Fumio Negoro, president of the Institute of Computer Based Software Methodology and Technology

Date: Friday, May 11, 2001

Place: Head office of the Institute of Computer Based Software Methodology and Technology

Moderator: Hamid Fujita, professor of the Iwate Prefectural University

### Biography of Professor Bochmann

1941 Born in Niebull, West Germany

1971 Awarded Ph.D in theoretical physics from McGill University in Montreal

1972 Appointed assistant professor at the Université de Montreal, Département d'informatique et de recherche operationnelle

1983 Became full professor at the same university

1998 Became full professor at the University of Ottawa

Prof. Hamid: Now, I would like Mr. Negoro and Prof. Bochmann to talk about new software methodology.

Negoro: Let me begin by talking about the background of the methodology. What kind of background did give rise to UML, for example? What kind of background did Lyee have? We did not much discuss these issues. That hinders us from thinking when we bring the philosophy into software science. How to form the methodology is a philosophical issue, while how to utilize the methodology is not.

Prof. Bochmann: However, we learn a lot from experience. In the

software world, we come across what others have made, which will eventually become one form of guideline. By looking at what they can do and how they do, we make certain evaluation. At the same time, we try to prove that it is actually useful.

\*\*\* NOTE this paragraph makes no sense

Prof. Hamid: UML was created from the experience in Object-Oriented Analysis and design, which came out in the 1990s as an advanced form of OMT (Object Management Tool) that came out in the 1980s. Unfortunately, OMT itself was not successful but contained new ideas. Followed by this, various new ideas were born. Though they were very important to the point that they defined innovative phenomena, it was difficult to use them because there was not a single standardized method.

Then, CORBA came out and standardization has become a kind of trend. At present, various E-commerce activities are conducted everywhere and people are becoming aware that they cannot continue without having standardization. Under such circumstances, Grady Booch \*\*\* Note: usually one says that UML was created by three people \*\*\* created UML.

I want to emphasize one thing here. That is, UML and Lyee have similarities in a sense. I heard that Mr. Negoro has led more than 80 projects in 30 years and I assume he received various feedback. This feedback must have been derived from problems of software itself at the time. I think this is why Mr. Negoro wanted to create new ideas and software.

UML is based on the idea of Object-Orientation and there are many approaches such as OMT and CORBA \*\*\* Note: COLBA does not want to solve the same problem as UML, they are quite different in nature \*\*\*. Everyone was seeking something new in them. When Booch tried to tackle problems with innovative ideas by analyzing many methodologies in the past, eliminating their problems, and utilizing only good points of them, UML was born.

I think that UML has similarities with Lyee in this point.

It is difficult to say that UML itself is successful. Anyhow, UML was created with such ideas at this backdrop. Even now, evaluation and research of UML has still being conducted and people are trying to discover new ideas.

Prof. Bochmann: I think UML is important in another point. UML is not a process but indicates notation. The notation is used in a process. Expressing a process is very important. It also collects different things, processes them, determines which notation should be used in which phase. If a generation tool should be used, what kind of tool is necessary for that, and so forth.

There is another trend. In general, people do not want to learn very many languages because it is a lot of work. I, myself, used formal methods for communications. Though I am good at languages, I used the standardized language established by ITU. Many people understand from their experience that they cannot use very many languages.

Advantages of UML (Unified Modeling Language) are that it is standardized and there are only a few dialects. It is like everyone is using a common language. Therefore, it is easier to learn.

It is possible to apply the ideas of UML to Lyee by embedding elements of UML notations in Lyee.

Negoro: First of all, I concentrated on analyzing source codes of various programs for a while. That was about 30 years ago. I was lucky in a sense because at that time there were no methodologies prevailing in the world. But there was much software and I made thorough analyses on those source codes which did not have methodological background. Those source codes mirrored a reflection of a natural way of human thinking. Needless to say, there were neither global standards nor rules to make software those days. I was lucky to have analyzed the source codes of the day. As a result, I could say that I know well what source codes are. Recently, various methodologies as well as standardized rules have been created, thereby making source codes. Nevertheless, the source codes in 30 years ago are almost the same as those of the present in terms of substance. A programming language may be different, but even this does not make a big difference, because source codes expressed in a certain programming language do not show a significant difference when they are re-expressed in a machine language. I believe this notion is very important. Turning to an issue of productivity, now it is a lot lower than those days. In this sense, I think software has

not much evolved, rather it is in a stalemate. In actuality, whether we can stand for this notion or not may be also important, because that affects how to understand software problems. Of course, I think UML is also a wonderful idea, but from this perspective, I could not see any progress in its fundamental level. I think what I said should go through verification. However, once we agree with a view point that software is in a stalemate, we cannot discuss it furthermore without examining why this has happened.

Prof. Bochmann: It is true that we cannot say that efficiency of software has improved in the same way as for hardware. However, there are improvements in the way we produce software and, in some cases, in the software quality that is produced. A few years ago, I listened to a presentation at an international conference given by Prof. Hoare who is very famous in software. He said in his presentation something like software world is making progress. What he meant was that what we want cannot be realized like miracle but it is slowly making progress by taking gradual steps. I agree with this idea.

I also listened to another presentation on formal methods. The speaker of this presentation also said that software is making progress.

Negoro: I feel that there is a gap between academicians and practitioners. This was already pointed out in a paper written by an American around 1960. He presented some data on the number of conferences, of participants, of conference hours, and of research hours, and how the situation of software has been changed by those. He concluded there was no big change, anticipating such a gap between the academia and the industry will not be narrowed. It is amazing that such a paper was written in America 30 \*\*\* Note: 40 ??? years ago.

When Object-Oriented programming appeared in the 1980s, I thought how thoughtless this methodology was. I clearly remember how I felt. Its structure was not much different from the module structure proposed by Dijkstra in the 1950s. Both of them keep the same weak points.

Before Object Oriented came out, a French mathematician, Warnier, advocated a programming method. This was not a language, but a development methodology so that it was one of the early-days'

methodologies. Whenever new methodologies were publicized, I make it a custom to use them for system development. I found that Warning's programming method was remarkable so I wonder why this has not been well used since then. One of the reasons for this is that IBM took an initiative in the software field. I think we are not in an environment that we can have a free and frank discussion on software in its true sense.

Prof. Bochmann: In the last 30 years, many interesting and outstanding methodologies were born. However, the improvement is not remarkable at all and almost nothing is changed. This means that the improvement itself in this field is extremely difficult. People can achieve what is easy. The methodology Mr. Negoro is proposing now was born in such difficult period. What is the ground for you to say that your methodology is superior to others that were also born in the field where making progress is difficult?

Negoro: I do not mean to maintain that Lyee is one of the very best. All I want to say is that Lyee is a method to express the intention. In order to do so, it is necessary to define what the intention is, but the intention I mean differs from that in a conventional manner. Lyee's world view is different from conventional ones. For example, a concept of class is needed both for Object Oriented and UML. In other words, a set theory is used for them. Why we cannot have a perfect axiomatic system may be explained by the fact that the axiomatic system connotes the same conceptual problem as a class does. I think it is quite unreasonable to incorporate a logically undue concept into software. As soon as you put forward a concept of the class, you are almost declaring that you would not pursue a deterministic manner any more. This is one of the reasons why I was sort of surprised to see first Object Oriented.

Prof. Bochmann: I do not understand why you say that the concept of class is a problem.

Negoro: A concept of the class is deeply associated with our way of thinking or feeling. In other words, we cannot define it. A concept of the class is varied among individuals.

Prof. Bochmann: You mentioned the intention earlier. I think people have different intentions. So it seems unnatural to say that the class varies from person to person.

Negoro: We have to define it by using another concept but the class.

Prof. Bochmann: Suppose there is a certain kind of formalization to get fundamental understanding for creating software, and that is mathematics. In that case, is it a problem to use mathematics to understand software?

Negoro: It should not be easy to express a concept of the class with mathematics. How to handle an issue of the class can be analogous to how to grasp the intention.

Prof. Bochmann: Set aside a simple system. When it comes to a complicated system, it is impossible to know if there are different objects in it without using the concept of the class.

Negoro: I am just talking about my own stance. I would rather not criticize the status quo directly. What I want to emphasize here is that we have to be released from a way of distinguishing an easy intention from a difficult one when we tackle the intention. We should search for a way of grasping the intention regardless of being difficult and easy. I believe that this will also provide a key to solve a problem of the class. You showed us a sample in which you are supposed to first define the class and make it into source codes gradually. This represents a human way of cognizing things, and naturally requires a complicated procedure. There are various classes which appear when they come to source codes, and those classes are replaced by entities and activities. Thus, the whole picture looks complicated in the end. I wonder if UML provides a method to put brakes on its complexity.

Prof. Bochmann: UML does not have a mechanism to avoid complexity, of course. UML is merely a notation, not a system. But

UML has notations to introduce abstraction which may be used to hide complexity.

Prof. Hamid: I think it is important to focus on how we conceive new software in the new century. In other words, ideal software in the 21<sup>st</sup> century should be flexible and intelligent so that it can adapt to various changes of the environment. Looking back to the history in software, Object-Oriented is often mentioned but it does not have sufficient power to handle such challenges. At this backdrop, I would like you to talk about the positioning of Lyee as new generation software development methodology in the new century.

Prof. Bochmann: I am not in the position to answer that question since I do not fully understand Lyee. I guess you have opinions about that rather than myself. Everyone is interested in what you just said and how to tackle such issues. There are many ways to do it. When we choose one method, it is necessary to measure its productivity and efficiency. The easiest way is to develop one project with multiple methodologies, and then to compare the results. But there is a budget issue so we cannot make a simple comparison. In other words, it is like the situation where we have to compare an apple and an orange though they are substantially different.

Negoro: But we have to overcome the problem and we need to be fully aware of the importance of theory for this purpose. It is important to realize that theory should exist for comparison.

Prof. Bochmann: How do you measure it?

Negoro: Therefore, we have to construct a theory and use it as measurement. Taking an example of UML, it cannot be compared with others until it is constructed with a theory.

Prof. Bochmann: I do not think you can compare things without the measurement having common scales. There should be a parameter for calculation. For example, there is a complicated system. The parameter

for this system is development period, error ratio, and how easy it is to make changes in programs, and the measurement is conducted based on these.

Negoro: In my view, that can be measurement in a narrow sense. There is one more thing I want to say about it. Although measurement is an independent property of the theory, the theory itself is an open and public entity. To compare the methods of software which does not have common measurement, theories become very important. UML has a concept of the class and that could be measurement of UML, but because it is totally depending on arbitrary individual judgement, it cannot be a true measurement.

Prof. Bochmann: What would be the measurement then? Do you mean how many classes there are?

Negoro: It is considered to be measurement to assess this method whether UML clearly defines the class or not. With the Lyee method, the predicate structure is one measurement. If we try to find out common items in the measurement, first that should be defined with C++ and next be defined with the predicate structure. It is then that we can compare these two. The class cannot be described by source codes. As previously mentioned, neither UML nor the Object-Oriented Architecture are not founded by theories. That means they do not have measurement. Productivity and maintainability that you mentioned cannot be measurement.

Prof. Bochmann: What would it be, then?

Negoro: For example, it is already proven that Lyee's productivity is five times higher than that of conventional methods, but if all the software in the world were developed with Lyee, the productivity would not be measurement any more. In this sense, productivity and maintainability cannot serve as measurement in its strict sense. Turning to the theory, it provides definition of measurement. From this point of view, most of software methodologies can never be theories, but products of software

politics.

Prof. Bochmann: As I listened to your talk about the theory, I remembered what I heard in the sessions in these last few days. With Lyee, a system can be developed with seven rules whereas conventional methods require many more rules. From what I heard, I had the impression that a system can be easily developed with this method.

Like an issue of languages, a system with less complexity and a simple structure always seems better than a complicated one. This kind of thinking has become one measurement.

For example, the language named PL/I came out, but later disappeared because of its complexity. Then, the language named Pascal came out. This does not cover much but is still used because of its simplicity. C language has a relatively simple structure. That is why C is widely used. I think this is one standard. When many people are involved, it is necessary to have a certain standard to draw a line. Though Mr. Negoro claims that there is no measurement, I received the impression that the simple set of rules of Lyee are kind of measurement which you mentioned in your talk about Lyee. You may say that Lyee has an advantage that it develops a system with simple rules.

Negoro: In my opinion, if the methodology has the theory, it is then qualified to be compared. I think this statement is applicable not only to software, but also to all other academic fields. Let me talk about how to compare different theories, then. Lyee has a concept. The Object-Oriented Architecture has a concept, too. If a concept is expressed with source codes, we can compare the theories in source codes. Lyee's concept can be represented with source codes, whereas a concept of the counterpart cannot. This is the point I want to make. In this sense, I think that the Object-Oriented Architecture does not possess a measurement scale. As for programming languages, approximately 70% of software developed in the world uses COBOL, which has about 3,000 variation. Because it adopts dialects to its own language, the number of variation amounts to as many as 3,000, making up 70% of the world share.

As for PL/1, Toyota, a Japanese automotive corporation, uses it. I would like to stress here that people have different opinions to evaluate

programming languages. I suppose people in Toyota decided to use it, believing that there are not any other superior languages to PL/1. I personally think that this language is one of the superb. But partially because mainframe manufacturers did not promote this language, it was not well spread out. I also want to mention ALGOL, another superb language which was not well used for the same reason. As these suggest, programming languages are placed in a precarious position. This includes problems which cannot be overcome only by technology. Lyee's program structure does not depend on programming languages. We have developed systems with C, COBOL, assembler, and RPG. Of course, PL/1 was also used for a system development. Through these experiences, I am tempted to conclude that a machine language would be the most suitable for programs. Oh, I should not miss JAVA. The programs expressed in JAVA have redundancy, causing lots of problems. Thus, I suppose that we have unique data which leads to different evaluation of programming languages from that in general.

Prof. Bochmann: That is because it is unnecessary to read code itself since it is automatically generated, isn't it?

Prof. Hamid: In any case, the fundamental issue is that how to conceive new software development in the 21<sup>st</sup> century and how to solve problems associated with it. In other words, it is an issue of how to measure the best way.

Prof. Bochmann: Yes, I think everyone is thinking about it but the point is how we can do it. The important thing is that software is developed to achieve a certain objective in any system. There are cases where quality is strongly pursued, cost effectiveness is focused, or achieving deadline is the priority depending on circumstances. This is the reality.

Under such a situation, you cannot say this one is the best, it depends on the context. As I mentioned earlier, developing one project using multiple methods is the effective way to clearly identify the best method. If we could compare different methods used in different companies, it is the most effective way.

As Mr. Negoro mentioned earlier, when we hear that one system was developed in such a short period, we do not have a clear idea about it. But if this can be compared with the development using other methods, it is the simplest way to see its validity.

Negoro: Would you give some such a theme to us? Unless it is too big, say, a system with 1000 man month may do.

Prof. Bochmann: I currently do not have any project of that scale unless I ask some company to collaborate with us.

Prof. Hamid: Mr. Negoro said that because he wants you to believe that Lyee is very productive in terms of development time and cost, and he is not just saying it. Lyee has a universal structure. Being universal means that it can be applied to any software. This is a priority matter for universality. Since it has this kind of idea, it eliminates others rather than combining with others. Do you think we need this universality? And what do you think of this idea?

Negoro: I believe that human depth psychology is constructed in an axiomatic manner so that humans will be sure to pursue a universal method to express it.

Prof. Bochmann: However, people need different methods and representations to think about different aspects of things. For example, a different notation is needed to do different things, therefore UML provides different notations.

Negoro: But those are just referring to phenomena. We should not underestimate a concept of universality to resolve these issues. Usually, we take the OS for granted and discuss a lot the quality of the OS as well as that of application systems. Right now, I am thinking that if we could make a role of the OS lighter, we can have a deeper discussion on quality of software. Just thinking of such possibility needs change of our way of thinking.

Software has already shown some breakup even though it has only 50-year

history. We should face this. It is time for us to consider the quality of software even more important than before. Software science should undergo some changes as well.

Prof. Hamid: We have already spent one year for preparation for the Lyee international network project, meeting professors from all over the world. The main objective is to introduce Lyee methodology into the academic world. Lyee is a new style born from actual system development or business world.

We have been trying to start the project on how to evaluate Lyee from academic point of view and how to conduct research of Lyee. What do you think of this?

Prof. Bochmann: While I was listening to the explanation of Lyee, I naturally felt that I want to understand this method as a scientist. In order to do that, I compare Lyee with what I know already. Of course, people may look at Lyee from different perspectives, Lyee may look different in a different context, and different persons have different opinions when comparing Lyee with other technologies.

Like we have been saying this in this interview, the point is how do we evaluate or measure this and with which standard or parameter.

Prof. Hamid: What do you think of the point that Lyee may be able to solve the problems that software science is having difficulty to solve? For example, software science is trying to find a solution for formal models and formal methods. But I do not think it is implemented effectively.

Prof. Bochmann: Formalization is meaningful because formality is needed for tools to operate. However, formalization of the entire processes of software has not been achieved yet.

As for languages, I think that each language has universality in a sense. What I am interested in about Lyee is its scope of application.

I want to add one thing to what I said earlier. I think it is very true that formalization has not been achieved in that even with the notation, the meaning cannot be defined yet. As for UML, its notation is defined but there is ambiguity about the meaning and some misinterpretation arises

there, which will take different patterns. There is no notation that can define formal meaning. I think it is very true that formalization in that sense is needed.

I have been involved in many developments for communication protocol and used various languages. I think SDL and LOTUS are quite simple languages and were successful in formalization at a high level but were not perfect. I agree with what Prof. Hamid said about problems of formalization in that sense.

I have a question about Lyee. When you first visited Ottawa and gave the presentation, I could not understand Lyee at all. But I came to understand a lot in these last few days as I saw concrete examples and found understandable explanation in papers. Nonetheless, there is still a lot of ambiguity. That is because the language is different from what I am familiar with. So if you could use the language that I am familiar with to explain the content, my understanding will improve quickly.

For example, both SDL and UML have input/output messages and they probably refer to the same thing. You explained about the base structure. Which part in it is equivalent to what I call dynamic behavior? If I can understand such, I will have clearer understanding of Lyee.

Negoro: You mentioned a dynamic behavior, but Lyee does not have a concept to define it. This is because it has a unique program structure called a predicate structure. This indicates a totally different characteristic from that of conventional methods.

Prof. Bochmann: I think there is a part that is equivalent to a default behavior. What do you think of it?

Negoro: That is in effect a process route diagram. The predicate structure is considered to belong to a default behavior in this sense.

Prof. Bochmann: In my opinion, that should be defined as input/output and what kind of dynamic behavior follows next should be considered.

Negoro: Input operations and output operations are not regarded as a dynamic behavior. These are categorized into asynchronous structure

with Lyee. They are contained in action vectors.

Prof. Bochmann: From my point of view, Lyee has a very special language using various terms. When Lyee is used with other languages and methods, it will need common terms for exchanging common messages, for example. Therefore, I think the common formalization will be necessary for that purpose.

Maybe, you could give the detailed explanation such as theorem and axiom later in the presentation. At the moment, I do not understand that part at all. If you can use terms used in other axiomatic systems or things that can be shared in terms of the meaning when explaining your ideas, it is helpful in understanding Lyee and creating system environment.

Of course, we should be able to understand the meaning of the structure that you created, which is very important. It is also important to understand the formality behind that idea.

Negoro: I am fully aware that how to explain a concept of Lyee to others reflects my personality. When I talk to you this time, I learn a lot from you. If you can join the collaboration project and work with us, that will be quite productive. This will also help contribute to society.