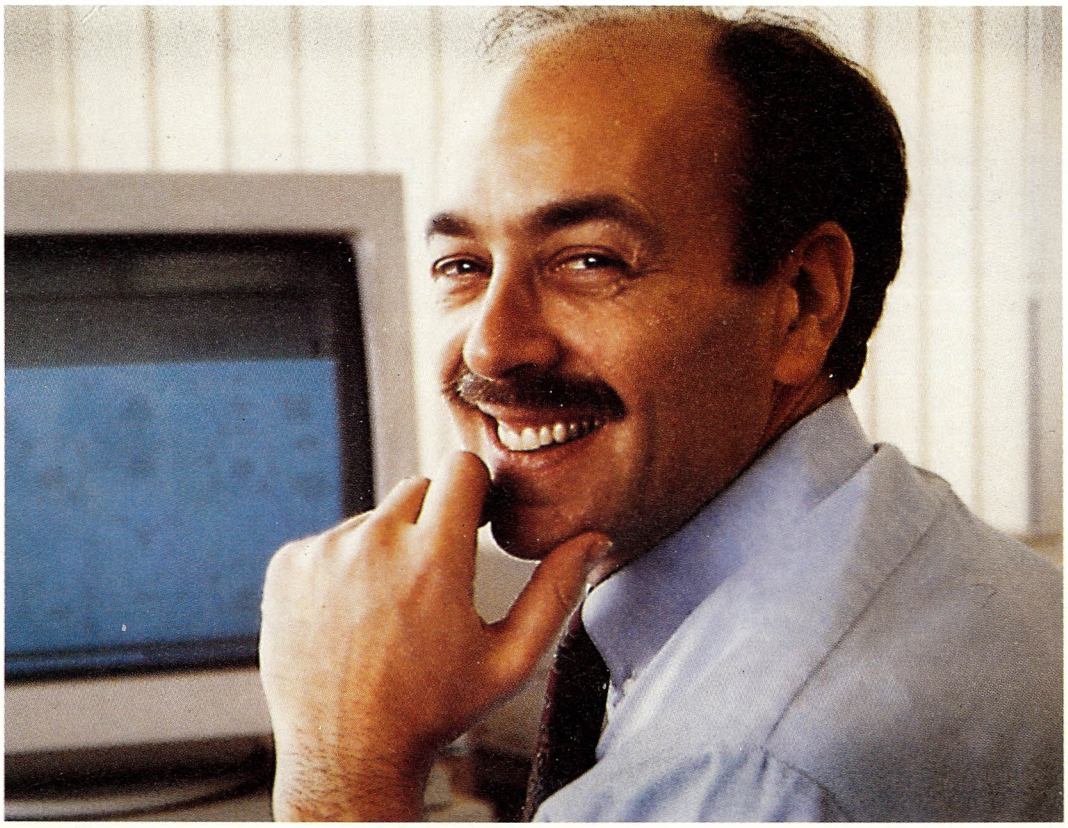
Im OUTPUT-Gepräch: Dr. A.I. Wasserman

Software engineering and Case environments

Interview: Louis A. Venetz

In the last couple of years, the importance of Case tools has grown. Companies building software systems must be careful to select a multi-user Case environment that can be adapted to continuing hardware and software advances.
Dr. Anthony Wasserman, one of the most famous Case architects, was interviewed in Geneva about his vision of using Case and the structure of the Software through Pictures (StP) Case environment. Also, his message for the future may help some people in their choice of the right way to develop software.

OP: *I welcome you Dr. Wasserman here in Geneva. I would like to ask you first about the status of the Case market today, and then how it has changed in the last couple of years?*
Wasserman: The status of the Case market has changed significantly in the past couple of years. The first users of Case tools in the mid-1980's were experimenting in the same manner as early adopters of any technology. They evaluated product features, and tried them out on small projects. But they did not use it in a serious way for projects that were a strategic importance to their organizations. This is now changing very quickly. Companies are paying attention to the need to improve the quality of their software and to improve their process of development. As a result they are looking to software engineering methods and to Case tools to help them do that. And so we are seeing higher-level management becoming involved in Case decisions, and they are making strategic decisions that have a greater influence on their organizations.
OP: *What kind of people will use Case in general?*
Wasserman: Most of the users of products like StP in the Case market are historically starting on the design of a new product. These are often analysts, designers, sometimes they are programmers. And now the market is gradually shifting to involve a broader set of people associated with software development.
OP: *What about the ability of programmers to learn and use software engineering methods and Case tools?*
Wasserman: It's like any new technology or any new approach to problem solving. There is a body of knowledge, there is a way of practising methods to analyze software systems or to design them. And many people who have done software development have done this very informaly. They have taken systems and done decomposition, top down design, step by step. Everyone taught this approach. Many of these ideas, of course, originated here in Switzerland. Methods like Structured Analysis and Structured Design, which are now quite old, are based on these ideas of partioning and decomposition. But structured methods or Case tools don't come automatically to people; they have to study them expressly.
OP: *Must your organization train your clients if you want to sell StP?*
Wasserman: There are two


Anthony Wasserman

issues. The first one is very traditional. People who work for an organization have to learn the process that is used in that organization for software development. And with Case that method has gradually changed. So there is a need for the organization to support the training of their people so that they can be the most productive. The methods that are used in Case in general are not taught today at Universities in USA or – as far as I know – in most other countries. So this becomes professional development material that they learn after they are working. We as a company at IDE place a lot of emphasis on making customers be successful with methods and with tools. And we offer a very comprehensive set of courses, training and consulting. We use for our publicity something like a traffic sign which says: Caution Case learing curve ahead. And we give our clients a road map which shows all the different courses that we offer to help them overcome the difficulties of introducing and using this technology.

OP: *But in comparison with learning programming languages isn't it much more complicate to learn Case or methods of Case?*

Wasserman: I don't think so. Of course it depends on the language and the method. But throughout history the methods that are the best and the most widely used are those that are comprehensible and that can be learned. If you try to learn a programming language like PL/1 or Cobol, they have a lot of reserved words, a complicated syntax and most programmers use only a very small percentage of the features. To me those are horribly complicated languages. One reason Pascal was designed was to bring simplicity, to have a nice clean and simple notation. So there are some software engineering methods that are very complex, but others where the principles are straight forward and where people can understand the notation and they just need some practice to use the method well. In the same way, a person using a new programming language must write several programs before being able to use the language effectively.

Dr. A.I. Wasserman is founder and President of Interactive Development Environments, Inc. (IDE). Before starting IDE, he was a Professor at the University of California. In 1983, he was a Visiting Professor at the Université de Genève (CUI II). Wasserman has a Ph.D. degree in Computer Sciences from the University of Wisconsin, Madison. He has edited 7 books, e.g. one about software engineering education, published in Europe by Springer and he has written many technical papers on software design and software development environments. He is an architect of IDE's Software through Pictures Case environment and codeveloper with Dr. Peter Pircher of object-oriented Structured Design (OOSD).

OP: *You invented also a design method called OOSD. Can you tell me the principle of it?*

Wasserman: Sure. We actually invented a notation. A notation and a method are not quite the same. But the idea of the OOSD notation is to support the architectural design of systems. We wanted to be able to provide a single notation that supports many different kinds of designs, including both hierarchical functional design and the more modern object-oriented design. So we produced this notation and have now built some tools that support that notation. We are also starting to teach people how to design their systems in an object-oriented way. And that´s very different from the traditional approach to design.

OP: *Do you develop your own applications with StP?*

Wasserman: Of course! As the founder of IDE and one of the two architects of StP I pay more attention to how to build the environment. Over time I have built some designs, some small systems with these methods. The methods are quite old. But today I do not do much design and implementation of software systems. There are other many things to do, to make this company of cours be successful.

OP: *Would you mind giving a short scenario how you would develop an application?*

Wasserman: Some years ago, application development followed a straightforward traditional life cycle model, where you began by doing Structured Analysis with a set of data flow diagrams and/or data model. You would validate this model with a walkthrough or review, and then move to design to build the software architecture, and so on – very traditional kinds of things. Today, we try to look at system design more from the object-oriented spiral approach. This approach is based on identifying classes and relationships between classes putting together an object-oriented design. The iterative steps involve high level identification of classes, refining those class definitions using implementations where they are available or building your own. This is a very different process for software development than the traditional structured approach.

The status of the Case market has changed significantly!

OP: *Do you have an example of very famous applications in the world made by StP?*

Wasserman: The one I like best is an application which was developed by Raytheon in the USA. And the application has to do with aircraft landing. When an airplane is landing it is very sensitive to defects of so called wind shear. A group in Raytheon built the system Terminal Doppler Weather Radar which is used at Logan Airport in Boston. And this is a system that was designed with StP. And since I fly to Boston a lot, I'm very happy to have that system there.

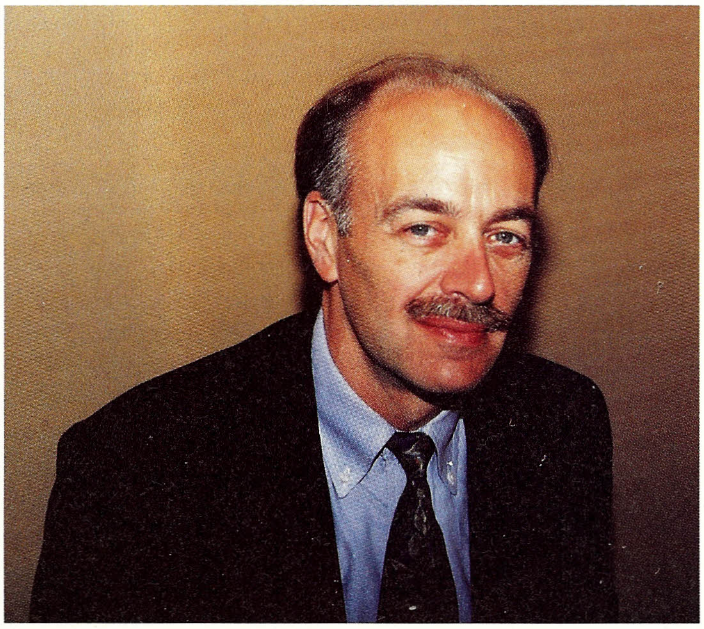
OP: *In general people are not always satisfied with software. Can StP stop dependencies between programmers and users? Is it possible – one day – to develop software like a shell in an expert system that users will be independent?*

Wasserman: Yes, it's possible. But it's typically for very specific kinds of applications. If you look today you'll see that database companies have designed application development tools in fourth generation languages and forms-oriented systems that are intended to be used by non-programmer. If you look at some of the spreadsheets products they have their own macro languages which are in fact like programming languages. If you look at Hypercard there is a scripting languages Hypertalk for people to create their own stacks. So in some kinds of application domains there has been an attempt made by software builders to give users the ability to write their own applications using this technology. It's interesting that some of these mechanisms are programming languages themselves. And so sometimes the endusers write very good programs in this specialized language and sometimes they don't. And they discover some of the same problems in their spreadsheet programs that programmers discover in their Fortran or C programs. There will always be something for the professional programmer to do.

OP: *What is the highest expectation with Case? Is it possible to redesign, reengineer an operating system like Unix?*

Wasserman: Your question opens a lot of possibilities. One thing that we have built very recently is the C Development Environment. It is aimed at people who are both: Developing new applications in C – which is a fair number – and there is a larger group of people who are doing continuous development – evolving and reengineering applications that were written in C. So we have combined parts of StP with Saber C which is a very good C programming environment. And we have built a tool that allows us to generate code from StP into Saber C and take C code and to generate the design information into structure charts and data structure diagrams with parameters, to do this both as a one time operation and incrementally. Somebody can change a piece of code, then generate only the new part of design. This is a very nice technology. But now, the question of can you use that to reengineer Unix? In principle: Yes. Because this is aimed at any piece of C code. It's not clear that one wants to reengineer Unix but could you use this to better understand the structure of the operating system, source code and the data structures – absolutely. And to navigate back and forth between the code and the design and ask queries.

OP: *Unix could be developed forward for the use of multiprocessor or parallel systems?*


Case tools don't automatically to people.

Wasserman: I think the goal that we had in StP and in the C Development Environment is largly independent of the specific applications that people are doing. So again, in principle the tools can be used for that. Indeed many of our users are building realtime applications, embedded systems and designing multi-processing kinds of applications. But the design and applications that run on parallel processors, this is quite a new area. I'm not aware of any of our users who are doing just that today.

OP: *What is the meaning of the word «repository»?*

Wasserman: With the big R it is IBM's proposed database for the information produced and used by other tools. In the sense of a little r definition the repository is the place where you store all the information that is produced by and available to all the tools that must work together in a Case environment. When we first built StP we included a relational database system, we defined a schema for the repository for StP. So we shipped this multiuser repository to our custumers over 7 years ago, in early 1985. So we had at that time created a repository specialized for the need of StP. Now there are quite a number of repository proposals, some of them have a predefined schema as available for IBM's repository for example. And others are just generalized capabilities that defined a nonsharable repository without a specific structure.

OP: *Is the repository a pool of tools where you can get the information for all the parts of software development?*

Wasserman: It's not the pool of tools, but the pool of information shared among the tools. So if you think about all these tools you want them to share data with one another. If I generate code from my design tool I can send it to a programming tool. If I have a design diagram I can send it to a technical publishing tool, if I have source code I can reverse engineer it into the design tool. Rather than having the tools talk directly to one another through a secret file format you use the repository as the place where this data interchange occurs. You make sure that all the data that is available to all the tools is in the repository. The advantage to this is that now you can build other tools that are able to share that data as well.

We as a company at IDE place a lot of emphasis on making cutomers be successful with methods and with tools.

OP: *Is the repository the reason of the ability to recover design form source code?*

Wasserman: It's not the reason because we could have taken the source code from Saber C and created our own private file system, file structure and achieved the same objective. But we used the repository because that is the mechanism that has multi-user access so that different tools and users can concurrently get access to and navigate through the design to the code in both directions which you can not do with the private file connection anywhere near as well.

OP: *Will heterogenous networks disappear in the next century?*

Wasserman: Sometimes it's hard to guess what will happen in the next century although it's only 9 years away. I think that heterogenous networks will be with us for quite some time. And there are a lot of tools and environments being developed that make the same assumption. All of these client-server applications that you see being developed are aimed at doing part of your task on one machine and part on another. If you look at IBM's SAA strategy they made the assumption that they have different hardware architectures that must work with one another. If you look around various hardware architectures you see that there are a lot of companies that are committed to the Intel architecture, others to IBM's Risc 6000, others to the Mips chip in the Ace Consortium, others to the Sun's Sparc platform. And many organizations have several different machines and as the network capabilities become greater and greater we see that users want to interconnect everything. They want to have whatever is on their desktop connected to every other machine, including PCs, workstations, minis, and mainframes, both at their local side and accross a geographically-distributed wide-area network. I think that the need for support for these kinds of distributed networks and heterogenous networks may be come greater rather than less over the coming years.

OP: *What is the meaning of «Visible Connections» then?*

Wasserman: Visible Connections refers to modularity which is the single most important design principle. Modularity is an important principle in software design and when we built our environment we published all the interfaces to all of our programs, because StP is not one huge executable program. More than 100 programs work together. Our tools can call one another and other people's tools can call some subset of our tools. So Visible Connections is a principle by which we built StP. We published the tool interfaces, the file formats, database schema, the templates that generate documents, code, database schemas, we published the annotations for every symbol and diagram and these are all accessible to users and modifiable if they want to do that. It makes our product very open and makes it very easy for people to integrate our tool with other people's tools, to built a complete environment. We used our own principles of Visible Connections to built the C Development Environment. So it's a very powerful principle for creating software systems.

OP: *Are you planning to port StP to platforms like PCs with Microchannel, Eisa or Macintosh architecture as well?*

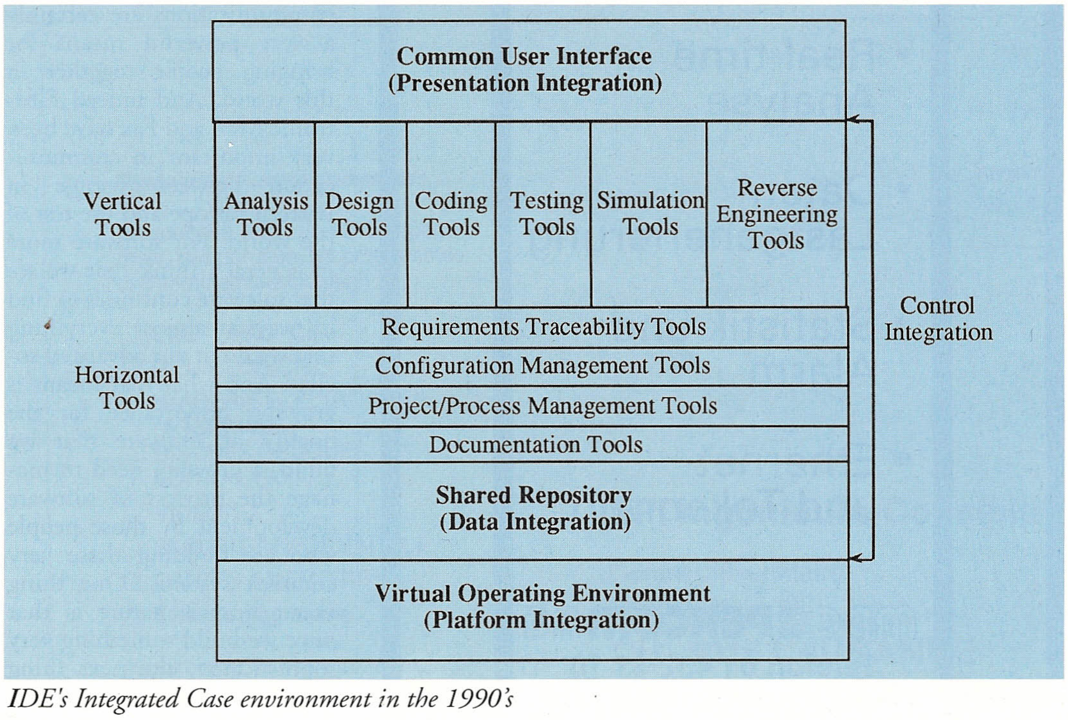
Wasserman: We have ported StP to a number of different hardware platforms. It runs on about 10 different hardware platforms. But from the beginning we designed StP to be a multi-user environment. We never built StP to run on MS-DOS or the Mac Finder or OS/2 which is also a single

Visible Connections is a principle by which we built StP.

user system although it has multi-tasking. So it is not the hardware architecture that is the problem. We can make StP run on an Intel 80386 or 80486 running SCO Unix on those machines. The standard PC architecture is not the problem; it's the fact that StP is a multi-user environment.

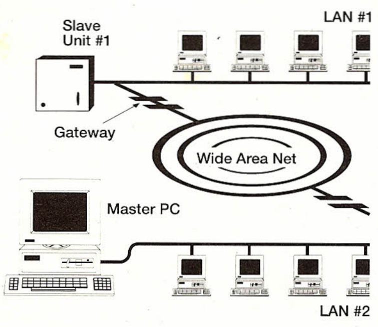
OP: *Would you explain IDE's Integrated Case environment for the 1990's?*

Wasserman: The Integrated Case environment is an attempt to model what a future integrated Case environment will look like. Users have a lot of different tools they want to use throughout their software development process: tools like StP's analysis and design, Saber C for programming, various testing tools, documentation tools and configuration management. And every organization has its own particular favourites. But what's important for everybody is to have a collection of tools that work nicely together. We talked earlier about the data integration. Users would like to have all these tools able to share data with one another through a repository. So the lower level in the diagram shows Shared Repository. So the implication is that all these tools will access the repository. I have shown the tools as vertical and horizontal tools. A vertical tool is typically used at one stage of the life cycle – compiler or analysis tool. And a horizontal tool is used across the whole life cycle. You should always do documentation, early and late in the project. You must always do project management to keep track of resources and your schedule. But you still want all these tools work together. We talked about data integration. There are three more kinds of integration in the diagram. Platform integration means that you want this whole environment to run on your network of machines transparently. You don't care where the tools are, what the hardware is, you just as a user of the environment want it to work. Then there is Presentation integration. What that means is it you'd like all the tools to have the same look and feel. Because when you begin to get a lot of different tools, if you have a different user interface for each one you go crazy. And the principle of the Macintosh is the right one. Every tool that you use has a consistent user interface, it has a consistent way of saving things to a clipboard which you can paste into another application. You want to have a similar capability in a Case environment. And we are seeing in the Unix community ideas like Open Look and Motif come forward as de-facto-standards for presentation integration. The last piece I called Control integration. You notice this on the diagram that it connects all the tools and the repository. And this is the capability of tools to dynamically communicate to each other. For example, if I'm using StP to do a design and I change the design, I want to be able to signal the technical publishing tool which has a document with that design and somehow tell it that you updated the design. If I generate code, I want to notify the compiler that there is some



IDE's Integrated Case environment in the 1990's

Netzwerktechnologie
Management Systeme



Lösung für Network-Management und Protokollanalyse

LANVista
bringt Sicherheit ins Netz

- Kontrolle über 7-OSI Ebenen
- Real-time Analyse
- Daten-Lastgenerierung
- Statistik und Alarm
- Ethernet und Tokenring

MDC PERCOM AG
Telefon 01 432 43 10
Téléphone 021 29 77 44

new code. So what I need is a dynamic tool to tool communication mechanism. Now there are several of these. The best known is the Broadcast Message Server in Hewlett-Packard's Softbench. Digital's Fuse and Sun's Tooltalk are very similar. The idea is that you can broadcast a message from a tool which is handled by the Tooltalk, Fuse or Broadcast Message Server, and then send out to other tools that will use that information to take some action. To conclude when you think about what you want this future integrated case environment to be: They must share date through a repository, they must run on heterogenous networks, must have a consistent user interface and it must be able to support tools that dynamically interact one another. That's the vision that we have of integrated Case environment.

OP: *What is your message for the future to all software users? What vision do you have for the future?*

Wasserman: As one thinks about the future in 1991, one has to think not only about software but about the society at large and about some of the amazing events that are happening in our world. Software plays an important role in all of this, the computer and its communications are certainly a very powerful means for bringing people together in this world. And indeed Electronic Mail and Fax have been very important in communication between people in Eastern Europe and the rest of the world. For software more in general I think that we see that software continues to find its way in almost everything that we do in our advanced society. And what that means is growing importance for the quality of software that we build, a growing need to manage the process of software development by those people who are building these very complex systems. One thing about human nature is that once we build something very sophisticated, the next thing we are going to do is to build something more sophisticated. And so the need to understand software design goes up continually. People are going to find that they need more and more standard methods and improved methods for designing and building the kind of complex systems that we expect to have. And here I think the technology that we have helped to bring to people can play a major role in building those systems.

OP: *As a Case developer you have quite a lot of responsibility that human beings will have it better in the future?*

Wasserman: We would like to think so. When we look at the applications that people are building, they affect the environment and they affect communication. We like to think that we can provide tools that can help people built software and applications that will be positive forces in the global community and society over a couple of years.

OP: *I wish you good luck for your product and your organization and I thank you for this interview with quite a famous person in the area of Case.*

Zusammenfassung

In den vergangenen Jahren ist die Bedeutung der Case-Werkzeuge stark gewachsen. Unternehmungen, die sich mit Software-Entwicklung beschäftigen, müssen sich vorsichtigerweise auf Multiuser-Case-Umgebungen konzentrieren, die sich jederzeit bei neuen Hardware- und Software-Vorteilen anpassen lassen. Dr. Anthony Wasserman ist einer der erfolgreichsten Case-Architekten und wurde in Genf über seine Sicht, Case zu gebrauchen, und über die Struktur von Software through Pictures (StP) befragt. Seine Meinung dürfte vielen Menschen helfen, die richtige Wahl bei der Software-Entwicklung und -Anwendung zu treffen.