8-25-1995

MANAGING INTERDEPENDENCIES IN VERY LARGE GROUPS

Nancy Staudenmayer
Massachusetts Institute of Technology

MANAGING INTERDEPENDENCIES IN VERY LARGE GROUPS

The Case of Design Changes in Large Scale Software Development

Nancy Staudenmayer
Doctoral Candidate, Management of Technological Innovation
Massachusetts Institute of Technology, Sloan School of Management

PROBLEM STATEMENT:

In developing new products for rapidly evolving markets, a basic problem companies face is how to manage changes in a product design during the development process. Conventional approaches to managing the development of large, complex technological products have largely advocated a linear development sequence. In software production, this is epitomized by the so-called "waterfall model" which portrays software construction as a logical series of cascading activities (Royce, 1970). According to the logic of this paradigm, work should always begin with a complete and detailed design because the cost of correcting errors (making changes) once implementation begins is very high. This approach was first applied in large software system projects for the aerospace and defense industries where it integrated well with their corresponding hardware component development and is generally considered to be an effective way of imposing discipline on such large scale efforts (Blum, 1982).

But Royce (1970) himself noted that the linear development sequence practically "invites failure" when confronted by an unexpected design change. Other researchers have substantiated this observation (Boehm,1981). Most research in software engineering has in turn emphasized ways to prevent or minimize design changes *before* implementation begins. For example, numerous studies have focused on the design decision process, using protocol analysis or observations of design meetings to understand how to best construct a complete and accurate design (Guindon, 1990; Waltz, Elam and Curtis, 1993).

Yet in practice the goal of proceeding without design change is often difficult to achieve for a variety of reasons. One set of factors has to do with the sheer scale and complexity of many development projects, in conjunction with high levels of uncertainty and interdependence. As software systems become increasingly complex, sometimes growing to millions of lines of code and hundreds of tightly coupled modules, it becomes virtually impossible for software engineers to comprehend *existing* systems, much less accurately predict their structure and internal interactions in advance. Such systems must also incorporate changes from the environment in the form of user feedback, adapt to changing scenarios of use and respond to competitive actions in the marketplace. For these and other reasons, it is useful to have a development process that allows engineers to alter specifications and detailed design plans during a project (Cusumano and Selby, forthcoming 1995).

Yet despite this reality, relatively little attention or investigation has been devoted to how software projects should manage the unanticipated changes that inevitably do occur. That is, the process of managing design change has been largely neglected. An important underlying premise of the present research is that design changes are not inevitably disruptive or inefficient (Hayes and Clark, 1985). Changes sometimes need to occur in order to build good products but when they occur and how they are implemented make a difference. One set of technical issues is the degree of flexibility in a product's architecture, which defines how components interact and how much change a product or individual feature can absorb without conflicting with other features. Another set of technical and managerial issues relates to how projects identify, test and communicate design changes among team members. The latter issue becomes particularly problematic in view of the scale of many of these system efforts.

To explore these issues, this project examined how groups of software engineers in two companies make design changes to complex software products. The goals of the research were (1) to acquire a better understanding of how and why design changes occur in practice, what formal and informal rules companies use to coordinate and control them, and how projects enforce (or sometimes circumvent) those rules (2) to evaluate the implications of those management approaches and (3) to consider theoretical interpretations of the data that could inform future research on change management.

RESEARCH METHODOLOGY:

The overall research approach was that of multiple case studies of software design change (Yin, 1984). At two leading software companies, two types of system products were examined (one legacy and one first time development project), reflecting the hypothesis that both the reason for a design change and its implications may vary in terms of the product life cycle stage.

There are three principle components of the research methodology (Leonard-Barton, 1990):

- a retrospective longitudinal study of design changes at the feature/system level
- a collection of focused case studies of real time enactments of different kinds of design change (yoked to the above, as described below)
- a description of the context of change at each study site

The longitudinal study established the what or chronology of change. It identified, for a given feature/system, what changes occurred when. Although the above data was informative, successive snap shots failed to provide information on the mechanisms and processes through which change occurred- the how and why (Pettigrew, 1990). They also failed to include the frame of reference necessary to understand the dynamics confronting people actually involved in the design change effort. March and Simon (1958) suggest three ways to uncover the programs used by individuals and groups in organizations: documents describing standard operating procedures, interviews and observations. All three forms of data collection were used here.

Finally, Pettigrew (1990) has called for a "return to embeddedness" in our theories on innovation, writing that "theoretically sound and practically useful research on change should explore the contexts, contents and process of change together with their interconnections through time." The two components above address the content and process of change, respectively. This third leg of the methodology addressed its context.

THEORETICAL CONCEPTS AND MODELS:

Existing contingency theories of technology management argue that there is no one best way to organize. Rather, effective organizations match their structures, policies and mechanisms to the relevant factors of their task or environment. Most of these theorists have conceived of organizations as information-processing networks (Galbraith, 1973). Focusing on variables such as uncertainty, complexity, ambiguity and interdependence, they seek to explain why and through what structures and mechanisms uncertainty and information relate to organizational design (Perrow, 1967; Thompson, 1967; Lawrence and Lorsch, 1967; Daft and Lengel, 1986).

The present research builds upon and extends the contingency framework in three ways. First, it examines the nature and implications of interdependence at the level of individual tasks. In contrast to most prior theories, however, it focuses on the existence of multiple, dynamic and possibly conflicting contingency factors. Furthermore, rather than trying to predict "fit" this model considers the implications such dynamic interdependencies have for misfit (Alexander, 1964; Gresov, 1989)

Second, the research identifies a taxonomy of alternative coordination strategies for managing such interdependencies during product development. Prior work on the structural or procedural solutions to managing coordination dilemmas is integrated with theories from social psychology which predict how people are likely to feel and behave when working in highly interdependent or "crowded" settings. (Engineers and managers working on large system development projects often refer to the challenges posed by their work as examples of "crowd management" problems.) Three of the core concepts which characterize that experience are social overload, goal interference and behavioral constraint. For example, conditions of high density often create more unwanted and unpredictable intrusions. Social overload reflects the fact that people's inability to regulate the nature and frequency of their social interactions with others taxes their information processing capacity (Milgram, 1970). The presence of many others can also block goal attainment either directly, by physically disrupting behavior, or indirectly by making purposeful action more frustrating and therefore less effective (Baum and Valins, 1979). High density can also be accompanied by a number of potential constraints, inconveniences and threats which limit the number of behavioral options and adaptations possible. All of these mediating processes have implications for how people solve problems and work together over time. The framework proposes that by combining knowledge of interdependence and organiational design strategies with an understanding of their likely impact on individual and group level processes, we will be better situated to identify appropriate managerial responses.

Finally, the theory considers the performance implications of these results. It suggests that there exist patterns or sets of interdependency which characterize and differentiate different kinds of product development projects. The appropriate organizational response strategy differs depending on the circumstances of interdependence, and projects which fail to adopt the necessary mechanisms and structures are likely to suffer in terms of performance.

THEORETICAL AND MANAGERIAL IMPLICATIONS:

In 1975, F.P. Brooks Jr. described software construction as inherently a system effort, "an exercise in complex (human) inter-relationships" (Brooks, 1975). Yet now, twenty years later, many studies of software development still focus on the mechanical or technical aspects of programming more than the social or organizational. As mentioned earlier, most research on software design change has primarily emphasized those variables which effectively minimize change. The present study offers a different look at the same phenomenon which builds upon and complements these prior research streams. Understanding why design changes occur and the social/organizational factors and reactions that complicate their management are important. Considering elements of technology, organization and process is likely to yield a more complete picture of this complex and important technology and thus enrich our understanding of the theory and practice of technology management.

This work is likely to have immediate practical value for practitioners. Change management is done in some form by all manufacturing companies, and many companies spend millions of dollars making changes to their products whether they expected to or not. Change management and propagation therefore represent high leverage points if we can identify ways to makes such activities more efficient. These are also activities that many (software) managers feel the need to understand and do better. The ability to have a flexible development process that incorporates design changes can enable a firm to better meet customer requirements and changing customer needs, as well as respond to mid-stream competitive activities. Unplanned and poorly managed changes, on the other hand, can result in late products delivered with defects and inconsistent features.

REFERENCES

Alexander, C. *Notes on the Synthesis of Form.* Cambridge, MA: Harvard University Press, 1964.

Baum, A. and S. Valins. "Architectural Mediation of Residential Density and Control: Crowding and the Regulation of Social Contact," in *Advances in Experimental Social Psychology*, Vol. 12, L. Berkowitz (ed.). NY: Academic Press, 1979, pp. 131-175.

Blum, B.I. "The Life Cycle- A Debate Over Alternative Models," *ACM SIGSOFT Software Engineering Notes*, Vol. 7, No. 4, October 1982, pp. 18-20.

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

Brooks, F.P., Jr. *The Mythical Man Month*. Menlo Park, CA: Addison -Wesley Publishing Company, 1975.

Cusumano, M.A. and R. Selby. *Microsoft Secrets: How the World's Most Powerful Company Creates Technology, Reshapes Markets and Manages People.* Free Press/Simon and Schuster, Inc., forthcoming 1995.

Daft, R.L. and R.H. Lengel. "Organizational Information Requirements, Media Richness and Structural Design," *Management Science.* Vol. 32, May 1986, pp. 554-571.

Galbraith, J. *Designing Complex Organizations.* Reading, MA: Addison-Wesley Publishing, 1973.

Gresov, C. "Exploring Fit and Misfit with Multiple Contingencies," *Administrative Science Quarterly*. Vol. 34, 1989, pp. 431 - 453.

Guindon, R. "Knowledge Exploited by Experts During Software System Design," *International Journal Man-Machine Studies* , Vol. 33, 1990, pp. 279-305.

Hayes, R.H. and K.B. Clark. "Exploring the Sources of Productivity Differences at the Factory Level," *in The Uneasy Alliance* , K.B. Clark, R.H. Hayes and C. Lorenz (eds.) Boston, MA: Harvard Business School Press, 1985.

Lawrence, P.R. and J.W. Lorsch. *Organizations and Environments: Managing Differentiation and Integration.* Homewood, IL: Irwin Publishing Company, 1967,

Leonard Barton, D. "A Dual Methodology for Case Studies: Synergistic Use of a Longitudinal Single Site with Replicated Multiple Sites," *Organization Science,* Vol. 1, No. 3, 1990, pp. 248 - 266.

March, J.G. and H.A. Simon. *Organizations.* NY: John Wiley and Sons, 1958.

Milgram, S. "The Experience of Living in Cities," *Science.* Vol. 167, 1970, pp. 1461-1468.

Perrow, C. "A Framework for the Comparative Analysis of Organizations," *American Sociological Review,* Vol. 32, No. 2, 1967, pp. 194-208.

Pettigrew, A.M. "Longitudinal Field Research on Change: Theory and Practice," *Organization Science.* Vol. 1, No. 3, 1990, pp. 267 - 292.

Royce, W.W. "Managing the Development of Large Software Systems," Proceedings of IEEE WESCON, August 1970, pp. 1-9.

Thompson, J.D. *Organizations in Action.* NY: McGraw-Hill, 1967.

Waltz, D.B., J.J. Elam and B. Curtis. "Inside a Software Design Team: Knowledge Acquisition, Sharing and Integration," *Communications of the ACM*, Vol. 36, No. 10, 1993, pp. 63-77.

Yin, R.K. *Case Study Research.* Applied Social Research Methods Series, Vol. 5. Beverly Hills, CA: Sage Publications, 1984.