Using software Development Progress Data to Understand Threats to Project Outcomes

Tracy Hall
University of Hertfordshire
United Kingdom
t.hall@herts.ac.uk

Austen Rainer
University of Hertfordshire
United Kingdom
a.w.rainer@herts.ac.uk

Dorota Jagielska
University of Hertfordshire
United Kingdom
d.jagielska@herts.ac.uk

Abstract

In this paper we describe our on-going longitudinal study of a large complex software development project. We discuss how we used project metrics data collected by the development team to identify threats to project outcomes. Identifying and addressing threats to projects early in the development process should significantly reduce the chances of project failure. We have analysed project data to pinpoint the sources of threats to the project. The data we have used is embedded in the project's fortnightly progress reports produced by the project team. The progress reports are part of the software measurement program this company operates. The company has highly mature development processes which were assessed at CMM level 5 in 2004. Our analysis shows that standard project progress data can generate rich insights into the project; insights that go beyond those anticipated when the metrics were originally specified. Our results reveal a pattern of threats to the project that the project team can focus on mitigating. The project team is already aware of some threats, for example that communication with the customer is a significant threat to the project. But there are other threats the team is not aware of, for example that people issues within the software team are not a significant threat to the project.

1 Introduction

In this paper we show that secondary analysis of software project data can yield rich insights into the threats to a successful project outcome. In particular we show how project data can be used to identify the nature and scale of threats to the project. In this paper we describe our on-going longitudinal study of progress and outcomes in a large complex embedded software development project (LEDS[1]). We show how we re-analysed project metrics data, collected by the software development team, to identify and quantify threats to the success of the project.

There remains a compelling case for building an understanding of what impacts on project outcomes. Many software project failures continue to be reported in the press. Indeed the Standish Group reports that in 2000 only 28% of U.S. projects were completed successfully (ie, on time and on budget with all features and functions that were originally specified). This means that 72% of projects were either challenged (completed and operational, but over time, over budget or with fewer features or functions than originally specified) or end as a failure (cancelled before completion or never implemented). Furthermore the Standish Group reports that 137,000 projects were late and/or over budget in 2000, while another 65,000 failed completely [32].

Examples of projects with less than satisfactory outcomes are commonplace. Microsoft's search engine, Search Beta, experienced technical problems which made it unavailable for consumers on its first day (Computer Weekly, 16/11/04). Microsoft is also expected to drop several features from the update to its Windows 2003 Server so that it can be released in 2005 (Computer Weekly, 16/11/04). In July and August 2004 mobile network operator O2 sent about 1.5 million incomplete bills to its customers due to problems with a new multimillion-pound integrated billing system managed by IBM Global Services (Computer Weekly, 05/10/04). The £390m Libra project, the fourth attempt in fifteen years to build a unified case management system for magistrate's courts across England and Wales, is drastically over budget and time, and its final date of implementation still remains unknown

[1] Pseudo name used for reasons of commercial confidentiality

11th IEEE International Software Metrics Symposium (METRICS 2005)
1530-1435/05 $20.00 © 2005 **IEEE**

(Computer Weekly, 16/11/04; Computing 06/02/03; Computing 05/10/04). Libra was called "a shocking waste of money" by Public Accounts Committee chairman Edward Leigh, as it has already cost nearly three times more then originaly expected (Computer Weekly 29/01/03).

It is therefore critical that software development companies are able to identify and control threats to software project outcomes. Furthermore, it is critical that the project we discuss in this paper, LEDS, is successful. The project is both safety and business critical. LEDS is a safety critical embedded defense system, the success of which is critical to the continued success of the development company. The company is a large UK defense contractor and the project is a novel multidisciplinary development project. We are studying the development of the software component of the project. The software development team has high maturity development processes which were assessed at CMM level 5 in 2004.

In this paper we report on an approach to identifying the threats to this project by re-analysing existing project data. This is an on-going longitudinal study of LEDS, the development of which has a planned duration of 60 months. Currently the project is at the end of month 24 having completed requirements capture and started design. During this study we have used a combination of qualitative and quantitative research methods within a triangulated research strategy [29]. However in this paper we present our analysis only of the project progress data produced fortnightly by the software team as part of their metrics program. These progress reports contain a wide variety of qualitative and quantitative data which track a range of project factors. We have re-analysed the data contained within the first 34 progress reports over a period of 18 months with the objective of identifying threats to the project. Overall in the study we are investigating the following research questions:

RQ1: What impact do a variety of technical factors have on project outcomes?

RQ2: What impact do a variety of social factors have on project outcomes?

In section 2 we provide some background to this work. In section 3 we present our approach to collecting and analyzing the data. In section 4 we outline our findings and in section 5 we discuss the implications of our findings. In section 6 we draw conclusions and summarise.

2 Background

2.1 Success factors

There are many factors that are reported to impact on the outcomes of software development. Goldenson and Herbsleb [10] identify a set of six project success indicators: meeting budget commitments, meeting schedule commitments, product quality, customer satisfaction, staff productivity and staff morale. El Emam and Birk [7] specifically add satisfying specified requirements to this list of indicators. The Standish Group's[2] "Recipe for Project Success" presents ten factors for project success [32,33] Each factor has been weighted according to its influence on a project's success and assigned an appropriate number of points. The factors are summarised in Table 1.

Table1. Factors for project success

FACTOR	IMPORTANCE WEIGHT
Executive Support	18
User Involvement	16
Experienced Project Manager	14
Clear Business Objectives	12
Minimized Scope	10
Standard Software Infrastructure	8
Firm Basic Requirements	6
Formal Methodology	6
Reliable Estimates	5
Other *	5

* Other factors include small milestones, proper planning, competent staff and ownership.

The Standish Group found that projects do not require all ten factors, but the more factors present, the higher the level of confidence in a successful project outcome. Table 1 shows that the Standish Group identifies lack of executive support as the number one contributor to project success or failure. User Involvement closely follows as a critical factor to a successful outcome [32,33]. It is clear that underlying these factors are a complex range of technical and social issues that will impact on the ability of companies to satisfy these success factors. In our

[2] It is important to note the concerns that Molokken and Jorgensen [20] report regarding the nature of the data reported by Standish

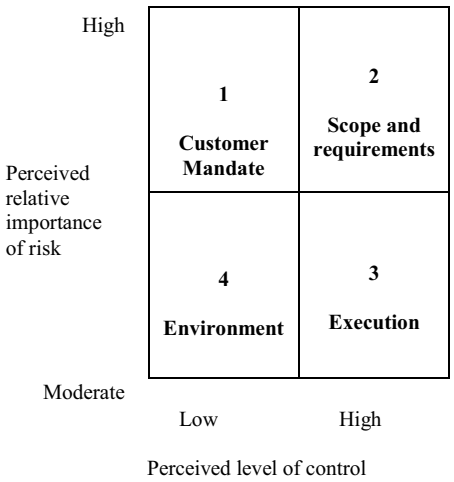
previous work we discuss various approaches to evaluating the success of a software project [30].

Three "pillars" of project success are identified by the Standish Group: project size, project duration and team size. The group believes the smaller each pillar the more likely the project is to succeed. In 1996 the Standish Group recommended 6 people over 6 months and not more than $750,000 [32,33]. In year 2000 they suggested the need for a further reduction of resources to only 4 people, over 4 months and at a cost of less than 500,000 [32,33]. They recommend "growing" software rather than "developing" it (shorter timeframes, with the delivery of software components early and often) as key to raising success rates [32,33]. Clearly such an approach is not always feasible and this implies that many projects are at risk of failure.

2.2 Risks

Risk analysis is used extensively by companies to control and manage risks to projects. In many companies risk management is a large part of the software development process. A study conducted by Keil et al [16], in which experienced software project managers reported and ranked the most important risks to the project, revealed a list of 11 risks to projects. The three most important risks are highly related to the success factors reported by the Standish Group. The top three risks are: lack of top management commitment to the project, failure to gain user commitment and misunderstanding the requirements. The two rated as the most important to the project are risks that the project's management has little or no direct control over.

Keil et al [16] also grouped risks into four factors and placed these factors on two dimensions: perceived importance in relation to the other factors and perceived level of management control over them, creating the following grid:



Quadrant 1: Customer Mandate includes the two top risks mentioned above. The success of the project often depends on the commitment of people outside the development team, such as senior management and the customer. The risks in this quadrant cannot be directly controlled by management, only indirectly influenced.

Quadrant 2: Scope and Requirements focuses on risks related to ambiguity and uncertainties which arise whilst establishing the project's scope and requirements. For example misunderstanding the requirements or not managing the scope of the project properly. The threats from this quadrant can be reduced by educating customers about the impact of requirements changes and skillfully managing ambiguity and change.

Quadrant 3: Execution concentrates on risks connected with actually carrying out the project, such as whether there are enough people and whether they are skilful enough to execute the project. Examples of risks are: insufficient staffing, lack of effective development process methodology and poor estimation. Risks from this quadrant fall generally within the control of the project's managers.

Quadrant 4: Environment includes risks coming from the project's environment (internal or external to the organization). Examples of these are conflicts between user departments, changes in the competitive environment or changes in senior management. These risks are the most difficult to predict and they can be very dangerous to the project. However the likelihood of their occurrence is low.

In this paper we look in detail at the threats to the LEDS project. These threats are the factors that contribute to the high level risks reported in the literature. Our analysis of threats to project outcomes is at a finer level of granularity than is typically reported in the literature. Our findings identify clusters of lower level issues that lead to higher level risks.

3 Methods

In this paper we present our analysis of longitudinal metrics data from 34 project progress reports. These reports track progress over an 18 month period on the LEDS project. These reports are used by software and project managers to track the progress of the project and to support project management. The objective of this longitudinal study is to investigate factors impacting on the outcomes of the LEDS project. We are tracking a range of technical and social factors over time and relating these to interim project progress and final project outcomes. This is an on-going study the logistics of which are now described (the methods used are described in more detail in [12]).

3.1 Longitudinal studies

Longitudinal studies consist of collecting data at a number of points in time from the same data source [4, 27]. They combine the benefits of field studies, ie studying the phenomenon in the complexity of its natural environment, with benefits related to time, ie capturing the dynamic nature of the phenomenon and observing changes over time [27]. Longitudinal studies also allow the direction of causal relationships between investigated variables to be identified [4].

Extensive use of longitudinal studies has been made in a number of disciplines, for example medicine (e.g. [27]) and organizational psychology (e.g. [15]). Longitudinal studies have also been used in a few software engineering research studies. Waltz, Elam and Curtis [34] undertook longitudinal studies to investigate knowledge acquisition, sharing and integration in a single software design team at MCC. Maximilien and Williams [19] conducted a year-long study with an IBM software development group to examine the efficiency of test-driven development. Porter et al [23] used longitudinal studies to observe variation over time in the effectiveness of software inspections. We have previously used longitudinal studies to investigate progress in two software projects at IBM Hursley Park [25]

3.2 LEDS

LEDS is a complex and novel engineering product which is being developed for the defense industry. The overall project is multidisciplinary involving the development and integration of a range of sophisticated hardware and software. In this study we track the development and integration of the software component of the project. Software development on the LEDS project is highly mature. In 2004 the software department was assessed as operating at CMM level 5. The department operated at level 4 for the previous 2 years.

The success of LEDS is very important. The company considers LEDS to be a prestigious project with a high internal and external profile. Managers have set up the project to show-case the high quality work of the company. Consequently managers have carefully selected highly skilled and experienced developers for this project. The project team is made up of 10 developers who are led by a software project manager. The software team is one of 4 teams in the overall project. During the project the team remains part of the software department.

The software project started in 2003 and is scheduled to complete in 2008. The software requirements specification is now complete and design is underway. LEDS development uses a waterfall approach to software development.

3.3 The logistics of the study

In this study we analyse project metrics data collected on a fortnightly basis. Project progress reports are produced by the software team as part of their normal high maturity development process. The reports provide low-level data relating to the day-to-day progress of the project. These reports are not produced specially for this study and so there is no incentive for the data to have been sanitised for our benefit (though clearly we do not know whether it has been sanitised for other purposes). Figure 2 provides an example outline of such a project progress report.

3.4 Project progress metrics data

The data in these reports covers all aspects of the projects' technical progress. Most of the data is quantitative and includes: full schedule data, effort data, risk data and inspection data. However the reports also contain a variety of free flowing qualitative data relating to technical, personnel and project management issues that have arisen during the previous two weeks. We have used conventional statistical analysis to analyse the quantitative data, mostly simply using frequency counts and measures of central tendency. We used content analysis [18] to analyse the free flowing qualitative data to identify every threat reported. One author read

through every progress report and identified all issues reported that may threaten the project. We then identified categories of threats in the project using a grounded approach [3]. This means that we grouped all similar threats together to identify themed categories. One researcher then classified each threat according to a single category. Each threat has not been classified in multiple categories. This could be considered a limitation to this analysis. We then performed an informal inter rater reliability test to ensure repeatability to the classification of threats.

Figure 2. Outline of progress reports

Software Progress Report

1. **Progress & Schedule Data**
2. **Process Improvements**
3. **Dependencies & Coordination**
4. **Resources**
5. **Risks**
6. **Actions taken this session**
7. **Effort & Cost**
8. **Requirements, Size & Critical Computer Resources**
9. **Reviews**
10. **Quality Assurance**

4 Findings

4.1 Overview of threats to LEDS

We analysed each project progress report to identify a total of 81 unique threats to project progress reported in the 34 progress reports. Many of these threats were resolved quickly and the risk to the project by such threats was dissipated. Consequently we analysed only those 29 threats that were on-going for more than 3 sequential reports (ie more than 6 weeks). We assumed that these were the threats that put the project in most danger. However we did not rate the severity of each threat, this may be a limitation to our findings as we have no evidence at this stage that threats going on for more than 6 weeks are the most dangerous to the project. Indeed it may be that threats were left unresolved for long periods as they were considered minor threats. This aspect of threats will be further analysed later in the project.

We used a grounded approach to identify the categories which describe the origin of the threat. Table 2 shows the classifications we derived from the data.

Table 3 shows the number of threats according to each category. Table 3 clearly shows that problems in requirements pose most threats to LEDS.

Table 3 suggests that the majority of threats to the project are related to requirements capture. There are significantly more requirements threats than any other threats and requirements threats go on for prolonged periods. This may be expected given that the majority of the work on the project so far has been on requirements. The nature and substance of requirement threats is discussed in more detail in the next section.

Table 3 also suggests a low occurrence of other types of threat. Technical and tool based threats are the next biggest threats followed by threats from external entities[3].

Organisational and people issues have relatively low occurrences. This is contrary to what the software team expected. The team anticipated that people issues would pose a more significant threat to the project than our analysis of the data suggests. It may be that all people issues are not recorded fully in the project progress reports (though there is no evidence to suggest this). It may be that there are many minor people-related threats which have been factored out of this analysis. Alternatively it may be that the two people-related threats shown in Table 3 are seriously problematic. The first people threat relates to a key developer leaving the company and exposing an experience gap in the team. The second people threat relates to the seconding of 2 members of the software team to another project team. This was to stop that team falling further behind schedule and thereby impacting on the software team's ability to progress.

Table 3. Scale and distribution of threats

Threat	**Unique threat occurrences**		**Mean duration of each threat**
	number	percentage	weeks
Requirements	12	41	14
Tool	5	17	10
Technical	5	17	11
External	3	10	15
Organisational	2	7	8
People	2	7	9
Total	29	100	13

[3] Many of the 'requirements' threats are related to the client and so could also be classified as external threats

Table 2. Categories of threats

Threat	Description	Example threats from progress reports
External	These threats emanated from outside the project. These included threats from other departments or from external contractors. Such threats are largely beyond the control of the software team.	*"Permission to progress" letter not signed by senior manager in development organisation*
Organisational	These threats were issues from within the company developing LEDS. Such threats are largely beyond the control of the software team.	*Funding to an element of the project not released*
Requirements	These threats related directly to problems in requirements collection. Many of these threats were, in practice, beyond the control of the software team.	*Changed requirements will have a knock-on impact on two important elements of the system.*
People	These issues related to any people or human issues that were threatening the progress of the project.	*Experienced developer left the company and left an experience gap*
Tool	Threat related to the tool use of the development team. A wide variety of sophisticated tools are used which are new to the team	*Problems running a version control tool on a secure network*
Technical	Any threat that is of a technical nature related to developing the system. Includes overcoming system limitations and implementation difficulties associated with the novelty of the application.	*Software needs to be made more efficient*

Table 4 shows the pattern of threats over time. It shows a large block of requirements-related threats in the first 19 progress reports. After this these requirements threats seem to be resolved and drop off the threats agenda. Section 4.2 discusses the reality of the pattern of requirements threats in more detail. Table 4 also shows that there is a regular pattern of technical and tool related threats throughout the project. Similarly there are a number of regular external and organizational threats shown. A few of these threats appear to be resolved, only to re-appear several months later.

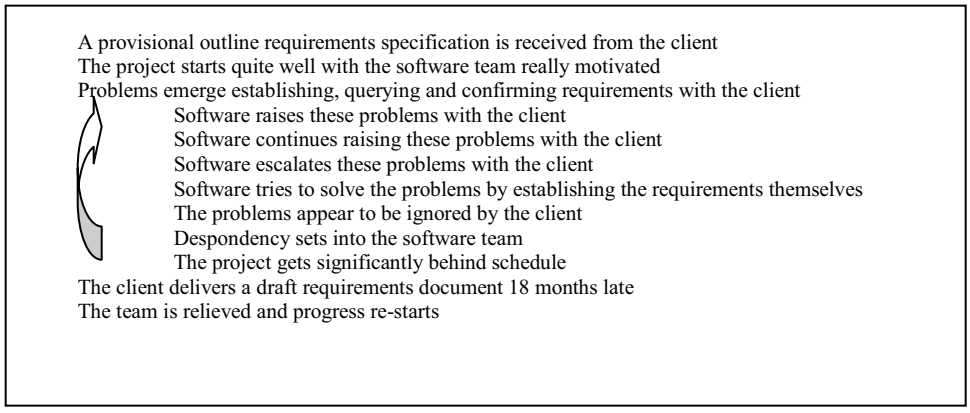
4.2 Requirements-related threats

Tables 3 and 4 clearly show the significant threat that requirements-related issues pose to the successful outcome of LEDS. The first 6 months of the project seems to be in crisis because of various problems with requirements. Most of these requirements problems relate to the client company not engaging with the developers to identify the requirements for the software. In particular the client company:

- Did not respond to requirements queries from the software team
- Did not provide the software team with critical information
- Did not seem to understand how critical they are to the success of the project
- Provide apparently random pieces of requirements information
- Provide a critical requirements document 18 months later than originally agreed

The requirements threats reported in the progress reports strongly suggest that the developers and the client are unable to understand each other. The data also suggests that the working methods of the two organizations are not highly compatible. This means that a critical high profile project has got off to a worrying start.

Figure 1. The relationship between the software team and the client



Motivated by the progress data indicating that there was a severe problem in requirements we investigated the project documentation more thoroughly. We looked at all the interactions the software team had with the client company and Figure 1 shows that the reality of the relationship with the client company was a major threat to the project. Figure 1 suggests that this cycle of poor communication between the developers and the client may put the outcome of LEDS in jeopardy. If this cycle is repeated the project is probably in severe jeopardy. It is not currently clear how much additional interaction is required with the client and therefore what level of risk the project is currently in. This will become clear as our longitudinal study progresses.

4.3 Response from the software team on our analysis of the metrics data

In this section we outline the explanations and responses from the software team on being presented with this analysis of their progress data. We presented our findings to members of the team during a formal feedback presentation session. Their reflections add context to the results we present.

Overall the software team was not surprised by the pattern of threats we identified. They interpreted most threats as being related to the team perceiving a 'lack of control' in the project. The team had the following specific comments:

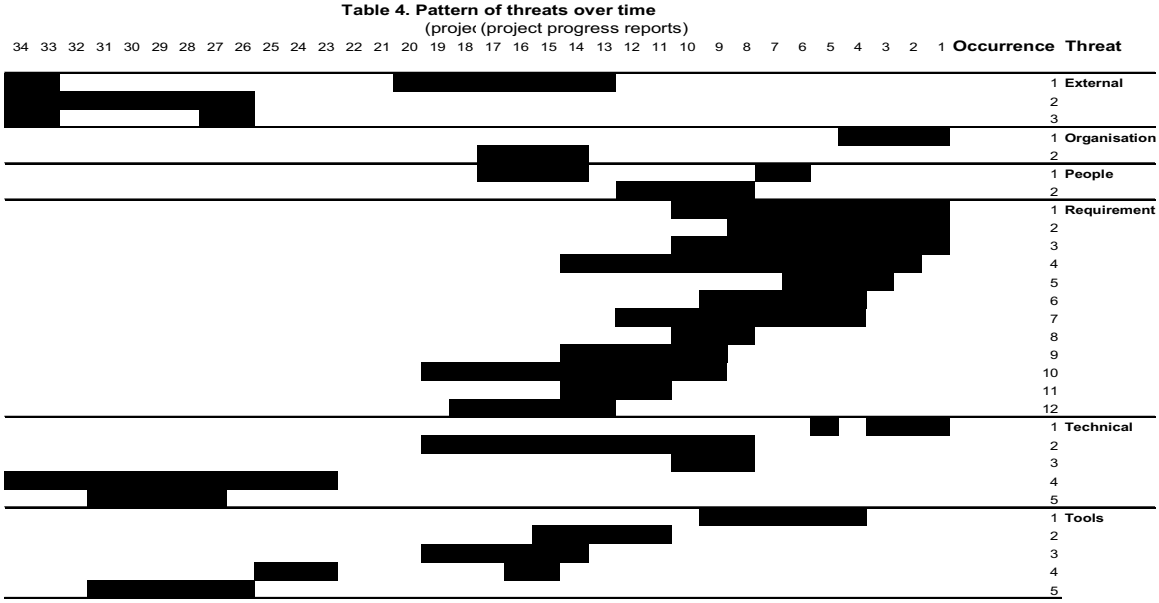
Requirements threats. The software team were not surprised that requirements was the most significant threat to the project. However they were relieved to see the problem clearly emerge in a quantified analysis. They also explained that the dramatic reduction in reports of requirements threats at month 7 was not entirely due to the requirements problems easing. Reports had also reduced because of a reduction in expectations. Under the circumstances, the software team decided not to continually record these requirements problems in the project reports. This may be a limitation to our study given that long standing important threats stopped being reported.

The team also reported that they were still not happy with the relationship they had with the client. They believe the client does not understand the software team. They also believe that the client's working methods are difficult to work with. They believe that the situation will improve when they have completed requirements and are able to 'cut off' from the client.

Tools/technical threats. The software team explained that there were more of these types of threats in the LEDS project than they normally experience. This was because a new toolset had recently been introduced. Consequently the team is still on a learning curve. In addition the project was breaking new technical ground and a variety of unexpected technical issues had arisen but had been resolved.

People threats. The software team reported that subsequent to our analysis more people problems had occurred. This had resulted in people being moved to other projects to make room for new people on LEDS to fill skill and experience gaps.

Table 4. Pattern of threats over time



5 Discussion

The findings we present here occur midway through a longitudinal study of LEDS. As such we can report only preliminary findings on interim project progress. However we speculate on some preliminary answers to our research questions:

RQ1: *What impact do a variety of technical factors have on project outcomes?*

Requirements is the major technical process that is currently threatening the success of LEDS. This threat can be located in the second quadrant of Keil et al's [16] grid, it is therefore perceived as bringing a high risk to the project. However we suggest that social and communication problems motivate and underpin many of the problems in this technical process. Other technical issues so far in the project are one-offs that do not seem to pose a significant threat to the project. Most threats seem to be short term technical or tool-based problems that the software team is able to resolve. This may change as the project progresses and we will particularly monitor the proportion of requirements threats we report in this paper with threats that emerge in subsequent development phases.

RQ2: *What impact do a variety of social factors have on project outcomes?*

Most of the threats to LEDS have some roots in the social and communication aspects of the project. We argue that the root causes of many of the requirements threats are related to social, organisation and political issues. Again the balance of social as opposed to technical issues may change as we continue to track the progress of LEDS and move into phases of technical implementation.

Our findings confirm many of the critical success and risk factors reported in the literature. In particular our findings confirm those presented by Keil et al [16], in which they report that the three most significant risks are: lack of top management commitment to the project, failure to gain user commitment and misunderstanding the requirements. Our findings show that failing to secure all of these poses a serious danger to the project. Top management commitment, even when supported by experienced project management, is unable to make up for deficiencies in the other two areas. Our findings also suggest that some of the significant threats to the success of LEDS are beyond the control of the software team. Successful software development may be dependent on external entities, in particular the client organization. However an alternative explanation could be that the software team is more likely to worry about threats beyond their control, and therefore unnecessarily emphasise those threats.

In addition our findings throw light on client behaviour during requirements. In this study the LEDS' client organisation does not seem to appreciate the important role that they play in determining a successful project outcome. Using Keil et al's [16] grid categories the project is featuring in risk quadrants 1 and 2: the project does not seem to have a clear customer mandate and the requirements of the project are problematic. Furthermore there is no evidence to suggest that the client organisation understands their role in the identification of requirements. However it may be that the client organisation is unable to participate in requirements in the way the software team expect. This might be a common problem underpinning the plethora of requirements problems reported in the literature.

Our findings also raise other issues related to how the software team interacts with the client organization. We suspect the software team has very high expectations of the client organization. Expectations that may, or may not, be realistic. The software team's high expectations are probably related to their high process maturity. Their software processes operate at a very high level of maturity and the software team seems to expect other entities to be able to interact with them on the same basis. This may be very threatening to external entities which are not at this high process maturity. The software team is rightly proud of their CMM level 5 status. But this may make the team highly demanding of other entities. The software team seems to perceive the lower maturity entities that they interact with, as a threat to their ability to optimize software development. Certainly the delay to software development caused by the slow responses from the client organisation was perceived as damaging to the high maturity status of the software team. These high expectations of others may need to be managed more explicitly by high maturity companies as they are potentially counter-productive and damaging to projects.

The high maturity status of the company may also explain the pattern of threats we report. Most threats to the LEDS project have social, organisational or communication underpinnings. Such issues are largely outside the scope of the CMM. The CMM addresses the design of development processes and the management and control of those processes. Our findings suggest that this high maturity team has their development processes well under control. Issues that the team perceive as most threatening to them are beyond their control and therefore beyond the scope of their high maturity process. Our findings suggest that high maturity does not insulate projects from external influences. Indeed these external influences may be most threatening to a successful project outcome.

The high maturity status of the software team means that, as a part of their process, the team collected a comprehensive set of project metrics data. Our findings show the immense value of the metrics data collected. In particular we have shown that there is additional value in the data collected when it is re-analysed from other perspectives. Furthermore our findings complement the formal risk management process the software team operates. The data we analysed emanates from another process yet this data could be used to populate and validate the risk process. We did not investigate the software team's risk assessment process as it was not part of the week-to-week management of the progress of the project. Data used in the risk process was said to be less dynamic than the threats data we used. Our data was also at a lower level of granularity than that used in the risk process. However we plan to compare our findings on threats to the formal risk assessment later in the study.

6 Summary and conclusions

In this paper we show our analyses of metrics data that was collected by software developers to track project progress in the LEDS project. We show how re-analysing existing data can be used to identify and quantify threats to progress and outcomes in software projects. We also confirm that the comprehensive metrics collected as part of a high maturity software process provide rich insights into aspects of the project that are otherwise difficult to understand. Furthermore it may be that companies are not using the data collected as effectively as they might.

Overall our analysis of the progress data suggests that requirements problems and maintaining an effective relationship with the client pose significant threats to the successful outcome of the LEDS project. This is the type of problem frequently reported in the requirements literature, but not what might be expected from a high maturity company. Our findings suggest that high maturity companies may not be correspondingly performing in their requirements process. We are currently planning the next phase of this study where we will be talking to the client to understand their perception of requirements. This will generate a rich context for the issues reported by the software team.

Problems with requirements overwhelm, and may ultimately cancel out, the many strengths of the LEDS project. The project was designed to be staffed by highly experienced and skilled developers, there is strong senior management support for the project and the project is meant to show-case the high quality work that can be done by the company. Despite these important critical factors being in place the team has no control over the responsiveness of the client regarding requirements. This lack of control seems to be a major worry to the software team who are used to having control over their development processes.

Our findings suggest that there might be weaknesses in the way high maturity processes interface with external lower maturity processes. This is especially important in an interdisciplinary project such as LEDS where extensive liaison with other entities is necessary. We speculate that high maturity may encourage teams to try and operate in a high maturity 'bubble'. They do not know how to relate their high maturity processes to other project entities and the interface between processes at different levels of maturity may be problematic. Furthermore high maturity teams seem to have high expectations of others. These demanding expectations of others may need to be managed more explicitly as they are potentially counter-productive and damaging to project outcomes. This is another factor we are planning to investigate later in this longitudinal study.

References

[1] Baddoo N, Hall T, Motivators of Software Process Improvement: An Analysis of Practitioners' Views" Journal of Systems & Software, 62(2), 85-96, 2002

[2] Baddoo N, Hall T, Practitioner roles in Software Process Improvement: An analysis using Grid Technique Journal of Software Process Improvement and Practice, 7(1), 17-31, 2002

[3] Bernard, H. (2000). Social Research Methods, Sage

[4] Carver J, Jaccheri L, Morasca S, Shull F, Issues in using students in empirical studies in software engineering education, Software Metrics Symposium, Proceedings. Ninth International , 3-5 Sept. 2003

[5] Denzin N, The Research Act: a theoretical introduction to sociological methods, Englewood Cliffs, NJ, Prentice-Hall, 1989

[6] Dyba T, Kitchenham BA, Jorgensen M. Evidence Based Software Engineering for Practitioners, Software, IEEE,22(1), 2005

[7] El Emam K, Birk A (2000) "Validating the ISO/IEC 15504 measure of software requirements analysis process capability" IEEE T Soft Eng 26(6)

[8] Gielbert N (Ed.), Researching Social Life, SAGE Publications, 2001

[9] Giele J, Elder G, Life Course Research. Development of a Field. in: Giele, J., Elder, G. (Ed.), Methods of Life course research. Qualitative and Quantitative Approaches. SAGE Publications: 1998

[10] Goldenson DK, Herbsleb JB (1995) "After the appraisal: A systematic survey of process improvement" CMU/SEI-95-TR-009-ESC-TR-95-009

[11] Greene JC, Caracelli VJ, Graham WF, Towards conceptual Framework for Mixed Method Evaluation Designs, Education Evaluation and Policy Analysis, 11(3), 1989

[12] Hall T, Baddoo N, Rainer A, Jagielska D (2005) "Longitudinal Studies in Evidence-based Software Engineering" IEEE ICSE Workshop, Realising Evidence-based software engineering, in review

[13] Hall T, Rainer A, Baddoo N, Implementing Software Process Improvement: An Empirical Study, Software Process Improvement and Practice, 7(1), 3-15, 2002

[14] Harrison R, Baddoo N, Barry E, Biffl S, Parra A, Winter B, Wuest J, Directions and Methodologies for Empirical Software Engineering Research. Empirical Software Engineering 4(4), 1999

[15] Houkes I, Janssen P, de Jonge J, Bakker A, Specific determinants of intrinsic work motivation, emotional exhaustion and turnover intention: A multisample longitudinal study. Journal of Occupational & Organizational Psychology, 76(4), 2003

[16] Keil, M., Cule, P.E., Lyytinen, K., Schmidt, R.C. (1998), A framework for identifying software project risks, Communications of the ACM, Vol.41, No.1

[17] Kitchenham BA, Pfleeger SL, Pickard LM, Jones PW, Hoaglin, DC, El Emam K, Rosenberg J, Preliminary guidelines for empirical research in software engineering, IEEE Transs on, 28(8), 2002

[18] Krippendorff, K. (1980). Content Analysis – An Introduction to Its Methodology. Sage Publications.

Krueger RA and Casey MA, Focus Groups: A Practical Guide For Applied Research. Sage, 2000

[19] Maximilien E, Williams L, Assessing Test-driven Development at IBM, International Conference of Software Engineering, Portland, OR, 2003

[20] Molokken, K.; Jorgensen, M.; A review of software surveys on software effort estimation**,** IEEE Proceedings International Symposium on Empirical Software Engineering. Sept, 2003, 223 - 230

[21] Morgan DL Krueger RA, When To Use Focus Groups And Why, in Successful Focus Groups: Advancing The State Of The Art (Morgan DL ed), Sage, 1993

[22] Pettit B, Western B, Mass imprisonment and the Life Course: Race and Class Inequality, Incarceration. American Sociological Review, 69, 2004

[23] Porter AA, Toman CA, Siy HP, Votta LG, An Experiment To Assess The Cost-Benefits Of Code Inspections In Large Scale Software Development,. IEEE Transactions on Soft Eng,. 23(6), 1997

[24] Preece J, Rogers Y, Sharp H, Interaction Design, John Wiley, 2002

[25] Rainer AW, An Empirical Investigation of Software Schedule Behaviour, Doctoral thesis. Depart of Computing, Bournemouth University: UK. 1999

[26] Rainer A, Hall T, Key success factors for implementing software process improvement: a maturity-based analysis Journal of Systems & Software, 62(2), pp71-84, 2002

[27] Remsberg K, Siervogel, A life span approach to cardiovascular disease risk and aging: The Fels Longitudinal Study. Mechanisms of Aging & Development, 124(3), 2003

[28] Sarantakos S, Social Research, Macmillan 1998

[29] Seaman C, Qualitative Methods in Empirical Studies of Soft Eng, IEEE Trans Soft Eng, 25(4), 1999

[30] Shah M, Hall T, Rainer A, Baddoo N, Software engineering projects: How to evaluate success?, University of Hertfordshire Technical Report, Computer Sciences Department, number 349, 2003

[31] Stewart V, Stewart A, Fonda N, Business Applications of Repertory Grid, McGraw Hill, 1981

[32] The Standish Group International Inc., ed., Extreme Chaos Report (2001), sourced from http://www.standishgroup.com.

[33] The Standish Group International Inc., ed., Chaos: Recipe for Success (1999), sourced from http://www.standishgroup.com.

[34] Waltz D, Elam J, Curtis B, Inside the software design team: knowledge acquisition, sharing and integration. Communications of the ACM, 36, 1993