Im OUTPUT-Gespräch: Edward Yourdon:

Case, Object-oriented Analysis and Reuseability

Interview: Louis A. Venetz

Case, object-oriented analysis and reusability were topics of the 1st Software Symposium Surselva in Flims. Edward Yourdon, one of the most famous methodologist in the field of analysis and design, has been invited as the main speaker during two days. OUTPUT accompanied the symposium and on the 3rd day Mr. Yourdon gave us a very interesting presentation of these topics.

Edward Yourdon, methodologist and independant consultant

OP: I welcome you, Mr. Yourdon, here in Switzerland and would like to ask you first about your methods. You have become a very popular methodologist in computer science. Would you give us an explanation for your success?

Yourdon: The method that most people know me for is called the structured analysis and design methodology. What all of my friends and I did in the 1970s and 1980s was to begin focusing the attention not on the programming issues which most computer science people are primarily concerned about. In the early 70s we begun shifting our attention to issues of software design, architectural issues, don't worry about the programming instructions instead worried about building blocks and how they are connected. And then we became probably best known, because we shifted our attention even further and said we should have better ways of understanding the user requirements. Because if you do a bad job there it doesn't matter what kind of design and programming you do. And we developed a very simple graphical notation for modeling the user requirements of a system. I think that was the key point that the notation was so simple that you could communicate in such a way that you could show it to a business person or some other user, without teaching.

OP: Could we say that the structure of diagrams is obvious to everybody?

Yourdon: That's the point that just as an architect usually does not have to explain the notation of a blue print (engineering drawing). As the architect is designing a home for you he shows you the diagram and you are immediately involved in the issues. He doesn't have to say this is our notation for how we represent the door or window. We had the same idea for modeling software systems that the notation that we used first showing data and functions was so simple and transparent that when we presented it to the user he could focus his attention to the issue of his system rather then to the notation. So it became a kind of obvious. And that's really were I became fairly well known. We published this ideas in a number of text books which were distributed all around the world. And then, in the middle of the 1980s, when the Case industries begun this to develop, many of the Case tool vendors showed my systems analysis to operate in their tools. So many of the Case vendors today will advertise they support the Jackson, the Warnier or the Yourdon method. I think it also helps explain the popularity of my work.

OP: But, which was the motivation to invent such methods. Did you want to make it better than it was?

Yourdon: Yes, we wanted to make it better than it was, but to be perfectly honest I started in this profession as a computer programmer. I wrote programs and I was perfectly happy doing that. But I got involved really through serious of accident in teaching. And I was really lucky to get my education in my very early career, in an area in the United States that has a lot of good ideas about software. The areas of Boston, Massachusetts, and the Silicon Valley near San Francisco has a number of people in the software field who are very well known. So I was shown many ideas by teachers or experienced people. And then I begun working in the field with people who were quite familiar with some of the standard techniques for program design. I begun teaching programming, and I guess, the motivation for my work is the gradual realization that the most difficult problem that we are trying to solve was not programming. May be, the most difficult problem is design, architectural design and we concentrated on that. Then we decided, well that wasn't really the most difficult problem either. You can have a brilliant design and brilliant programming and still have a failure in the project if the system that you built did not actually solve the users real application. So for me, the motivation was to provide more successful methods for helping computer people succeed in their projects.

OP: Which books did you write about your methods?

Yourdon: Well, it's hard to say, I have written 17 books myself. When I had my training and consultant company which had my name, Yourdon Inc., we had a publishing division, because we were so anxious to get this books into the public in the 1970s, that we couldn't wait for the two years that most of the publishing companies required. So we acquired a typesetter, we begun using the Unix at 1970 before anyone it does. And so I actually had a publishing company called Yourdon Press, which was a division of the whole company called Yourdon Inc. All together that division published 50 different text books.

OP: Is the book Object-oriented Analysis from Yourdon Press?

Yourdon: Yes it's from Yourdon Press, but it's now a part of Prentice Hall. Five years ago I begun to find that I was spending all of my time to administrate and material issues. I wasn't very happy about that. So I sold the publishing company to Prentice Hall. And it's now the Yourdon Press Division. Then I sold the training part to some other people. But in any case I have written some books, probably the one it's best known of mine is Structured Design with Larry Constantine in the middle of the 1970s. And I think it was probably the first book really focusing everyone's attention on architectural design issues rather then lower level programming issues. Then I wrote a number of other books in between modern structured analysis which describes an analysis approach. The book Object-oriented Analysis, with Peter Coad, is one the most recent because I begun to shift my attention to a much newer methodology in this object-oriented area.

OP: By the way, what is really object-oriented analysis?

Yourdon: Perhaps to answer that I should first describe briefly the two most popular methodologies before this. The one that I developed, called structured analysis, turns out that kind of systems analysis, concentrates primarily on the functions that the systems has to carry out. So we will go to the user to tell us about the system he want to built. And he would say it's a payroll system. And we will identify that is the major function. And then we would ask him to help to decompose that major function into smaller. And all this would be represented in a data flow diagram.

We developed a very simple graphical notation for modeling the user requirements of a system.

But the primary emphasis was functions. Meanwhile another group of people were focusing primarily on the data. So there were information engineering and data modeling methodologies in which the most common is the entity relationship diagram. And that is a very good methodology, but unfortunately it only emphasizes the data, where my methodology only emphasizes the functions. And we begun to see during the 1980s that either methodology was biased, or prejudiced, in one direction and there were some other problems as well. Neither of the methodologies provided any assistance in the area of reusability. With both of these methodologies when you begun working on a new system, you would start as if nobody had ever developed any kind of system before that you can take advantage of. And more and more today we can see that there is a great benefit from reusing existing software components so that you don't have to do it again. So this object-oriented approach is one that in many ways was suggested from some of OOPL programming languages that are available now. Primarily it is an attempt to bring together the data side and the function side into a single package. So when you ask him to describe a customer for example on typical business data processing system, you are not only interested in attributes or described elements of the customer, name, address, phone and number, but also what are the services that have to be required for the customer. Well, you have to create a new customer, sometimes you have to delete a customer, perhaps you have to provide certain inquiries about customers and so on. That's the basic idea.

OP: And what does inheritance mean?

Yourdon: Well, inheritance is the aspect of object-oriented analysis that allows us to take advantage of reusability. And it has a profound impact on the way software engineers work. It means that when a user tells us about a new system, instead of thinking about as completely new, we are looking for other systems that we have already developed in the past that could be used as a starting point. So that we can perhaps define some new objects which are subclasses of existing objects. The technical aspect of object-oriented analysis is this inheritance idea where we can define a new object in such a way that it automatically takes advantage of the attributes and also the functional services of higher level objects that we defined previously.

OP: So the entity relationship model cannot use this inheritance?

Information Technology in the 90s:

Software Symposium with Edward Yourdon

L.A. Venetz

Organized by M. Uffer, Ralph Kirkley Associates, St. Gallen, the 1st Software Symposium Surselva took place at the Hotel Adula in Flims. The main speaker was Edward Yourdon. At the conference topics and practical experiences were the educational event with participation of the Institute Inforge, University of Lausanne. The workshops were organized in coordination with sponsors, software schools and the Institute Inforge. The participants were given the opportunity to assist in small groups at technical presentations and pursue more informal discussions within the workshops. The success of this first conference, will make a second Software Symposium happen.

Edward Yourdon suggested what Case tools of the 90s should provide to support. Other topics about the Case technology were new developments, successfully implementing in organizations and the influences by AD-Cycle. In the area of software methodologies he mentioned the characteristics of different software engineering methods: structured analysis vs. information engineering vs. object-oriented methods, requirements on database architecture for a repository and the opportunities for other approaches than AD-Cycle. For the software re-engineering and reverse engineering topics, he gave informations about the current technology, future trends and how to get started. Emerging software technologies concerned about software metrics, software reusability, and software quality assurance. The new word "Peopleware" explained how leading MIS organizations are selecting, training, and motivating personnel; new ideas in team formation; "breakthrough" performance concepts; "performance management" – an approach that aligns corporate objectives with the employee's perception of the personal consequences of his actions.

Other famous speakers from Europe and USA completed the conference. C. Stricker from the University of Lausanne showed how the use of methods and tools in Switzerland is and what the results from an overall survey in 100 organizations are. F. Vassalli, Denner AG, demonstrated a successful 4 GL project for a major commercial application and its impact. Dr. K.H. Hauer, EBH Konstanz, enjoyed the audience with the implementation of a Software Factory, from a training organization to a productivity center. For example, Lansa a so called lower case tool, greatly contributes in reducing the normal development time of software, and also allows to diminish the costs of software maintenance. Some application experience with Case stated A. Stewart, Aspect Computing Europe Ltd. Topics about reconciling functional decomposition, conceptual modeling, modular design, and object orientation for application development were presented by G. Maksay, Institute Inforge, University of Lausanne, who is together with Prof. Pigneur active in the fields of information systems design, application development and man-machine interface. Goals and principles of quality management for application development were discussed by Dr. E. Wallmüller, ATAG Informatik. A well-defined methodology is the key for all quality activities and has to support a variety of components for quality management. Furthermore there was again a development experience using X-Window for a resource planning system, which was introduced by R. Varonier, Openware. And a pragmatic approach for realizing a communication project in an international industry suggested D. Witteck, Focke & Co. Last but not least, there was Ralph Kirkley himself who discussed the question «Software contracting, why an increasing need?» He was able to show the best way to manage «external» people.

Mr. Uffer and Mr. Ralph Kirkley

Yourdon: Well, I think some Case vendors might disagree with you. It's more appropriate to say that this object-oriented approach has taken some of the best ideas of entity relationship diagrams, the concept of an entity and the concept of relationship between entities. We also use the idea of inheritance, the idea of packaging function and data together. Most of the methodologies that are based on the entity relationship diagrams only allow a description of the entity itself, not the associated processing units. Most of the methodologies based on entity relationship diagrams don't have the concept of one object sending messages to another. So we are beginning to see some extensions of the entity relationship approach to try to incorporate some of this features. To some extends there is a question of weather object-oriented analysis is a revolutionary new technique that requires to throw away entity relationship diagrams and throw out dataful diagrams, or is it possible perhaps to start with entity relationship diagrams and extent them to add some of this very useful ideas. As I say, some of these database management vendors as Software AG with their database technology certainly they already have the idea of superset or superclass and subclass. And some of them are beginning to add inheritance capabilities.

OP: You have mentioned already Case. What does it mean to you?

Yourdon: Because of my background in developing methods I have always been interested primarily in Case tools that provide automated support. For that area of systems development which is only a subset of the entire spectrum of systems development activities, the area that has always had some automated support is the down stream activity of programming, testing and debugging. Today that's called Lower Case at least in the United States. The significant thing is that the automated support has moved off the mainframe,

the workstations and PCs. That's a very powerful, very valuable idea. But to me Case really means Upper Case that is automated support tools to provide support for the analysis and design ideas. Particularly because my and most of the other successful methodologies in the field they are primarily based on graphical models, diagrams of the design, diagrams of the specification. And diagrams are obviously easier to do if you have automated support. So that's what Case means to me. As either workstation, PC or theoretically even mainframe technology they can provide automated support for the diagrams and all the details behind diagrams.

OP: What do you think about the package "Software through Pictures"?

Yourdon: That's one of the very popular and successful Case tools in the United States. And of course it is used all around the world. In the United States there tends to be a much greater distinction between Unix and other operating systems than what I have seen in Europe. In the United States, if you say Unix, typically your are talking about an environment where people are building scientific or engineering systems, where they are using programming languages such as C or Ada. For the business data processing people that is usually Cobol, IBM mainframe and today of course PCs and workstations that use MS-DOS, Windows and OS 2. So Software through Pictures has typically been one of the very powerful Case tools in the Unix environment in the United States, for people building realtime and engineering systems. There are two or three other similar and popular, but one nice thing about this Software through Pictures product is that the man who is in charge of the company, Anthony Wasserman, is a very distinguish computer scientist himself, he has been very active in this area of object-oriented technology, he proposed some graphical notation of his own at the object-oriented design level, something called object-oriented structure design notation which he has built into his product. It's interesting that most of the other Case vendors have very good products, but they are not able to take a leadership role in the actual methodology themselves. Usually they are simply providing automated support for methodologies written in a book like mine or some others. But the Software through Pictures people are methodology leaders themselves, I think. They have a good product.

OP: While using your methods, what would you call a typical analyst?

Yourdon: Well, it's hard to say, because the structured analysis methodology has been used all over the world. Most commonly I suppose on business data processing or MIS systems the object-oriented methodology is still relatively new. The typical analyst is someone with perhaps five years of experience, typically it's someone who is working on large systems in a group of 10 or 20 people. I say that because there is another world of people building relatively small systems on PCs or workstations, people who works alone or with 1 or 2 other people where the problem they are working on are simple enough that sometimes they don't feel that is necessary to use any formal methodology for documenting the user requirements. The typical analyst who uses my methodology is somebody in a large engineering or business data processing environment working on big systems.

Case really means Upper Case that is automated support tools to provide support for the analysis and design ideas.

Edward Yourdon is one of the most famous methodologist and author of 17 books. He was born in Florida, USA. He studied mathematics at MIT and begun working at Digital Equipment Company. Now he lives in New York City for about 20 years. He has three children. Until 1986, he had a training and consulting company with offices in London and all over the United States. One reason his methodologies are fairly well known is that he trained about 250000 people all around the world published about 50 text books and different aspects of software systems engineering. But he sold that company about five years ago and now he is working as an independent consultant, still running text books, developing methodologies. But he has his office at home, which he likes to call an electronic cottage, with local area networks and computers everywhere.

OP: What are the difficulties you see to be a perfect analyst?

Yourdon: Ah, well normally the most difficult thing is to have a good communication with the user to really understand his needs, to distinguish between how the user is currently operating his business and what the underlying essence of the business is. That's the distinction we often make between physical, logical, implementation and essential models. Normally, if I were a systems analyst and you were an user regardless of this methodologies I would still come to you and say, tell me what kind of system you want me to built. But you would describe how you currently carry out the business today, in the manual fashion, in the automated fashion, or how you imagine it should be carried out. And one of my most difficult tasks as a systems analyst is to eliminate all of these arbitrary implementation details and try to distill the essence of what are the essential functions have to be carried out, what is the essential data that is involved. How I represent this with structured analysis or object-oriented analysis is always secondary. The most difficult job that distinguishes a good analyst from a mediocre is really being able to understand the essential message he is getting from the user.

OP: Does an analyst the opposite of abstraction? Does a philosopher constructs words and the analyst does the reverse to find out what it means?

Yourdon: Yes, that's true. You know, an analyst discovers the current, the business environment, what the user is presently doing. In our approach it's very important for him to be able to do that, but also to carry out this abstraction. I think that's just another way of describing this problem I mentioned before. If I asked you to give me a very detailed description of your current activities to some extent I am carrying out analysis, but I also have to be able to abstract from that, a very pure model of the key functions that have been carried out in your application.

OP: What do you think about education and leadership of analysts?

Yourdon: Certainly, education is important. There is no question about that. When all of these methodologies were first developed in the 1970s and in the early 1980s they were often considered very advanced and very esoteric. Some of them were derived from research activities. As a result they tended to be studied and practiced at very advanced university level initially. They were graduate students activities. And then over the 1970s they begun dropping down to

the under-graduate level. So that in most universities now somebody who studies computer science or software engineering is exposed to all of this methodologies to the 2nd or 3rd year of study. And in two or tree cities in the United States it dropped below that now in towards the secondary school level. Fourteen year old children are instructed in analysis in New York City. But that's understandable, you know. In the Middle Ages, long division was a university subject, only the most educated people could divide numbers. In the long term I don't see education has been much of a problem. It's still some of the problems today because we have a large number of people that are already in the field who may have received some computer science education 20 years ago, but they were not exposed to methodologies. So training that people in companies today is still very important. In terms of leadership, probably the most important aspect of these methodologies is to simply take a leadership role in terms of the importance of doing a good job. There is a tremendous pressure all around the world to build systems more quickly because the user has his own business pressures. Because of that pressure and because of the power of prototyping technology and fourth generation languages there is a temptation in many organizations that has existed for – as long as I have been in the field – to start programming by the way: Don't worry about defining the problem exactly and precisely, start ready programs. In many organization that's the only tangible evidence of progress. When I'm a programmer and I sit there and going to write 25 Cobol statements, my manager thinks that I am competent for something. If I draw three bubbles and entity relationship diagrams it doesn't seem to be tangible evidence of progress. It's abstract thinking, you see. So it has not as much value. So the most important leadership role I think that a project manager can play is to defend the importance of carrying out some of these analysis and design steps, regardless of which methodology it is, to defend that against the pressure that he is certainly going to get from users and senior management, keeps in why are you programming that? And the answer has to be it's important that we have a good understanding of the requirements of the problem itself and the overall architecture of the system before you begin the implementation.

It's important that we have a good understanding of the requirements of the problem itself and the overall architecture of the system before you begin the implementation.

OP: What is your message to all computer scientists? What should they avoid for example?

Yourdon: Well, I think the most important thing they should avoid is stagnation. Unfortunately there is tendency with this methodologies to treat them as a religion. You have fanatical followers of information engineering methodology, and fanatical followers of the structured analysis methodology. One thing is beginning to be clear as our software industry has shown the last 30 years is that we are going through periods of change, and that's certainly true with methodologies as well. If you become a an expert, or familiar with a particular development methodology today you should expect that in five years there will be a better one. Personally I think that object-oriented analysis is a step forward beyond information engineering and structured analysis. But even if that's true which I believe the most important thing is to remember that by 1995 or the year 2000 there will probably another be beyond that. At the moment for example object-oriented analysis encapsulate functions and data but it's probably going to be true that the next generation beyond that will do a better job of capturing knowledge, business rules, to incorporate some of the ideas of expert systems and artificial intelligence, to really have it done in this methodology. So to me, the future is one of continued change in evolution of the methodology. And it's also something that is more possible by the ongoing evolution of hardware technology. All of the hardware research that I looked at indicates that the kind of hardware improvement that is happened over the last 20 years is going to continue for at least another 15 years. The predictions suggest that the hardware of the year 2000 will be somewhere between a 100 and a 1000 times more powerful that we have now. That provides for more powerful Case tools, more powerful languages which makes it possible to have more abstract modeling tools that we can use in the methodologies. So I think that's my view of the future.

OP: This was certainly only a short excursion with a very famous methodologist and I suppose that all the missing questions here in this interview you will further answer in your books and speeches all over world. Thank you for answering to the questions and for coming to Switzerland. And we may hope that you will come next year again for the 2nd Software Symposium Surselva.