Teaching XP for Real: some initial observations and plans

Mike Holcombe,
University of Sheffield,
Regent Court, Portobello Street,
Sheffield, S1 4DP, UK,
+44 114 222 1802
m.holcombe@dcs.shef.ac.uk

Marian Gheorghe
University of Sheffield,
Regent Court, Portobello Street,
Sheffield, S1 4DP, UK,
+44 114 222 1843
marian@dcs.shef.ac.uk

Francisco Macias,
University of Sheffield,
Regent Court, Portobello Street,
Sheffield, S1 4DP, UK,
+44 114 222 1800
f.macias@dcs.shef.ac.uk

ABSTRACT

Fourth year students run their own software house which involves them in carrying out real projects for real business clients. This year we have introduced them to extreme programming and we examine the initial impact that this has had on their business. The philosophy has been adopted with much enthusiasm and seems to have delivered in a variety of contexts, including maintenance and new projects. Some plans for a more rigorous experiment looking at the possible benefits of XP are also described.

Keywords
Real projects, computing curriculum, empirical software engineering, extreme programming

1 INTRODUCTION

Teaching computer science and software engineering students is greatly enhanced if they can be introduced to the real issues relating to software design through the mechanism of projects for *real business clients*. For more than 10 years we have required students to take part in team projects in their second year where the teams compete with each other to produce a solution for a business person's current problem. Each student is required to work for 100 hours on this project during the semester. This equates to 9-10 hours per week on the project for each student over 12 weeks. Typically there are 80 students in the class and there are three business clients each with a specific problem relating to their business. The student teams are each allocated to one of these clients. The teams comprise 5 students and each client deals with 5 or 6 teams. At the end of the Semester the client evaluates all of the software solutions produced and selects the best one for use in their organisation. The winning students receive a prize.

This framework really transforms the students' learning because it emphasises two of the most problematical issues when teaching software design, how to communicate with a client and capture the real requirements and how to deliver a really high quality, bug free system.

It is very hard to introduce either of these dimensions into the curriculum using projects specified by academics. Students know that once the software has been marked it is usually thrown away. With our approach which we call the *Software Hut*, students are much more motivated because they know that someone *wants* their work and *will use it*. They also learn quite a lot about the way businesses work. It is always the most popular course and the one that they say teaches them the most!

More details can be found in[1], [2] and [3].

A recent extension of this approach occurs in the 4th year where the students run their own software company and spend approximately one third of their time working in it. The company, they call it *Genesys Solutions*, [5], [7] (formerly called VICI), has a wide variety of clients requiring database systems and e-commerce applications. About 25 students work in the company. The students run the company, take all major decisions, operate their own premises and network, and carry out R & D as well as specific industrial projects. As part of this the students negotiate the details of a contract with a client - cost, delivery as well as the detailed requirements specification. As one might expect, estimation and planning is a major issue in running the company and one of things that we are trying to do is to collect suitable data on projects that would help us to do this better. The estimation of resources for XP-driven projects needs to be considered in a different way to traditional projects so we are starting from a position where we need to think about things rather differently We believe that this student-run company is a unique innovation but one which the students are incredibly enthusiastic about. As an aside, a number of former members of this company have successfully set up their own real software houses.

2 PROBLEMS THAT MOTIVATED CHANGING TO EXTREME PROGRAMMING

These sort of projects are not without their problems, we do not want clients coming back with complaints about software quality and inappropriate functionality, we and we cannot afford to spend all our time in maintenance. Consequently we must ensure that we deliver extremely high quality solutions. The Genesys company can do some

maintenance, particularly where the client wants some new functionality, but we have to focus very hard on the software quality. The students are steeped in the conventional software engineering methodologies by the time they reach the 4th year but it is clear that these have not been able to guarantee the level of quality that we need and in many cases these methods seem to get in the way!

This year we decided to introduce the 4th year class to Extreme Programming, none of them had heard of it before. The response was overwhelmingly positive and they decided to apply the ideas, as far as they could, to all their projects.

3 INTRODUCING XP INTO THE COMPANY.

There were two types of current project when we started, some major testing and debugging of existing projects and some new developments.

In past projects we had organised ourselves in such a way that the teams would test each other's software, relying on the view of many test experts that independent testing is the most effective approach. This didn't really work since the teams' main priority was to their own project and, with deadlines fast approaching, they would concentrate on their own development work at the expense of testing another team's system. Coupled with the problem of teams trying too hard to satisfy their client's late requirements changes this was a clear recipe for disaster. Thus, we were unable to deliver, when the academic year ended, software of an appropriate quality. (After the end of the year the students graduate and leave so we do not have the flexibility of extending deadlines or of the benefits of continuity in the teams since next year's company come from the next cohort of students.) We discussed these problems with the next cohort when they arrived in September 2000 and then looked at the ideas behind Extreme Programming, primarily using [4] and the main XP websites. It was immediately clear to them that this new technique could be a big improvement on what they had done before. They therefore decided to adopt this way of doing things as far as possible.

The idea of pair programming was very well received and has proved extremely effective in debugging code, the construction of functional test sets from the requirements also had a big impact on the process. It highlighted the need for suitable testing software, so both test generation and test application tools had to be built for the specific applications, since these tools were based on generic concepts they can be adapted to other projects. We describe some work on developing extremely powerful test case generators in another paper.

The other important influence was in the management of client expectations and this is now realised to be a vital factor, delivering a high quality basic system rather one with lots of extra, mainly unnecessary, features, was very instructive. These students are very enthusiastic about satisfying their client's requirements and sometimes they can try too hard and the project is then put at risk because they cannot deliver it all in time and of a high quality.

For the projects that involved brand new projects we introduced the students to a new approach for organising stories and for the creation of provably powerful test sets. This approach was tried out on a web-based project and immediately produced excellent results, being both simple to use and very powerful in its ability to capture the essence of the system. However, the projects are still on-going and so it is perhaps too early to make any firm conclusions.

Risk management.

Part of any successful company activity is the management of risk. In both Genesys and the Software Hut these are important activities. XP raises a number of different issues to the traditional design-led approaches. We have, in the past, carried out two phases of risk analysis. Initially at the start of the project the teams are asked to carry out a risk analysis for their project and to record the results and create their workplan in the light of these results. After 7 weeks there is a second risk analysis exercise, which is clearly informed by the problems and successes of their project over the intervening period. The plan is then altered to suit the circumstances. At this stage the scope of the project is usually reduced due to the original plans being downgrading some of the desirable requirements to optional. With XP this process may need to be rather more continuous and is an area we wish to consider.

4 FUTURE EXPERIMENTS

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.

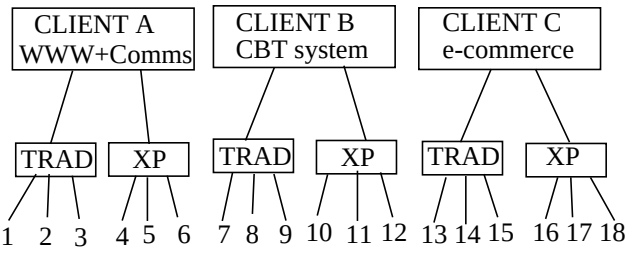

Fig.1 The organisation of the 18 teams.

The three clients are as follows:

Client A is a company of management consultants specialising in human resource business for the automotive production industry. They wish to have a mobile communications based system that interacts with their Web pages and their management databases, planning and sales systems. The main focus of the project will be the web pages but designed with these other interfaces in mind.

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, 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.

One area that we have to address concerns the advice we give about the form of the project plan. Clearly the XP-based plans will be very different to the traditional approach and it will be a new phenomenon for the tutors to be managing a set of projects which are at very different stages at any one time. The students will also compare notes to some extent and I hope that the teams using XP will be discreet about what they are doing so as not to influence the other teams too much. we have found, in the past, that the competitive element has minimised this.

Part of this trial run will be learning about the sorts of metric and data we need to enable us to carry out proper comparisons. We will then be able to run better experiments subsequently.

5 CONCLUSIONS AND FURTHER WORK

Clearly, it is early days and there is much work to be done. What is different about our approach is that the student teams are building real systems for real clients. Thus they face, immediately, the issues of communicating with their client and of trying to understand the client’s business context as well as their problem. *This is vital*. Normal student project experiments are rarely valid because the whole exercise is something of a sham and everyone knows this. Nobody really wants the products to use in real life. The Software Hut approach also creates the desire amongst nearly all the students to do it properly as they realise that delivering software full of bugs, or with an unusable interface just will not do. They have some professional pride and don’t want to let the University down. We are convinced that this means that we can really carry out legitimate empirical experiments in controlled conditions and that the results will be meaningful.

This is just the start. We are bound to see, as XP evolves, the emergence of different ways of doing it, using different tools, methods and notations. This will give us further opportunities to test out the ideas in what we call our *Software Engineering Observatory*: the *Software Hut* for

detailed comparative experiments and *Genesys* where we are investigating how new ideas and methods can be introduced into a working software company.

ACKNOWLEDGEMENTS

We would like to acknowledge our colleagues, Tony Simons, Tony Cowling, Gerald Luettgen and Kirill Bogdanov for many useful discussions and suggestions. We would also like to thank our clients who agreed to working with our students on their problems.

REFERENCES

1. A. Stratton, M. Holcombe and P. Croll "Improving the quality of software engineering courses through university based industrial projects." in "*Projects in the Computing Curriculum*", (eds.) M. Holcombe, A. Stratton, S. Fincher, G. Griffiths, Springer, (1998). 47-69.
2. M. Holcombe and A. Stratton. "VICI: experiences and proposals for student run software companies." in "*Projects in the Computing Curriculum*", (eds.) M. Holcombe, A. Stratton, S. Fincher, G. Griffiths, Springer, 103-114.
3. M. Holcombe & Helen Parker, “Keeping our Clients Happy: Myths and Management Issues in "Client-led" Student Software projects.” *Computer Science Education*, 9 (3), 230-241, 1999.
4. K. Beck, *Extreme Programming Explained: Embrace Change*, Addison Wesley, 1999.
5. http://www.genesys.shef.ac.uk
6. XProgramming Web site, On-line at <http://www.XProgramming.com/ >
7. Genesys Solutions Web sit, On-Line at <http://www.genesys.shef.ac.uk/ >