Cost Estimation in Agile Software Development Projects

Michael Lang[1], Kieran Conboy[1,2] & Siobhán Keaveney

[1] Business Information Systems, J.E. Cairnes School of Business & Economics, NUI Galway, Ireland

[2] School of Information Systems, Technology and Management, Australian School of Business, UNSW, Australia

Email: Michael.Lang@nuigalway.ie

Abstract. Numerous studies over the years have shown that information systems development (ISD) projects often run over budget or fail entirely. Such failures are not restricted to certain industry sectors or project types; rather they occur with some regularity in systems development projects and organizations of all types and sizes. Cost estimation has long been a difficult task in systems development, and although much research has focused on traditional methods, little is known about estimation in the agile method arena. This is somewhat ironic given that the reduction of cost and development time is the driving force behind the emergence of agile methods. This study looks at how classical problems which adversely affect cost estimation in traditional ISD are managed within the agile paradigm. A qualitative approach was followed, based on data collected from four companies. Amongst other findings, the study revealed that estimation inaccuracy was a less frequent occurrence for these companies. A number of recommendations can be drawn from the research: estimation models are not a necessary component of the process; fixed price budgets can prove beneficial for both developers and customers; and experience and past project data should be documented and used to aid future estimation efforts.

1. Introduction

A decade has now passed since the chief proponents of what were then called "lightweight" software development methods, – including eXtreme Programming (XP), Scrum, Crystal, Dynamic Systems Development Method (DSDM), Feature-Driven Development (FDD), Adaptive Software Development (ASD) and Pragmatic Programming, – famously convened in Utah to form the Agile Alliance. The outcome of that meeting was the proclamation of a "Manifesto for Agile Software Development" which called for a profound shift in the underlying philosophy of traditional ISD approaches (Highsmith 2001). The Agile Manifesto embodies twelve guiding principles and a declaration of values which places individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan. Fundamentally, the agile movement is based on a new paradigm which argues for a departure from so-called "plan-driven" or "heavyweight" ISD methods on the basis that they are not appropriate in the modern era of rapid change, a viewpoint that increasingly was gaining support in the academic literature through the 1980s and 1990s (McCracken and Jackson 1982; Baskerville et al. 1992). At the turn of the millennium, practitioners and academics alike were calling into question the underlying assumptions upon which traditional ISD methods were based (Highsmith 2001; Russo and Stolterman 2000) and it was broadly acknowledged that there was a need to move from the past imperfect to a better future way of building software (Fitzgerald 2000).

Over the course of the past decade, the notion of "agile" information systems development has found tremendous favour, as evidenced by the increasing number of practitioner and academic conferences, the high rate of uptake of agile methods within industry, and a rapidly growing body of research activity. However, few studies of agile methods in actual use are based on strong theoretical or conceptual foundations (Abrahamsson et al. 2009; Conboy 2009). In the absence of systematic research there are few lessons learned across studies, and thus the existing body of knowledge is fragmented and inconclusive. This is particularly problematic for agile project managers who have trained and worked with traditional, plan-driven development approaches (Dybå and Dingsøyr 2008; Tan and Teo 2007).

Regardless of the methodology adopted, the ISD process requires effective management and planning. A large part of this planning is the creation of estimates so that resources can be appropriately allocated during projects. Numerous cost estimation techniques and models (e.g.

COCOMO, SLIM, ESTIMACS, COBRA, Checkpoint) have been proposed over the years, an extensive taxonomy of which can be found in Boehm et al. (2000). Across their systematic literature review of 304 software cost estimation papers, Jørgensen and Shepperd (2007) identified regression, function point, expert judgement, theory-based, and analogy as the main cost estimation approaches. However, notwithstanding the vast body of cost estimation literature, the chronic problem of cost and schedule over-runs on ISD projects indicates that accurate estimation remains elusive.

One of the main principles of agile methods is to "welcome changing requirements", but changing requirements are a major cause of software cost estimation problems (Jones 2003; Conboy 2010). Alford and Lawson (1979) pointedly remark that "in nearly every software project that fails to meet performance and cost goals, requirements inadequacies play a major and expensive role in project failure". As yet, the issue of cost estimation in agile software development projects has received very little attention in the academic literature, the only previous empirical studies that we discovered in our literature search being those of Cao (2008), who conducted an in-depth longitudinal study on an agile project in which estimates were compared versus actuals, and Ramesh et al. (2007), who investigated agile requirements engineering practices within 16 US software development organizations. Interestingly, whereas Ramesh et al. (2007) observed that "the agile approach towards RE makes the estimation of costs and schedules more difficult than with traditional methods", Cao (2008) found that "estimation in agile development is more accurate than that in traditional development even though agile developers still underestimate the effort". The research discussed in this paper was part of a broader study, but in view of the aforementioned gap in the literature, the aspect that we have chosen to concentrate on here is: how do agile ISD approaches cope with the problems that have traditionally plagued software cost estimation?

2. Literature Review

Estimating the cost of an ISD project is one of the most crucial tasks for project managers but unfortunately it is a persistent weak link. ISD projects have a long history of being delivered over time, over budget, and failing to satisfy requirements. As early as 1958, concerns about information systems project failure were expressed in the inaugural edition of *The Computer Journal* (Caminer 1958). By the time of the 1968 NATO Conference on Software Engineering, the high incidence of failure had reached such proportions that the now infamous phrase "software crisis" was first uttered (Naur and Randell 1969). Brooks (1987) uses the metaphor of "a monster of missed schedules, blown budgets, and flawed products" to convey the essence of this problem. In a US study conducted by The Standish Group (1995), it was found that only 16% of software projects were completed on-time and on-budget, with 53% of projects costing approximately double their original estimates. Somewhat more positive findings were reported by Lang and Fitzgerald (2007), whose survey of 164 Web development companies in Ireland revealed that 67% of projects were delivered within the agreed budget and 33% were delivered on time. However, even though those figures are more favourable, they still indicate that two-thirds of projects for whatever reason are delivered late, and a third are over-budget, which might even be more if the real cost of fixed price contracts was considered.

Lederer and Prasad (1995) conducted a survey of 112 ISD project estimators and implementers, and based on an exploratory factor analysis they classified the principal causes of inaccurate cost estimates in traditional ISD projects into four categories, namely *methodology*, *politics*, *user communication* and *management control*. We use these classifications here to present the main strands of literature and the corresponding findings.

Methodology Issues

Cost estimation problems attributable to methodology issues include the techniques and guidelines employed to produce the estimate, the means by which estimates relating to past projects are examined and reviewed, the setting of standard estimation durations, insufficient analysis when developing estimates, and lack of co-ordination of systems development activities (Lederer and Prasad 1995).

Over-reliance on intuition and personal memory is a concern for project members trying to increase estimation accuracy. Estimation inaccuracy can also be caused as a result of a lack of policies on how

to learn from past experiences and properly deal with failures and mistakes (Ewusi-Mensah and Przasnyski 1995). In agile software development, estimates are normally produced on an iterative basis for each "sprint" of activity, typically 2-4 weeks turnaround. One very commonly used agile requirements specification technique is a "user story", a feature of both the eXtreme Programming and Scrum methodologies. This technique, which has been referred to as "just-in-time analysis", asks users to very succinctly communicate requirements in the form of short, simple task descriptions. Each user story therefore represents a distinct piece of functionality that can be plugged into a system. An overall expected time for each of these stories is estimated by the developers, and the customers then prioritise the stories based on these initial estimates and on the business value of each one (Lovaasen 2001).

The frequency with which estimation is performed, typically at the beginning of every iteration, leads to progressively more accurate estimation by the developers as they become more and more skilled at estimating the tasks (Abrahamsson 2003). According to Highsmith (2003), the nature of agile methods often results in fixed budgets and a fixed schedule, and it is the scope of the project that remains flexible throughout. On the other hand, Ceschi et al. (2005) report that companies using agile methods usually lean towards "flexible contracts instead of fixed ones that predefine functionalities, price, and time".

Agile methods "welcome changing requirements, even late in development"; however, in terms of estimation, the requirements are finalised to a certain extent at the start of each iteration and so developers can devise their estimates being reasonably safe in the knowledge that the scope for the iteration has been broadly agreed (Taber and Fowler 2000). The impact of changes in scope and requirements within ISD projects can vary greatly depending on the stage at which the change is introduced. The cost of change rises phenomenally throughout traditional development (Boehm 1981) while in agile projects the impact of change levels off (Neill 2003). Agile methods aim to reduce the cost of changes throughout the development of a system, but not necessarily to reduce the occurrence of changes (Highsmith and Cockburn 2001).

Political Factors

According to Jørgensen and Moløkken (2003), estimation is typically fraught with "tug of wars" and "political games", therefore high accuracy may not be the only goal or perhaps not even the principal goal of the actors involved. Chapman and Ward (2002) refer to a "conspiracy of optimism" whereby political pressures from within the organisation can lead to unrealistic estimates or reluctance to report the actual outcome. Moløkken and Jørgensen (2003) suggest that software managers may over-report causes of inaccuracy that lie outside their responsibility, such as customer-related causes. Project managers therefore have to be aware of the implications that political factors can have on ISD estimation (Winklhofer 2002).

Lederer and Prasad (1995) identified pressures from managers, users or others to increase or reduce the estimate, or removal of padding from the estimate by management, as political factors that can negatively impact the accuracy of software cost estimation. It is quite common for software developers to experience stakeholder pressure to stay within the original base estimate, but if those estimates were initially pitched or subsequently manipulated in order to satisfy managers or customers, they will usually lead to over-runs and shortfalls (Lang 2009). Within "self-organizing" agile teams, the delegation of responsibility to developers to estimate their own tasks can cause inaccuracies if a developer feels pressurised into underestimating his workload in order to gratify managers or customers. This agile practice can also lead to reluctance by developers to expose themselves to the risk of developing a reputation as having poor estimation/time management skills, or perhaps even seen amongst peers as having limited technical capabilities (Elssamadisy and Schalliol 2002).

Lederer and Prasad (1995) also identified reduction of project scope or quality to stay within the cost estimate resulting in extra work later, and "red tape", as two other political factors. The agile principle of simplicity, which is defined in the Manifesto as "the art of maximizing the amount of work not done", would seem to directly relate to these issues. Additionally, the Manifesto's emphasis on "customer collaboration over contract negotiation", which resonates with Jones' (1988) call that "both parties (designers and clients) have to give up the use of the requirements as a semi-legal basis of control and measurement and agree to work together", means that at least in principle agile ISD cost estimation aspires to be less prone to the ills of adversarial politics. Indeed, one of the criticisms of the traditional heavy-weight methodologies was that they could descend into political "rituals which enable

actors to remain overtly rational while negotiating to achieve private interests" (Robey and Markus 1984). In their study of Web development practices, Lang and Fitzgerald (2007) found that methods could serve a variety of covert political motives, such as being seen to have followed a process in the event of a "blame game" arising between stakeholders about over-runs. While one would hope that the spirit of the Agile Manifesto would precipitate a change of culture in this regard, it is perhaps too optimistic to expect that political factors and their potentially disruptive impacts can be entirely eradicated.

User Communication

Brooks (1987) famously declared that "the hardest single part of building a software system is deciding precisely what to build … no other part of the conceptual work is as difficult as establishing the detailed technical requirements ... no other part of the work so cripples the resulting system if done wrong". Lederer and Prasad (1995) identify users' lack of understanding of their own requirements, frequent requests for changes by users, users' lack of IT knowledge, and poor or imprecise problem definition as major contributory factors to inaccurate cost estimates.

Poor communications with users is one of the most prominent reasons why project estimates tend to be inaccurate (Jørgensen 2003). It is quite normal for users not to fully understand what they want and to be unable to clearly articulate their needs (Brooks 1987; Walz et al. 1993). Boehm (2000) refers to the fickle and rapidly changing nature of user requirements as the "I'll know it when I see it" (IKIWISI) phenomenon. If customers have a limited awareness of the potential and limiting factors of information technology, as is typical, they cannot be expected to be in a position to clearly state their requirements at the outset of an ISD project (Orr 2004; Stamelos and Angelis 2001). This leads to difficulty in producing a complete set of requirements and thus estimation inaccuracy is inevitable.

Keil and Carmel (1995) therefore recommend that a substantial portion of the time assigned to systems development activities should be given over to learning and knowledge exchange between customers and developers. Moreover, they call for direct links between customers and developers, because where communication passes indirectly through intermediaries, it is likely to be less effective because they can filter and distort messages (Keil and Carmel 1995). Similarly, Grudin (1991) makes the point that "go-betweens or mediators often discourage direct developer-user contact … and are often ineffective conduits". Bringing this point forward, the Agile Manifesto states amongst its principles that "business people and developers must work together daily throughout the project" and that "the most efficient and effective method of conveying information to and within a development team is face-to-face conversation". The eXtreme Programming methodology recommends that an on-site customer should be attached to the development team, though in practice this often does not happen or perhaps the role is filled by a customer proxy. This close working relationship between the project team and the customer in agile software development approaches mitigates the traditional problems arising from poor user communication, but on the other hand the effectiveness and success of agile methods is very dependent on customer co-operation and availability, which if not forthcoming threatens to unravel the whole process (Paulk 2002). For example, if a customer was not available to clarify and elaborate on confusing user stories, development activities might have to proceed misguided or stall altogether.

Management Control

Problems caused by management control include management reviews and comparison between estimates and actuals. When management fails to participate in the preparation of the estimate, and does not monitor the accuracy of the estimate, this can contribute to the estimate being inaccurate. Inaccuracy also occurs when management does not refer to the estimate when conducting performance reviews of estimators and other project personnel (Lederer and Prasad 1995).

In order for an estimate to be accepted and adhered to, it must consider and include all members of the development team and in particular the project manager (Agarwal et al. 2001). It also must be communicated clearly to the project team before the development begins. Research has shown that if the estimator is somebody who will be involved in the development, the estimation accuracy is likely

to be higher than if an estimate is produced by a senior executive or a staff member from a different department (Jurison 1999).

Within the agile paradigm, each developer takes responsibility and ownership for the stories that he estimates and so management involvement is less of an issue in agile ISD as it is in traditional development (Schalliol 2001). Management involvement in agile projects tends to be less "hands-on" than on traditional projects and their involvement is at a higher level, enabling them to oversee the estimation process from one iteration to the next (Abrahamsson 2003). Evaluation of team members based on their ability to meet the estimates is less appropriate for agile projects because it is the developers themselves who estimate their own tasks (Schalliol 2001).

3. Research Approach

The notion of cost estimation in agile ISD projects combines an important and much researched project management issue with the relatively new topic of agile development, where comparatively little empirical research exists. In order to gain a deeper understanding of the core issues, we therefore chose to follow an investigative approach based on semi-structured qualitative interviews. For feasibility reasons, the four companies that participated in our study were drawn from a convenience sample, but they were purposefully selected so as to obtain breadth and diversity e.g. indigenous small-to-medium businesses versus larger multinational organisations, well-established companies versus recent start-ups. A comparative profile of the companies is shown in Table 1.

Company name	Year founded	No. of employees	No. of concurrent projects	Typical project length	Team size	Development methodology	Estimation techniques
Travtech	1999	70	15-20	2-3 years	2-5	Tailored version of XP	Expert judgement; Regression
BrightSoft	1995	12-15	4-5	4-6 months	12-15	MSF for Agile Software Development (MSF4ASD)	Analogy
MobilApp	2002	13	1-5	1-2 months	1-10	Variant of XP	Expert judgement
HPG	1971	500	2-3	4-8 months	6-7	Tailored version of XP	Expert judgement

Table 1: Case Study Company Profiles

At each company, interviews were held with project managers / team leaders. A list of interview questions and topics for discussion was emailed to each interviewee in advance. All interviews took place on site within the companies' premises and lasted between one and two hours. Conversations were audio-recorded by agreement of interviewees, and observational notes were also made during and immediately after the interviews. The interviews followed the general course of the pre-planned questionnaire schedule but, where appropriate, elaboration was sought on points that were of high relevance to the research. Upon conclusion of each interview, provision was made for follow-up meetings, phone calls or emails on points that required clarification or further investigation. Immediately after the interviews the recordings were fully transcribed. All interviewees were subsequently contacted via email requesting clarification or elaboration on points made during the interview. The analytical procedures employed followed the general principles espoused by Miles and Huberman (1994), and concentrated on the transcribed interview conversations, notes made during the interviews, email correspondence from before and after the interviews, and any available secondary information about each company's activities.

4. Findings

The experience of the companies in our study was that reasonably accurate estimates for agile projects are easier to produce because of the frequency with which estimates are required. Typical agile iterations spanned two weeks, with estimates being produced at the beginning of each iteration. This not only helped to keep a high degree of accuracy but also honed the estimation skills of the team members and developers involved. Notably, estimation inaccuracy was not a substantial problem for any of the companies and where it was, they typically saw it as an opportunity to learn and inform their future estimation activities.

Fixed price projects where a budget is agreed at the beginning seemed to be the most common project type. In some cases the schedule was movable and in others it was the functionality that could be revised. Typically when the cost is determined, a number of developers can be assigned and the delivery date calculated from this. On the other hand, if the schedule is set by the customer then the cost can be calculated from the number of people available to work on the project. Either way this enables the project to be run in a manner that delivers increasingly more features as time progresses until the scheduled delivery date has been reached.

Methodology Issues

The main estimation techniques used across the four projects were analogy and expert knowledge with varying degrees of formality and structure between the companies. In some cases project data was stored and in others it was simply assigned to the developers own memories. Estimation models, despite their popularity in the literature were not used by the companies and for the most part were not even recognised. “User stories” were very commonly used as the unit of work for which developers were asked to return estimates, and all four companies followed the “planning game” practice.

Procedural flaws and shortcomings

As regards causes of inaccurate estimates related to flaws within the execution of estimation processes, Travtech’s experience is that the requirement for appropriate expertise, and in particular domain and technical expertise, as part of their estimation process is something that has given rise to estimation errors. For example, it has occasionally happened that somebody might be asked to produce estimates in an area that they are not particularly familiar with, or with a technology or development language with which they have limited experience. Travtech’s approach in situations where they find themselves in unfamiliar territory as regards application domain or development platform is to build a risk factor into the estimate to compensate for the amount of time and effort that will be required to come up to speed with the intricacies of new technologies. Interestingly, Travtech also commented that eXtreme Programming, if indeed taken to extreme limits and “applied rigorously with little up front documentation” could lead to costly situations further on where “there is a lot of refactoring which has to be done and this can create real inefficiencies when having to rewrite software”.

The estimation method that HPG use on their projects is quite an informal process and they feel that this lack of formality may contribute somewhat to the discrepancies in their estimates (of the order of 10%), although another possible reason for the variance is not necessarily that the original estimates were wrong, but rather that subsequent change requests were not properly tracked and the initial estimate would therefore appear to be out of line. The fact that they are using agile development practices has also led them to focus less on formality overall. With regard to the development process changes that HPG have experienced, there have been significant repercussions from the adoption of agile processes. The team members have had to adapt to the new practices and learn new skills such as pair programming and test-driven development. This has impacted cost estimates, particularly in the early stages of the project, because they now have to take other factors into account such as refactoring and acceptance testing.

Use of guidelines to counteract under-estimation

Consistent with the findings of Cao (2008), we discovered that although cost estimates are reasonably accurate, developers continue to have a tendency to underestimate. Both HPG and BrightSoft spoke of how they compensate for this by means of the concept of the "perfect engineering day". Interestingly, although agile methods have a strong focus on productivity issues such as "maximizing the amount of work not done" and "maintaining a constant pace indefinitely", HPG and BrightSoft both have a policy of treating a developer's estimate of "8 hours" of work as the equivalent of two working days. As the HPG project manager explained, "team members tend to think that they have spent a full day at a task but in reality will have only spent 3 or 4 hours because of interruptions". BrightSoft combine this "perfect engineering day" rule-of-thumb with their "relative size table", which is a reference guide to the effort taken to complete similar work assignments in the past. This enables them to produce very accurate estimates. For example, if a developer estimates that a task will take one day (i.e. 4 hours), but the relative size table suggests that it will take 2 days (i.e. 8 hours), the average of the two is usually taken as the estimate.

Political Factors

Neither HPG nor MobilApp raised any cost estimation issues related to political factors. BrightSoft explained that they are a small closely-knit crew with a strong collegiate culture and as a result have never experienced any internal political divisions. For example, there is never any suggestion of apportioning blame on individuals for estimates going astray, and though estimates are audited during post-implementation reviews, that is emphatically not for the purposes of evaluating the personnel involved in either the estimation process itself or the development effort.

As regards "playing it safe" with customers, if BrightSoft find themselves placed under pressure to come up with a fixed deadline, they generally react by building two weeks of slack into the estimate as a risk buffer. It may turn out that only a certain number of features are required and the project can be delivered before the risk buffer has been expended, so they recognise that "it is important politically to get the balance right because over-estimating can cause problems as well".

Pressure to reduce estimates

Of the four companies interviewed, only Travtech appear to be experiencing estimation problems arising out of the types of political factors identified by Lederer and Prasad (1995). Pressures from Travtech managers and customers can cause unrealistic estimates to be produced in order to keep a customer on track or prevent a manager from pulling the project altogether. There could also be a combination of both the management and the customers adding to the pressure for lower estimates.

It can be very difficult if the team leader calculates an estimate that represents the capabilities of their team and they know that the customer will not be willing to accept the length of time or the cost required for completing the project. This problem arises more frequently with customers that are not IT-savvy because they are not as appreciative of the effort required to implement certain requirements. This can sometimes lead to under-estimation by team members as they may be conscious in the back of their minds that the customer wants better value.

As regards pressures from managers giving rise to poor estimates, this can occur when there are certain tasks included as part of the estimate and a team member ends up cutting corners to produce a lower estimate than what can be realistically expected. For example, if the testing phase is left entirely until the end of the project, it is often the part that will be omitted in the formal estimates in an effort to bring the estimate down. This will render the estimates inaccurate because effort will always need to be expended on testing regardless.

"Red Tape" issues

Travtech have also occasionally ran into "red tape" political problems, such as where trade unions within the customer organisation can dictate work practices and distribution of tasks, or where the

customer organisation has its own zealously protective internal IT department acting as a gatekeeper. In such scenarios, the political "tug of wars" spoken of by Jørgensen and Moløkken (2003) can arise as stakeholders attempt to wrest control of certain aspects of the project. This presents a serious risk to the integrity of time and cost estimates.

User Communication

Instances of all four of the main types of user communication problems identified by Lederer and Prasad (1995) were evident in our study. As mentioned in the previous section, users' lack of IT knowledge was not just a communication problem, but it also gave rise in some cases to political pressures being heaped on developers by technologically-naïve customers. The other three issues are described in the following sub-sections.

Poor or imprecise problem definition

The Travtech project manager felt that poorly documented requirements and insufficient management of the relationship with and involvement of the customer can leave too much room for misinterpretation, resulting directly in inaccurate estimates. Expectations may be based on an impression that the customer got from a meeting and their methodology might fail to ensure that clarification was sought. Similarly at HPG, estimation problems have arisen as a result of poor problem definition from the customers, or where the customer comes to the team with a change request and the scope of a particular story needs to be revised.

Agile methods aspire to address poor problem definition by placing developers and an "on-site customer" into direct daily communication. However, of the four companies that we visited, only one (HPG) had an on-site customer. As explained by the project manager, "our customer wasn't really on-site up until a few weeks ago when there were a few issues that were coming up in the retrospective regarding communication problems between the development team and the customer, so we decided to set up a machine for the customer in our lab area and now she's on-site probably 80% of her time, which has helped". Both Travtech and BrightSoft mainly supply to the export market so it is not feasible for them to maintain a customer on-site. In order to stay close to their customer, which they feel is important from the point of view of gaining good feedback and getting to know how to handle key individuals, BrightSoft have set up an international office in California close to their main customer base.

Users' lack of understanding of their own requirements

Travtech have experienced inaccurate estimates in situations where the customer knows their own business so well but cannot articulate it in a form that the development team can understand. Customers can often find it difficult to explain or even remember some of the intricate details that can be required in order to produce a concrete set of requirements. They typically know what they want the system to do but they may not be capable of getting this information across to the developers at the early stages of the project and it will often take a prototype version before they can provide a decent specification of the requirements. Of course, there is nothing new about this type of situation (Brooks 1987; Walz et al. 1993; Boehm 2000), but what it clearly demonstrates, lest we forget, is that the benefits provided by the agile development paradigm are negated unless we continue to use good old-fashioned user interface design principles and techniques.

Frequent requests for changes by users

Travtech's approach to handling change requests depends on whatever agreement is in place with the customer. In a fixed price contract the scope will have been signed off and anything after that will probably be charged separately in addition, except if the change is very small. If the requested change is substantial, the additional cost will be negotiated with the customer. Revision of the estimate

therefore is ad-hoc and depends on a number of factors. The stage of the project at which the change is requested is a major factor in deciding whether or not to refine the estimate, as is the effort required and the relationship and contract agreed with the customer.

When asked if BrightSoft adhere to the agile principle of "welcome changing requirements", the project manager assuredly responded "Yes, fact of life for software engineering!", but their experience however is that constant feedback and streams of minor change requests can be overwhelming to the point where initial estimates get completely thrown out and the profitability of a project can be seriously threatened. In one such case, they have conceded that a project has effectively become a "loss leader".

MobilApp's approach is that if the customer submits a "must-have" requirement midway through the project, they will typically do their utmost to include it without affecting the schedule. Changes such as these are not caused by problems in the estimation process, but rather because of problems in the requirements specification and poor definition of needs on the part of the customer. The schedule will typically have to be extended and the customer will be informed that this is due to their late change request. If there is an instance whereby the deadline cannot be extended then the functionality will have to be revised and the time will be made up by omitting some other feature from the original set of requirements. This will all happen with agreement from the customer because it was their lack of understanding of their own requirements that drove the change.

This is one of the advantages that MobilApp find with agile methods, i.e. that the customer can see the mid-results and at any stage have an input and reassess their requirements. If the customer does want to change the requirements they can go back to the team who would then thrash out new requirements for that particular area and give an estimate for it. MobilApp find that once they have a project plan in place and have signed off with the customer on requirements, then these can be changed once the customer is made aware that the schedule for release will be pushed out as a result of the late change in requirements. In this way, costs can be controlled effectively.

Management Control

In addition to expert judgement, Travtech also use regression-based estimates where they look at previous data to compare the actuals with forecasted values and then conduct a variance analysis. This approach is especially useful on projects that have a number of iterations because data from earlier iterations can inform estimates for subsequent phases. Travtech have managed to improve their estimation capabilities not just by comparing whole projects to one another, but also by comparing different phases within the same project. With agile iterations, it is possible to quickly identify through post-iteration reviews if the initial estimates have slipped, and subsequent estimates can then be revised accordingly to absorb any over-runs. However, few customers are prepared to agree to an open-ended budget and most want to sign fixed price contracts up front. In those circumstances, the development company takes all the risk but Travtech find that the combination of experiences gained from previous projects, coupled with the greater control that comes from using atomic use cases or user stories as the basis of estimates, places them in a position where they can confidently price contracts.

Inaccurate estimates caused by IS management approval and control do not affect MobilApp to any significant extent. Generally if a team member comes up with an estimate this will be accepted by the management because they feel that everybody in the company has the experience and skills necessary to estimate fairly accurately without having it checked or validated by management. However, if a post-implementation audit reveals that the estimates for a project were 15% or more out of line with actuals, management regard it as very important to use this information to try to rectify the inaccuracies. Whether it is early or late, they always try to find out where the issues were, what problems arose, why they got ahead or fell behind so much, and what are appropriate reference points for future project estimation.

Similarly, BrightSoft use estimates to audit projects afterwards to determine how successful the estimation techniques were. They conduct a post mortem analysis on all of the estimates for each project and this helps them to review the accuracy of the projects that have just been finished, and it also enables them to create a new set of figures for future estimation.

At HPG, the estimates that are produced for projects are continuously evolving and they are monitored closely at all times. When new pieces of work emerge in later iterations the original estimate is re-examined and this helps to keep the team members focused on the bottom line. In the past when

projects were developed using traditional development methods, they often found that by the end of the project the original estimates were redundant because of the changes that had occurred during the project. On projects that are using agile approaches, the estimate is continuously checked and although inaccuracies still occur the project team are aware of this as it is happening and can react accordingly.

Other Factors

Regarding the personnel who are involved in the projects, Travtech would be conscious of the skills and experience of those assigned to various tasks as this is likely to have an impact on their estimates. The estimate could change if customers were not willing to pay for more experienced people at a higher rate, in which case there would be a need to use people with less experience. In the preparation of the estimate, it is therefore important to have a good sense of who is actually going to be doing the work, or at least the skill level or expertise that they will have.

Similarly, BrightSoft see new people and new technologies as a principal threat to their ability to produce accurate estimates. When people are learning on the job or learning a new technology, it is very difficult to know what to expect in terms of the time and effort required for them to become familiar and comfortable with the technology. A major factor that has contributed to the accuracy of BrightSoft's estimates is the maturity of the programmer, not only in terms of their skills and experience but also their own estimation capabilities. The accuracy of the entire estimate can depend on how well each individual estimates the work that they have to do themselves. If the person does not know their own capabilities or if they have not used the particular type of development process before, they are very likely to come up with an unrealistic estimate.

5. Summary and Conclusion

In their study of inaccurate estimates in traditional ISD projects, Lederer and Prasad (1995) identified 16 items, grouped under 4 factors, that were problematic. These items and the corresponding solutions advanced by the agile paradigm, as evidenced in our study, are presented in Table 2.

Traditional ISD cost estimation problem	**Response of companies in this study**
Lack of an adequate methodology or guidelines for estimating	Short iterations, expert judgement
Inability to tell where past estimates failed	Post-iteration audits/retrospectives
Lack of setting and review of standard durations for use in estimating	Normal iteration = 2 weeks
Insufficient analysis when developing estimates	Planning game
Lack of co-ordination of systems development activities	Lean, lightweight processes, small teams
Pressures from managers, users or others to change estimate	Reduce scope, swap in cheaper labour
Removal of padding from the estimate by management	No evidence
Reduction of project scope/quality to stay within estimate, leading to extra work later	No evidence
Red tape	Intractable, requires patience and diplomacy
Users' lack of understanding of their own requirements	Rapid prototyping, on-site customer
Frequent requests for changes by users	Change request process, renegotiate estimates
Users' lack of IT knowledge	Closer working relationship with customer
Poor or imprecise problem definition	Rapid prototyping, on-site customer
Performance reviews don't consider whether estimates were met	Post-iteration audits/retrospectives, Stand-ups
Lack of project control comparing estimates and actuals	Post-iteration audits/retrospectives, Stand-ups
Lack of careful examination of estimate by ISD management	Know your staff's capabilities, need to tweak unrealistic estimates, build in risk buffer

Table 2: Summary of findings: "agile" responses to traditional ISD cost estimation problems

While agile methods have come some way to mitigating these cost estimation problems, there is little evidence to suggest that the agile paradigm is any less prone to falling victim of shoddy analysis,

stakeholder politics, or disengaged end users than traditional ISD cost estimation approaches. Although Travtech were the only company to have experienced political pressures from customers or managers to reduce estimates, that is not of course to say that similar issues could not foreseeably arise in the other companies. Management control factors were not found to be a major cause of inaccuracies, although the need to keep a watchful eye on over-optimistic estimates by new staff, or staff moving into unfamiliar territory, was noted. All of the companies experienced user communication difficulties at some stage or another and this is potentially a very serious threat to accurate estimates.

Travtech find that when inaccuracies do occur, it is typically due to some lack of understanding between the customers and developers regarding the requirements. It can also be due to a lack of technical expertise in a particular area which would prevent the accurate estimation of certain tasks. BrightSoft seem to be the most confident in their estimation abilities. Typically the estimates produced are relatively on target and if not, the discrepancy is usually negligible. They have found the major potential threats to accurate estimates to have been the introduction of new people, new technologies and too much feedback from their customers. MobilApp find that change requests from customers and lack of estimation expertise can cause problems on some projects, particularly if a new development language is being used. HPG have found that their estimates are typically accurate to within 10% of the actual figures, however they feel that their inaccuracies may be due to their lack of formality in the estimation process.

To conclude, research on project cost estimation has been conducted for decades with a vast number of models and tools in existence. This study has looked at the estimation process in the emerging field of agile development and examined causes of inaccurate estimates and steps to improve the process. From the four case studies, a number of recommendations can be summarised as follows: estimation models are not a necessary component of the process; fixed price budgets may be the best option for both developers and customers; and a critical success factor for agile cost estimation is that experience and past project data must be documented and used to guide the estimation of subsequent projects.

References

Abrahamsson, P. 2003. Extreme Programming: First Results from a Controlled Case Study. In *Proceedings of the 29th Euromicro Conference*.

Abrahamsson, P., K. Conboy, and X. Wang. 2009. "Lots Done, More To Do": The Current State of Agile Systems Development Research. *European Journal of Information Systems* 18 (4):1-7.

Agarwal, R., M. Kumar, Yogesh, S. Mallick, R.M. Bharadwaj, and D. Anantwar. 2001. Estimating Software Projects. *ACM SIGSOFT Software Engineering Notes* 26 (4):60-67.

Alford, M.W., and J.T. Lawson. 1979. Software Requirements Engineering Methodology (Development), *RADC-TR-79-168*, U.S. Air Force Rome Air Development Center, June 1979.

Baskerville, R., J. Travis, and D. Truex. 1992. Systems without Method: The Impact of New Technologies on Information Systems Development Projects. In *The Impact of Computer Supported Technologies on Information Systems Development*, eds. K. E. Kendall, K. Lyytinen, and J. DeGross, 241-269. Elsevier Science Publishers.

Boehm, B. 2000. Requirements that Handle IKIWISI, COTS, and Rapid Change. *IEEE Computer* 33 (7):99-102.

Boehm, B.W. 1981. *Software Engineering Economics*. Englewood Cliffs, NJ: Prentice Hall.

Boehm, B.W., C. Abts, and S. Chulani. 2000. Software Development Cost Estimation Approaches: A Survey. *Annals of Software Engineering* 10 (1-4):177-205.

Brooks, F.P. 1987. No Silver Bullet / Essence and Accidents of Software Engineering. *IEEE Computer* 20 (4):10-18.

Caminer, D.T. 1958. And how to avoid them. *The Computer Journal* 1 (1):11-14.

Cao, L. 2008. Estimating Agile Software Project Effort: An Empirical Study. In *Proceedings of Americas Conference on Information Systems (AMCIS)*.

Ceschi, M., A. Sillitti, G. Succi, and S. De Panfilis. 2005. Project Management in Plan-Based and Agile Companies. *IEEE Software* 22 (3):21-27.

Chapman, C., and S. Ward. 2002. *Managing Project Risk and Uncertainty: A Constructively Simple Approach to Decision Making*. Chichester, UK: John Wiley & Sons.

Conboy, K. 2009. Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development. *Information Systems Research* 20 (3):329-354.

Conboy, K. 2010. Project Failure En Mass: A Study of Loose Budgetary Control in ISD Projects. *European Journal of Information Systems* 19 (3):273-287.

Dybå, T., and T. Dingsøyr. 2008. Empirical Studies of Agile Software Development: A Systematic Review. *Information and Software Technology* 50 (9-10):833-859.

Elssamadisy, A., and G. Schalliol. 2002. Recognizing and Responding to "Bad Smells" in Extreme Programming. In *Proceedings of the 24th International Conference on Software Engineering*. Orlando, Florida.

Ewusi-Mensah, K., and Z.H. Przasnyski. 1995. Learning from abandoned information systems development projects. *Journal of Information Technology* 10 (1):3-14.

Fitzgerald, B. 2000. Systems Development Methodologies: The Problem of Tenses. *Information Technology & People* 13 (3):174-185.

Grudin, J. 1991. Interactive Systems: Bridging the Gaps Between Developers and Users. *IEEE Computer* 24 (4):59-69.

Highsmith, J. 2001. History: The Agile Manifesto, http://agilemanifesto.org/history.html.

Highsmith, J. 2003. Agile Project Management: Principles and Tools: Cutter Consortium.

Highsmith, J., and A. Cockburn. 2001. Agile Software Development: The Business of Innovation. *IEEE Computer* 34 (9):120-127.

Jones, C. 2003. Why Flawed Software Projects Are Not Cancelled in Time. *Cutter IT Journal* 16 (12):12-17.

Jones, J.C. 1988. Softechnica. In *Design after modernism: Beyond the object*, ed. John Thackara, 216-226. London: Thames & Hudson.

Jørgensen, M. 2003. How Much Does a Vacation Cost? Or What is a Software Cost Estimate? *ACM SIGSOFT Software Engineering Notes* 28 (6):1-4.

Jørgensen, M., and K. Moløkken. 2003. A Preliminary Checklist for Software Cost Management. In *Proceedings of the 3rd International Conference on Quality Software*.

Jørgensen, M., and M. Shepperd. 2007. A systematic review of software development cost estimation studies. *IEEE Transactions on Software Engineering* 33 (1):33-53.

Jurison, J. 1999. Software Project Management: The Manager's View. *Communications of the AIS* 2 (3):1-50.

Keil, M., and E. Carmel. 1995. Customer-Developer Links in Software Development. *Communications of the ACM* 38 (5):33-44.

Lang, M. 2009. The Influence of Short Project Timeframes on Web Development Practices: A Field Study. In *Proceedings of 18th International Conference on Information Systems Development*. Nanchang, China.

Lang, M., and B. Fitzgerald. 2007. An Empirically-Grounded Conceptual Framework of Situated Web Design Practices. *Requirements Engineering Journal* 12 (4):203-220.

Lederer, A.L., and J. Prasad. 1995. Perceptual congruence and systems development cost estimation. *Information Resources Management Journal* 8 (4):16-27.

Lovaasen, G. 2001. Brokering with eXtreme Programming. In *XP Universe 2001*. Raleigh, North Carolina.

McCracken, D.D., and M.A. Jackson. 1982. Lifecycle Concept Considered Harmful. *Software Engineering Notes* 7 (2):29-32.

Miles, M.B., and A.M. Huberman. 1994. *Qualitative Data Analysis: An Expanded Sourcebook*. 2nd ed. Thousand Oaks, CA: Sage.

Moløkken, K., and M. Jørgensen. 2003. A Review of Software Surveys on Software Effort Estimation. In *Proceedings of the 2003 International Symposium on Empirical Software Engineering*.

Naur, P., and B. Randell. 1969. Software Engineering: Report on a Conference Sponsored by the NATO Science Committee, Garmisch, Germany, 7-11 October 1968. Brussels: Scientific Affairs Division, NATO.

Neill, C.J. 2003. The Extreme Programming Bandwagon: Revolution or Just Revolting? *IT Professional* 5 (5):62-64.

Orr, K. 2004. Agile Requirements: Opportunity or Oxymoron? *IEEE Software* 21 (3):71-73.

Paulk, M.C. 2002. Agile Methodologies and Process Discipline. *CrossTalk, The Journal of Defense Software Engineering* (October):15-18.

Ramesh, B., L. Cao, and R. Baskerville. 2007. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal* 20 (5):449-480.

Robey, D., and M.L. Markus. 1984. Rituals in Information System Design. *MIS Quarterly* 8 (1):5-15.

Russo, N.L., and E. Stolterman. 2000. Exploring the assumptions underlying information systems methodologies. *Information Technology & People* 13 (4):313-327.

Schalliol, G. 2001. Challenges for Analysts on a Large XP Project. In *XP Universe 2001*. Raleigh, North Carolina.

Stamelos, I., and L. Angelis. 2001. Managing Uncertainty in Project Portfolio Cost Estimation. *Information and Software Technology* 43 (13):759-768.

Taber, C., and M. Fowler. 2000. An Iteration in the Life of an XP Project. *Cutter IT Journal* 13 (11):13-21.

Tan, C., and H. Teo. 2007. Training Future Software Developers to Acquire Agile Development Skills. *Communications of the ACM* 50 (12):97-98.

The Standish Group. 1995. *The CHAOS Report*.

Walz, D.B., J.J. Elam, and B. Curtis. 1993. Inside a Software Design Team: Knowledge Acquisition, Sharing, and Integration. *Communications of the ACM* 36 (10):63-77.

Winklhofer, H. 2002. Information Systems Project Management during Organizational Change. *Engineering Management Journal* 14 (2):33-38.