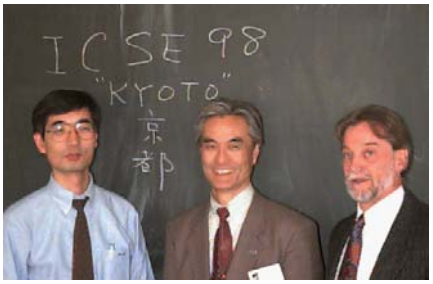
More ICSEs on the Way!

...continued from page 1

a venue for such an *international* event. He also stressed the importance of holding such an event in Asia, which he said was an area with many talented and productive software engineering professionals and researchers. From a technical perspective, Futatsugi and Kemmerer highlighted their intent on expanding the ICSE programme to address *applications* such as networking, multimedia and databases, all of which impact the way software is engineered. More details on ICSE-98 will be available in next year's WOW!



ICSE-98 Organisers: *(from left to right): Futatsugi, Torii, and Kemmerer*

After Kyoto, ICSE returns to the USA with Prof. Barry Boehm as General Chair. The proposed programme co-chairs are Profs. David Garlan and Jeff Kramer.

I've heard some rumours about ICSE-2000, but I'm afraid I can't say more! I have my own views about where I'd like to see ICSE in the year 2000, but you will have to talk to me

Personals

Lonely academic (swm) seeks industrial partner for long-term, fulfilling relationship. Sense of humour essential. Send photo. Box 96.

Like to experiment? Fed up with just thinking about it? Experienced s/w designers are waiting to indulge your fantasies. 1-900-EMPIRICAL.

Why are we here?

By Jyrki Kontio

We surveyed a sample of participants in the ICSE-18 in order to find out their primary motivation for attending the conference was. We set out our reporter disguised as a professionally looking conference participant. We randomly picked 20 participants and asked them whether the primary reason for attendance was to learn some new technical information or to work on your contact network. It turned out that for most people, the primary reason for attending the conference is networking; i.e., to establish or maintain contacts withother people. Sixty percent of the participants reported this as the main reason for participation, whereas 40% said that new technical information was the reason for attending the conference. According to our pocket calculators, there did not seem to be statistically significant differences between the responses from industry and academia, the same 60/40 ratio for contacts/technical information seemed to hold.

We also asked whether any outstanding or new piece of information had been particularly interesting. Most respondents were able to mention at least one new technical piece of information that they had found useful during the first day of the conference.

Since many of the purists in the conference may want to find out more about the sampling method we used, here are some details: we only included participants who seemed to be awake and aware of what is going on in the conference. We did have to exclude some responses, one person could not make up his mind, one person said that he came to spy on a vendor (who failed to show up), one person claimed to come just for fun, and two respondents refused to take this survey seriously.

ICSE-18 Window on the World

Editor-in-Chief: Bashar Nuseibeh (Imperial College London)

Reporters:
Boris Bokowski (FU Berlin)
Steve Easterbrook (NASA/WVU)
Wolfgang Emmerich (City Univ, London)
Christian Kanele (FU Berlin)
Jyrki Kontio (Univ. of Maryland)
Stefan Tai (TU Berlin)
Scott Tilley (SEI)
Will Tracz (Loral Federal Systems)
Ken Wong (Univ. of Victoria)

Photographer: Keng Ng (Imperial College)

Technical Support: Sabine Lembke, Lutz Friedel and **Stephanos Mantoulidis** (TU Berlin)

Technical Coordination: Ingo Claßen (Fraunhofer ISST, TU Berlin)

Letters to the Editor

Dear Editor,

I regret that my long-standing, general concerns about the development of our field have been incorrectly interpreted as a personal attack on the planners of this conference. I hope that others will look at the real issues.

David Parnas

Contest Reminder!!!

■

neers were familiar and had used before, rather than trying to come up with something fundamentally new. Harel's major contribution was to resolve a scalability problem. Equipped with a solid theoretical background, he went further than only defining a diagrammatic notation, but developed a visual formal language including the formal specification of its behavioural semantics. This enabled engineers to discuss specifications and provided the foundation for a company (i-Logix Inc.) he co-founded to develop the Statemate product. Without a reliable environment, such as Statemate, the complexity of specifications could have never been handled in an industrial setting.

Even with all these measures, his approach would not have been bound to success. 'Back in 1987', Harel said, 'when we tried to bring statecharts into practice we felt as through we were trying to sell silver cutlery to an African tribe'. It was a certain degree of stubbornness and persistence, combined with the patience required to get industry to appreciate statecharts that made the approach take-off.

In his tutorial yesterday, Harel was explicit in mentioning the repeated failure of publishing his first article on statecharts in (major) software engineering journals, including CACM, IEEE Computer and IEEE Software. It was eventually published outside the community in Science of Computer Programming.

Metrics on Participants

By Christian Kanele

In the spirit of current software engineering trends, WOW conducted a survey of 70 ICSE-18 delegates. Shown below are the questions asked and the percentage of "yes" answers.

(1) Have you heard about this year's ICSE-newsletter WOW?

YES: 90%.

(2) Did you have breakfast this morning?

YES: 71%.

Careful analysis of these numbers clearly reveals that WOW is more popular than breakfast.

Microsoft's Magic Formula

By Ken Wong

Industrial experience reports of large, real-world software projects are an important focus of ICSE-18. Chip Anderson presented a lively talk about how Microsoft ships software on time. Actually, Chip began with some history of his development work at Microsoft, which includes the Windows 3 shell, program manager, file manager, and task list. If you're going crazy using these programs, you can blame it on Chip. In 1991, he joined Microsoft consulting services, and advised other companies on matters such as "development discipline"; his talk was essentially a cut down version of a three-day course.

The Windows 3.0 product is a graphical environment for Intel PCs, with 80 percent written in Microsoft C and 20 percent in Intel assembler. Internal tools are used for source control, issue tracking, and automatic testing. Frequent, once-a-week builds of the software form the heartbeat of the development process. Combined with quick, automated tests, and one tester for each developer, there is a focus on stability and "zero defects".

The talk also addressed Microsoft's team-of-peers model, which highlighted the importance of people and organisation over tools and metrics. Moreover, there appears to be a mindset of teams working on products rather than "systems". In a product-oriented mindset, people tend to work harder, with more pride and personal commitment. For systems, especially legacy ones where you did not design it, there is very little feeling of ownership.

The team for Windows 3.0 is relatively small, about 50 people organised as a team-of-peers model. In this model, there are six goals and six corresponding, well-defined roles. The main idea is that everyone knows each other's responsibilities, there is no overseer, there is a strong sense of ownership, and the process is controlled via consensus.

The development life-cycle follows a spiral model, with versioned releases. That is, it is often possible to produce 80 percent of the functionality in 20 percent of the time. This functionality can be released earlier, with the remaining portion scheduled for the next cycle through the model. When a product ships, there is no "finality" about it; you revisit and re-analyse the software in a post-implementation review to continuously improve the product.

Anderson's talk is available at http://www.mindspring.com/~chipa/icse96.html.

Badge Colouring

By Will Tracz

Some of you may be wondering about the algorithm that was used to assign attendee badges.

White is the most predominant badge; if you are attending the conference and (optionally) any other event, and are not an author, PC member, or organizer the conference, then you should be wearing a white name badge. *Yellow* indicates ICSE "staff" and includes PC members serving as session chairs. A *Blue* badge distinguishes you as an author or PC member. A *Green* badge means you went to a workshop only; *Gray* means you went to a Tutorial only (or a Tutorial and a Workshop). Finally, *Red* is reserved for Exhibitors. Now you know.

results. Process technology is setting in place the foundations upon which these capabilities are being built. They are the foundations upon which process evolution, and the systematic, demonstrable, engineered improvement of processes will be based. They are quintessentially about software development agility and prudent innovation.

The CMM can support the testing of processes for certain quality attributes, but relying solely upon post hoc testing is always risky. Software process technology provides the needed complement to this approach by making software processes tangible, engineerable, rapidly evolvable to assure quality improvement and cost and risk reduction. This boils down to engineering quality in – to products and processes – by treating them as first-class objects of scientific and engineering inquiry and discipline. Over the past decade ICSE has been a premier venue for presentation of work that is leading to the maturation of these scientific and engineering disciplines. The community would do well to distinguish it from other work in process that is in need of just this scientific and engineering complement.

Audience Comments

- "I cannot disagree with DeMarco because generally the "giants" of Software Engineering are not to be disagreed with."
- "I wonder, if we would change to more mobility in the next years, what will be the 'armour' phase that follows afterwards?"
- "I agree with DeMarco: This is what I put forward all the time in my organization."
- "The last part of DeMarco's talk was a little weak. What exactly is the 'essence'? He could not give a sensible definition for it."
- "As I work on process improvement, I don't want to agree with DeMarco, especially since he likes to provoke people. But it made me think..."

Wojtek on DeMarco

By Wojtek Kozaczynski

If I were to paraphrase Tom's message I would write: "Don't mind the technology, the methods and the processes, those are easy, or the accidental aspects of software systems development. What we [I assume Tom meant software engineering profession] should concentrate on, are the business and social and the psychological aspects of systems building instead".

There is a large body of evidence that many (if not the majority) of software projects fail due to non-technical reasons – there is no controversy here. Does it mean, however, that software architects, designers, and engineers should neglect the tools of their trade and become businessman and psychologists. I don't believe that it does. The systems are constantly becoming larger and more complex. Building them will require more and better software engineering skills, not fewer and more "mechanical" skills.

What I hope Tom meant to say, is that a successful software project can be completed only by a team which collectively possesses the right mixture of skills. These skills should come from roughly three domains: process engineering (I mean our client business process engineering), change management (I mean helping the client jump through all kinds of people-related hoops while developing and deploying the system), and software engineering. Certainly, a good software engineer should be trained to recognize that some of the issues s/he is facing are non-technical and should be handled by differently trained people. Recognizing this fact is, in my opinion, a much better approach than pretending that we can be as good businessmen and psychologist as we can be software professionals. This will never be true and there is no reason to try to make it true.

I strongly agree with Tom that large and complex systems cannot be built by engineers alone, but have to be built by multi-disciplinary teams. But guess what, other engineering domains discovered this one a long time ago. For example, the car industry employs the best engineers and artists to design car bodies together.

So instead of trying to become jacks of all trades, we should invite other professions to work with us. By the way, Andersen Consulting (which happens to be my company) has been practicing this rule for many years with a great success.

On Selling Statecharts

By Wolfgang Emmerich

Is David Harel the software engineering underdog?

His statecharts are a visual, yet formal, technique that made a successful transition into industrial practice. Statecharts are used in avionics, automobile industry, medical informatics, telecommunications, chemical process control and business processes. What were the keys to this success?

Harel conceived of statecharts while consulting at the Israeli avionics industry. Rather than trying to sell a piece of his research as clothing that would not fit, Harel invented a fitting dedicated solution. It evolved from improving finite state machines, with which avionics engi-

David Harel: *They didn't buy it!*

Conflict Resolution, Win-Win, and Architecting

By Barry Boehm

Not surprisingly, I agree totally with Tom DeMarco's conclusions that conflict resolution and negotiation techniques are critical to software success, and that stakeholder Win-Win models are good approaches for realizing them. I've been using Win-Win techniques at TRW, DARPA and USC since the mid-80's. I've found them effective in anticipating and resolving potential stakeholder conflicts which otherwise would have sunk the project.

Since joining USC, I've been evolving a set of methods and tools to support general use of a Win-Win approach to software requirements engineering and architecting (an architect is a good role model of a technical expert who uses a stakeholder Win-Win approach to, in Tom DeMarco's words, lead from a position of no institutional power to successfully satisfy multiple stakeholders in the definition of a building complex.)

The Win-Win approach involves the following basic steps performed by stakeholders and an architect-facilitator (details and feedback loops are suppressed here).

1. Identify stakeholders' Win Conditions.
2. Identify Issues involving Win Condition conflicts.
3. Formulate and evaluate Options addressing the Issues.
4. Formulate, vote on, and adopt Agreements on mutually satisfactory Options.

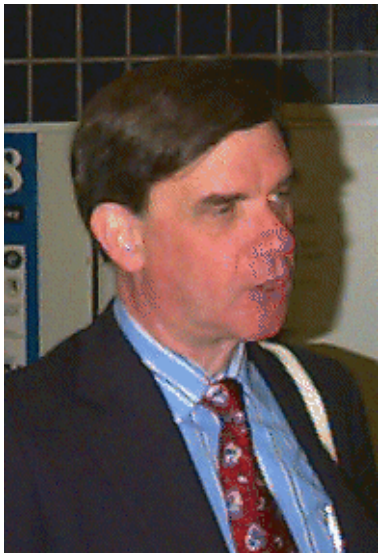
At USC, we've built a groupware support system to help distributed stakeholders operate in this way. We've also defined a Win-Win Spiral Model to integrate Win-Win with risk management. More information on those is available at http://sunset.esc.edu/

Some further good sources are Fisher and Ury's *Getting to Yes* (Penguin Books), Eberhardt Rachtin's *Systems Architecting* (Prentice Hall), and Christopher Alexander's *Notes on the Synthesis of Form*. The European NATURE project has also done good research in the area.

Process Technology and the CMM

By Lee Osterweil

Well, now I'm really starting to get mad. After all these years during which all of us have been writing all of these papers about software process technologies, so many people are still equating "software process" with CMM-based process improvement. Maybe the skeptics and critics are right and people aren't paying much attention to ICSE – and all of those papers about developing a (software-like) technology of engineering software processes.



Note to Tom DeMarco (and the legions who make the same mistake): the CMM should be thought of as an acceptance testplan for software processes. But there is also a complementary, strong, maturing discipline of developing software processes as first-class scientific and engineering artifacts. Those who develop their software (process software included!) with the primary aim of just passing an acceptance test are making a big mistake. It leads to gaming, conserva-tive, play-it-safe mentalities and strategies that do indeed ossify one and one's organization. No great new insight there. That's an old story. But process technology is a different story.

Helping software development (and other) organizations become nimble, resilient, adaptable, and able to turn on a dime (or pfennig) is a key goal for many (including most of us process technologists). And the key to reaching this goal seems to us to be having a strong, sure sense of what you are trying to do (your software product requirements) why you are doing it (your process and product requirements), and how you are doing it (your process itself). Being sure about these things leads to increased comfort about taking risks, experimenting with new software development approaches, and being able to tack with the winds of change. Being sure about these things means being able to reason about them with confidence, and that means being able to rely upon tangible realizations of process requirements, models, code, and test

quent decisions about which projects to adopt. Projects that are very similar to previous projects will be easy to adopt; riskier projects will not. The company that decides to experiment with (say) Java might have to discard its existing process, but is likely to reap bigger pay-offs than a company that carries on doing the same thing year after year.

This relates to DeMarco's next paradox: humans are able to react quickly to rapid change, but tend to ignore slow change. Hence "we may be getting better and better at doing the things that are less and less worth doing". Because the change is gradual, we tend to ignore it.

The remaining paradox is that we do not feel we are reaping the benefits of the effort put into reuse, when in fact we are. The benefits of reuse show up in packages such as PowerBuilder and Visual Basic, which allow us to reuse past experience. This is, in part, because reuse is only possible if you invest heavily in the thing you want to reuse.

Perhaps the key to DeMarco's analysis lay in his re-interpretation of the notorious baggage handling system at Denver airport. The system is now up and running, and has already repaid the entire development costs, including hardware. 'Failure' lay not in any technical problem (although there were such problems), but in the failure to plan for the risk that the software might be late. And of course we all know that software is often late. DeMarco is a recent convert to risk management, but now holds it up as one of two areas of optimism in software engineering.

His second source of optimism is the recent work on conflict resolution. Software engineers tend to think that they are developing software, but really they are designing organisations. This must be accomplished with a great deal of negotiation, especially now that software projects often lack management backing. Software developers must lead from a position of low (organisational) power, and hence need to become mediators.

An interview with Tom DeMarco

By Steve Easterbrook

I had an opportunity to interview Tom after his keynote address yesterday morning. He declines lunch: he and his wife want to make the most of visiting Berlin. As we chat I become aware that we're dancing around the central theme of conflict resolution, and especially the importance of negotiation in software engineering.

Before we get our teeth into this central issue, I ask him about his new book, "Why does Software Cost So Much?". He describes the contents: 24 essays on organisational change, management, measurement, team structures and other such "soft" issues. We discuss one of the essays, "Mad about Measurement". He describes it as a sombre retrospective on the history of measurement. His argument is that extrinsic metrics tend to interfere with intrinsic motivators. This leads us on to more pressing matters: the themes of his keynote address. He is keen to advertise the book he mentioned in his talk, over and above his own. The book is Thomas C. Schelling's "The Strategy of Conflict", and we immediately get sidetracked into a discussion of the history of warfare. We both spend some time admiring Schelling's work at Harvard, and especially the initiative to teach kids negotiation and mediation skills.

The discussion moves on to the benefits (or otherwise) of process technology. Of course, I'm hoping for a scoop, picking up on the question of whether people are more likely to be laid off in a more mature (by CMM standards) organisation. DeMarco treads a fine line here. He denies that the focus on process is harmful: rather his position is that it only makes a marginal impact on the problems of software engineering. Again he uses Brook's distinction between accident and essence for software problems. I ask whether academic research is doomed to tackle only the accidental problems, because of the need to bite off a manageable chunk. He agrees to some extent, and describes a need for boldness in research. Inevitably this means focusing on the softer problems of software engineering, and less on the (easier) technological problems.

Above all, DeMarco expresses a desire to get beyond the simple arguments of what the CMM is good for. Such arguments are fruitless. Improving software process is a good thing; there's no denying that. But he doesn't want to focus on the process argument. He wants instead to see the community 'switch gears' and focus on the 'essential' problems of software engineering. His twin hopes are risk management and conflict resolution. He expresses admiration for Carr's work at the SEI on risk management, and tells me stories of projects that failed because of a day by day failure to manage development risk. I chip in with stories of my own, and we converge on the Challenger disaster as an object lesson in failure to manage development risk. He also expresses admiration for Humphrey's work on the personal software process: "anything that helps to equip people with better skills is important".

I begin to see the future sketched out: Schelling teaches high school kids how to be good mediators, and when they grow up to be software engineers, they hone these skills through the personal software process. And of course, they take risk seriously. Lister summed it up well: "Risk management is project management for adults".

As DeMarco heads back to the hotel, I am left with the impression that a conflict with the process technology advocates is resolved. Or if not resolved, at least we're getting to the real issues, which is an essential first step in any negotiation process.

– ICSE-18 WOW! –

18th International Conference on Software Engineering

Window on the World

A Fraunhofer ISST Sponsored Publication **Volume 2 Number 2** March 28, 1996

DeMarco: Process considered harmful?

By Steve Easterbrook

In his opening keynote address, Tom DeMarco stirred up controversy yesterday with an entertaining diatribe on the "decade of process". Without actually saying that process technology is a bad thing, DeMarco managed to infuriate its advocates, by marginalising their work. Citing examples of companies that improve their software processes, and then lay off their developers, he pointed out that a well-defined process can act as a disincentive to take on high risk (and hence, potentially high payoff) projects.

He began his talk with a brief history of his career, including an entertaining account of how as a hardware engineer, he was given the job of implementing microcode for a jump instruction. The instruction involved copying the argument into the program counter, which appeared to DeMarco to be crazy: if the argument was less than the current program counter, the program would loop forever, while if it was greater, the program would merely miss a chunk of code. They concluded that the only possible use was to jump over data segments. The implied reference to Dijkstra's paper on GOTO statements was clear; but was DeMarco about to give a "Process considered harmful" talk?

DeMarco presented four paradoxes of software development, covering themes such as process improvement, mobility (or flexibility) of software development organisations, re-use, change management and cost/benefit analysis. He gave both a pessimistic and an optimistic view of the future of software engineering; a future in which software engineers seem to have less and less organisational power to help them get on with the job.

In the event, his theme was not so much that process technology is bad, but rather that it does not address the really difficult problems of software engineering. In Fred Brooks' terms, process improvement only addresses the 'accidental' problems of developing software; the harder, 'essential' problems remain. This is the first paradox: the result of process improvement is that developing software gets harder. The easy parts are formalised and mechanised, leaving just the harder parts, such as introducing change and defining requirements.

The second paradox is that a focus on process makes organisations risk averse. DeMarco pointed out that while this doesn't have to be true, unfortunately it usually is. An investment in process definition involves a commitment to the defined process. This in turn affects subse-

More ICSEs on the Way!

By Bashar Nuseibeh

With so much discussion on the future of ICSE, some of you might be curious to know what is in the pipeline. In yesterday's issue of WOW, Rick Adrion *et al.* reported on plans for ICSE-97 in Boston. In 1998 the roadshow moves on to Japan. I caught up with ICSE-98 General Chair, Prof. Koji Torii, and his programme committee co-chairs, Profs. Kokichi Futatsugi and Richard Kemmerer, to chat with them about their plans for the event. Torii painted an attractive picture of the conference venue – the historic city of Kyoto. Kyoto was the capital of Japan for over a thousand years and is famous for its ancient cultural and religious attractions. Torii was keen to emphasise the appropriateness of Kyoto as

...continued on page 6

CONTENTS

■