Sheffield Software Engineering Observatory.

Abstract.

Software development methodologies are a source of much inconclusive research primarily because the evaluation of design notations, methodologies, management frameworks and other process aspects is difficult to do in any sort of scientific manner. It is not usually possible to carry out large scale comparative experiments with many different design teams using different approaches in realistic industrial scenarios. Most results are drawn from small scale internal projects involving university students. Some useful information can be obtained from these but it is limited. The projects rarely involve a real industrial client and the pressures of delivering an easily maintainable and usable product to a client organisation is invariably missing. This proposal is built round the Department's long experience - 14 years - of running industrial software projects with competing teams which provides as realistic an environment for such evaluations as is likely to be possible. Because the supervisors of these projects insist on a highly professional approach with the collection of many statistics and data it provides a unique facility for observing realistic, if relatively small scale, industrial projects. Since many of the projects operate with a number of competing teams it is realistic to carry out comparative evaluations of different methodologies involved in the production of solutions for the same external client. The Department also operates a commercial software house which can be used to experiment further with different approaches to software construction and which can be used to examine how new approaches can be introduced into a working company. The Software Engineering Observatory would be a unique, in international terms, facility which can be used to both explore different approaches to software construction and to see how successful strategies can be embedded into a commercial company.

Aims and objectives.

The aim of the Observatory is to carry out carefully designed and scientifically based experiments on software engineering methodologies, management approaches and innovative software construction techniques with a view to their comparative analysis and evaluation.

The objectives of the Observatory are:

to design and execute experiments involving teams of trainee developers working on industrial design projects;

to develop and evaluate empirical methods and analysis techniques appropriate to small to medium scale software projects;

to develop expertise and software environments that exploit modern information extraction technology for analysing repositories and databases of design information;

to develop and maintain expertise in empirical and scientific approaches to software engineering;

to offer the facilities of the Observatory to other research groups working on empirical analysis and to software companies wishing to carry out detailed comparative evaluations of emerging software construction technologies and tools.

Background.

The construction of complex software systems is a major part of the world economy and the pressures on the industry to produce high quality software, rapidly and efficiently are great. The shortage of skilled software engineers coupled with the increasing demand for software are creating serious problems. Attempts to introduce new methods of designing software seem to be based more on fashion and conviction than on any scientific evaluation of the best ways to engineer software systems. This is compounded by the problem that there is clearly a disconnect between academic researchers in software engineering and the needs of industry. A fact that has been pointed out by EPSRC's senior committees as well as by industrial commentators.

Is there a problem?

Software engineering research - principally looking at methodologies, tools and management support has been beset by problems of evaluation. Software companies are unlikely to be persuaded to take the risk in adopting a new way of working unless they are convinced of the benefits. One hope of the academic community was that the issue of quality would become paramount and this would drive companies to use better founded methods of design. Far from this happening the main driver is *time to market* and quality is sacrificed as a result. The legacy of poor software and the phenomenon of premature release is the phenomenon of users having to do much of the testing involuntarily and installing corrective patches! The demand for software is so strong and the capacity to deliver is generally insufficient that companies are cushioned from the consequences of this approach. Almost everyone now delivers incrementally - with subsequent releases mainly providing *corrections* to previous versions, sometimes on a monthly basis, and increased functionality or better usability/performance being the purpose less frequently.

In the UK some potentially useful work has been done on issues relating to several major bottlenecks and problem areas in software engineering. *Requirements engineering* evolved to try to deal with the difficulties of eliciting what clients actually want and this relates to business analysis and process re-engineering. [***] The problem with this is that often the domain chosen is general business software - databases, information systems etc. Most large companies have their own in-house techniques and these have evolved with their business and their clients. They are often weak but the house documentation needs to be continuous - many systems are not “green field” or “new builds” but extensions and adaptations of existing systems driven by the need for some new functionality, or the platform technology has changed - green screen to windows to web based end user interfaces or the non-functional requirements - performance/reliability/usability etc. need to be updated. The point is that the core of the system remains and is not re-engineering very significantly - it is too risky from a business point of view. There are few benchmarks available against which techniques and methodologies can be judged simply because coherent and reliable data from rigorous scientific experiments is just not available.

Is the problem changing?
Many new challenges are emerging as the technology improves and changes, as new application domains open up and as previously impractical applications become possible. There seems to be little work on requirements capture for these other application domains - domains where the data is much more complex such as speech, language and what is traditionally referred to as AI. In fact Software Engineering has little to offer the AI community who are building large, complex, sometimes critical software systems using rather *ad hoc* techniques. This should be a major area of research. Intelligent agents are being proposed as potential solutions to the problems of dealing with complex data, changing environments and needs. These are not always well understood and modelling and testing this technology is still in its infancy. Nevertheless there are strong market pressures to use this technology - even in critical areas. Software engineering research has not addressed this area properly.

A further issue is that the *dynamism* of the business world means that extensive requirements elicitation processes result in the definition of target systems which are out of date before we start implementing them. Few, if any, of the software engineering methodologies can support rapid requirements change effectively. In many ways the methodologies proposed are so heavy weight and unattractive to developers that companies tend not to use them. [Worboys, **] This has a consequence when it comes to maintenance and quality assurance. New movements such as eXtreme Programming (XP) and Feature Driven Development (FDD) are attempts, originating from design practitioners in the industrial sector primarily, which attempt to address a number of issues that relate to practical design considerations rather than elegant academic principles. There is scope here for some innovative research to understand, evaluate and underpin these approaches.

Quality is still a problem.
Another major issue is testing. This, and related activities - review, debugging, perfective maintenance - now often consume more than 50% of the budget. Testing OO systems and web-based systems, particularly, is a very difficult area - witness a number of recent disasters that received much press coverage - boo.com, egg.com etc. Significantly there is very little research going on in the U.K. relating to software testing - there are about 8 academics in this field in the UK. Providing practical and efficient methods for testing these new generation applications would make a major impact. Even more effective would be a better utilisation of the knowledge we already have about testing and the tighter integration of testing into the design process. The V model [**] has been a convenient managerial framework up to a point but it has not highlighted the need to consider testing at an early enough stage so as to introduce a “design for testing” approach into the software design process (unlike the situation in hardware - VLSI - design).[Holcombe]

In order to address the disconnect between academic software engineering and the commercial software industry it is important that proper scientific evaluations are carried out into practical and transferal practices which have demonstrated usefulness in real development projects.

The philosophy of the proposal.
There are three major themes in the basic problem of trying to understand the software construction process which needs to be addressed in such foundational research. Firstly the *nature and structure of the process* being examined; secondly the *quality of that process* measured in terms of its fitness for purpose, this involves not only

the coherence and consistency of the process but its human dimension and its impact on and dependencies upon the management environment and thirdly the *quality of the product* which is primarily dependent on the judgment of its clients and users.

In order to carry out any scientific investigation into the software construction process the experimental framework must be appropriate. Science, generally requires experiments to be hypothesis-based and repeatable, providing the environment that is required is itself reproducible in a reliable manner. One of the most powerful types of hypothesis is that of a comparison between rival proposals which means repeating the experiment for the different manifestations of the conjecture under controlled conditions so that valid comparisons and conclusions can be drawn. In many cases this involves a *novel treatment* and a *control*, the hypothesis being that the novel treatment has some identifiable and measurable benefit over the control - or not - as the case may be. Here is the key problem, being able to run trials which involve more than one process under controlled conditions, something that is generally impossible in industry where the sheer cost of development is such that repeated trials and concurrent project teams working on the same problem is completely prohibitive. Where university experiments have been carried out using teams of students the criticism is that the simulation of the industrial context is extremely weak and thus conclusions drawn from the experiments have little industrial credibility.

Another major issue is that the personality of the individual programmers, their cultures, the management environment and other parameters make comparative analysis difficult unless the trials are carried out simultaneously under identical conditions and amongst an homogeneous programmer population.

The proposal.

The Software Engineering Observatory will be focused around three software development companies, The Software Hut and The Maxi project and Genesys Solutions.

The *Software Hut* comprises around 90 trainee software engineers (second year undergraduates) and three senior managers. Typically it produces three real systems each year for real clients from business companies and other, public, organisations. The core business expertise of the company is bespoke databases with custom front ends and web based applications, usually with an e-commerce aspect. The mode of operation of the company is to have one third of the members working with each client. Within this division into sections there are 6 teams of 5 developers each of which build a version of the product in competition with the others. The clients choose the best system from the 6 produced and these are used in the client’s organisation. The IPR is negotiated separately for a fee. What this means is that there is an opportunity to use different design approaches with the different teams and this can be the basis of a detailed scientific comparison of different approaches. Currently the company is planning to operate with the next projects an experiment whereby one set of teams applies “standard” software engineering methods based around UML and a significant design and analysis phase. The other set of teams will use the extreme programming approach - a lightweight, test driven approach [see the Appendix]. We hope to collect and analyse some data from this exercise to see if there is a significant difference in the performance of the teams and the quality of their products. However, it will not be possible to carry out the in depth research and analysis that is needed to make any definitive conclusions. It will be regarded as a basis for seeing what data could be practically collected with the current resources available and for examining the methodological issues associated with this sort of experimental research in this context.

The *Maxi Project* has been operating for 12 years in a similar manner to the Software Hut, except that it involves MSc students studying for the Software Systems Technology and Data Communications MScs. The numbers have fluctuated between 100 and 50 and each team is managed by an experienced industrial consultant, Mr. S. Price, who provides an experience for the students similar to that they might expect when being managed by a senior manager in a large software house. As before there are multiple teams competing to deliver a software solution for an outside client, [Kent paper etc.].

Genesys Solutions is a partnership of 25 qualified software engineers (4th year undergraduates and senior masters students) carrying out a similar range of contracts but also providing consultancy and other services. The company has its own R & D section and provides infrastructure support to the design process. The way that this company will be used will be to consider how the results of the comparative evaluations of the methodologies and processes derived from the Software Hut experiments can be introduced and embedded into an existing software house. This is an important area because of the apparent reluctance of many software companies to introduce new methods of working.

All clients of these companies will be asked to permit the results of these experiments to be published. In the past a number of the projects carried out by these companies have been the basis of publications and so far all the clients have been happy with this - provided any “in confidence” commercial details are hidden. It is very difficult to persuade software houses to release the results of any analysis of their design projects because of commercial reasons.

The proposal requests funds for support of a long term nature in order to assemble a team of experts who can monitor and analyse the results of these experiments. Since software development is a rapidly changing industry and the introduction of new methods, tools and languages is likely to continue, once the observatory has been established it will be able to provide a service to the software industry based around realistic and rigorous scientific approaches to the evaluation of different methodologies and processes over a long term. The Observatory will be reactive to new methods and technologies and receptive to new demands from clients.

During the course of the Software Hut projects each team generates between 200 and 300 pages of documentation, this includes requirements documentation, detailed designs, source code, test sets and results, debugging reports, quality review reports, time sheets, meeting minutes and team evaluations. The source code is typically of the order of ***** lines of code for the sort of systems built. The clients and managers also produce reports on the products delivered and evaluated as well as the judgements on the design processes.

This documentation, taken over 18 teams and archived over several years so far, provides us with a major resource for the analysis of many aspects of the investigation.

Part of the role of the researchers will be the development of better tools for collecting and archiving the material in the future, tools for collecting specific metrics and, in particular, the creation of an *observatory environment* where project methodologies can be specified, metrics defined and the results from the projects processed and analysed. This activity will evaluate various alternatives for building flexible repositories of project data, using XML-based archiving methods and tagging the source material in suitable ways so that information retrieval and data mining techniques can be applied. Some of the *state of the art* machine learning techniques of Professor Mahesan Niranjan will also be applied in order to identify key discriminators that characterise the processes from the data investigated, [refs***]. This will enable us to evaluate which metrics prove to be the key ones which can provide information in the most efficient and effective way.

The collaboration of colleagues in the Department’s world leading Natural Language Processing (NLP) group of Professor Yorick Wilks will be a major asset. We plan to use some of their generic tools for information extraction, including the DARPA approved GATE environment for building our observatory repository environment and their information extraction techniques to enable us to recover important information about the design process and quality out-turns from the project data. [Refs ****] We also intend to use information agents to dynamically explore the archives and to report on key phenomena which we characterise. This will represent a major new development in the conduct of empirical software engineering research.

The Observatory Environment produced will be evaluated carefully and be made available to other empirical software engineering laboratories for their use.

Method of working.

The Observatory team will define a number of experiments each year based around the use of the Software Hut, Maxi and Genesys Solutions projects. These experiments will be carefully designed and suitable clients and projects obtained to act as the focus of the experiments. The framework and facilities required will be set up by the Observatory team, this will include suitable tools, repositories and appropriate recording software and techniques. The definition of the metrics and other evaluation techniques that are to be used will be done before the experiments begin. The training of the student teams in the appropriate technology and facilities will be carried out under carefully controlled conditions. Since all clients pay for their solutions there is an important motivational driver for the student teams and this has always ensured that the teams are appropriately motivated.

Key activities will include the estimation by the teams of the resources needed for their projects and this will be done using appropriate methods - in fact estimation techniques could be a particular focus of some experiments as well as risk analysis and planning technique [Boehm,**]. Other suitable metrics such as object point analysis [Kauffmann, ***], weighted methods per class [Chidamber & Kemerer,], as well as measures of software quality will be available. The time spent per team, per team member and how it is spent are also important factors that will assist in the analysis of the efficacy and expense of any method. The relationships between all these variables will be major areas of interest.

These experiments, if they are to be effective, need to be carried out under carefully controlled conditions, the scale of the experiments, in terms of the number of teams and clients, will be a major headache unless there are sufficient staff to support the Observatory.

Funds for the appointment of an Observatory Director are requested. The director's role will be to manage the overall activities of the Observatory which will include the following:

1. Identifying specific experimental methodologies to be evaluated in the context of software development;
2. Identifying and liaising with potential and existing clients;
3. Identifying which of the methodologies evaluated in 1 can be applied to the work of Genesys Solutions.

Resources required.
Three further research associates will be required to give a team of four engineers. The roles of these four research fellows will be as described:
:
1. *Metrics engineer.*
The metric engineer will research into the use of appropriate metrics for the experiments, with the particular aim of introducing, wherever possible, automated methods for collecting and analysing the data and deriving statistics from the experiments. The metrics engineer should have experience of research into metrics and empirical software engineering and be well qualified in statistical sampling and analysis.

2. *Test process improvement engineer.*
This researcher will be familiar with state of the art software testing and testing tools. They will play a role in evaluating the quality of the software built by the various design teams, the effectiveness of fault avoidance strategies and in debugging techniques. This engineer will have a strong quality focus and included in their remit will be the issue of review of all team deliverables, including documentation. The reason for having such a specialist is because of the fundamental importance of testing and review, the most time consuming and expensive part of software development.

3. *Requirements engineer.*
This researcher will be familiar with the most recent and current developments in requirements elicitation, processes, tools and experiences. Their role will be to observe the client-designer interactions and develop and adapt methods for the effective identification of business processes and user requirements. The ways in which client's requirements change over the period of the project - a major problem in the industry - will be a major focus of attention. The reason for having a specialist in this area is because of the utmost criticality of the requirements capture process and the inherent difficulties of this task in specialised application domains. The person appointed will interact strongly with the Natural Language Processing, Speech Recognition and Bioinformatics researchers in the Department as new methods for building AI systems are being developed as well as working in traditional software systems areas.

4. *Process improvement engineer.*
This researcher will be familiar with modern design methodologies and CASE tools. He/she will be providing an evaluation of current versions of design notations, processes and supporting tools. Of particular interest will be UML based approaches, but also novel approaches used in AI and other application domains where the data is more complex and difficult to describe. Experience with Natural Language Processing or Speech Recognition software development will be an important extra dimension since some of the more experimental and evolutionary approaches and competitive corpus-based evaluation methods used successfully in these areas would provide alternative design paradigms that could be trailed in the Observatory.

The Director would be responsible for one of these tasks as well as the overall Observatory management.

Contributions for the costs of a secretary to support all of the above activities and in particular to act as a communication link between the Observatory and its clients both in terms of its software development activities but also in terms of the software industry - we envisage the Observatory being in a position to act as consultants to the industry, being able to carry out evaluations of methods and tools for industrial collaborators. A software technician is requested to provide software (and some hardware) support for the Observatory's activities.

All of these posts are vital if the ambitious programme of research to be conducted by the Observatory is to be feasible and the liaison and dissemination of results into industrial software houses is to be achieved.

The management of the Observatory will be provided by weekly meeting of the Management Board which will comprise of the Principal and co-investigators, together with the research associates, the Observatory Director and representatives of the associated companies (Genesys Solutions and The Software Hut)

An Observatory Board will be set up comprising leading members of the software industry, leading academics and representatives of EPSRC as well as senior members of the Management Board. This group will review, annually, the activities of the Observatory and provide critical feedback and support to its ongoing activities. An annual report will be produced and circulated widely throughout the commercial and academic communities. A number of eminent figures have agreed to join the Board, these are high profile researchers from USA and the travel costs include provision for their expenses in coming to the Observatory once per year for these Board meetings.
References.

G.Kotonya & I.Sommerville. Requirements Engineering: processes and techniques, Wiley, 1998.
B.Worboys, P.Kawalek, I.Robertson & M.Greenwood. Business information systems: a process approach, McGraw Hill, 1999
R.Kauffman & R.Kemerer. Modelling estimation expertise in object-based CASE environments, NY University, 1993,
R.D.Baukner, R.J.Kauffman & R.Kemerer. An empirical test of object-based output measurement metrics in a computer aided software engineering environemnt, Jour. Man. Inf. Syst. 8(3), 127-150, 1991.
B.W.Boehm, B.Clark, E.Horowitz et al, Cost models for future life cycle processes: COCOMO 2.0, Annals of Software Engineering, 1(1), 57-4, 1995.
Basili VR. Cleanroom
Basili VR. & Selby RW. Comparing the effectiveness of software testing strategies, IEEE Trans. SE, SE-13, 1278-96, 1987.
Beck, K. Embracing change with extreme programming, IEEE Computer, 32, 70-8, 1999.
Beck, K. Extreme programming explained, Addison-Wesley, 2000.
Holcombe M. & Ipate F. ?????
Holcombe M. & Lafferty H. Kent,
Price S.(Project 99)
Stratton A. (Project 99)
Holcombe M. (Project 99)
Chidamber S.R. & Kemerer C. F. A metrics suite for Object oriented design, IEEE Trans. SE, 20, 476-493, 1994.

Appendix 1.

An example of a potential experiment for the Observatory.

This coming semester (February 2001) will see the start of the next Software Hut exercise. There are, as usual 3 clients each dealing with 6 teams and what we plan to do is to divide the 6 teams into two groups, one of which will be given some reinforcement in traditional software design techniques and the other will get a crash course in Extreme Programming. We will then monitor the progress of the two cohorts of students, some using XP others not, as they attempt to build their solutions. This will be done by studying the way they manage their projects. Each team has to produce and maintain realistic plans, keep minutes of all their project meetings, and by interviewing them weekly. We will also get all of their working documents, requirements documents, analysis, test cases, designs, code and test reports. These will provide many further opportunities for measuring attributes of their output, ranging from function and object point analysis to bug densities. The XP experiments suggested by Ron Jeffries on [7] will be helpful in this respect.

At the end of the Semester, the clients evaluate and mark all the delivered solutions, they use a structured marking scheme that we construct for them and this provides a final level of measurement relating to how well the solutions did - usability, installability, functionality, robustness etc. These are the key attributes since they will be applicable to all the solutions no matter how built. We will use this information in a statistical analysis to see whether there are any significant differences in the quality of the final products between XP and traditional “heavyweight” methods.

Finally we will require each student to give both a team evaluation and a personal commentary on how the project went, the strengths and weakness of what they did and how they did it. In the past this has proved to be very useful in identifying issues and problems with approaches to software development.

After delivery we will be able to track the performance of the delivered systems to gain further information about their quality in their working environment.

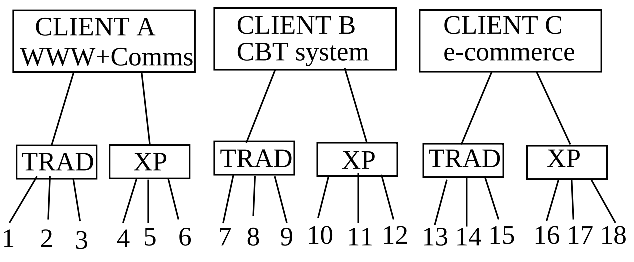

Fig.1 The organisation of the 18 teams.

The three clients are as follows:

Client A is a company involved in the analysis of biological data arising form the various genome projects. Algorithms are to be developed and integrated into an easy to use system which will exploit data from genome databases available on-line and specific outputs will be classification and categorising information about the interrelationships between genetic properties and disease.

Client B is an organisation, a legal practice centre, that provides specialist training for the legal profession, that aspect that is post academic qualifications and deals with the experiential learning related to legal practice in solicitors' offices. The system required is a computerised assessment system to provide a mechanism for tracking and evaluating individual student's performance on the course.

Client C is an organisation which brokers waste. A waste exchange provides a facility for industrial companies to offer their waste products to other companies who might be able to reclaim something of value from it. The waste exchange maintains a database of current waste products and arranges for the exchange and payment of deals in waste. The project is to build a web based system that interfaces to the existing database and allows clients the opportunity to browse the database.

The overall arrangements are described in Figure 1.

In all of this the students will be basing their approach on what they have learnt in the course so far. In the first year they will have taken part in a group project which involves them building a small software system specified by the course lecturers. The students do this as one-sixth of their work over the year and it integrates what they have been taught in formal lectures dealing with requirements and specification; Java programming; Systems analysis and design (essentially UML). This exercise helps them to start understanding some of the issues relating to working in teams, keeping accurate records and producing high quality documents, some of the problems of dealing with clients (a role p[layed by their academic tutors) and the problems of delivering quality software, and the need for thorough review and testing activities.

Before they get started on the Software Hut projects they attend a practical course on team work organised by the University's Careers Services Department.

They will then be split into two cohorts, the XP teams and the Trad teams, for further specific training in methodology and approach to software construction.