The Year of Informatics

Niklaus Wirth, March 2008

Abstract

This tale begins with my personal impressions of the opening event of the Swiss Year of Informatics and ends with those about the opening of the new laboratory of Google. In between, I offer some remarks about the development of IT in general and software engineering in particular.

The Opening Event

It was to be a grandiose event, this January 28, the opening of the Year of Informatics, proposed and inaugurated by the Swiss Informatics Industries and sponsored by foundations, companies, and a big bank. Between 200 and 300 people convened in the Hyatt Plaza Hotel in Zurich's banking district, adjacent to the city's concert hall and surprisingly inconspicuous.

I had been lulled by dear colleague Gusti Zehnder to join the Patronizing Committee. I was assured that this did neither carry duties nor responsibilities. Hence I must consider it an honor. Later I learnt that about two dozen people had received the same honor. In any case, I thought it might be a good reason to attend this opening ceremony, furthermore being in the good company of the Federal Minister of Commerce and Trade. ☺

First on stage was the head of the Swiss IT-Industry Association, Stefan Arn, immediately followed by the moderator, who would introduce the speakers through the evening. They had engaged none less than the anchor man from Swiss television who heads the late news, a show of gossip and irrelevant excitements. He did his job very well, though, and gave interesting and witty, brief interviews with all speakers. The first of them was Zürich's mayor Elmar Ledergerber. He emphasised how important this event is for this country and (t)his city, and that already in the past Zürich had not been idle. Imagine – the mayor even mentioned my name, although surely he has not the faintest idea of who I am and what I did. Colleague Zehnder must have acted as his speech writer ☺.

This very same minister, preceded only by the welcoming address given by

The mayor was followed by the Minister of Commerce, Doris Leuthard, driven from Bern by Federal Limo, now delivering the opening speech. Her talk started with only a single minute's delay. It was very well conceived and very well presented, a blessing that people in informatics circles rarely can enjoy. She drove home the central point with refreshing clarity: Informatics had become a highly relevant subject in the country's business and economy. Everybody's life was becoming influenced by it, a field particularly attractive for young people seeking employment and career. Therefore, she said, it was surprising that we perceive a growing shortage of people in industry, and a fast declining number of students at universities. She urged that this

dichotomy must be explained and counteracted, and she declared this to be the purpose of this Year of Informatics.

The next speaker was Urs Hölzle, former student of ours and now a vice-director of Google in California. Also he spoke very well (something that he must have learnt in the US ☺). He emphasized the importance of informatics worldwide and the urgency of not letting this country slip behind. He wittingly pointed out that by scheduling this event at the end of January rather than its beginning, we had already lost one month out of twelve. He hardly needed to mention the success of Google. It is obvious: Within only three years, the Google Zürich Lab grew from 6 to about 400 employees.

The last speaker, the CEO of Migros, the largest Swiss retail store chain, presented a power-point sequence of bare facts and figures. The numbers of stores, daily customers, products, sales, logistics of supply, etc. were all staggering. They convinced the last doubter that without Informatics, everything would break down. Well done! The plain message: No informatics → no food → chaos.

This presentation was followed by a panel discussion with seven participants from industry, commerce, banks, city government, and a young assistant professor from the University of Zürich. They delivered their opinions about the causes of the menacing lack of qualified people in our prosperous profession, proposed what to do about it, and vowed that this year would be decisive.

The subsequent glass of wine was excellent, the snacks lousy. I met former colleagues from Lausanne, Bern, Berlin, and from various industries.

Some Personal Afterthoughts

During the subsequent nights I could not help but pondering about this event and about the deplored state of affairs. The reported facts were undoubtedly correct, the need for action and remedy undisputed. But was the diagnosis right? If not, how could actions be successful? No patent is cured by vitamin pills, if he has broken a leg!

A comprehensive answer to the raised questions is impossible. One of the reasons is that the word *Informatics* is sufficiently ill-defined. Different people mean different things when using the term. Certainly the majority sees the subject from a very different angle than I did, when I became excited about it some fifty years ago. Then the excitement was caused by the emergence of a new subject, a new technology with scientific underpinning. The excitement came from the challenge to construct powerful computers, and to develop methods and tools for programming them into useful artifacts.

Today, the term encompasses everything that has to do with computers. And this is almost everything. It ranges from repair services to preparing a slide show, from analyzing weather data to creating web pages, and in particular any office activity from typing to document preparation, budget control, and logistics. Out of ten thousand people working in IT, as informatics is called today, perhaps one is still designing computer hardware. And perhaps one out of a hundred is still designing software, is still programming. As an aside, this means we are making poor use of this most flexible

machine ever invented. Anyway; when talking about IT, today people think about business, money, and trade, and neither about engineering nor science.

Programming

The shortage of people, however, lies in the technical branch. Its core, essentially, is programming. Contrary to many claims, programming is difficult. It is a constructive, exacting activity, requiring precision, a high ability for abstract thinking, a care for details, experience and skills. Not everybody is suited for it. Add to it the fact that today's computer systems are enormously complex. They consist of thousands, even millions of lines of program text, and they are developed by teams. They are not designed once and then left alone for use. Instead, they *grow*. The demands change, their environment changes, and accordingly parts are added; mistakes appear, and accordingly parts are modified. Side-effects of these additions and changes are easily overlooked, and new errors are introduced. They may perhaps not have any immediate effects, go unnoticed for a long time, and remain hidden. But on a certain occasion they may pop up, with possibly disastrous consequences. Their diagnosis may be difficult, as it requires detailed familiarity with much of the system. Those who had designed it had meanwhile successfully moved to other places and tasks. Therefore, the job of maintaining software has become a nightmare.

A friend of mine, a highly competent designer, working at a large bank, employed to correct the worst cases, wrote to me recently the following:

> The bank is keeping me very busy with the depressing task of maintaining and amending existing software which is of very poor quality. It is really fire-fighting. Sometimes, I wonder if I have the stamina to take such a bad situation and improve it dramatically - there is simply too much to do! While a system is still working (just), even like Windows for example, there is little incentive or support to rip it out and start again with something of higher quality. What is more important for the business is what they call "time to market", in other words, the ability for a development team to deliver (perhaps mediocre) software rapidly, to satisfy an immediate business need. The cost of its ongoing support afterwards is not factored in.
>
> As I am a bit of a perfectionist, I find this to be deeply unsatisfying. Developers of varying skills come and go, there is no overall strong architectural guidance, and software must be delivered as fast as possible, with no time to develop tools to manage the self-inflicted complexity. These factors conspire to a very poor (but just sufficient) result. And often, I see the same mistakes being repeated. □

This picture certainly contrast severely from the future of IT so highly praised. In fact, the situation is often even worse. Huge programs are messy, and industry is inextricably caught in this nightmare of complexity. To try to disentangle such systems is worse than descending into a coal mine! The best idea would be to discard them and restart designing from scratch. But this is impossible too, because there are so many odd requirements to satisfy, and so many standards to conform with. The standards are a huge problem themselves. They are baroque and complicated, messy compromises to please everyone.

The only escape from this situation is, supposedly, employing tough managers and, if they fail – and they will - to hire more "coal miners", just like in the army, where more soldiers are mustered when a war threatens. However, in programming, this approach cannot succeed. Fred Brook, top manager to rescue IBM from a calamity with its OS/360, in 1965 wrote pointedly: "When a project is late, adding manpower makes it later". But in reality, we observe that the poorer programmers work, the more will be employed. An ideal insurance against unemployment!

Some Remedies

The inevitable conclusions on what is needed to salvage the sad situation *in the long run* are:

1. Attract more young people to the profession, and
2. Provide better education for programming, and
3. Eliminate the complex systems that are flaky and impossible to "maintain".

The first item seems to be easy: Send recognized experts and enthusiastic fans to our high-schools, and let them praise the glory of IT. I have my doubts, as to whether this measure will bring the expected success. There exists in our country an intrinsic aversion against technology. To use it is quite acceptable, but to engage in its spread is suspect. Technical subjects are also seen as too boring, too exacting, and no fun. The crux behind this attitude is the aversion against hard subjects, subjects that require precise thinking, where questions can be answered with "no" or "yes", where "perhaps" is not good enough, and where sentiment and emotion are eliminated. I do not have a ready recipe against this *malaise hélvétique*.

Education

Back in the 1980s Informatics finally entered curricula at high-schools in the form of a 24-hour course, i.e. two hours per week during a term. This has all been replaced by skill courses - still under the heading of Informatics - in which the use of computers is learnt. The students get exposed to popular programs like *Word* for text editing, *Excel* for the use of spread-sheets, and *PowerPoint* for the construction of sequences of slides for presentations. This is all quite useful, but it has not much to do with understanding computers and their principles. Such courses provide skills for immediate use, rather than insight and long-term knowledge.

Changing subjects in our school system appears most difficult. There is indeed reason for doubts about the quality of our education in scientific and technological fields. The Swiss traditionally display a stubborn pride in their schools. However, perennial pride carries the danger of causing insensibility, even blindness against change and emerging deficiencies. The Swiss are particularly proud of their unique system of apprenticeship, where the youngsters learn a trade under a master, while attending school two days per week. This was – and perhaps still is – a sound practice in traditional subjects from carpentry, metal works, office work, cooking or construction. But it works badly in informatics. Qualified masters teaching out of decades of experience hardly exist. If they were technically masterful, they were promoted to a management position long ago. Programming, the basis of our trade, is learnt by osmosis. Often the young people

finish their apprenticeship without understanding the basic principles, not to speak of good style.

At universities, matters are hardly brighter. Informatics has become a coarse conglomerate of widely differing subjects, almost without a core. The idea that computer scientists are basically engineers, designers of abstract machinery, is gradually fading away. Universities have lost touch with what industry needs, they have shifted from teaching basics of lasting value and truth to specific applications and specialties of transient nature. The trend of Informatics away from its foundations towards applications is explicable. It is the applications that make computers ultimately useful, and thus may provide the students motivation, fun, and the experience of success. Yet, in the end the trend is disastrous. It is as if physicists would skip over the laws of Newton, and those of aerodynamics, and immediately discuss cars and airplanes, or as if chemists would from the beginning concentrate on perfums and rocket fuels. I am afraid I do not have a recipe to reverse this trend either.

In short, our students do not learn serious programming anymore. It is as if mathematicians would never prove any hypothesis, electronics engineers would never design a circuit, chemists never analize a substance, and cooks never cook a meal. Programming is notably more than coding. It is the careful, deliberate, structured discipline of building a complex artifact.

Programming is not only hardly taught at universities, it is looked down upon as a menial task, although it is difficult to do well. As a consequence, a professional pride in the product is missing, and the programs are concocted, lacking style and quality, and remain impenetrable. Thus, no teacher is keen on carefully analyzing and criticizing them. Without feedback, the student can hardly make progress. How, then, can you expect solid craftsmanship when they leave school? As an aside, note that the obscurer a program is, the less the danger of it ever being inspected. ☺

Far away from this battle ground thrive the professors. They would crucify themselves rather than descend into the catacombs of coding and inspecting the students' programs. Having seen these concoctions, one can hardly blame them! The crux is that the professors must maintain the impression of being at the forefront, and they decide which languages, tools and methods are to be taught and used. But without practicing the craft themselves, how can they truly decide what is best? The easy solution is to select what everybody else uses, to join the bandwagon. And these are the utterly complex languages and tools that supposedly cater for the needs of everybody.

System Software

Under the term system software we understand the conglomerate of operating system, compilers, text editors, utilities, communication and mail system. The requirements imposed on system software are constantly expanding. Generality is the key and the main sales argument. System software must cater to all needs of all people. This is hard to achieve, perhaps impossible. But it is also the wrong goal.

As a result, system software is typically not planned, designed, and then built. Instead, it is constantly evolving. Extensions, changes, and corrections (!), called *updates*, follow each other rapidly. People who create these updates come and go. No wonder is

the outcome rarely satisfactory. Systems that outgrow their original purpose and design inevitably become bulky, inefficient, and unmanageable. If locomotives were constantly updated, if they were as complicated to use and as often malfunctioning as software does, they would not be allowed to operate in public service. If airplanes were built and "maintained" in this fashion, they would be barred from take-off.

The plain fact is that in general our software could be vastly more economical and effective than it actually is. Vastly! It is indeed surprising with which equanimity this inefficiency, this waste is tolerated, accepted as a fact of life. For example, even for small application programs, perhaps taking a few thousand bytes, the basic operating system is necessary that occupies dozens if not hundreds of megabytes. As a consequence, many computers quickly become insufficient and must be upgraded or replaced, when a new software release enters the house. There seems to be a malignant tendency to let software grow in order to keep up the demand for more hardware. People hardly complain. It is quite incredible.

Why are we forced to buy and load so much that we will never need and use? Why are we forced to carry along so much that is only wasteful? The unavoidability to use bulky monster systems is a sure sign of the immaturity of software engineering. We are still compelled to use monolithic monsters.

So what?! If memory capacity and processor speed incessantly grow and their cost (almost) vanishes, why not make use of them and be happily wasteful? Well, why not indeed? This is how modern reasoning seems to go. It thereby ignores that growing size carries with it growing complexity, complexity beyond the limits of intellectual manageability.

There is yet another aspect of the unhealthy role of monster systems. Applications must be built on top of a base system, a "platform". As the border line between system and application software vanishes, we see a whole hierarchy of building blocks, called *modules*. The higher modules draw service from the lower ones. This picture makes it evident that the quality of the clients, the applications, can at best be as good as that of the ones below. If the base is not trustworthy, how can the top ever be? The dire conclusion is that the monster systems that have invaded all our computers – and all our minds – must be replaced, if we insist on building software of high quality. The sooner the better!

A brief Interlude

But how can this ever happen? It is about as likely as the survival of mankind without oil and gas. We are in a terrible bind! I feel sad about these irreversible developments which had started so exciting 50 years ago.

The trouble is that I am still and often identified with Informatics and all its wonders and evils. I receive invitations to give talks, and it is unpleasant to decline. Recently I was asked to present a keynote at a Java programmers' European meeting, another at an Austrian society for the advancement of software engineering, to deliver speeches at universities in various countries, mostly about software engineering and its "future". I have the suspicion that I wrote the above diatribe to let my misgivings be known. Apparently invitations often come from people who have no inkling of neither my

attitude nor my work. Hence, on such occasions I have the choice of being either cooperative and flattering, or honest and disappointing. Also now: What the devil can I do to end this tale in a less pessimistic mood? Well, if you don't mind, read on!

The Google Event

In the evening of March 6 I followed an invitation to attend the opening ceremony of the new research and development center of Google. It was quite an event! I took the tram 13 and left one stop after Bahnhof Enge, as I had not quite managed – inept as I am - to determine which stop would be nearest to Brandschenkestrasse, using Googlemaps.

There was already a crowd in the reception lobby of the very large, new building on the grounds of the previously famous Hürlimann brewery, now transformed into an industrial park, all neatly paved with cobblestones. The gigantic, colorful neon sign on the roof quickly guided me to the right building.

Google started a small office four years ago with four people. They expanded and moved three years ago, first occupied a single floor, gradually took over more floors, and then the whole house, until this wasn't large enough either. Now the lab has expanded to 400 people from 40 nations, and they moved into this new location a few weeks ago.

The first person I met was Mr. Golliez, head of the Informatika 2008 effort. Then I spotted the President of ETH Zürich, a physicist, and I had an informative conversation. He introduced me to Mr. Fritz Schiesser, formerly Senator in Bern and now Chairman of the Board of the two ETHs and several federal research institutions. And, well, then he introduced me to the head of Google's archrival Microsoft Switzerland. When finally the call came to move into the lecture hall, I ended up in the front row next to Mrs. Fontana, heading ETH's fund raising office. But I did not see a single soul of the Department of Informatics of ETH. Once again, it celebrated its glorious absence. - I had had no idea the event was so prestigious; so, with a certain relief I noticed that many of the men did not wear a tie either …

The first speaker was none less than Mrs. Rita Fuhrer, President of the government of the canton Zürich, and minister for trade and commerce. She gave a very good speech, emphasizing how important it is for our city and canton that prestigious companies choose this city for their expansion into markets of the future. After her speech she discreetly vanished (probably moving to a next event), leaving behind her purse ☺. The next speaker was the head of the Zürich Lab (Straumann), and then the vice-director for Google's developments in Europe, Middle East and Africa (Mattos). They both did a good job, although perhaps a bit too long, which point they aggravated by repeatedly mentioning all the goodies that would expect the hungry folks during the following tour through the house.

The environment was already rather googlish. The room, much wider than deep, with about 150 people, was bathed in continuously changing colors emanating from various strong LED beamers. A strange glass construction suggesting mountain peaks in the middle, two projection areas left and right, colored lights from above and below. Even if the speaker says nothing, it's a show!

After the serious parts came two persons on stage clad in jeans, the man a true entertainer and salesman, with a strong American accent and style, talking fast and constantly with

jokes and gestures, and a woman, blond and radiant, who was introduced as the architect of the house and its new style of googlish culture. I enjoyed both, because they were really enthusiastic about what they had done. She talked about the new experience of designing something quite different from the usual, and about how to satisfy best the wishes of an unusual customer. I don't remember a thing of what *he* had said.

Finally, the audience was asked to congregate in groups around the tour guides. The guide to whom I was directed was a young woman who first talked French to me, but quickly revealed herself to be a genuine American from Seattle, having worked for some time in Belgium. She was not a computer scientist – and hence no technical talk was likely – but a former student of political economy (or economic politics). We passed more than half a dozen locations, including the massage room – with a pretty lady -, the room of rest – all dark with several aquaria with colorful fish in the wall and chairs like in a spa, the library with comfortable chairs and divans, dim lights, and a bar, panelled walls and books painted on it. There was the Swiss corner with ski lift gondolas in which you could sit and have a tête-à-tête, iglus equipped with a telephone for discussions, and at the very end the game room with game tables and a hamburger and hot dog stand, and beer, Budweiser, I guess.

At each stop there was food offered and something to drink. This ranged from seafood to cheese, but thank heavens also something more ordinary, like Gerstensuppe and sandwiches, wine and fruit drinks with a shot of vodka, and hamburgers at the very end.

Although there was a Dutchman in the group who apparently was a technical guy, and also a Zoogler – as the Zürich employees are called – there was no discussion of technical matters. The visitors seemed to be rather from the legal, journalistic, and business communities. Still, I looked around corners to the sides and saw several large rooms with computers, obviously work places for a dozen people. And indeed, there were some guys still working, having a late dinner at the same time. Quite obviously Google did not intend to present much of their work, but rather to show off with their flashy places and flexible working conditions, where work, fun, and relaxation are inextricably intertwined, entangled. Still Renée, the guide, showed me quickly how to find the nearest tram station on Googlemaps; so I learned something after all ☺.

Yes, it's a new culture, and probably difficult for established Swiss to swallow. I overheard some rather critical comments within the tour group, like "I don't need this" or "I couldn't stand this". Time will tell, how it works out in Zürich. So far, the majority of employees come from the US. – Yet, I was enjoying the show. Must confess that I had been somewhat prepared for it some 30 years ago. Then I entered the Xerox culture during a year's sabbatical leave. True, there were not so many colors, no Googlemaps, but there were beanbags in place of tables and chairs in the meeting room, but – how nice – individual offices with personal computers!

At the end, everybody gathered downstairs. Ample food and drinks again. This day must have cost Google a fortune. The celebrities seemed to have left before the tours had begun, but here were the Zooglers! Among them I met a Swiss, the one who had actually started the map idea, was successful, and then sold his start-up to Google. He had not come from the division for land surveillance (at ETH), not from Computer Science. And then I saw that blond architect and could not resist approaching her, asking where she had

acquired that typical American way of talking on stage. It turned out it had been in London, which I found odd. I detected an accent also in her German, and then she admitted being Russian. I told her I had been battling with the Russian languages for a few years, which seemed to please her greatly. Thereafter we continued chatting in Russian. A bit less lively, though! She had studied architecture in Moscow, and is now married to a Swiss with the unmistakably Swiss name Rüegg: Tatjana Rüegg-Basheva.

It was after 22:30 when I decided I had had enough drinks and it was time to sneak out. I walked through the nightly industrial park toward the next tram stop on a railway bridge. No. 13 ferried me to Paradeplatz, No. 11 to Stadelhofen, and the Forchbahn S18 to Forch. A 15 minute refreshing walk downhill brought me home. It was exactly midnight.

Next morning I read in the newspaper that Joe Weizenbaum had died in Potsdam. I had met him first in Berkeley in 1963. He then was a protagonist of the infamous Artificial Intelligence scene, of which I was and remain skeptical because of its boasting promises. As time went on, he also became skeptical, even antagonistic, a fervent critic of the glorification of computers and our fast growing dependence on them, their stultifying effect on many people. Lately I felt that he sometimes exaggerated. Yet I wonder what he would have thought of the Google event.