MONITORING AND CONTROLLING SOFTWARE DEVELOPMENT PROJECTS.

McBride, Tom, Faculty of IT, Department of Software Engineering, University of Technology, Sydney, Australia mcbride@it.uts.edu.au

Henderson-Sellers, Brian, Faculty of IT, Department of Software Engineering, University of Technology, Sydney, Australia brian@it.uts.edu.au

Zowghi, Didar, Faculty of IT, Department of Software Engineering, University of Technology, Sydney, Australia didar@it.uts.edu.au

Abstract

Software development projects are seldom able to be planned so accurately that the plans predict exactly what will happen during the project. Project managers must monitor the project and make changes either to the project's activities or to the plan itself. Yet much of the project management literature gives only cursory attention to the problem of monitoring and controlling the project. To investigate how project managers regard software development projects, this paper examines software development from the perspectives of two extremes: software development as a production problem and software development as a design problem. Empirical evidence was analyzed and it was found that software development is viewed as a production problem by most project managers. The research highlighted that a relatively simple test could guide project managers toward tailoring their project monitoring and control activities to better suit the project type.

Keywords: Software development; project management; production problem; design problem.

1 INTRODUCTION.

Most project management literature e.g. (Hughes *et al.* 1999; Project Management Institute 2000) has little to say about monitoring and controlling a software development project in progress. The assumption is that the project unfolds as planned and that adjustments to the project should realign the project with the plan (McConnell 1998; Thomsett 1989). While this is a reasonable overall strategy suitable for relatively stable technologies, its suitability for many *software development* projects is questionable. Certainly there is little to advise new project managers on just how to monitor and control software development projects.

The way software development projects are structured and planned depends very much on whether they are viewed as a production problem (Boehm 1988; Curtis *et al.* 1987; Krishnan 1998) or as a design problem (Gasson 1998; Smith *et al.* 1993). If they are viewed as a production problem, then it is assumed that the problem at hand is largely understood and that its implementation in an information system is a matter of producing the required documents, code, tests and other artefacts in a reasonably predictable and controlled manner. On the other hand, if a project is viewed as design problem for which the solution is unknown and far from certain, there will be more emphasis on understanding and solving the problem at hand and less emphasis on steady progress toward project completion.

The main contribution of this paper is the conclusion that most projects are structured, monitored and controlled as if they are production problems. By applying a relatively simple test, project managers could tailor their project monitoring and control activities to better suit the project type.

This paper is structured as follows. Section 2 describes the general characteristics of production problems, and how they would be monitored and controlled. Section 3 describes the general characteristics of design problems, and how they would be monitored and controlled. Section 4 proposes the research question then describes the research method used to answer it. Threats to validity are discussed in Section 5 and conclusions are presented and discussed in Section 6.

2 PRODUCTION PROJECTS.

There are a range of production processes varying from mass production through to one-of-a-kind product production. With mass production, the objective is to produce a perfect product every time. The production process should become a steady state with highly predictable processes, both automated and manual. Changes in the processes have measurable effects on either the processes or the finished product. The key activities toward achieving a highly predictable process are given in an introduction to process characterization (Croarkin *et al.* 2002):

> "* *identify the key inputs and outputs of a process*
>
> * *collect data on their Behavior over the entire operating range*
>
> * *estimate the steady-state Behavior at optimal operating conditions and*
>
> * *build models describing the parameter relationships across the operating range* "

The result of this activity is a set of mathematical process models used to monitor and improve the process.

Production that is less frequently repeated in the sense of, for example, building construction is characterized by extensive schedule planning to reduce unknowns. Then, scheduled activities are executed in pursuit of the final goal (Muller 1982; Yates *et al.* 1982). Essential to this type of planning is that the techniques used are well known and the problems to be solved have precedent solutions.

Many writings on project management reflect this, frequently assumed, production approach to software development (Bauch *et al.* 2001; Boehm *et al.* 2000; Hughes *et al.* 1999; ISO 16326 1999; McConnell 1998; Project Management Institute 2000; Scarola *et al.* 1982; Thomsett 2002). The assumption is that planning will be done once and, with only minor corrections, will accurately predict how the project will be carried out. This requires detailed estimates of how long each task will take and the resources required for its completion. It is argued that, with the software development project divided into component parts, it becomes possible to treat software development as an engineering process, amenable to standard monitoring, management and improvement techniques (Humphrey 1994; 1989; 2000).

2.1 Monitoring production projects.

If software development is viewed as a production process and the project is encapsulated in a planned schedule of activities, there is likely to be an emphasis on monitoring the project against planned progress. The delivered artefacts may be evaluated to determine whether or not the task was completed as planned and whether they are of acceptable quality. Monitoring software development projects in this way is quite common (Hughes *et al.* 1999; McConnell 1998; SEI 2000; Thomsett 1989).

2.2 Controlling production projects.

Corrective actions, arising from observations made during project monitoring, tend to realign the project progress with the planned progress. The most common modification to project tasks is to work longer hours or, less popularly, to add more people to the task in an attempt to bring the project back to the planned schedule. Adding people to a late task doesn't always work and, as pointed out by

Brooks (Brooks 1995), can delay the task still further. Moving work from one task to another, or from one group to another, to balance out the work load is also a common corrective action. However, if none of those can be accomplished then the scope of work can be reduced to something that can be achieved in the scheduled time or else the scheduled time is expanded to accommodate the scope of work (Project Management Institute 2000; SEI 2000).

3 DESIGN PROJECTS.

The way in which a software development project should be treated as an exercise in design is viewed differently by different groups. A systematic approach developed by Chen (2002) in the frame of Engineering Science believes that there are a finite number of steps that will obtain a design solution. However, Chen argues that the representation of the domain knowledge lies at the heart of the design problem. This accords with Gasson (1998) and Curtis (1987) who see the design problem as one of understanding the problem, which would necessarily involve the problem's representation. Smith and Browne (1993), however, believe that the elements of design problems are more than its representation, however important, and include goals, constraints, alternatives, representations and solutions.

The design process is widely seen as a collaborative process that is largely opportunistic rather than orderly (Chen 2002; Curtis *et al.* 1987; Gasson 1998; Henderson *et al.* 1992). Performance is understood in terms of how individuals systematically affect the behaviours of each other (Henderson *et al.* 1992). Gasson further argues that the traditional model, based on the rational model of problem solving and involving problem decomposition, requires that all the requirements are defined before problem decomposition begins. Obviously, this is seldom true when, on average, only 58% of requirements are specified before beginning product design (Thomke *et al.* 1998).

Volatile requirements compound the problem of designing software systems but should not be confused with the design itself. Volatile requirements for a comparatively well understood problem present different challenges than stable requirements for a new and little understood problem. It is entirely possible that both challenges can be addressed by the same mechanism, that of various forms of agile development (Beck 1999; 2000; Fowler 2003; Highsmith *et al.* 2001; Moore 2001). Agile development is usually distinguished from "plan-driven" development (Boehm *et al.* 2004).

3.1 Monitoring design projects.

Gasson (1998) observed that the nature of design is an alternating cycle of opening up the design problem and narrowing down potential solutions. This process terminates "*when the majority of the design team feel that the distributed design model matches their individual design model in sufficient detail*". In other words, the team believe the design has converged. Jagodinski *et al.* (1997) observe that planning and control in the early stages of design "*places more demands on seat-of-the-pants control which requires tacit knowledge and meta knowledge acquired through experience.*"

Such subjective measures do not generally sit well with software development professionals who are only too aware that such measures tend to be optimistic (Brooks 1995; McConnell 1998). Sooner or later the subjective assessment must be tested. Brooks observed that the "*incompleteness and inconsistencies of our ideas only become clear during implementation.*" Life cycles that develop the system in stages achieve such objective demonstrations, and their use seems likely to be favoured when the problem to be solved is not well understood.

Monitoring is then performed on two levels. The first level is the day-to-day interactive monitoring of tasks related to the project plan's work breakdown structure. This monitoring only checks if the planned tasks are being carried out and completed as intended. The second level is through the structure of the project life cycle which, if incremental in some way, affords opportunities to examine whether the work completed so far demonstrates increasing an understanding of the problem.

3.2 Controlling design projects.

Managing, that is, taking action over something revealed during monitoring, is likely to reflect what was monitored. Agile projects afford two types of monitoring; interactive and structural. Interactive monitoring is monitoring of progress compared to a plan, possibly for the particular increment. It is likely to be the familiar task-oriented corrective actions to realign the tasks with the planned activities for the particular increment or release. Structural monitoring occurs at each increment boundary, either at the start or the end. It reviews such things as the state of the requirements and lessons learnt from the recently completed work. It will result in broader corrective actions to the already developed product or the intended product development. Beck (1999) and Highsmith and Cockburn (2001) refer to this as the feedback cycle and advocate that it should be as short as possible.

4 RESEARCH QUESTION.

Software development projects are not so homogeneous and not so simple that they can be treated as if they were all one type or another, but it can be helpful to consider extremes of characterization to gain some insights. On the one extreme, software development processes could be considered as production process where most aspects of the problem to be solved are known and all that is required is to actually do the work. This would match the comparison that McConnell (1993) makes to constructing a building. At the other extreme, software development could be considered as a design problem where very little is known about how to solve any particular problem.

If software development conformed to either of these extremes, how would a software project be monitored and how would it be managed? When the problem to be solved is thought to be well understood, a plan-based approach is more likely to be adopted. Project monitoring will be schedule-based, such that adjustments to the project are likely to realign the project with the plan.

When the problem to be solved is not well understood, a phased or agile life cycle uses the normal project management and monitoring techniques while allowing the problem solution to emerge over the longer review cycle of the increments or phases.

4.1 Research question

The research question is then:

Do organizations and project managers monitor and control their projects as production problems or design problems?

4.2 Research method.

Structured interviews were conducted with project managers from software development organizations in Sydney, Australia, between February and September 2003. Organizations were approached initially by phone and asked if there was a project manager involved in software development and willing to be interviewed. Structured interviews allowed questions and responses to be clarified, or amplified, during the interview and also allowed for unexpected information and findings to emerge rather than directing responses to preconceived models.

4.3 Sample characteristics.

4.3.1 Organizational size

Organizational size was judged largely on the number of personnel. This estimate included the whole organization, not just the software development part because past experience indicates that a small division within a large organization more closely resembles the large organization than a small, independent company of similar size to the division. Table 1 gives the distribution of organization size.

Small (< 30 staff)	11
Medium (31 – 120)	4
Large (>120 - 1000 single organization)	3
Multinational (> 1000 or Multinational)	12

Table 1: Organization size

4.3.2 Process maturity.

The process maturity is a very approximate guide based on the ISO 15504 (SPICE) or CMMI scale of process maturity. Process maturity ratings are given in Table 2. The single instance of a maturity level of 5 came from an organization that had recently undergone a CMMI assessment and was accredited with that level. Organizations were adjudged at level 3 if they were ISO 9001 accredited or had undergone a SPICE or CMMI assessment and had achieved that rating. Level 2 was assigned if the organization had documented software development processes, particularly those dealing with project management and document control.

Informal - Level 1	6
Managed - Level 2	7
Defined - Level 3	16
Measured - Level 4	0
Optimizing - Level 5	1

Table 2: Process maturity

4.4 Monitoring the project.

Subjects were asked how they monitored the project, either to see that it was going right or to detect if it was going wrong. The responses are summarized in Figure 1. The majority used some form of progress measure such as milestones as the first indication that something in the project needed attention.

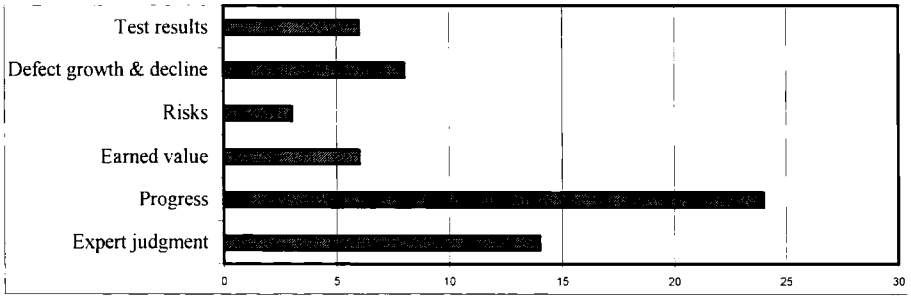

Figure 1: Project monitoring technique

Approximately half of the subjects reported using expert judgment to monitor the state of the project while 80% used some form of progress measure, either milestones or comparison to a schedule. Very few, only 20%, reported using some form of earned value to monitor the project's progress against plan or the need to modify the plan. Other devices such as monitoring the growth and decline of defects or monitoring risks were seldom mentioned as part of project monitoring.

In response to tasks taking longer than planned, the most common response was to add resources (63%) followed by reassigning planned work to another part of the schedule such as a later increment (43%).

4.5 Meeting the schedule.

When asked if functionality would be dropped or rescheduled, 16.7% responded that functionality would always be retained, while 63.3% responded that changes in functionality would be negotiated with the stakeholders. A similar question about trading quality (in terms of the number of known defects in a delivered product) against schedule, 36.7% responded that quality goals were never relaxed to meet the schedule. 30% of the respondents said that changes in the quality goals would be negotiated with the stakeholders while 20% of respondents said that it would be the engineers who would decide whether to compromise the delivered quality to meet a delivery schedule.

For performance goals, 13.3% of respondents said that no such goals were ever set, 30% said that performance goals were never relaxed once set while 33% said that performance goals, like quality and functionality, would be negotiated with stakeholders. These responses are summarized in Figure 2

Only one respondent declared that neither functionality nor quality nor performance goals were relaxed. The schedule simply expanded and, since the project was to develop a first release of a product, there was the capacity to choose to do this.

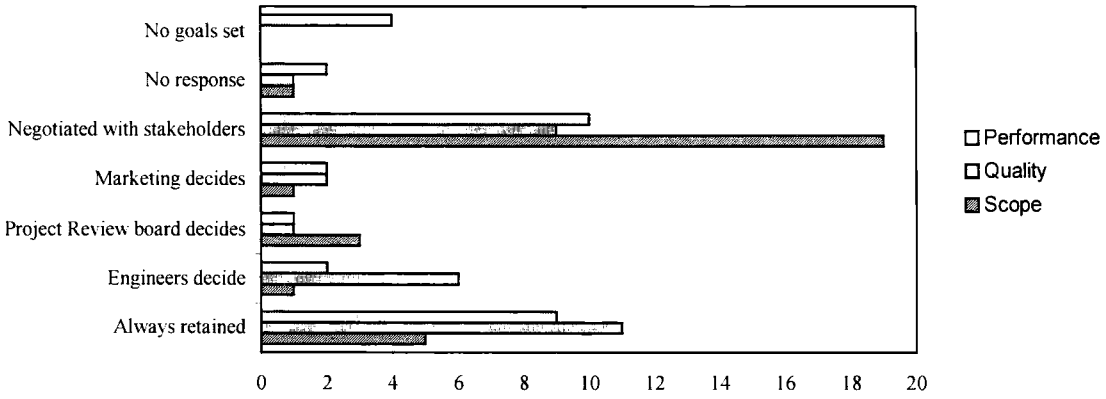

Figure 2: Goals compromised to meet schedule commitments

4.5.1 Team meetings

All but two respondents said there was a regular project team meeting, usually weekly. Such meetings were not necessarily in person because some of the teams were dispersed. The main subjects discussed at the team meetings are summarized in Figure 3.

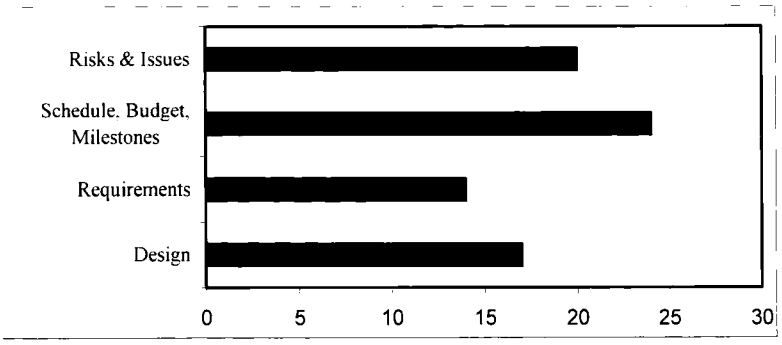

Figure 3: Subjects discussed at team meetings

Many project managers also hold regular formal meetings with their management or with the customer (53%), or write a report for distribution to their management (20%).

4.6 Production or design.

An estimated novelty rating was developed and assigned to each application. Novelty was rated simply as low, medium, high.

- If the application was well known and being implemented using familiar technology, the assigned rating was low. An example would be an accounting application being customized for a specific customer or a new version of a product.
- Familiar applications using familiar technology, but pushing the boundaries slightly was rated as medium, for example, developing an internet based accounting system.
- A new application that involved new ways of doing business delivered using newer technology such as an internet based application in a new business area, was rated high.

Discussing design issues at team meetings was not correlated with novelty. No project manager specifically listed "understanding the problem" or conveyed a similar sense of gaining increased confidence in the correctness of the design or solution being implemented.

5 THREATS TO VALIDITY.

Small sample size. The sample was relatively small at 29 and many statistical tests suffered from having insufficient cell counts, usually less than 10.

Non-random sample. The participating organizations were those listed in the Sydney, Australia, Yellow Pages who agreed to be interviewed when approached by telephone. Soliciting started at the beginning of the list of those organizations listed under "Computer Software and Packages" and proceeded until sufficient data had been gathered to provide a useful, if limited, source. Such *accidental sampling* is considered to have very weak external validity and likely to be biased (Trochim 2001).

Weak external validity. Organizations with low maturity, chaotic project management processes are less likely to be willing to reveal to a researcher just how they manage, or don't manage, projects. Consequently the findings of this research are likely to be biased toward the more mature organizations. However, given the conclusions, the weak external validity is of less importance.

Localized sample. The research sample was from organizations in Sydney, Australia. While there were a significant number of multinational organizations in the sample, it is possible that the same research findings are localized and the study would need to be replicated in another country to test this.

6 CONCLUSIONS AND DISCUSSION.

We have described the characteristics of a production process and how it would be monitored and controlled in Section 2, described the characteristics of a design problem and how it would be monitored and controlled in Section 3, and described the research question along with the method used to investigate it in Section 4.

From the research results we conclude that the majority of interviewed project managers managed their projects as if they were production problems rather than design problems. That is, the assumption was that the problem was amenable to decomposition and solvable with known or readily available techniques. This is despite Gasson's assertions that most software development projects should be treated as if they were design problems rather than production problems if only because the requirements are so volatile (Gasson 1998).

There was very little evidence, statistical or anecdotal, that software development projects in this research sample present novel problems, problems that require periodic checks on how well the project team understood the problem itself and problems that require progress measures of awareness toward an acceptable, implementable solution.

6.1 Importance of the conclusions.

In this research sample, most software development problems were neither new nor unique. They did not require special project monitoring and controlling. Project managers did not evaluate problem novelty before planning the project and did not consciously change their intended methods because of it. It seems sufficient for project managers to monitor adherence to the project plan and to correct any deviations from that plan with no expectation of significant replanning. A simple, if crude, measure of novelty could be used during project planning to decide whether the project should be treated as a design or production problem. Project monitoring and control could then be tailored accordingly.

6.2 Acknowledgements

This is Contribution number 04/15 of the Centre for Object Technology Applications and Research.

References

Bauch, G. T. and C. A. Chung (2001). 'A Statistical Project Control Tool for Engineering Managers.' *Project Management Journal* **32**(2): 37-44.

Beck, K. (1999). 'Embracing change with extreme programming'. *Computer* **32**(10): 70-77.

Beck, K. (2000). 'Extreme Programming Explained', Addison-Wesley.

Boehm, B. and R. Turner (2004). 'Balancing Agility and Discipline: A Guide for the Perplexed', Addison-Wesley Pub Co.

Boehm, B. W. (1988). 'A spiral model of software development and enhancement'. *Computer* **21**(5): 61-72.

Boehm, B. W., C. Abts, et al. (2000). 'Software Cost Estimation with Cocomo II'. Upper Saddle River, New Jersey, Prentice Hall PTR.

Brooks, F. P., Jr (1995). 'The Mythical Man-Month', Addison Wesley Longman.

Chen, D. (2002). 'Developing a theory of design through a multidisciplinary approach'. *IEEE International Conference on Systems, Man and Cybernetics*, Bordeaux, France, IEEE Computer Society.

Croarkin, C. and P. Tobias (2002). 'Engineering Statistics Handbook'. Available: http://www.itl.nist.gov/div898/handbook/index.htm, Accessed 3 November, 2003

Curtis, W., H. Krasner, et al. (1987). 'On building software process models under the lamppost'. *Proceedings of the 9th international conference on Software Engineering*, Monterey, California, IEEE Computer Society Press.

Fowler, M. (2003). 'The New Methodology'. Available: http://www.martinfowler.com/articles/newMethodology.html, Accessed October 2003

Gasson, S. (1998). 'Framing design: a social process view of information system development'. *International Conference on Information Systems*, Helsinki, Association for Information Systems.

Henderson, J. C. and S. Lee (1992). 'Managing I/S Design Teams: A control Theories Perspective'. *Management Science* **38**(6): 757-777.

Highsmith, J. and A. Cockburn (2001). 'Agile software development: the business of innovation'. *Computer* **34**(9): 120-127.

Hughes, B. and M. Cotterell (1999). 'Software Project Management', McGraw-Hill.

Humphrey, W. (1994). 'A Discipline for Software Engineering', Addison-Wesley Pub Co.

Humphrey, W. S. (1989). 'Managing the Software Process', Addison-Wesley Publishing.

Humphrey, W. S. (2000). 'The Team Software Process', Software Engineering Institute: 37.

ISO 16326 ISO/IEC 16326:1999:1999 - Software engineering - Guide for the application of ISO/IEC 12207 to project management

Jagodzinski, P., R. R. Parsons, F., et al. (1997). 'Use of ethnography to acquire an insider's view of engineering design teams'. *Workshop on Soft Approaches to Product Introduction Improvement*, Birmingham, UK, IEEE.

Krishnan, V. (1998). 'Modeling ordered decision making in product development'. *European Journal of Operational Research* **111**(2): 351-368.

McConnell, S. (1993). 'Code Complete'. Redmond, Microsoft Press.

McConnell, S. (1998). 'Software Project Survival Guide', Microsoft Press.

Moore, R. J. (2001). 'Evolving to a "lighter" software process: a case study'. *26th Annual NASA Goddard Software Engineering Workshop*, Maryland Univ. USA, IEEE Computer Society.

Muller, F. (1982). 'Definition of Construction Management'. *Specialty Conference on Engineering and Construction Projects*, New Orleans, American Society of Civil Engineers.

Project Management Institute, Ed. (2000). 'A Guide to the Project Management Body of Knowledge', Project Management Institute.

Scarola, J. A. and C. B. Tatum (1982). 'Definition of Project Management'. *Specialty Conference on Engineering and Construction Projects*, New Orleans, American Society of Civil Engineers.

SEI (2000). 'CMMI for Systems Engineering/Software Engineering, Version 1.02'. Pittsburgh, Carnegie Mellon University/Software Engineering Institute: 606.

Smith, G. F. and G. J. Browne (1993). 'Conceptual foundations of design problem solving'. *IEEE Transactions on Systems, Man and Cybernetics*, **23**(5): 1209-1219.

Thomke, S. and D. Reinertsen (1998). 'Agile product development: managing development flexibility in uncertain environments.' *California Management Review* **41**(1): 8(2).

Thomsett, R. (1989). 'Third Wave Project Management', Yourdon Press Computing Series.

Thomsett, R. (2002). 'Radical Project Management', Prentice Hall PTR.

Trochim, W. M. K. (2001). 'The Research Methods Knowledge Base'. Cincinnati, Atomic Dog Publishing.

Yates, M. K. and C. B. Tatum (1982). 'Definition of Engineering Management'. *Specialty Conference on Engineering and Construction Projects*, New Orleans, American Society of Civil Engineers.