Lyee Internet Information

Interview with Dr. Hamido Fujita

Curriculum vitae

1955 Born in Canada

1979 Graduated from the Faculty of Engineering, Department of Electrical Engineering, University of Manchester, U.K.

1982 Studied as Research Associate at the universities of Tohoku and Tokyo in Japan

1988 Awarded doctorate in engineering from Information Engineering of Tohoku University

1989 Studied the object-oriented approach as Associate Professor at the Research Center for Advanced Scientific Technology (RCAST), University of Tokyo

1991 Appointed Associate Professor at the Faculty of Information Engineering of University of Montreal

1994 Appointed Professor at Tohoku University of Art and Design, Japan

1998 Appointed Professor of the Faculty of Software and Information Science at Iwate Prefectural University; appointed Chairman of the Information System Division of the Media Center at Iwate Prefectural University

2000 Appointed Professor at the Graduate School of Iwate Prefectural University

Q： What is your main research subject?

A: I have mainly studied software engineering. Particularly “evolve software” has long been a primary focus of my research.
Namely, “intelligent systems” such as artificial intelligence and the object-oriented approach. In order to solve software problems and make smarter evolving specifications, I also studied more intelligent systems which were supposed to be established through adding the object-oriented approach and the artificial intelligence one to existing systems. A reflective language is one of its examples. When changing the system and adding new requirements to an existing system, we have to examine its related parts and its affected modules, and dynamically change those as well. A reflective language is a method of completely understanding the system and advise not to alter it if the effects generated by that change are overly large, and not consistent with pre-changed systems. Such language helps to change the system in order to dynamically take into account the new system adaptation.

Q: Would you give us a simpler explanation on that?

A: It is a support methodology for dynamic development. It is a tool to generate protocol specifications and system specifications, for example. One of the representative reflective languages is RMODEL or Reflective Modeling Language, and is used for system development in the object-oriented manner. When a user wants to change some parts of a system, it will reflect almost automatically how the changed parts will affect other parts, and how the other parts will be changed. So within the language there some invariant that helps to approve the system after its specifications are changed for specific adaptation.

Q: How did you find out about Lyee?

A: It was in the Parallel Computing Technologies (PACT99) held in Russia, in September, 1999 when I listened to the lecture of Mr.

Negoro, president of the Institute of Computer-based Software Development and Methodology. I also had a chance to talk with him at the international conference, held in Appi-kogen in October of the same year.

Q: What did you think?

A: I could not understand the theory of Lyee, but as a researcher, I felt there was something in it – something very interesting, because I could totally agree with Mr. Negoro's idea of generating a program directly from a user's intentions, which is the basic concept of Lyee. It is extremely difficult to establish an agreement between what a user wants and the specifications and programs produced. It is always uncertain whether the programs agree to the user's request or not. Each step for development requires an enormous amount of labor and the operations are complicated. They say that Lyee method does not need such annoying steps, so I became very interested in it. I have never heard of such a method in any academic conference. Is it truly possible or not? I found it interesting.

Q: Did your suspicions cause you to reject Lyee?

A: Suspicious feelings are put on the back burner. I wondered how the theory is realized? Through discussions and questions, I want to judge whether it is right or wrong, or true or false. That way is more constructive than to ignore it, and helps to deepen the understanding of new things.

Q: Which part of Lyee is most interesting to you?

A: Mr. Negoro claims that, with Lyee there is no need for testing. But, in software engineering, testing is fundamental and mandatory. I was wondering why he could say so? He has changed the way he explains Lyee now, but when I met him first, his explanation did not meet what scholars in the software engineering wanted to hear, rather he took a philosophical or physic explanation. His wording was totally different from ours which is commonly accepted by academia. His explanation did not sound convincing. I thought

the explanation should be adjusted to be more user-friendly. There are in fact many ill-explained parts in it. I have encountered many of those cases. But I did not take a negative stance toward them because I did not understand it clearly, rather I would take a positive approach to look into it because it looks intriguing. Incidentally, Mr. Negoro started to write papers for academic conferences, so he asked me to give some advice on writing. I was happy to help him writing, and at the same time I was given opportunities to learn about Lyee. Since then, we have maintained a give-and-take relationship: once or twice a month, we had researcher-level of discussions and papers on Lyee.

Q: Any good results?

A: Of course. Mr. Negoro's paper was accepted by the IS2000, 2000 International Conference on Information Society in the 21st Century. He will make a presentation about it in November, in Aizu.

Q: How important is it?

A: Very much so, because numerous referees and scholars usually take three months to read all papers carefully and evaluate them in terms of originality and benefits for international conferences. It is worth an Olympic gold medal. Also, only about 30% of papers are accepted. You have to jump over a very high hurdle. The fact that Mr. Negoro's paper was accepted indicates that Lyee is well presented as a new research idea in a paper so that academia can accept it.

Q: What happened after the paper was accepted?

A: It can be referred to by anybody. The conference book including his paper will be distributed to universities and public libraries so that anybody can read it anywhere, anytime. Before this happened, Lyee was recognised just by its patent, but now it is given another form of recognition by academia. The presence of Lyee as a new software methodology for development is officially accepted by the academic community.

Q: Are there any other interesting stories related to the academia?

A: Mr. Negoro handed out another paper to the ICSE, the International Conference on Software Engineering. This conference is one of the top conferences in the world. If his paper is accepted here as well, we will be then allowed to put higher expectation on Lyee's advancement in the academic field. Let me also tell you about the tour to Canada. In June, Mr. Negoro with his staff members went to Canada to visit the University of Ottawa and other universities (cf. Lyee Internet Information Vol. 7) and succeeded in bringing their attention to Lyee. Unlike Japanese, westerners do not pay flattering compliments to Mr. Negoro and his team unless they find it interesting and meaningful. They are quite straightforward. If not, they will never again give us the time of day. If they think that Lyee theory is groundless and boring, they will never give us a second chance. Instead, they said, "It was very interesting, but something is missing. We want to know more details about it." Then, we decided to go back to Canada again in October. This time, we visited the Université de Montréal, the Université de Québec à Montréal, the Université Laval, Ottawa University, McMaster University, and other organizations. We had a nice fruitful discussion, as well as feedback from the professors in those universities. Some of these professors expressed a great willingness to learn more about Lyee, by coming to Tokyo and study more about Lyee. The schedule of this visit has already been established.

Q: We have heard that you are going to make a research network worldwide. Tell us about the outline of this academic circle and the way of its development.

A: The participating professors and researchers will number more than 40, who will be coming from ten countries including the United States, Canada, Japan, Sweden, Italy, Denmark, Australia, and England. Each member of the club is expected to choose its own theme on Lyee and compare it with other methods, and deepen it. That is the image that I have now about it. Graduate Students will also join the club. Once every six months, each member will

make a presentation about their themes and hold open discussions. I am thinking of workshop-type of meetings.
There are a lot of details here but you will hear about them soon.

Q: When will you start to work on that?

A: Actually we have already begun to prepare for it. We will have finished a preliminary survey by the end of March 2001, and will allocate roles to each member of the circle and decide a detailed schedule. Around that time, we will hold a session to make an interim report. After April, we will get ready to launch an official activity or implementation of the plan.

Q: What kind of research will they conduct?

A: Briefly speaking, it is a study of Lyee and scientific survey of other related methods. For example, they might investigate what the scenario function in Lyee really means, and what can be achieved with Lyee. They will explore the possibility of Lyee. For example, can Lyee be applied to a switchboard system or an auto-pilot system or real-time control system of an airplane? What happens if Lyee is applied to OS management? Will Lyee be successful as an OS? How is Lyee different from the object-oriented approach? With these in mind, we would like to assign various topics to each member of the circle. Through this process, we would like to explore extended possibilities of Lyee, and strengthen and enhance them. Anyhow, in April, 2001, we will launch this project officially, and accordingly as a result of such collaboration, we will hold an international conference of Lyee in 2002 in Kyoto.

Q: Let me ask you about the technical side of Lyee. It seems users have hard time grasping what Lyee is. We often receive questions like how Lyee differs from DOA, and whether it is the same as the object-oriented approach. What do you think of these questions?

A: If something new appears in Japan, it is likely that people do not accept it without comparing it to something (laughs). Certainly, Lyee is difficult to access. In the conference in Appi last year, several professors made the same comments, namely that Lyee is

sort of the object-oriented method. The scenario function is similar to the "specialization" in the object-oriented approach. From a viewpoint of its coverage or the scope of application, what I think is right. The outcome of both methods turned out to be the same. However, they are quite different from each other. Source programs generated by the object-oriented approach have not determined the details, whereas the output or automatic generation of the source programs generated by the scenario function determines the details clearly. Another characteristic of the scenario function is to use subsets. But, the specialization of the object-oriented method just focuses on abstract matters with no details. It is a totally abstract approach. People may say, "America, America, America!" for a generalized concept. But America is just a frame maker. They may also want to say, "Ask God about details" (laughs). On the other hand, Lyee has a more concrete approach. But those who only know the object-oriented approach may feel Lyee is similar to it.

Q: Elaborate on the difference between Lyee and the object-oriented approach.

A: When you develop programs by using UML (Unified Modeling Language, constructed by Rational Rose) in the object-oriented manner, you utilize UML's use case diagrams. The diagrams determine what kind of communications should be held between users and their system. There are nine diagrams in which requirements are determined. With the Lyee method, requirements are determined by users. This is a significant difference between the two. In the object-oriented method, requirement engineers will listen to users and make specifications based on their needs. In UML, there is another methodology, which is called OMT(Object modeling technique). As you know, we are not sure whether descriptions of requirements do match the actual needs of users' or not. Why? Because we do not have users' models. But requirement engineers try to make programs into a system based on the requirements, that is, based on nothing because we do not have users' models. It becomes a total contradiction. Lyee does not

have this process. With the OMT, on the other hand, they use various diagrams such as state transition diagrams and data flow diagrams to design a system. Based on the system design, they make specifications, conduct testing, and make documentation. Like Lyee, UML also has an automatic tool. But, we do not know the messages among objects so that a programmer decides details between objects what data to be sent to the other object and what to receive. But a programmer does not know the requirement model so that he/she is not sure whether the implemented system meets the requirements or not. Lyee puts importance on users, rather than on engineers, whereas the object-oriented approach does on engineers, because engineers determine specifications. So, the gap between the two approaches is big.

Q: What do you think of "non testing" in Lyee?

A: I think testing is still necessary. But the testing Mr. Negoro means is different from ordinary testing in conventional methods. From a Lyee's structural point of view, there is no need for testing in a conventional sense. When you design software with Lyee, all you have to do is to define the user's requirements, namely words, using LyeeAll. The scenario function sets the sequential order of operations of the W04, W02, and W03 Pallets or programs. Even while defining, the W03 Pallet effectively conducts that testing whether users' intentions are appropriately dealt with or not. How? In the mechanism of the scenario function, the W04, W02, and W03 Pallets are operated in this order, and the operation is continuing. During this repetitive operation, users are supposed to unconsciously confirm their intentions expressed as requirements. Operations iterate until users' intentions expressed on the W03 Pallet agree with the model. In other words, this method conducts virtual tests, so this is another difference from conventional testing. What is testing, then? With conventional methods, systems engineers listen to various needs from users and write them out as specifications. They pick up what users want and what kind of system they want to make, and make specifications. But Lyee iterates operations. With conventional methods, things are

finished once system engineers listen to the client. In such a way, they do not understand the programs to be implemented so that even though they produce a prototype, it is not easy to make the system work satisfactorily. After a prototype is made, it is still difficult to satisfy the needs of the users. Then, those involved started to wonder if something is wrong. However, this part is being covered by an iterated operation of the scenario function in Lyee. With conventional methods, systems engineers listen to the needs of the users, make specifications and a sort of system. Once the system is completed, they need to check it. They also have to check the specifications of the system. What points do they confirm, then? They are supposed to examine logical issues and what could become a problem in terms of logic. With the Lyee method, this phase of the work is not done by engineers, but by the mechanism of the scenario function. The scenario function is composed of the W04, W02, and W03 Pallets. The intention of the function is what the user actually says.

Q: With conventional methods, do you think the testing burden is bigger?

A: Of course. Testing in a conventional manner is very troublesome. Why? Requirements of the system is based on users' talk. Systems engineers try to match the requirements with the specifications. Therefore, they have to test whether it works satisfactorily or not. There are so many inconsistencies. It is not certain whether the system agrees with the user's requirements or not. We have to do logical checking. We also have to check design documents and conduct mechanical proving as well. Even after these are finished, we are still not sure whether everything is completed or not and we may have to come back to the beginning. This is one of the hardest part in conventional methods.

Q: Should the same things be happening in Lyee as well?

A: Lyee also requires checking of human errors – but only human errors. With conventional methods, however, the original meaning of testing is to conduct logical examination before the document

design phase. People do coding based on the design papers. Removing data entry errors and bugs is not considered to be testing in the software engineering world: it is called confirmation. Of course, confirmation of total system specification correctness is necessary.

Q: Is the statement of "non testing" theoretically correct?

A: Yes, it is correct. The testing part is very simple in Lyee. Why? With conventional methods, there is no complete mechanism for testing. For example, conventional methods employ use case diagrams for testing. They are not perfect. They are not always right so that this cannot be a full test. Some use cases are good; others are not. Even though we use certain mechanisms, they can be used only in certain cases: they apply case-by-case, and are not almighty. But thanks to the scenario function, testing is conducted in a hidden way. Lyee does not need formal models for testing, due to the scenario function, whereas conventional methods need an essential testing phase to be set up at great pains.

Q: What kind of system do you think Lyee is suitable for?

A: I have checked Lyee's adaptability for various applications. Specifically speaking, the 401K system is developed as a business software. Controlling systems of digital watch and protocol etc. are also developed. Confirming these, particularly the protocol controlling system, I think it is possible for Lyee to be applied to any sort of software. As you know, specifications are very important and mandatory in the development of software programs, because we believe we have to make a realistic computer system which would satisfy various conditions to reflect users' requirements. For this purpose, we have to wrap up the requirements in some form. This process itself is called specification, and is expressed as a use case by UML, for example. But, we cannot write specifications which can grasp all events by such use cases, because the specifications are written by engineers whose knowledge and experience are limited. Specifications are usually based on the assumption of various events which

requirements would bring about, and on the way of handling those presumed events. Specifications are therefore similar to the concept of protocol in a sense. It is, in principle contradictory, and impossible to grasp the whole with the limited. After all, it is extremely difficult for humans such as engineers or users to write accurate specifications to grasp all the possible events. In spite of that, you are supposed to make an operative system from those specification. To this end, you have to avoid conflicts among various events caused by the requirements, deadlocks and loops which make the system lifeless. The specifications are given a significant role in the process, and formal models, standardization, verification, and testing are needed as tools to confirm the specifications. These tasks are mandatory in conventional methods. Anyway, all the possible events are too hard and complicated to grasp. But with the Lyee method, these processes are automatically handled using the scenario function. That is, Lyee does not need analysis of specifications and designing. The scenario function structure contains two permanent loops. One of them is a loop of W04 > W02 > W03 > W04, which the Pallet control function executes. This loop includes definition of the user's intentions or specifications between W04 and W02. The other is to restart the Signification Vector which the Pallet function executes. This loop is supposed to check that the user's intention in the aforementioned loop is significant or meaningful. Namely, it conducts a semantic check. As you know, loops are a big no-no in conventional methods: they are to be avoided, meaning specifications, formal models, verification, testing, and standardization became necessary. Nevertheless, Lyee's scenario function has loops in its core structure. The loops are naturally untied, however, when it reaches the single state where the user's requirements are fully met. In other words, first, a word (the minimum unit composing the user's intention) is provided to the scenario function. W02 Pallet receives it, and the Pallet function of W03 restarts the signification vector for the word by using loops until the meaning of the word is established. When the meaning is formed, the loop is untied and it goes to the user via W04. The

user confirms whether his/her requirement is fulfilled by the given word or not. If it is not satisfactory, the user changes or adds words. The Pallet control function works and makes a W02, W03, and W04 loop. If the result is satisfactory to the user, the loop is undone. Loops can be undone without relying on the reset function, unlike conventional methods. This mechanism could be called automatic specifications, standardization, formal models, verification, and testing. Once a word, which is a component of the user's requirement, is assigned to the scenario function, a source program is mechanically produced. As for the protocol, engineers design it by using state transition diagrams and decision tables in conventional methods. But, those diagrams and tables cannot grasp every state and every action. Also, when specifications defined by state transition diagrams and decision tables are to be made into programs, the programs usually become large-scaled and complicated. On the other hand, Lyee's programs are small-sized and simple. Even though the specifications of the protocol are not defined in the state transition diagrams and decision tables, something like those diagrams and tables are to be made in W03 Pallet. All software could be said to be synonymous to protocol control. When I had a chance to look at a sample program of the protocol control using Lyee's scenario function with the above-mentioned mechanism, I was impressed to see it worked perfectly.

Q: Do you think Lyee is appropriate to the development of a large-scaled business application?

A: Yes, I think so.

Q: What points are most suitable?

A: Usually, it takes time and money to work on projects like 401K in a conventional method. The most time-consuming part of making specifications is omitted in Lyee, because Lyee does not need this process, and thus it can take the quickest way. Both with W03 where business specifications are to be expressed, and the mechanism of the iteration of the scenario function (W04 > W02 > W03 > W04…), we were able to check the appropriateness of a

word given to a screen. With conventional methods, we cannot do so. When systems engineers listen to the customer's words – requests and specifications – they naturally make what Lyee calls W03 logic in their minds as a black box. Afterwards, they try to make programs based on the image they have of the logic of processing order and business requirements. Therefore, nobody is sure whether their product complies with the requirements of the customer. On the other hand, it is of no consequence if errors occur with the Lyee method. All they have to do is to pick up a word, check the W03, and increase words one by one. Thus, the system is produced so that we do not have to worry about whether the system will be out of focus from the user's demands.

Q: Compare the waterfall and prototyping methods with that of Lyee.

A: The conventional method follows a model of the waterfall model. If something happens which demands us to go back to some steps earlier, we in fact have to remake programs from the very beginning. As for the prototyping, it is certainly similar to Lyee in terms of its expression. Based on users' requirements, they make an imaginary prototype and revise it, and make it again and revise it again. Thus, the product is gradually approaching the user's requirement. And it is indeed a methodology. However, it does not have a mechanism. With conventional methods in general, everything is determined depending on the situation.

Q: What is the greatest advantage of using Lyee?

A: One of the strongest points would have to be the fact that testing is not necessary. It is an awful amount of work to examine whether the product meets the requirements or not. With conventional methods, testing methods are also changing depending on the system, the specifications, and the purpose. That is, as I mentioned earlier, everything works depending on the situation. In contrast, Lyee has nothing to do with these. Whatever system is used, it does not matter. This is very good. Also, the Scenario function semantically produces validated specifications from the user intention by its iterative nature.

Q: Are there any weak points or contradictory points in Lyee's method?

A: I cannot judge it yet. I am just wondering, however, if the same scenario function works in every case? I want to verify this. I also feel this part needs studying. I guess other researchers or professors feel the same way.

Q: How did you find LyeeAll?

A: Interesting. Before I knew Lyee, I always made the greatest effort to use my imagination in order to work on projects. In a conventional manner, while listening to users' requirements very carefully, I sort of believed I understood what they wanted and went to the next step. But, if my imagination works inappropriately – even a little bit out of focus – that's it. With LyeeAll, however, we can correct the errors immediately if they occur. But in actuality, what we expect is what we get most of the time.

Q: Do you permit your students use it?

A: Well (laughs)… I am worried because it is too easy for them to make source codes by using it. Given too a simple way, students will not take a tough road. I would not let them use only LyeeAll.

Q: Do you think maintainability will improve?

A: It is true that the automatic generation of source codes are very simple, but I have reservations about commenting on this. I want to make judgement after studying the mechanism of Lyee more and verifying what Lyee can be really applied to. I have not yet reached the point where I can make a conclusive judgement about it. My purpose is not to develop the system, rather to study it. On the other hand, from a business point of view, they have to choose the best method for their system development. Changing development method will affect their businesses as well. But the change may be painful to the people in software business. For example, after development of some parts of the system, its platform becomes antiquated, so the part which adopts it needs to be changed into a

new one. It is a pain. As the computer industry depends on human resources, they should also be replaced by new staff. It is also extremely painful to switch from one methodology to another methodology. There are many problems. It is said that Lyee can resolve all these problems. We need to examine it. In business, theory matters very little and output and outcome is the be-all and end-all.

Q: The scenario function is said to be a universal function. What do you think of this?

A: It should be proven as a mathematical former. It is understandable as an abstract term. But I wonder if the scenario function can do anything related to software development once it is applied to it. We need proof, not just words. I think it must be proven in many use cases such as protocols, banking systems etc. Various cases should be investigated. Mathematics and applications are totally different matters.

Q: What do you think of the idea that source codes are to be determined by the solution of the scenario function?

A: Lyee's way to develop a system is very interesting as an idea. It is totally new in the software industry. This is a key point. Software was in effect generated from hardware, but Lyee has nothing to do with it. Hardware is not an issue to Lyee. This characteristic is essential to this new invention. The waterfall type is a concept generated from hardware so that designing and specifications are necessary. The state transition diagrams and the data flow diagrams are also concepts to originally produce hardware. I am repeating the same things (laughs).

Then, how different are the object-oriented approach and Lyee?
The similarity between them is that both of them employ objects as development targets. Words are considered one of the objects. Similarity ends there: there is nothing else. In the object-oriented approach, everything is handled as a unit of objects. System engineers determine attributes of generality, specialty, and class of

objects including words. Concepts of the words are established by exchanging messages among words. This part appears similar to Lyee. Then how are the programs implemented by the object-oriented approach? For example, they use UML. But even UML produces abstract classes. Various diagrams are necessary. While working on it, we sometimes lose the track of relationships among the diagrams. But Lyee is not like that: it's simpler. As an overall model, Lyee may look like the object-oriented approach, but the latter has no rule to determine each single step of the programs in the object-oriented method, but Lyee has. As a concept, Lyee seems object-oriented, but it does not utilize its approach.

Q: Is Lyee superior to the object-oriented approach?

A: I have not yet come to a conclusion regarding that. I want to clarify strong and weak points of each method and compare them. I would also like to find out weak points of Lyee (laughs). But I have a feeling that Lyee is superior. I truly sense so. But since I am a scholar, I do not want to say things based on my feelings. However, I dare to say so, because I have used both the object-oriented method and Lyee. Something is wrong with the object-oriented method; it's strange. I've used them both and had both easy times and hard. Lyee is able to generate concrete source codes, but the object-oriented method does not provide answers. With the object-oriented method, I did various things including documentation, programming structures, as well as determining the exchange of various messages in the empty class, entering various processes and attributes, using no less than nine diagrams, and entering the value that users want to use. Still, I could not get what is called W03 logic with Lyee. That part of the logic (W03) is the next phase of development, which is a job for programmers. On second thought, however, I find that this phase is nonsensical. Once the product is implemented, it will be shown to a user. If the user says, "No, this is not what I want." Then, a programmer changes the programs. But this process itself is contradictory, as I mentioned. Taking this into consideration, I feel Lyee is better.

But I want to emphasize that this is simply my gut feeling. Please look forward to the results of research about these issues, which will be done by the new Lyee international research network.