Institution: BRUNEL UNIVERSITY (H0113)

Unit of Assessment: 11 – Computer Science and Informatics

Title of case study: Software Engineering Impact through Fault Analyses

1. Summary of the impact (indicative maximum 100 words)

The impact of the research is evident in two areas of software engineering practice connected through software fault-proneness: (i) improper use of 'design patterns', recognised reusable templates for how to design code; and (ii) the *real* benefits of 'refactoring', a technique whereby code is intentionally changed by a developer to improve its efficiency and/or make it easier to read. Application of the research findings has led to significant impacts on software development at BancTec Ltd., a medium-sized, international IT company which, as a result, has changed its practices, challenging established approaches in industrial IT. The research has had, and continues to have, direct and sustained impact at BancTec through changed commercial practice and raised awareness of internal standards; this has led to increased training of developers and rollout of new internal software development standards in the UK and India, and as a template world-wide for 2,000 employees in 50 countries.

2. Underpinning research (indicative maximum 500 words)

The research was undertaken at Brunel as part of a partnership with BancTec Ltd. between October 2008 and June 2012. Dr Counsell (Senior Lecturer then Reader at Brunel during the whole of the eligible period) supervised the research. Dr Tracy Hall (a Reader at Brunel during the whole of eligible period) provided additional expertise on fault profiling and empirical studies. BancTec was a member of the Refactoring and Testing (RefTest) EPSRC network EP/E055141/1 [1] (on which Dr Counsell was PI and Professor Hierons, also at Brunel throughout the eligible period, was CI). The remit of the network was to address industry-based problems in refactoring and to test aspects through explicit collaboration between industry and academia.

The relationship between design patterns and the preponderance of faults in code has, anecdotally, been assumed to be an inverse one: applying design patterns leads to fewer faults. From this perspective, developers should therefore seek to maximise the use of software design patterns to minimise later maintenance and faults in their code. An analogy to describe the context of design patterns would be that of a car engine, whose design, irrespective of model or make, is broadly similar and well-understood, but which can also be modified to accommodate different capacities, car sizes, etc. The underpinning research [2] challenged this accepted wisdom through the mining of a large repository of 250,000 C# lines of code, the identification of the set of faults exhibited by that system, and the set of changes that had been applied to that code base over a 12 month period. The findings showed that use of design patterns by developers was not as effective as theory suggests in terms of their change- and fault-proneness. Code developed using design patterns caused *more* faults over time than code that was 'non-pattern' based. The research also found that design patterns caused more changes to be made by developers than code that did not use design patterns. The reason for this was revealed by follow up interviews with the developers: if a code-based artefact is well-understood by developers (as patterns are), it becomes a target for change. It then follows that if that same artefact is changed frequently (relative to other code), then it will induce faults.

The relationship between refactoring and faults has anecdotally and theoretically been assumed to be positive: appropriate refactoring reduces the number of faults in a software system. There had, though, been no empirical evidence of this. Refactoring is the process of making changes to code to improve its efficiency and/or readability without changing what that code does. Returning to the earlier analogy, refactoring is equivalent to an annual service of a car by mechanics in which small parts are replaced due to 'wear and tear'. Developers should 'refactor' code whenever possible and, in theory, faults in classes will be minimised as a result. The research in [3] examined thousands of classes (pre- and post-refactoring) in a large C# system; the classes were systematically 'tracked' to determine whether the same classes when refactored showed a higher propensity for faults. Classes that had been refactored over the period showed a lower fault propensity than classes that had not. The same set of refactored classes were also changed less over that period. The research therefore supported the theory and anecdotal evidence that

refactoring does provide subsequent benefits. No empirical study before this research had demonstrated that undertaking refactoring provides quantifiable benefits later on.

3. References to the research (indicative maximum of six references)

[1] RefTest Network: http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/G031126/1

[2] M. Gatrell, S. Counsell, T. Hall, Design Patterns and Change Proneness: A Replication Using Proprietary C# Software. 16th IEEE Working Conference on Reverse Engineering. Lille, France, 2009, pages 160-164 (http://doi.ieeecomputersociety.org/10.1109/WCRE.2009.31).

[3] M. Gatrell, S. Counsell, The Effect of Refactoring on Change and Fault-Proneness in Commercial C#, second round of revisions for Science of Computer Programming Journal, 2013.

4. Details of the impact (indicative maximum 750 words)

We treat each of the two findings from Section 2 separately in explaining the impact that has arisen from the research. The research has had far-reaching implications for the way that BancTec thinks about and develops its software and the role that programmers play in that development.

The research into design patterns uncovered features of BancTec's software systems of which they were wholly unaware, yet which had significant implications for resource allocation if left untreated. Concurrent with its publication, the research directly caused further internal company investigation/research and has prompted the Commercial Research Director and team to allocate time each week to explore how the problems caused by the use of design patterns could be prevented and to identify and implement mechanisms to effect this. The developers at BancTec directly benefitted from the research because it provided knowledge about the systems they maintained that they did not have before, and would not have had the time to explore themselves. The solution to the problem of design patterns exhibiting excessive faults (and the solution actually implemented as a result of the research) was simple – standards in the company should insist that design pattern classes be extended (through inheritance mechanisms) rather than being modified. The impact of the research has therefore been a change in the way that developers use and deploy design patterns at BancTec. As the Director of Development at BancTec states [4]: "*Increased training and guidelines have been given on the role of design patterns in an attempt to break the trend, identified during the research, of design pattern participants becoming bloated and fault prone*". This has been implemented by BancTec, in their London and Delhi sites, through training for their developers, and is being used as a template that is being rolled out world-wide – BancTec has offices in 50 countries across the US, Asia and Europe. The research has therefore had impact on company policy worldwide.

In relation to the second research finding, commercial fault data is exceptionally difficult to obtain for the simple reason that sensitive commercial data is protected for financial and company image reasons. As such, very few studies have explored refactoring and its link with faults. While no concrete evidence exists to condone its use, refactoring by developers is encouraged. In other words, developers refactor because they believe it to be of both short and long-term benefit. The impact of the research was in providing developers with concrete evidence that refactoring is a beneficial activity and thereby reinforced usual practice at BancTec. This encouraged refactoring and reinforced the benefits to BancTec of the approach - promoting good practice on the basis of an evidence base showing refactoring's effectiveness. As the Director of Development at BancTec notes [5]: "*We have continued our practice of informal developer-led refactoring, following the research indicating the benefits of refactoring*". This practice has also been implemented by London and Delhi and as a template for world-wide roll-out of 'good practice'.

Both the pattern-based and refactoring-based studies were able to demonstrate empirical results through the automated collection of measurement from the source code at BancTec. The use of measurement (or metrics) has resulted in extended use by BancTec and has had a significant impact on their approach to system development and maintenance. As the Director of

Development at BancTec states [6]: "*Since the research was undertaken we have recorded a richer set of metrics that we have started to rely on during the development and maintenance of our software. Specifically change and fault metrics, but also internal quality characteristics and source code analysis*". The research has also had an impact on leading BancTec: *"towards a more pro-active approach to maintenance - using the metrics as a guide*" [7], having a substantial impact on their organisational policy and approach to software maintenance. The outcomes of the underpinning research on refactoring associated with the case study also embraced analysis of software test artefacts. In particular, it showed that there were problems with different 'types' of object-oriented classes of concern to BancTec. The impact here has been a re-assessment of practices at BancTec, as recognised by the Director of Development at BancTec who states that [8]: "*We are embarking on a study* [at BancTec] *to identify the impact of different testing techniques, following the research indicating that the maintenance of our test code was significant*".

The effect that the introduction of new training and standards is having on designed and developed code is currently being monitored by project managers (as the initiatives are rolled out worldwide) and is expected to lead to sustained impact as this becomes more widely embedded beyond the initial activities in London and Delhi. In terms of economic impact, the Director of Development states that [9]: "*Since our research began and we started to focus on faults and practises surrounding faults ~2008, we have reduced the size of the team working full time on maintenance (as opposed to new development) by 2 developers. The fully loaded cost of two developers would be about $200k"* [approximately £126k]. Importantly, this reduction would allow those two developers to shift their focus away from maintenance to other creative development activities. While BancTec accept that there are other factors which have influenced that reduction (e.g., the platform becoming more stable, the team becoming more experienced on the platform, the rate of new development decreasing), the Director states that [10]: "*a proportion of that $200k can be attributed to the improved practises and greater focus on faults and their contributory factors as a result of the research*".

Although it is impossible to say exactly how much of the annual $200k is due to improved practices because of the various stated contributory factors, if we take a conservative percentage attributable to the improved practices to be 15%, then over the past five years this will have saved BancTec $150,000 (approximately £95k) in maintenance costs alone; world-wide over the next five years (at all 50 offices) it will save BancTec $7.5 million (approximately £4.72 million) in costs as well as the concomitant benefits of a significant number of BancTec staff worldwide being able to focus on other aspects of development.

5. Sources to corroborate the impact (indicative maximum of 10 references)

Quotes 4-10 reported in section 4 were taken from two written corroborating sources provided by the Director of Development, BancTec.