# Closing Remarks

## 1. INTRODUCTION

I was supposed to give the closing remarks, maybe because it's part of my name (Mark Plesko). I would first like to reiterate why should we have a PCaPAC in the first place. The obvious answer is that PCs are standard, cheap, stable and multi-platform. But so are screws. And we don't have a conference on using screws in control systems. So there must be something more to it. The main reason why PCs are so important for the control community is that PC technology brings a new paradigm to everybody. Concepts that were known for ages, like Windows components, now really are a standard of our everyday life and, compared to the ICALEPCS conference, this is much more obvious here in this workshop.

We use PCs in very small machines where we have one or two PCs or in very big systems, like the KEK where we have hundreds of PCs. In both cases, the same technology and the same concepts are used. That's why in this workshop everybody can learn from everybody. It's not just the big guys from the big guys and the small guys from the small guys.

So why PCaPAC at KEK?

Many small Japanese machines are completely controlled by PCs. And also PCs are used in many big machines. This workshop shows that Japan is very strong in the use of PCs. Actually I was amazed by the sophisticated control systems used in Japanese accelerator laboratories. Even in very small systems and small experiments. And I was very happy to come here, so I could see first-hand and talk to the people who really do the work. This is something you should look forward to at the next PCaPACs. Some people remember me looking into the crystal ball at ICALEPCS97 and predicting the future, when all I had to do was go to Japan and see the future in front of me. So I would gladly accept Abe-sans's offer - it's not an official offer, but he mentioned it - that maybe KEK will host PCaPAC again in 2002.

## 2. CITATIONS

I will try now to go over the highlights of this workshop. I have listened to all the talks very carefully and I have looked at all the posters. I have tried to point out what is good and what is really very good. But it was very difficult, because, first of all, PCs were really used everywhere. And, second, there were so many clever and ingenious uses of PCs and PC technology that it's very difficult to single out one or two papers for which one could say "this is the way to go, this is the right approach". This workshop really proved that there is tremendous energy in the people using PCs and the community is more alive than ever in producing excellent work and excellent results. So, instead, I tried to pick a few citations made by people during the conference:

- There was a very interesting one that I liked very much by Nishimura-san. He said, "Java is for young people". And there is probably not a single person here who has not used Java yet. So we are all young and we will stay young.
- Duval-san gave a very interesting definition of components. However, Hill-san proved to us that old office PCs are reusable components, too.
- Finally, the one, which I like most, was by Hajima-san. I have modified it slightly to make it sound even nicer: Learn Java once, be happy forever.

## 3. THE MAIN DIRECTIONS

Let’s still try to distinguish the main directions given here at PCaPAC. I really tried to cover all the talks, but it’s very difficult to go through all of them in this talk. So I’ll try to just briefly mention the concepts that were brought out:

- There was a fascinating list of applications. They are too numerous to be mentioned all, but this is definitely one of the main topics of people’s work, although we don’t present them really at workshops. At workshops we talk more about architecture, about concepts, and only briefly glimpse at the real work. For example, somebody showed a screen-shot of an application for a moment and then it disappeared while there is really a lot of work behind it. It would be very interesting to see all these applications in action and discuss what different people have done and why.
- Complex GUIs are the visual tools for everyone and everywhere. We have seen excellent charting tools. Many have been built in house, because it turned out that the commercial charting tools are not so applicable to control systems.
- Visual Basic and ActiveX are, of course, the technology of choice for people who use Windows NT. They are also used in browsers nowadays quite intensively.
- Still many Web servers are based on pure HTML, which can even be used for simple synoptic. There was a poster on using juts plain forms and HTML tags to present an, albeit simple, though active synoptic. However, many Web applications use active server pages, while seems that CGI is now more or less out. Of course Java displays of all kinds are used, from very simple input to very complex panels.
- This brings me to Java. The global consensus of this workshop is that “Java...yes, we will use Java”. Everybody uses Java. However, the combination with CORBA is not here - yet. There was some discussion on CORBA, but there is no real running control system based on CORBA yet. But it will come.
- Another interesting point is managing controls environments, which can be quite inhomogeneous. We have seen very small control systems where this is not a problem. But there are some very big systems, like these in KEK, where it is very important to have very powerful and clever tools to manage the network and the PCs. This is a topic that will ever gain in importance as the features of systems become more complex.
- We have seen that commercial SCADA systems can be applied. The question is, can we use commercial SCADA systems also for accelerator control or do we use them just for the GUI front end
- PC databases with ODBC or JDBC: There are a lot of control systems going through databases. Often, these are not even special runtime, memory resident, or whatever databases, but just pure Microsoft SQL, Access or similar. It is a well-understood and easy tool to make relatively easy and simple control systems.
- The PC, of course, doesn't have enough I/O channels, so it is standard practice to extend it with either fieldbuses, VME, PLCs, or a combination of them.
- Another very interesting topic that I saw at the poster session were some very interesting demonstrations from Japanese companies. They had very good products. We should check out the Japanese suppliers more than we are used to. Maybe there is a language barrier or we're just more used to American suppliers. But what we have seen here is very promising.

## 4. INTERESTING DEVELOPMENTS OUTSIDE THE MAIN STREAM

Now, let’s look at interesting developments that are not quite in the mainstream.

- There have been PCs put in robust chassises – and we turn a desktop PC into an industry PC.
- PCs were used in COD correction for special applications,
- PCs for voice alarms,
- Interlock control with PCs,
- we use EPICS on PCs and PCs as clients for EPICS,
- Java 3-D for visualization,
- Scripting tools: it’s a very interesting topic how to use scripting for simple control actions,
- touch panels for PCs and PCs as touch panels,
- fast compression algorithms, which are very efficient for data acquisition, because they have a very short compression time,
- and object-oriented databases, which are not yet there but it seems that something is going on and they are coming.

## 5. SMART SOLUTIONS

Now comes a purely personal pick. I have looked into a few things which I think are smart solutions. By smart solutions I usually consider something which is a very small trick: not to go the straight, hard way but to go around or to find a very cheap solution to a problem. There have been other smart solutions, but these are the ones I have found to be very smart.

- There was a poster on using the Web for remote display. Just by making a screen dump from a local control PC and sending it to the Web. There was no “PC anywhere”; no remote control. You use the plain Web, but you see the display of the controls PC anywhere in the world.
- Using HTML with script tags for synoptic. We talked about this already.
- There was a nice talk on distributed-shared memory, using it actually remotely over two PCs.
- We all know that writing drivers for Windows NT is not so pleasant, at least not the first one. It takes a few months. But there was a CAMAC-SCSI interface. You just plug your interface into the SCSI port of Windows NT. You don't have to write any drivers.
- Use of the mouse driver for analog knob control: You just disassemble the sensors of the mouse and make two very nice knobs. Everything can be controlled and it's a very cheap solution.
- People simulate devices to develop control systems. So we don't actually need an accelerator anymore. Sometimes this is convenient. We can kick out the physicist and just run our control systems as we want.
- People even cut cables of analog controlled power supplies to be able to insert a control system.
- Video blaster image processing: instead of playing games, we can control our accelerator - or the other way around. It depends how we feel.
- The last contribution that I mention, but far from the least interesting, is the talk we just heard recently. It's a very nice combination of a low cost, actually almost free, control system: commercial motor controller, GPIB to RS-232 converter, freeBSD operating system and Pearl as the application programming language.

## 6. OPEN ISSUES

There were also some open issues discussed at the workshop, which are important to mention. Again, they are my personal picks - I may exaggerate.

- Microsoft versus the world: When we talk about PCs, should we talk about Microsoft or should we talk about something else, too?
  - LINUX is a strong competitor to Windows NT. My good old friend, Bill Gates, has the same opinion. Maybe you heard about the Halloween papers. In November last year, some internal, highly confidential memos of Microsoft were released on the Internet. In those memos, it was revealed that Microsoft considers LINUX as the strongest competitor to Microsoft in the server market. The reason is that the turnaround time for one release of LINUX is much smaller than for Windows NT. Because LINUX, being in the public domain, has so many people working on it, 24 hours a day, in all the time zones. So Microsoft is lagging behind interesting development.
  - The other point is Active-X versus Java. Sun says "write Java once, run it anywhere". Microsoft says "write Active-X, buy Microsoft everywhere". It could also work. So we will see what the future will bring.
  - And the same applies for DCOM versus CORBA.
- Real-time operating systems of PCs are becoming strong. There were some interesting posters and talks on real-time operating systems with LINUX, but there are also some real-time kernels, which can be placed below Windows NT. There are some commercial solutions for that.
- Something which has not been mentioned at this workshop, but which is a trend I see in Europe, is the use of industrial PCs instead of PLCs. PLCs are propriety solutions. You can get an industrial PC that also runs the same ladder logic programs. So maybe PCs will replace even PLCs. In that case we would have an open solution also at the fieldbus level. Everybody will have industrial PCs and everybody can write their own programs, not limited anymore to propriety protocols.
- I mentioned commercial SCADA previously. I wrote that point yesterday. But from what I heard today at the presentation of Momal-san, it is a viable solution to just use a commercial SCADA system. Of course the question is where is the fun factor? Are we just going to implement commercial SCADAs ? Or do we want to have fun playing with our tools, Java, CORBA and all those nice things. So we don't tell our bosses that it can be done.
- An interesting design decision for control panels has been raised during this workshop. Should we use dedicated GUI builders, for example MEDM, which is the basis for EPICS, or DDD, which is used at DOOCS, or should we rather use components: Java beans or Active-X components and put them together in a commercial visual development environment. The answer was given by Mutoh-san: combine both. The builders that we make, like DDD and MEDM, should be able to accept foreign components. In this case, we can get the best of both worlds. We get all the commercial components, yet we can provide our own builders, which are accelerator specific.
- One question is when to upgrade hardware. Everybody now says Windows 98, Windows 99, Windows whatever. But we know Windows 3.11, even DOS, is working quite well. So if the systems are running well, what to decide, when to do it?
- What about moving code between UNIX and PC? There are many problems related to that. And there was a poster

that gave the answer. It said, Java is the future. Of course, it could happen that Microsoft becomes the only operating system provider and then there's again no problem.

- Looking at how to make things cheap, we saw two approaches. One is using free software, which comes from the outside world: LINUX, public domain, freeware. There is even a CORBA implementation, which is freeware. Java, in principle, is for free. At least the language, since you can use vi to write in Java. The other idea for saving money and time is to share. I have listed three consecutive steps. Sharing components is the most obvious one. For this workshop, it's ideal. Everybody says what he or she has done. They show their components and maybe even some commercial components. If they look nice, we take them. The next step is sharing controls components. Here we have to agree on some common communication, like the ACOP OCX. If you agree on something ACOP accepts, then we can share controls components. The third step would be that we share auto configurable components. So really we use components that we place somewhere in the builder and the control system runs. Of course here we need an agreement. Maybe the first step was suggested by Abe-san with a COACK, an accelerator kernel. We'll talk about this in a few seconds.

## 7. HIGHLIGHTS

Having gone over the plethora of contributions to the workshop, I still tried to look at what could be the highlights. So I have finally chosen three of them: for the first, second and third place or three equals, respectively – take them as you like.

The KEK control systems at the various accelerator systems, which you have seen on Thursday, really is the number one highlight of this workshop. The sheer number of component-based, visual-based applications, the diversity of PC usage is overwhelming. This conference was dictated by what we have seen here in this place. I have singled out two topics that were particularly surprising, because I didn't know of them before. It was the Plug and Play driver-less I/O modules, which were demonstrated. This is the way to go. It's not Plug and Pray as we say in Europe, but it's Plug and Play. In Japanese it sounds very similar, but the difference is very important. So you just get a component which has all the software already loaded into it. You plug it in, download and run it. Another interesting concept that I saw in the Photon factory was the integration of HP-VEE into control systems servers. So this is really an example of embedding a commercial SCADA or control system, not of using it separately, because the HP-VEE was inside the control system and attached to it from two sides. HP-VEE took raw data from an I/O point, did some complex calculation with it and then sent it back to the control system into a top level Java application. So HP-VEE became one component of the control system.

Highlight number two is the idea of a component-oriented accelerator control kernel - COACK. ACOP is only the first part of it, at its lowest layer. But other things also belong here, like databases, components, etc.. Many presentations that have been shown here implicitly fit into this scheme.

The third highlight, DOOCS, is really remarkable, because it's just the other way from the commercial component-oriented control kernel. First of all, DOOCS is, as has been said by Rehlich-san, an object-oriented system from the device up to the screen. This is an excellent concept, which is exactly the same as the components concept. The server provides all the data, including locations and properties. However, we have to imagine that only two people have done this work completely in-house. They didn't use any components. They didn't buy any commercial tools. They sat down and wrote the communication protocol themselves. This is in itself a remarkable task. Compare it to Microsoft who has I don’t know how many software engineers and they have two. This is the proof that we can still do something completely for ourselves. We don’t have to buy everything. And the fun factor is still there.

## 8. THE LESSONS LEARNED

Which lessons have we learned at this conference? One lesson, which I would like to point out as the most important one is that PCs provide a homogeneous environment. This fact accelerates development very much. I give only two examples, because I can quote the numbers that I have seen on the posters. The ConSys control system developed in Denmark: It took them one man-year to design the system and another half man-year to implement the control systems kernel and two and a half man-years to make all the applications. That’s all! Another example, which I have seen on practically all the posters and wherever I came to in the tour: I asked the people “OK nice panel, but how long did it take you to build the panel?” Well, with Visual Basic maybe two to three days. So, if you have a nice environment, which is very homogeneous, which is familiar to you because of other tools like Word and Excel, then it really helps you to develop a control system.

## 9. CONCLUSIONS

When we conclude, we look back not only at the technical part of the conference but also the organizational and human part of the conference. Here, I can find no words. I can only say that this was an excellent workshop to the third (cubed). By now we Westerners -- most of the people here have been to Japan before -- are used to Japanese perfection in all fields. Japan is not an exotic country for us anymore. It is the leader in technology. It's one of the leading countries in the world. However, I am pleased to say that the organizational skills, the friendliness, the kindness and the help given by the organizers to all participants really exceeded even the very high Japanese standards by far.

We should thank Abe-san for his personal involvement, for really doing everything that was possible. I could tell you the story that happened to my colleague, and me but the same thing probably happened to many people here. So this is the time to put our hands together and applaud to Abe-san. And who was backed by the excellent help of Kurokawa-san, Kishiro-san and Mrs. Hayashi-san, not to forget our ladies in the office. They were hidden all the time, but they did all the work. Now is the right time to thank them very much.   And I would like to give my personal thanks for their personal and technical help during the conference to Nigorikawa san, Kosuge san, and Shirakawa san.

It's time to say good-bye. I’ll try to say it in Japanese: PCaPAC 2000 de aimasho (See you at PCaPAC2000).   But life goes on. So, see you at PCaPAC 2002 and maybe even later. Of course technology will change, things change, PCs change, industries change. However, friendship is forever. Good friends are forever. People don't change. Whatever will happen with technology, I believe that this community will stay together. We will stay friends for all our lives. I'm deeply moved by the friendship of our hosts and also by the colleagues in the workshop. We will meet again and again. So see you at whatever “aPAC 2010”. Thank you very much.