Collaboration and telecollaboration in design

7th Turing Lecture: London, 20 January 2005

Professor Frederick P Brooks Jr, founding Kenan professor of computer science, University of North Carolina.

The Turing Lectures are annual joint events of the British Computer Society and the Institution of Electrical Engineers. The seventh Turing Lecture took place in the lecture theatre at the IEE's headquarters in Savoy Place, London, on 20 January 2005.

Frederick Brooks has a long and distinguished career both in industry and education. He worked for the IBM Corporation from 1956 to 1965, first as an architect of the Stretch and Harvest computers and then as manager for the development of IBM's System/360 family of computers and its OS/360 software. He was responsible for many key developments in computing, including the development of interrupt systems and the selection of the 8-bit byte, which made possible the use of a lowercase alphabet for the System/360 and subsequent developments in word-processing.

In 1964, he founded the Department of Computer Science at the University of North Carolina at Chapel Hill and chaired it for 20 years. As Kenan Professor of Computer Science, his principal research is in real-time, three dimensional computer graphics --'virtual environments'.

His research has helped biochemists solve the structure of complex molecules and enabled architects to 'walk through' buildings still being designed.

In 1975, he reflected on the successes and failures of the development of Operating System/360 in 'The Mythical Man-Month: Essays in Software Engineering' (enlarged Anniversary Edition, 1995). He further examined software engineering in his 1986 paper, 'No Silver Bullet.'

Frederick Brooks started his lecture by pointing out that not only has he experienced design issues in relation to computer software, but also graphics projects, and a few building projects too. The design process seems to him to be the same across all these fields of endeavour, and several times he used examples from art, architecture and civil engineering to illustrate his points.

Looking back across the last hundred years, what are the big changes that have taken place in design? Prior to the twentieth century, all designs and inventions in engineering were essentially the expression of a single person's creative mind. But consider the example of the Nautilus nuclear submarine -- who could you point to as its designer? It was designed by a team of people. Design by teams is standard for modern products. In this respect engineering has departed from most other creative processes such as the writing of plays, painting, or the composition of music.

Therefore the topic Brooks sought to address was understanding collaboration in design work. There is an assumption that collaboration is an undoubted good thing, that 'all of us are smarter than any of us’, and that therefore the more collaboration one has, the better. But is this true?

Historically, it has not been unknown to have two people working on a creative task, but when we examine these examples closely we find that they have been working on complementary aspects of a production (Gilbert and Sullivan would be an example, librettist and composer), or that one has been the master and director of the project, while the other has been the helper. For example, master painters might have their students work on the backgrounds and crowd scenes.

The complexities of modern engineering

Why is modern engineering design done by teams? Because of the complexity of every aspect of engineering. Brooks showed side by side photographs of two bridges: the 1779 Iron Bridge near Coalbrookdale in Shropshire, designed by Thomas Farnolls Prichard and built by Abraham Darby, and the Sunniberg Bridge designed by Christian Menn. Pritchard's design used a great deal of iron in a not particularly economical way, while Menn's bridge designs exploit the properties of pre-stressed concrete and very careful calculations of loading and deflection to create structures that are not only extraordinarily strong for their weight but also highly elegant, and easy to build with minimal scaffolding.

Brooks proposed this postulate: ‘There are no naïve technologies left in the West.’ He gave three examples of this. In shampoo manufacturing, he has learned, computer simulations of the flow of viscous liquids is used to make sure that the mixer blades don't separate the three-layered emulsion of the shampoo. As an American farmer, his brother now spends more time on the computer than on the tractor. And in a tour of the Sara Lee cake factory, Brooks learned that as each new batch of flour comes in, they change the cake recipe for the day based on a chemical analysis of its content, which varies subtly from one grain-growing region to another.

Therefore, in most modern engineering projects, the range of knowledge and skills involved is larger than any one person can master. Furthermore, there is a hurry to get groundbreaking new products to market; it is said that the first entry to a new market will take 40 per cent of the market share, and Brooks cited the iPod as an example of this.

Do many hands make light work, as the proverb says? Yes, quite often.

Do many hands make *more* work? Yes, said Brooks -- always!

This is because team working is complicated. The different jobs need to be divided, the interfaces defined, the interfaces between them interpreted and reconciled, the standard elements and common styles defined, and all the bits put back together again. And that's a lot of work.

Integrity is key

However, conceptual integrity is the key to success in design, and some examples Brooks gave of engineering projects which display great integrity at all levels are the supercomputer designs of Seymour Cray, the bridges of Christian Menn, and Christopher Wren's St Paul's Cathedral. All these come across as the consistent expressions of a single mind.

Brooks raised a laugh by showing two lists of computing products, divided into those that have fan clubs and those that don't:

Have fans: Fortran, VM/360, Unix & Linux, Pascal, C, Macintosh, APL.

Haven't: COBOL, OS/360, Microsoft NT, Algol, PL/I, PC, Ada

All of those products that have fan clubs were creating outside what Brooks called ‘the normal design process’; all those without fans were done inside that process. This suggests that the normal design process doesn't create products that are innovative and that get loved. Perhaps one can say that the normal process tries to minimize the creative (disruptive) element in design.

There is a textbook on design which suggests that engineering design is about ‘interdisciplinary negotiation’. But Brooks thinks this isn't good enough. Rather, he approved the suggestion of Mills, who says that for small programming projects you should have a single chief programmer who is in charge of the integrity of the project AND is responsible for writing ALL of the code that finally gets delivered. Brooks anticipated that many companies would respond: ‘We don't have anyone that good’; but he thought that the problems are chiefly sociological.

Organizing the team

As a teacher of computing, Professor Brooks organises his students into teams of four and gets them to elect a ‘team boss’ and a ‘system architect’. This division of labour is rather like having a director and a producer on a film, with the director being responsible for conceptual integrity and the producer organizing to make sure the whole project works.

Brooks recommended four key elements to a successful design project:

(a) The system architect
(b) The person in charge of the user interface
(c) Documenting the assumptions in the design
(d) Agreeing a common ‘style sheet’

The system architect has to be someone who can get his or her mind around the whole system. As an example of this, Brooks mentioned a discussion he had had with the person who was responsible for the system architecture of the Geographical Positioning System. This man grasped that the whole system was based on the complex inter-operation of many component satellites and other components, and their common currency was time, sliced into microseconds. Thus he saw it as his role to ensure that each process got its necessary share of the microseconds

-- with, as Brooks put it, enough spare microseconds in his trouser pockets to rescue parts of the project that were in difficulty!

The principle responsibility of the system architect is to be an advocate for the user. Brooks said not much more about this, but pointed out that four chapters of his book ‘The Mythical Man Month’ deals with this.

You also need to have one person in charge of the user interface. It is highly appropriate to have one person handle this, because at the end of the day there is one person -- the user, that is -- who is likewise going to have to wrap their mind around the system. In other words, the UI designer acts as a surrogate for the user. And in user interface design, consistency and predictability are key. When Brooks asked Ken Iverson ‘Why do people love APL?’, Iverson replied: ‘Because it does what you expect it to.’

Document those assumptions!

You also need to document the assumptions behind the design. There are all kinds of assumptions: assumptions about the characteristics of the user community; assumptions not only about what the application will be in the immediate future but also how people will want to use it further down the line; assumptions too about the technologies of implementation.

In any design team, everyone makes their own assumptions and they will all be different. Brainstorming is a good way to winkle out those assumptions and share them within the group. It is better to be wrong in one's assumptions than to be silent or vague about them! Documenting your assumptions is enables discussion of them. You can make a ‘sensitivity analysis’ of your assumptions (how much does this assumption really matter? not much? a whole lot?); and the process also directs ‘verification analysis’ (is there any evidence for these assumptions?).

Once this is done, ‘you can get everyone singing off the same page’, said Professor Brooks. The process enables a reference specification to be created, a vital tool for any engineering project. As Brooks wrote in 1987 in ‘No Magic Bullet’: The hardest part of the software task is arriving at a complete and consistent specification, and much of the essence of building a program is in fact the debugging of the specification.

The other shared reference a design team needs is a style sheet. This serves a purpose similar to when a team of authors write a book or manual together. It ensures that things are referred to consistently by the same nomenclature, that processes follow the same style and so on. This is not because the *surface* style is essential to the conceptual integrity, but it sure indicates that the team has thought about the integrity of the ‘skeleton’ of the project.

Why does the bazaar work?

Professor Brooks next turned to a software engineering phenomenon, which could be said to be an exception to his argument so far: the design process that takes place in some parts of the Open Source/Free Software community. Does this undermine his argument, or is it a special case?

Or, as Eric Raymond has written: The lore of software engineering is dominated by Brooks's Law, articulated in Fred Brook's classic ‘The Mythical Man-Month’. Brooks predicts that as your number of programmers N rises, work performed scales as N but complexity and vulnerability to bugs rises as N-squared. N-squared tracks the number of communications paths (and potential code interfaces) between developers' code bases.

Brooks's Law predicts that a project with thousands of contributors ought to be a flaky, unstable mess. Somehow the Linux community had beaten the N2 effect and produced an OS of astonishingly high quality.

Professor Brooks actually referred to Eric Raymond's paper on ‘The Cathedral and the Bazaar’ -- the essay, evolved from a talk given to the 1997 Linux Kongress, which contrasts the centrally planned design process that was also the prevalent mode of development in the GNU project led by Richard Stallman, with the seemingly anarchic Linux development process. As Raymond writes:

‘I have been preaching the Unix gospel of small tools, rapid prototyping and evolutionary programming for years. But I also believed that there was a certain critical complexity above which a more centralized, a priori approach was required. I believed that the most important software (operating systems and really large tools like the Emacs programming editor) needed to be built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time.’

Linus Torvald's style of development -- release early and often, delegate everything you can, be open to the point of promiscuity -- came as a surprise. No quiet, reverent cathedral-building here -- rather, the Linux community seemed to resemble a great babbling bazaar of different agendas and approaches...out of which a coherent and stable system could seemingly emerge only by a succession of miracles.

Professor Brooks suggested that the Linux bazaar is an evolutionary model. It works because despite the community and the collaboration, there is no design-by-committee: each component that is contributed has its own conceptual integrity. The model emphasizes early and large scale distributed ‘fixing’ -- and those who find the bugs are also the ones who propose how to fix them.

The hacker community that Raymond describes is one in which prestige is accorded to those who contribute, and contribute well; that is highly motivational for this particular group. It should be noted that the contributors are people who are getting fed anyway: it's not the day job. The will to create excellence is also driven by the fact that the contributors are also the intended users: they know what they need, and they really care about it.

So, perhaps the Open Source (especially Linux) experience is the exception that proves the rule. As Brooks asked: would you design an air traffic control system by the bazaar method? It would be a ‘bizarre’ method.

From the many to the few and back

Many brains are useful in the early stages of a design project, when you need to determine what the users' needs are. Brainstorming in these early stages helps to explore radical alternatives. However, when it comes to conceptual design, and the detailed design, you don't want heaps of contributors.

Brooks says that the evidence is that two programmers working together (pair-programming) takes about 1.4 times the time, the ‘man-months’ if you like, to get the job done -- not twice as long. Why should a company want to pay that extra 50 per cent? Because the evidence also suggests that when two programmers work together on the code, there may be as much as an order of magnitude fewer bugs. And that is real value for money.

Once a design has been produced, or rather prototyped, is the time to involve many minds again. The review process benefits from having many reviewers.

As an example of how insights may come from unexpected sources, Brooks told the story of an engineering project that was commented on by the people who were going to paint the heavy girders. ‘You'll need to make that bit out of heavy steel,’ they said. When they were asked why, they said ‘Look where it's going to be placed -- we'll never get another chance to paint that beam again.’ In fact the company didn't make that component super-strong -- but they revised the design to ensure that the painters would be able to get access to maintain those beams.

Textbook examples of design are almost always ‘way too simple,’ said Brooks. In particular, they ignore the fact that complexity often forces designs to change halfway through, and these changes then involves many other changes. Finally, there is no substitute for ‘the dreariness of labour and the loneliness of thought’ -- even though it has been joked that committees are a place where people seek refuge from that.

Telecollaboration

At this point in his lecture, Professor Brooks stopped to show us a video clip of some of the experimental systems he has been working on, which permit virtual meetings to take place among three people located far away from each other. The system gives an impression of three-dimensionality to the view of each participant's workspace by synthesising the video image out of input from two TV cameras. Each participant wears a device on his or her head that indicates head position and orientation to the system, and changes the way the TV images are merged, so that by moving one's head one is able to see behind objects in the rooms being displayed on the screens.

What is the motivation for this research into telecollaboration? Because complex projects need people with specialized skills, not all of whom want to move to where the work is; because you might want to have work going on all around the world and therefore around the clock; because labour may be cheaper in some places; and also for political reasons (e.g. one may have to involve Brazilians in a design for an oil rig for Brazil, at the insistence of their government).

The Airbus 380 project not only has its components being made in four countries; it has also been designed between four offices in different countries. Nor is this particularly new: on the IBM 360, design was taking place in the USA, Holland, Denmark and Sweden, with CPUs in four labs in three countries.

However, telecollaboration is not simply a substitute for meeting in person. ‘Face time,’ as Brooks called it, is crucial. The best and most productive telephone conversations are between people who have a long history of face time. ‘Travel is worth what it takes, which is why all the planes are full,’ he said.

This is true not just at the executive level and between the top design personnel. At IBM, they found it was productive to charter a bus and take all the secretaries from IBM Poughkeepsie to IBM Yorktown, because these were the people who played a critical role in the interface between the two institutions.

Although the system which Professor Brooks had shown in the video clip is ‘exotic’, it is by no means essential. If you and your collaborators have the shared face-time, very low levels of technology will do the job. Even email alone may be quite sufficient. If you can go further by sharing a document or drawing -- sent perhaps as an email attachment -- and then follow that up with a telephone discussion, it may be every bit as effective as videoconferencing in many cases.

When is the effort and expense of videoconferencing worth the trouble? Brooks suggested that it is valuable when participants are insecure and the issues are vital to one or more of them. It may also help when the participants are from very different cultures: getting as many cues as possible to what is going on (e.g. by observing body language) can help.

On one of his visits to the UK, Brooks had had the opportunity to ask the chief engineer at BAe Systems in Filton, Geoff Jupp, how collaboration at a distance was achieved in Airbus. Jupp explained that as well using as the obvious telecollaboration methods, the partner companies maintained ‘resident ambassadors at each other's courts’ -- people able to explain the political background to why discussions went the way they did. They exchanged personnel between Broughton, Filton and Toulouse -- and, being an aircraft company after all, they dispatched a plane each way every day.

In making some closing remarks about developments in telecollaboration, Brooks regretted that too much of the R&D is being driven by the interests of the toolmakers, not those of the people who want to collaborate. In other words the research isn't application-pulled; it is technology-pushed.

Conrad Taylor is information design & electronic publishing Secretary, BCS Electronic Publishing Specialist Group (www.epsg.org.uk)