Panel: Software Maintenance for 2005

Chairman: Keith Bennett, University of Durham
Panelists: David Griffiths, British Telecom Laboratories, Martlesham
Pearl Brereton, University of Keele
Malcolm Munro, University of Durham
Paul Layzell, University of Manchester Institute of Technology

1 Addresses

keith.bennett@durham.ac.uk
pearl@cs.keele.ac.uk
malcolm.munro@durham.ac.uk
griffigd@boat bt.com
p.layzell@umist.ac.uk

The URL of the BT-funded Centre of Excellence is: http://www.dur.ac.uk/dcs1nat/bt-coe/

2 Chairman: Keith Bennett - Panel Objectives

Research results in software maintenance over the past 15 years have led to substantial improvements in software maintenance practice. However, if we extrapolate our current rate of improvement forwards, it seems likely that it will fail to meet the needs of our users.

Software is becoming ever larger, and more complex, yet users are requiring it to become more flexible to meet ever more rapidly changing business needs. At the same time users do not want to create yet more legacy systems.

It is therefore likely that we need radical new solutions if software maintenance is to be viable in the year 2005. This is of great importance to academics, who are setting out their strategic research agenda; and to industrialists who need to exploit the latest technology to maintain competitive edge.

In software maintenance we need to reach a consensus about what our major goals are to be over the next ten years. This will help to focus and co-ordinate the community, and avoid its current rather fragmented approach.

Thus each panellist was asked to prepare a position statement in which the following points were addressed:

1. Where will the major growth areas in software maintenance and software engineering be for software in the year 2005?

2. What are the strategic research problems we need to work on to ensure that software can be produced and evolved in 2005?

3. What are the likely implications for industry?

3 Panelist: David Griffiths - The Future of R&D into Software Technology

R&D into software technology has for sometime led to little concrete results with, for example, disciplines such as artificial intelligence spawning many interesting concepts but little of real final application. For the longer term, consideration must be given to the more radical issues that will facilitate a real paradigm shift in systems engineering. We can no longer be bounded by the experience of systems development today: every functional enhancement seems to be based on the development of yet more software. The longer term vision - call it a challenge - for research in this domain is to find ways of conjuring the necessary systems functionality without recourse to armies of engineers and the generation of mountains of code. The impact of this longer term vision could be profound, possibly the total absence of the current problem set as a result of new paradigms such as self-generating - and even self-seeding - systems. The potential dangers could also be immense if such fast-breeder computer systems can no longer be protected or controlled - especially if the boundary of every given system becomes in effect as now with the Internet: global.

I would like to propose the following as just a few of the R&D opportunities that software engineers should be considering in order that we may be moving towards the required paradigm shift:

- Intelligent agent technology, because it includes techniques which support mobile, intelligent, self-contained entities that autonomously solve problems and perform tasks on behalf of other systems
• componentware, because it supports the on- and offline integration of prefabricated software building blocks and integration of legacy systems
• microkernels, because this technology is heading for small, effective, configurable operating software
• spontaneous functionality, to enable self-generating systems immediate delivery of required functionality without the time lapse and costs associated with generating software
• network-centric computing, where computing, communication, content coverage, and the network becomes the computer
• harnessing virus technology as a vehicle to self-propagate software

4 Panelist: Pearl Brereton - Configuring and Maintaining Virtual Systems

As outlined in the overview to these position statements, software-intensive systems are becoming even larger and more complex whilst users expect greater flexibility and more rapid delivery. Underlying these problems of size and complexity, current software development practices focus on the production of tightly integrated systems configured from large collections of part or components. This actual integration is carried out by service and product providers, as well as by customers/users. Product providers deliver (sell, hire out or give away) pre-integrated products which might be further integrated and customised by customers. A characteristic of such systems, from the perspective of an integrator (either a provider or a customer), is that the rate of change of parts is slow.

In order to satisfy the user needs for flexibility and rapid delivery, we need to liberate ourselves from the constraints of actual integration and move towards the development of systems and services that are configured as a result of the virtual integration (VI) of parts which might be obtained from a variety of sources under a range of different contractual arrangements. Because of the virtual nature of the integration it will become possible to integrate or substitute new (versions of) parts as soon as they become available. We can already see a trend in this direction with the rapid growth in the use of Java and there is the further potential to use the Internet for the purposes of component distribution and for providing the communications infrastructure to enable VI.

At the product level, in order to support virtual integration we need to be able to build, understand and maintain virtual systems (VSs). This will require a new approach to the way in which we create, describe and change systems, both in terms of their architectures and the parts from which they are composed.

The research challenges associated with configuring and maintaining VSs encompass architectural issues, component and connector issues as well as quality and business issues. Many problems in these areas are of course already being addressed in some way. However, if we are to make best use of VI we need to draw together this work and apply the technologies and ideas to the major problems of complexity that will inevitably arise from VI.

Some of the architectural issues to be addressed include: configuration languages for VSs; understanding VSs; measuring and controlling the complexity of VSs.

Component and connector issues include: describing and evaluating component and connector services; utilising (and taming) agent technology; detecting and using new versions (substitution).

Of the many quality and business factors that arise, particularly relevant are those relating to reduced requirements engineering and testing activities, consistency problems resulting from substitution and from behavioural changes in components that learn, transactional pricing, accountability and trust.

5 Panelist: Malcolm Munro - Self Fixing Software

Software is now central to the working of a large number of products and services provided by a large business. This research highlights the way forward for software development and evolution for the next decade so that companies can maintain a market edge in an ever changing market where the expected casual and embedded software growth is exponential.

There is still a lack of fundamental theories of software engineering that has led to the development, and continued evolution of, what are termed legacy systems. The legacy problems have not been solved by conventional software engineering nor are they likely to be improved by existing technologies and current practices. With the arrival of new business opportunities and technologies new problems will arise that will have to be overcome.

To overcome these shortcomings there has to be a fresh vision for software and software engineering in which systems will be:
• reconfigurable,
• flexible,
• with a fast development time to create or capture new business opportunities, and
• capable of being developed, with high quality, by the end user.
These attributes will support a high rate of change in business and aid global cooperative business for main stream markets and niche markets. This will require very flexible software not encumbered by legacy systems.

An important aspect of reconfigurable and highly flexible systems is the notion of Self Fixing Software. This concept can be looked at from two viewpoints, that of

- software that fixes itself;
- automatic fixing of software after testing.

The former implies that some form of agent based system is being used so that the can be self aware; and the latter implies that there is some form of intelligent testing oracle. Our research is investigating these new aspects of systems.

6 Panelist: Paul Layzell - Who Will Be The Software Developers of 2005?

The tools, techniques and processes for producing, evolving and managing software will have changed significantly by 2005. However an equally significant change lies in the manner in which such production and maintenance processes will be organized and who are the agents in such processes.

An earlier contribution to this conference highlighted some of the changes taking place within the organizational structures of IT departments, particularly in the service and commercial business sectors. It presented three models of the organization of development and support activities based upon the degree of integration of these activities within the underlying business. However current research by the Software Management Group at UMIST indicates that the underlying assumption that there are IT departments is being challenged, replaced by a view that software development and evolution will become a common, technical skill found across all successful business activities.

Service and commercial sector businesses need to produce systems considerably faster than can currently be achieved. One approach is to provide business managers with a range of pre-defined object classes, both for low-level objects such as windows and buttons, as well as higher-level, domain-specific objects and allow them to construct proof of concept systems in which traditional development lifecycles are neither appropriate nor fully applied. In many cases such systems will be dropped because the business opportunity has changed and they are no longer required or, in a few cases, will flourish as a competitive business-edge is obtained requiring possible rework or complete redevelopment as the needs of the business become better understood.

The role of IT departments and software engineers is radically different in this scenario as much of the construction work of an information system will be undertaken by agents other than IT professionals. Whilst it is likely that low-level object classes may be produced by software specialists, there will be a limit to the number of different types of object required, eventually leading to standard packages of objects which can be bought off-the-shelf and little in house development by IT departments.

A common assumption is that it will be higher level, domain-specific objects that will be produced by IT professionals; however this is unlikely to be true in the longer term. First the complex nature of domain-specific objects means it will be easier for domain experts to be trained in good software engineering practice and allow them to develop and maintain such objects rather than to train IT professionals in complex domain understanding. Secondly, business objects will be the source of competitive advantage and therefore unlikely to bought from third parties.

Thus the construction of future business support systems will be undertaken by an emerging class of business manager, well acquainted with software concepts and supported by high level development tools which enable them to produce rapid business solutions and information systems. The dynamics of business and the speed with which new systems will eventually be created will mean that maintenance (at least in its currently understood form) will not be required as we enter the age of disposable business software and as a result of which many organizations will be able to dispose of their IT departments.

This is a possible vision of the future, but is underpinned by many assumptions as to the competence of business managers and the capabilities of the tools provided for them. The panel will provide an opportunity to examine these assumptions and to identify to what extent the premise that there will be no IT departments developing and maintaining software in future holds true.

7 References
