2006

Using emergent outcome controls to manage dynamic software development

Michael Loyd Harris
University of South Florida

Using Emergent Outcome Controls to Manage Dynamic Software Development

by

Michael Loyd Harris

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Information Systems and Decision Sciences
College of Business Administration
University of South Florida

Co-Major Professor: Rosann Webb Collins, Ph.D.
Co-Major Professor: Alan R. Hevner, Ph.D.
Stanley J. Birkin, Ph.D.
Richard P. Will, Ph.D.

Date of Approval:
June 2, 2006

Keywords: Control Theory, Dynamic Capabilities, Software Development Methods, Flexibility

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

USING EMERGENT OUTCOME CONTROLS TO MANAGE DYNAMIC SOFTWARE DEVELOPMENT

Michael Harris

ABSTRACT

Control and flexibility may appear an unlikely pair. However, I propose that effective management of dynamic environments, such as systems development under conditions of uncertainty, must still provide clear control mechanisms to manage the progress and quality of the resulting products. This dissertation presents research to understand the types of control used in the context of flexible software development processes. The dynamic capabilities extension to the resource-based view of the firm is used to understand dynamic environments. Within those environments, control theory is used to understand how activities are guided and controlled to achieve management objectives. Specifically, control theory acts as a lens to contrast the control mechanisms found in plan-driven and flexible processes. I extend current thinking to include emergent outcome controls for team coordination in a taxonomy of control mechanisms. These phenomena are studied through a qualitative field study. The results show that organizations will choose more flexible management approaches as uncertainty increases, and that more controlled-flexible approaches managed with emergent outcome controls will lead to better outcomes than uncontrolled, ad hoc approaches.

CHAPTER 1: INTRODUCTION

This study examines flexible software development as an expression of the dynamic capabilities extension to the resource-based theory of the firm, and it uses control theory as a lens to examine the central tension between control and flexibility in managing software development. The control versus flexibility tension is evident in the software industry's long running debate over the relative merits of plan-driven development approaches versus flexible[1] approaches (Table 1). The goal of the paper can be summarized in the following research questions:

Why do organizations choose flexible methods instead of plan-driven approaches?

How do the control mechanisms used in flexible methods influence the dynamic capabilities of a software development organization?

---

[1] The term flexible was chosen over agile to be more inclusive. In practice, organizations may use flexible approaches that manage towards emergent outcomes, but they may not use a formally defined agile method.

Table 1: Plan-driven versus controlled-flexible development

	Description	Comments
Uncontrolled or Ad hoc	No constraints on developer.	May wander – lose sight of goals & timelines. Likelihood of bugs.
Plan-driven Development	Software development guided by detailed plan. All changes must be documented first in the plan and the developers follow the plan.	May impede creativity. Creates a barrier between users and developers. Slows down changes.
Controlled-Flexible Development	Developer has freedom to discover better solution during development, but is bound by process rules.	May lose focus on goal. Stopping rule for ending project may not be evident. May be difficult to forecast completion date.

From the earliest days, the industry recognized that uncontrolled development can result in software delivery problems (McConnell 1996). The search for a structured process led to the creation of the waterfall development method. The waterfall approach is the archetype of a class of methods referred to as plan-driven approaches. With the waterfall method, a plan is created at the beginning of the project, and the balance of the project is focused on execution of the plan. However, experience has shown that the upfront plan is often out-dated before development is completed (McConnell 1996). In response, researchers and practitioners have developed more flexible plan-driven methods that allow the software to evolve through multiple iterations. Despite the increased flexibility of these approaches, many methodology experts (Beck et al. 2001) feel that the best solutions come from interactions between developers and users, and that the formal plan impedes these interactions. As a result, a new class of agile methods has been created. These agile methods are not ad hoc. They contain processes to keep

development on track. However, they use mechanisms other than the formal plans used by the plan-driven approaches.

The need for agility is not limited to a few companies; in a survey of CIOs, 85% indicated that agility was part of their core business strategy (Ware 2004). As a case in point consider the FBI Virtual Case Project that was cancelled after $170 million in expenditures. According to SAIC, the contractor, the problem was an evolving design (Hayes 2005):

> *…And what the FBI called software deficiencies were really more changes in requirements. And users kept rejecting SAIC's software designs, taking what one SAIC executive complained was a "trial-and-error, we-will-know-it-when-we-see-it approach to development.*

The article's author offered his opinion regarding the issues:

> *And in the frantic days after Sept. 11, SAIC should have spotted that stable requirements for this project just weren't in the cards. The FBI needed results in the face of a crisis. SAIC should have shifted gears and methodologies to start producing working deliverables right away, no matter how far the project was from a complete set of requirements.*

One interesting part of this article was that the SAIC executive dismissed the FBI's design process as a "trial-and-error" approach. Agile methods clearly contain structures that differentiate them from "trial-and-error". However, these differences have been less explicitly discussed than the demarcation between agile and plan-driven methods. Consider the agile manifesto in the figure below. It clearly explains that agile

methods are not plan-driven approaches, but it does not explicitly describe the differences between an agile approach and a trial-and-error approach.

Manifesto for Agile Software Development (Beck et al. 2001)

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

I. **Individuals and interactions** over processes and tools
II. **Working software** over comprehensive documentation
III. **Customer collaboration** over contract negotiation
IV. **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	John Kern	Dave Thomas
Martin Fowler	Brian Marick	

Figure 1: The agile manifesto

The confusion is exacerbated because organizations that use these methods generally create adaptations to account for their idiosyncratic needs. Without theoretical guidance, it is difficult for adopting organizations to understand whether an appropriation of a method is faithful (DeSanctis et al. 1994). Furthermore, there is no consensus regarding the 'best' agile approach. Much of the research into software process improvement has consisted of studies that concentrate on a single method and that make their points through assertions and proof of concept studies (Zelkowitz et al. 1998), and agile research is not an exception to this trend. There is no established framework for comparing, analyzing, and evaluating flexible methods.

THEORETICAL BASIS

The theoretical basis for the study comes from several literature streams. The dynamic capabilities extension to resource based theory of the firm (Barney 1996; Eisenhardt et al. 2000; Teece et al. 1997) is used to explain the environmental context for flexible development and to explain why a purely plan-driven approach is not sufficient. However, the dynamic capabilities approach does not explain how these flexible controls operate. This study uses control theory (Kirsch 1996; Ouchi 1980) as a lens to examine existing systems development mechanisms, and to search for a possible explanation of the role of the mechanisms that are in use. The set of available mechanisms was extracted from the IS literature on systems development, especially the work in agile methods (Boehm et al. 2004). These three literature streams are discussed briefly in the paragraphs below, and in more detail in Chapter two.

The dynamic capabilities literature builds on the resource-based theory of the firm. Since the resource-based theory of the firm hypothesizes that a firm's competitive advantage is based on its control of certain resources, the dynamic capabilities extension to that theory starts with the assumption that a manager's job is to build and maintain those differentiating resources. However, in a fast moving environment, research has shown that it is not possible to determine a priori what resources are required. The dynamic capabilities approach explains that managers in fast moving environments will choose to build in the flexibility to quickly take advantage of changing conditions. This explanation is consistent with IS's demand for flexible processes.

Although the dynamic capabilities literature explains why flexible processes are needed, it does not explain either the processes that should be used, or how those processes operate to achieve their goals. If flexibility is taken to its extreme, the result is simply trial and error development whose problems are well known (McConnell 1996).

In order to understand the mechanisms used to deliver in a dynamic capabilities (DC) environment, we turn to control theory (Ouchi 1977). Control theory provides insights into the mechanisms that organizations use to achieve their goals. These mechanisms are classified into three categories of controls (Table 2).

Table 2: Three types of control

Control Type	Theoretical Intent
Outcome	Workers are evaluated based on their ability to deliver against some pre-specified outcome.
Behavioral	Workers are evaluated based on a comparison of their performance to a pre-specified behavior that is known to transform inputs to desired outcomes.
Clan	Rituals and ceremonies are used to promote common values among clan members. Clan members are then expected to self-regulate based on those common values.

The control types (outcome, behavior, and clan) represent the types of controls that can be used to manage towards a desired objective. This dissertation uses these control types as a filter for viewing the mechanisms used in systems development.

While the lens, or filter, for the analysis comes from control theory, the set of available control mechanisms is found in the IS literature on systems development, especially the work in agile methods (Boehm et al. 2004). As mentioned previously, the literature on software development suggests many alternatives for managing flexible

software development. Some examples include eXtreme Programming (Beck et al. 2005), Scrum (Schwaber et al. 2002), and the synchronize and stabilize approach (Cusumano et al. 1999).

The use of control theory to organize flexible development mechanisms does suggest that flexible development mechanisms can be thought of as expressions of outcome, behavior, and clan control. However, there are significant differences in the expression of these control types in flexible development vis-à-vis traditional control theory. As a result, the analysis also suggests adjustments to control theory to expand the concept to cope with dynamic environments. The following paragraphs briefly compare the expected control types and the observations of mechanisms used in flexible development.

Outcome Controls

According to control theory, outcomes are deterministic: produce to this specification and you will produce an acceptable product. A strict interpretation of this control type suggests that you must know the specification in order to use outcomes as a control. This could lead to the conclusion that outcome controls are inappropriate for flexible methods since specifications are not complete until the development is already complete. Surprisingly, however, control of outcomes is a primary concern of the flexible approaches. The difference is that the flexible methods are concerned with emergent outcomes. Emergent outcome control is similar to typical outcome control in that it focuses on managing and controlling the outcome or work product of the process.

However, typical outcome control looks to deliver against a fixed goal that is known a priori, at the beginning of the process. Emergent outcome control recognizes that the final goal is initially unknown, and that the goal will dynamically change as it emerges from the ongoing process. As the work product emerges it is actively managed by carefully defining the space containing the allowable outcomes, and by providing ongoing feedback on the progress.

Clan Controls

Ouchi (1979, 1980) explains clan control as a process of selection, training, and acculturation to attain common values. According to the theory, achievement of common values addresses any problems of goal incongruence. The importance is in development of common values, not in monitoring the actual work product. The flexible methods also use training, selection and peer pressure (acculturation) to control work. However, flexible methods use peer control to directly impact work products, not just to align attitudes. Although clan control has not been carefully studied, other researchers have suggested that clan control may include control of tasks (Henderson et al. 1992; Kirsch 1996).

Behavior Controls

Traditional control theory focuses on behaviors that transform inputs to desired outputs. Compared to the previous control types, the flexible development approach is congruent with the traditional control theory definition. However, in flexible development, even behavioral control comes in its own unique flavor. This consists of an

attitude that can best be summarized in the popular saying: “Don’t put off until tomorrow, what you can do today.” This mindset is engendered by the short time-frames of the flexible processes. At any given time, the developers must be ready to stage a user demonstration (Aebischer et al. 2003), they must prepare for daily software builds, and potentially for weekly product releases. The result is a very short term focus. Bugs are fixed immediately instead of being added to bug lists. Development is planned in bite-sized chunks that can easily be added to the daily software build. As the agile manifesto states, the focus is on ‘working software’, and the goal is to be able to demonstrate progress as development progresses.

RESEARCH APPROACH

As described in the paragraphs above, the theoretical findings were initially analyzed with respect to the existing literature on software processes. This process revealed a partial fit between theory and practice, but it also emphasized the need for theory to expand to include the management of emergent outcomes. Controlling to emergent outcomes allows managers to use outcome-orientated controls in situations where the desired outcome is not known a priori. These preliminary findings were validated and further explored through a qualitative-orientated field study. As a pre-test, the findings were discussed in interviews with experienced developers. Next these findings were investigated using a qualitative field study in situ with companies involved in building development products. Although the study gathered data on many control types, the focus was on emergent outcome controls. Emergent outcome controls represent

the largest departure from traditional control theory and, thus, offer the most significant opportunity for a contribution to the existing body of research. Detailed analysis of other control types will be left to follow up initiatives.

The unit of analysis for the study is the software project through the eyes of individual informants participating in a project. In most cases, there are multiple informants for each project. Organizations are chosen to participate in the study based on their use of flexible development methods. There is no attempt to identify organizations using some specific methodology – the goal is to study methods-in-use, not textbook approaches. A flexible approach is defined as one that encourages, and even expects, developers to discover and explore new capabilities. In contrast, a plan-driven approach may allow new changes, but the bias is towards implementing an existing plan, and any changes must be approved through a formal change-control process.

RESEARCH QUESTION

As stated initially, the general research questions for this study are as follows:

Why do organizations choose flexible methods instead of plan-driven approaches?

How do the control mechanisms used in flexible methodologies influence the dynamic capabilities of a software development organization?

Based on the discussion thus far, we offer the following research propositions:

Proposition 1: When uncertainty increases, flexible development methods will become more advantageous.

Proposition 2: Flexible development teams that manage emergent outcomes will be more successful.

These propositions will be developed into specific hypotheses in Chapters two and three of this dissertation.

FINDINGS AND CONTRIBUTIONS

This study validates the theoretically justified link between market and technology uncertainty and the need for a flexible development approach. Furthermore, it expands upon control theory to include emergent outcome controls and uses this as the basis for differentiating between plan-driven approaches and controlled-flexible and between controlled-flexible and ad hoc.

The description of emergent outcome controls came from an examination of the existing literature on development methods. Even though the existing literature is not theoretically oriented, it contains embedded wisdom in the form of mechanisms that have been tested for decades in the real world. The knowledge encapsulated in these methods offers lessons that can be used to expand the scope of control theory. The three types of control discussed earlier can be deployed via formal or informal control mechanisms. Formal control mechanisms include explicit rules for deterministic situations and informal control mechanisms include approaches such as socialization for ambiguous situations. Since flexible development involves considerable ambiguity, this would suggest the use of informal control mechanisms. However, the flexible development literature does not rely only on informal mechanisms; the literature includes many

explicit mechanisms that can be used to manage ambiguous situations. As a result, this process-oriented literature was used to offer insights that allow control theory to be adapted to better handle dynamic situations.

The aforementioned literature analysis and theoretical analysis offer a common language for analyzing and comparing flexible processes. The current dialog centers on individual mechanisms recommended by specific methodologies. The study suggests that control theory can be used to understand the goal of individual mechanisms, and suggests that emergent outcome controls will consist of either scope boundaries or ongoing feedback. Scope boundaries describe the shape of the area where flexibility is allowed – they contain the flexibility and insure it stays on course. Ongoing feedback provides communications both within the project and with stakeholders focused on the actual work products of the process as they emerge from the development efforts.

Because of this melding of separate literature streams, this study offers contributions to researchers in the areas of the dynamic capabilities extension to resource-based theory of the firm, control theory, and software process control. In addition it offers aid to practitioners who seek to adopt flexible methods. Adopted routines are generally adapted to the idiosyncratic needs of the adopting organizations (Eisenhardt et al. 2000). However, the adopting organization risks an unfaithful appropriation if the spirit of the mechanisms is not well understood (DeSanctis et al. 1994). This study can help adopting organizations achieve a better understanding of the role and purposes of the control mechanisms in a flexible development method.

SUMMARY

In dynamic environments, competitive and market forces result in continuously changing conditions. Given the constant changes, it is not feasible to develop a priori plans for development, and successful dynamic organizations build capabilities to react to changes. Control theory suggests that the best approach in a dynamic environment is an informal, or clan based approach. However, software researchers have suggested that purely informal development may not lead to working software. Many techniques have been suggested to actively manage in fast changing software environments. The existence of these techniques suggests that formal methods can be developed to manage in dynamic environments. These techniques encompass two types of activities: 1) evolutionary outcome controls that manage and constrain the development of emergent outcomes; and 2) formal group control techniques that help a group work together. This study offers practitioners advice on the range of mechanisms to pursue in implementing dynamic capabilities; it offers researchers a set of building blocks for analyzing and designing dynamic processes. Finally, this paper is consistent with the call for IS to become a reference discipline to other disciplines. It takes the embedded learning from years of research on flexible development methodologies and uses that learning to offer suggestions to reshape control theory to encompass control of dynamic capabilities environments.

The balance of this dissertation will proceed as follows. Chapter two will discuss the relevant literature and theories. Chapter three will describe the analysis of selected

methodologies from the viewpoint of control theory. Chapter four, will discuss the research design of the field study. Chapter five will discuss the analysis of results. Finally, Chapter six will present the contributions, limitations, and summary of the study.

CHAPTER 2: LITERATURE REVIEW AND THEORY

This study investigates the control of flexibility in information systems development. Establishing a theory of control for flexible methodologies has benefits for both researchers and practitioners. It offers researchers a basis for comparing, analyzing, and evaluating flexible methodologies. This will lead to easier synthesis of research across various methodologies. It will also help practitioners faithfully adapt (DeSanctis et al. 1994) agile methodologies by offering an explanation of the role of various control mechanisms.

At the heart of this discussion is a debate over the relative merits of flexibility versus structure. The need for flexibility is based on a belief in innovation-based competition as expressed in Schumpeterian economics (Schumpeter 1934), and based on the modern conceptualization that a dynamic capabilities approach (Eisenhardt et al. 2000; Teece et al. 1997) provides a means for competing in a fast moving environment. The importance of structure has been studied throughout the history of organizational and sociological literature, and it finds its modern expression in the discussion of control theory established by Ouchi (1977; 1978; 1979; 1980). In the following paragraphs we explore first the dynamic capabilities view of the world, followed by a discussion of the role of controls in reaching organizational objectives. Finally, we will discuss the

information systems literature on flexible methodologies, and the impact of the aforementioned theories on the IS research.

STRATEGY RESEARCH AND THE NEED FOR FLEXIBILITY

In the 1970's the popular paradigm viewed product development as a sequential process, similar to a relay race with specific phases that must be performed in order (Takeuchi et al. 1986). This view was challenged by the "ready-fire-aim" approach proposed by Peters and Waterman (1982). They suggested a more experiential approach to innovation whereby an organization would move quickly to try out new ideas and then extract learning from their experience in the market. Tekeuchi and Nonaka (1986) suggested that a more effective metaphor than a sequential relay is that of a rugby scrum[2] "where a team tries to go the distance as a unit, passing the ball back and forth". Over the following years, researchers began to view the product development world as a complex environment that required a faster moving, interactive development approach (Bhattacharya et al. 1998; Bourgeois et al. 1988; Brown et al. 1997; Dahan et al. 2001; Eisenhardt 1989a; Eisenhardt et al. 1995; Iansiti 1995; Karagozoglu et al. 1993; MacCormack et al. 2003b). The ideal approach involved overlapping phases instead of sequential development. It also recommended that decisions be based on user feedback obtained from multiple probes of the market (Brown et al. 1997). However, this research

[2] The Scrum software development method traces its inspiration back to the Tekeuchi and Nonaka (1986) article.

was often empirically based, and did not link to existing theory. Teece, Pisano and Shuen (1997) explained that the need for dynamic capabilities could be thought of as an extension to resource-based theory of the firm.

Resource-based theory (Barney 1996) hypothesizes that control of valuable, rare, inimitable, and non-substitutable (VRIN) resources are the basis for a firm earning rents from a market. In other words, control of special resources provides a source of competitive advantage as long as those resources are valuable and unique (VRIN). Given this resource-based approach, the job of a manager should be to acquire, grow, and maintain VRIN resources (Teece et al. 1997).

However, some organizations exist in rapidly changing environments. In these circumstances the set of important resources may be constantly changing. Furthermore, it may not be possible to identify and acquire resources a priori. Instead, firms develop dynamic capabilities that let them recognize and exploit new opportunities as they arise (Teece et al. 1997). According to dynamic capabilities researchers, high-technology product development is a prototypical example of the type of function that is best addressed with a dynamic capability approach (Eisenhardt et al. 2000). The dynamic approach would be to build a product development process that recognizes and reacts to new information that emerges during the development process.

Of course, the dynamic capabilities view of the world is not the only view that exists. During the 1980s, the field of strategy research was influenced by Porter (Porter 1980) who emphasized a logic of positioning (Sambamurthy et al. 2003). This view of

the world explained that rents would flow from privileged product positions (Teece et al. 1997). If corporations operate according to this theory then their goal will be to develop a superior product position in relation to their competition and they will attempt to maximize the advantage of their superior position.

Both Porter's strategic positioning approach and the dynamic capabilities approach allow us to make predictions concerning competitive behavior. If we examine the software marketplace through the lens of these competing theories, we can determine if they are applicable in the world of software products. Consider the case of new product introductions.

- Porter's product positioning approach suggests that a company's primary goal is to seize and maintain the upper hand versus the competition. Porter notes that an important element is information secrecy for new product announcements (Porter 1980). If this were true, then a company should hide its product details until the product is ready for the market. Pre-announcing a product gives the competition time to react and shortens the time before the competition is able to introduce a competitive response.
- A company that is focused on a dynamic capabilities (DC) approach will primarily focus on the changing and uncertain marketplace. They will be interested in gathering information on the emerging market conditions and capitalizing on that information. A DC company will launch multiple market probes to understand the market's evolving needs from various viewpoints (Brown et al. 1997). The way to beat the

competition is not to surprise them, but it is to build a better map of the marketplace and to deliver a better solution based on that market map.

The bottom line is that a positioning approach would suggest surprise product announcements to reduce signaling to the competition while a dynamic capabilities approach would suggest continuous exposure and airing of product plans and directions in the form of various prototypes, test balloons, and briefings in order to gain market feedback. The evidence suggests that a DC approach is alive and well in the software industry. Product pre-announcements are common. For example, Oracle began user trials of Oracle 10g at least 10 months before the product was released (Kao et al. 2003; Veitch 2003). Similarly, during the development of Navigator 3.0, Netscape released six beta versions of its browser. This allowed the development team to react to user feedback and marketplace changes (Iansiti et al. 1999). Clearly, these companies were not focused on hiding product details[3].

However, the DC approach is not limited to consideration of product introductions. It is focused on all conditions that create uncertainty in planning and execution. Specifically, a DC approach would also prove valuable when significant technical uncertainty exists (MacCormack et al. 2003b). Consider Microsoft's Longhorn release of its operating system. Microsoft made product announcements to developers

---

[3] I am sure that exceptions can be found to these examples. This argument is that at least some of the companies view the market through the lens of a dynamic capabilities approach – not that all companies operate this way.

and trade press starting years before the product release (Ricciuti 2004). Even after features had been announced, Microsoft later pulled the features due to development issues.

CONSTRAINING FLEXIBLE PROCESSES

One way to build in flexibility is to use an ad hoc process that has no constraints. Earlier organizational researchers (Ouchi 1980; Robey 1996) suggested that an 'organic' approach would be best in an uncertain environment. An organic approach lets the workers have freedom to produce and innovate without constraints or controls other than their own self-control. Yet other researchers have noted problems with organic approaches. One writer suggests that even in R&D, some structure is needed (Sheasley 1999). Researchers have suggested that researchers who do not follow product guidelines can engage in technological "wandering" resulting in a mismatch between product needs and the market (McDonough III et al. 1986). In one study of high technology companies the researchers note that (Eisenhardt et al. 1995):

"… calling this organic does not capture the sense of structure that we found."

An analysis of the characteristics of an organic approach (Robey 1996) leads to the conclusion that there is no real difference between an organic approach and the discredited (McConnell 1996) ad hoc software development approach. Furthermore, research has shown that organizations that invest upfront to prepare for change do better than those who wait until change occurs and try to react to it (MacCormack et al. 2003b).

In fact, Dynamic Capabilities researchers do not suggest a purely organic approach to handling dynamic situations. They predict that path-dependent routines will be developed by firms. Since these routines are path-dependent they will be idiosyncratic to the firm. However, the researchers also expect that below the surface there will be commonalities between firms in the underlying structures of these path-dependent routines (Eisenhardt et al. 2000).

CONTROL THEORY

The underlying structures in these dynamic routines can be thought of as control mechanisms. These mechanisms are formal or informal structures that guide the dynamic activities of an organization. For much of the twentieth century, a study of management control implied a cybernetic view of control. This was a mechanistic approach that involved setting a standard or a goal, offering feedback that compares accomplishment to the standard, and making adjustments based on the feedback (Diefendorrff et al. 2003; Hofstede 1978). However, the cybernetic view works best for short-term, programmable, and easy to access positions (Jaworski 1988). It does not work well if standards do not exist, accomplishment is not easily measured, feedback can not be used, or the proper corrective action is unclear (Hofstede 1978; Jaworski 1988; McDonough III et al. 1986).

Ouchi established a more behavioral view of control that is the basis for the current research in control theory (Ouchi 1977; Ouchi 1979; Ouchi 1980; Ouchi et al. 1978). Ouchi described three types of controls that organizations use to manage towards their objectives. These control types can be briefly defined as:

- Behavioral control: Appropriate when the behaviors that transform inputs to outputs are known.
- Outcome control: Appropriate when an individual's output can be measured.
- Clan control: Appropriate in ambiguous circumstances where neither the behaviors nor outputs can be predicted a priori. Clan members belong to a common organization and share values, beliefs, and attitudes (Ouchi 1979).

The relationship among the types of control can be shown in the following figure.

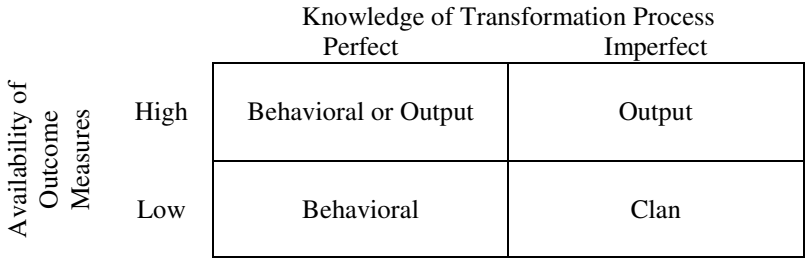


Figure 2: Organizational use of control types. Adapted from Ouchi (1977; 1979)

The diagram suggests that two factors will determine the proper control approach: the availability of outcome measures and the knowledge of the transformation process. Outcome measures are useful if an outcome can be specified a priori, and if the individual's contribution can be tied to the outcome. If a task outcome can not be described, or if an individual's contribution to the outcome can not be easily determined, then using outcome-based control will not be efficient. Alternatively, behavioral control can be exercised by prescribing the transformation behaviors that produce the end product and measuring adherence to those behaviors. This behavioral control approach not only requires well known transformations that will result in success, but it also requires that behaviors be observable. Furthermore, the controller must be knowledgeable

enough to understand the appropriate behaviors in order to observe them (Kirsch 1997). As jobs become more complex and more ambiguous the 'best' approach shifts from output or behavioral control to clan control.

In recognition of the differences between outcome/behavioral control and clan control, some researchers have called them formal control and informal control, respectively (Andres et al. 2001; Birnberg et al. 1988; Jaworski 1988; Kirsch 1997). This is generally thought of as a dichotomy, but Merchant (1988) points out that formal and informal may represent opposite ends of a continuum rather than a dichotomy. Formal controls are enacted through explicit rules enacted by the employee's manager (Andres et al. 2001; Jaworski 1988). On the other hand informal controls are worker constructed (Jaworski 1988) and explicit rules may not be present in an informal control (Birnberg et al. 1988).

The problem with formal controls is that each desired circumstance must be encapsulated in a rule or control mechanism and that any rules left 'uncontrolled' may not be supported by workers (Ouchi 1980). For example, in a study of retail sales personnel, Ouchi (1977) found that an over reliance on sales commissions (outcome control) resulted a lack of support for non-sales activities such as restocking or training new peers. On the other hand, clan control (informal control) relies on developing shared values, beliefs, and attitudes among employees. From this shared basis the employees can derive a limitless set of rules to cover unexpected situations (Orlikowski 1991).

However, a clan approach has many shortcomings of its own. First, development of a shared view of the world takes time. It starts with proper selection of employees, but it also requires training and socialization that takes time to complete. As a result, clan control will not work if turnover is high and it takes some time for new members of the clan to become fully indoctrinated with the clan's beliefs (Orlikowski 1991; Ouchi 1979). Furthermore, clan controls give significant discretion to the employee and all controllers may not be comfortable relying solely on clan controls (Kirsch 1997).

One problem with Ouchi's vision of control theory is that it does not fully describe the interactions among team members. A clan, as he has described it, is broader than a team. A clan may be "a profession, a labor union, or a corporation" (Ouchi 1980). Ouchi's clan works at the broad-level on attitudes, beliefs, and values, not on daily activities. It is assumed that the individual will make appropriate decisions based on the existence of these common, broad-level attitudes, beliefs, and values. However, as task uncertainty and task interdependence increases there is a need for more detailed coordination among team members (Van de Ven et al. 1976).

As an illustration, consider the following quote (Cusumano et al. 1999):

> *Developers who checked in code that broke the build had to fix their code immediately and face penalties, such as become the build master for the next day… wear a dunce cap, or pay a small fine.*

This instance involves clan-style feedback: peer level pressure and hazing. However, the feedback directly relates to task performance whereas Ouchi's clan control relates more directly to attitudes, values and beliefs. This points to two different modes

for clan control. In Ouchi's original definition, clan control involved establishment of common values among a community of clan members. A clan member who poses the proper values is expected to be able to make the proper task level choices without supervision or input from others. In the new definition, clan control allows direct team-member to team-member feedback regarding task-specific issues.

Some researchers have operationalized clan control as team member coordination (Henderson et al. 1992; Kirsch 1996; Kirsch 1997). One issue is with this approach is that it loses Ouchi's sense of clan control acting through values.

Researchers have also suggested a new type of informal control, called self-control (Choudhury et al. 2003; Kirsch 1996). In self-control individuals select their own goals and self-monitor their activities. However, consider Ouchi's (1977) definition of control as "… the mechanisms through which an organization can be managed so that it moves towards its objectives." If self-control is guided solely by the individual, then it will not necessarily advance organization objectives. Organizations that depend on self-control must also rely on selection, training, and socialization to ensure that individual goals are congruent with organizational desires. However, these are the same factors that are used to create a clan (Ouchi 1979; Ouchi 1980). From this viewpoint, it appears that self-control is another name for Ouchi's vision of clan control.

In fact, the researchers who talk about self control appear to redefine clan control to involve task-orientated feedback by team-members. It appears that the proposal from these researchers is that there are two types of informal control: team-orientated feedback

on tasks and clan-orientated feedback on values, attitudes and behaviors. In this research I continue the assumption that these two control types exist; however, I have chosen slightly different labels to describe these two control types.

Both of these control types are identified in this dissertation as variations of clan control. They both use clan-like mechanisms to enact their respective controls. These include activities such as hazing and socialization activities. When direct person-to-person feedback does occur, it does not involve a controller with legitimate power (Jex 2002), such as a manager. Clan-team control is the label I use for team-orientated feedback. This is consistent with researchers who operationalize clan control as a team orientated endeavor. It involves direct feedback on tasks that is delivered by peers belonging to the same team. Clan-attitude control is the label I use to encompass both Ouchi's original vision of clan control and the concept of self-control proposed by Kirsch (1996). This version of control recognizes that the clan can play a role in developing common beliefs, attitudes and values. Decisions that arise out of this common mindset are assumed to be congruent with similar decisions that would be made by other clan members.

Researchers have since expanded on Ouchi's initial view of the control theory in two directions. First, the initial formulation concentrated on identifying the single preferred control mode given a set of conditions. However, Ouchi discussed that multiple controls might be needed in some conditions. He noted that this was especially important when relying on formal controls, because explicit rules are needed to cover all

contingencies. Subsequent researchers have emphasized the use of multiple controls in a portfolio, or matrix of controls (Choudhury et al. 2003; Henderson et al. 1992; Kirsch 1997; Nidumolu et al. 2003-2004; Orlikowski 1991). According to the portfolio view, organizations will use a set of controls rather than just one. The need for a portfolio view became more apparent as researchers moved from conceptual discussions of control to real world examples and as the tasks studied became more ambiguous.

In the original view of controls, instituting a specific control would result in success. For example, if the desired outcome was production of a widget with certain specifications, then achieving that outcome (i.e. successful widget production) would be sufficient for control purposes. However, consider the case of production of systems software. The detailed software specification may not be known a priori. Instead, the controller may hold the parties to meeting certain project milestones. Just because these milestones are met does not assure successful delivery of the software. In this more ambiguous environment, controllers often use multiple means of control to more precisely constrain activities of the controllees. Furthermore, each mode of control (outcome, behavior, or clan) may be enacted through multiple control mechanisms (Choudhury et al. 2003).

The second expansion since Ouchi's original work has been in the area of dynamic controls (Cardinal et al. 2004; Choudhury et al. 2003; Kirsch 2004; Orlikowski 1991). Researchers have pointed out that previous control studies were cross-sectional discussions of the controls in place at an instance in time. Some researchers have begun

to explore how the choice of controls changes over time based on the phase of the project. However, the study of control changes over time is still early. One researcher (Kirsch 2004) specifically called for the study of control dynamics in further contexts and noted that her context was in custom software development, not in software product development. I would point out that flexible software product development may not have the same distinct phases, but that dynamic controls may still be needed. The difference is that flexible controls may need to automatically adjust with changing conditions, whereas current dynamic control studies focus on control changes that are enacted externally, through management intervention.

In this study, the view is that controls are enacted through a portfolio approach. Consistent with previous literature, I recognize that controls will need to adapt over time. However, I propose that flexible controls may contain inherent adaptability and there may be less reason to impose changes in controls used since the controls themselves can adapt. In a related paper, this need for flexible controls was recognized (Harris et al 2006). This paper called for a study of emergent outcome controls to manage development and the evolution of the development work product.

DEVELOPMENT METHODOLOGIES

The history of IT development methodologies reveals a central tension between control and flexibility. Initially developers used an unstructured, ad hoc development approach that led to numerous system failures. In response, the waterfall development

method was developed, and this method is still the most common approach in use today (Neill et al. 2003).

The key feature of the waterfall method is that it requires a planned approach to systems development. The development project is divided into phases. The exact number of phases may vary, but the general structure is the same. The process begins with planning and analysis, followed by development, and finally implementation and testing. During the first phase, planning and analysis result in a detailed development plan. Once the plan is in place the development team focuses on execution of the plan.

The waterfall method's emphasis on upfront planning is both its biggest strength and its biggest weakness. Clear upfront design and documentation leads to code that is cleaner, easier to maintain, and shortens the coding process. However, this approach relies on the ability of the development team to fully specify requirements during the planning stages of the project. The assumption that requirements can be specified upfront breaks down under two conditions. First, it assumes that users know and can articulate their needs. The desired knowledge about the target environment may be tacit and may not be readily accessible in a planning environment. Even if the users understand their needs, they may not be able to clearly communicate their desires to the development team. When different communities of practice exist there may be knowledge barriers that inhibit communication (Carlile 2002). Arguably, the differences between users and developers are such that their communities of practice are separate. The second condition affecting upfront planning is that it assumes an unchanging target environment. Changing

conditions can be a problem for any type of systems development, but they are especially prevalent in the development of software products. Software product developers may face additional dynamic factors that are outside the control of the development organization. For example, competitive product introductions or changing customer tastes can lead to changed requirements.

Clearly, the waterfall method is not a dynamic capability. This leads to the first hypothesis of the study:

H 1: As the market and technology become more uncertain, software development is less likely to be managed using a plan-driven method.

In the search for a more adaptable process, practitioners and researchers have proposed many alternative development methodologies. Some of these techniques build from the waterfall process in order to make it more adaptable. For example, a development team might use rapid prototyping during the planning phase of the waterfall development (McConnell 1996). This allows users to react to a mockup of a real system and helps uncover tacit knowledge about the new information system. Another alternative is the spiral development approach (Boehm 1988). The spiral approach breaks down the total system into separate iterations. Each iteration focuses on a portion of the total requirements. The requirements selected for a given iteration are those that have the highest risk remaining. One of the more modern alternatives to the waterfall approach is the Rational Unified Process (Kruchten 2000). Similar to spiral development, this is an iterative process that can adapt as the development proceeds.

These alternatives are more flexible than the traditional waterfall approach, but they are still 'plan-driven' (Boehm et al. 2004) approaches. In a plan-driven approach, the plan produces a specification that drives development. Any proposed changes must first be added to the specification and approved by management before they are reflected in development activities.

In contrast, agile methods do not rely on documented plans to direct change. In fact, documentation may be seen as an output of a development rather than an input (Cusumano et al. 1997). As discussed earlier, the Agile Manifesto (Beck et al. 2001) describes the differences between flexible processes and plan-driven approaches. It clearly emphasizes a more interactive development process involving dynamic tradeoffs between developers and users.

The benefits of this interactive process lead us to the second hypothesis:

H2: *In an uncertain environment, a controlled-flexible development method is more likely to match the market's needs than a plan-driven method.*

Furthermore, remembering the problems seen with an ad hoc approach leads to the third hypothesis:

H3: *In an uncertain environment, a controlled-flexible development method is more likely to match the market's needs than an ad hoc approach.*

Despite the common approaches of the agile methods there is considerable variation in the details of the processes. Compare two flexible approaches, the

Synchronize and Stabilize approach (S&S) (Cusumano et al. 1999) versus Extreme Programming (XP) (Beck et al. 2005).

In many ways, the S&S approach is similar to a plan-driven method. S&S has been used to manage large teams of hundreds of developers. Projects begin much like plan-driven development methods. Functional specifications are built, major milestones are established, and feature teams of three to eight developers are established. Standards, such as user-interface design rules, are also set (Iansiti et al. 1999). However, these startup processes stop short of specifying every detail. About 30% of the product design evolves from the development activities of the feature teams. Since the 70% of fixed requirements include infrastructure items, such as the system architecture, the amount of freedom given to the feature teams is considerable. Each feature team is expected to discover and implement the best possible solution for their product area. Many different structures are suggested to help guide the developers on the correct track, but many feature-orientated decisions are left to the developers (see Chapter 3 for more information).

XP takes a different approach and it bears little resemblance to a traditional plan-driven method. Upfront planning is minimized. The development team agrees on stories, which describe broad user requirements of the system. The actual design of the software evolves from the developers' efforts to implement the stories. It is the development team's responsibility to find working solutions to the stories' requirements. Although developers have considerable freedom regarding specifications, they are constrained by

practices such as pair programming, daily builds and testing. The software team works closely with users and the team must continually adjust their efforts to adapt to feedback.

Two differences between the S&S approach and XP relate to 1) their approach to architecture and 2) ownership of code. In regard to these items, S&S, unlike XP, resembles a traditional plan-driven approach. S&S develops an architecture before coding begins. The architecture constrains the flexibility and insures that any subsequent creative solutions are still true to the overall system needs. XP uses a minimalist architecture. For XP architecture is important, but for each iteration the architecture only supports features in that iteration. The XP philosophy is based on the belief that included features will change over iterations. Some planned features may be dropped and other unknown features may be added. Planning an architecture for future iterations is believed to be a waste of time since the feature set may change. The second difference relates to the ownership of code. S&S carefully segments the code. Each feature team has ownership of only its own code segment. In XP, the entire team has ownership of the entire code base.

Despite these differences, S&S and XP are both considered agile approaches. They both encourage the developers to pursue creative approaches and they both consider documentation an output of the process rather than an input. The differences are in the control mechanisms that each employs. In the following chapter, we apply the control framework to these mechanisms to determine if there are any common patterns in their control approaches.

SUMMARY

As noted in the first chapter of this dissertation, there are two primary research questions of interest in this study. They are:

Why do organizations choose flexible methods instead of plan-driven approaches?

How do the control mechanisms used in flexible methodologies influence the dynamic capabilities of a software development organization?

In response to these questions, this chapter explained the literature and developed the theoretical reasoning to support several hypotheses. These hypotheses can be illustrated in the following preliminary research model.

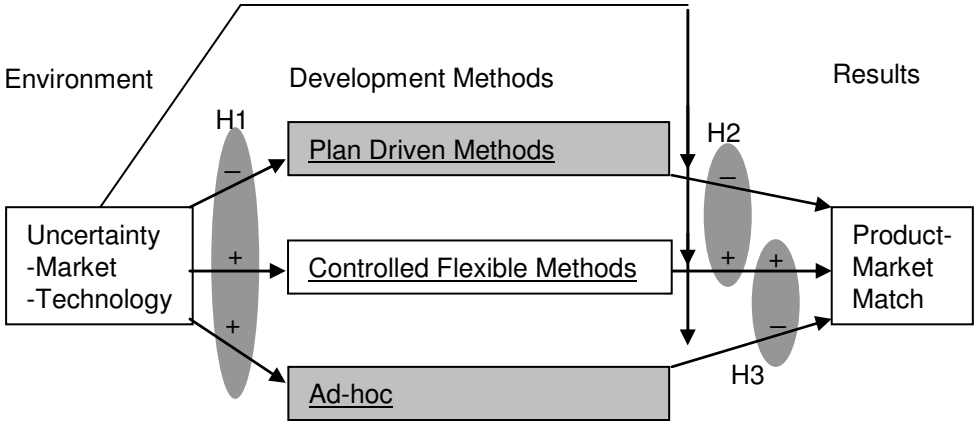


Figure 3: Preliminary research model

The hypotheses in this model respond to the first research question, as shown on page one of chapter one of this dissertation. The second research question relates to the central box of the diagram: "Controlled-flexible Methods". In the following chapter we will use control theory as a lens to explore existing literature on development methods. This allowed us to elaborate the model and to address the second research question.

CHAPTER 3: ANALYZING CONTROL MECHANISMS

This study examines flexible controls in action. Because of the path-dependent nature of dynamic controls, these controls need to be studied in context in order to understand how they operate. However, before beginning the analysis the model will be trained on data from textbook agile methods. Although these methods may be from textbooks they contain years of embedded experience.

Many methods can be used to train the model, but our goal will be to test the control model using a range of methods that cover the possible options for development. In order to differentiate between methods we focus on how the methods allow for management of change. This focus is logical since our governing theory, the dynamic capabilities approach, is a change focused theory.

The range of options is illustrated in Figure 4 below. At one extreme is the waterfall approach. It discourages changes during development. Since the early days of the waterfall, many iterative approaches have been developed. These methods were built with the expectation that change will occur during development and they are designed to deal with change. At the other extreme are the agile methods. They not only expect change, but they encourage it. It is the developer's job to explore options and to select the best solution. In comparison, the iterative approaches allow developers to suggest

changes, but the developers are not expected to implement design changes until they have been approved by a change management board.

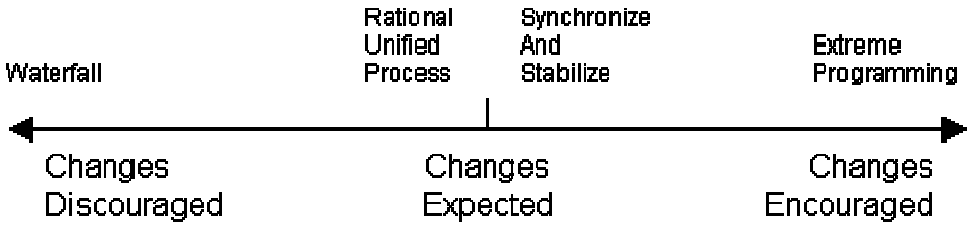


Figure 4: Attitude towards change

Figure 4 illustrates that the waterfall method and extreme programming anchor the ends of the change spectrum. The Rational Unified Process (RUP) (Kruchten 2000) and the Synchronize and Stabilize approach (S&S) (Cusumano et al. 1999) define the two sides of the middle. In fact RUP and S&S are similar at the highest levels. They have similar approaches regarding the use of major milestones, software architectures, and they both use an iterative development approach. The difference occurs in how they treat changes. In RUP, developers are expected to deliver according to the plan. If the developer has a new idea, the idea must be submitted to a change management committee where it is prioritized with all of the other changes. If the idea receives top prioritization from the committee, then it will be added to the official plan and the developer can work on the change. In S&S, developers are expected to try out new ideas and then to gather feedback from others based on the developed code.

The analysis began with the categories suggested by control theory: outcome control, behavioral control, clan-attitude control, and clan-team control. Summarizing from the literature, each of these is defined as follows:

- *Outcome:* Outcome control relies on a priori specification of desired outcomes and measurement of delivery versus those outcomes. It has been used to measure success against final deliverables, such as sales revenue (Ouchi 1977), as well as for interim deliverables and broad goals, such as project milestones (Henderson et al. 1992; Kirsch 1996).
- *Behavior:* The behaviors that transform inputs to desired outcomes. In order to use behavior controls the transformation behaviors must be known (Ouchi 1977) and observable (Kirsch 1996).
- *Clan-attitude control:* The individuals monitor themselves and decide on the appropriate action based on the individual's interpretation of attitudes and beliefs shared with the clan. These clan attitudes and beliefs take time to develop and are built through selection, and socialization (Ouchi 1980).
- *Clan-team control:* Team members advise each other on tasks (Henderson et al. 1992). They may work directly together or use socialization tactics, such as hazing.

Since the study is orientated towards understanding dynamic systems, the analysis begins with Extreme Programming (XP). Comparing the XP key practices and the initial control framework reveals a problem with the definition of outcome control. Traditional outcome control requires a priori specification of a planned outcome and comparison of the actual delivered product against the planned outcome. In the case of XP, the final outcome is unknown until the development is finished. Thus, traditional outcome control can not be used. However, XP includes techniques to manage *emergent outcomes*.

The term ‘emergent outcome’ describes the way that XP gradually builds towards a final outcome (software deliverable). In XP, this emergent process is an integral part of the development of the software and it can not be separated from the development process. Even though there is no a priori software design, XP contains structures to guide the outcome as it emerges from the development process. This insures that the final solution does not arrive as a surprise and that it meets the needs of the customers.

Emergent outcomes contain two kinds of mechanisms: mechanisms that create scope boundaries and mechanisms that provide ongoing feedback. Scope boundaries limit technological wandering by insuring that developers remain focused on the task at hand. For example, XP uses short iterations of one to three weeks. If the development team does wander off-course, these short iterations limit the amount of drift that can occur before feedback is received. XP also provides developers with user stories that define required capabilities at a broad level. This helps define the area within which the developer is to create a solution.

XP also contains feedback processes that allow the development team to correct their course on a real time basis. Researchers have shown the value of early and ongoing user feedback (MacCormack et al. 2001; MacCormack et al. 2003a). One XP requirement is the presence of a user co-located with the development team. The concept is that developers will have continuous access to a user representative as they make design decisions. However, XP also realizes that a single user may not be representative of the market as a whole. One important reason for the short release cycle is that it allows the

team to continually check their evolving design against the needs of the broader market of users.

Table 3 shows a mapping of the XP mechanisms to controls. As the analysis reveals, the extreme programming mechanisms are consistent with the expectations of control theory as adjusted to include emergent outcomes. Ongoing feedback is the primary concern of the process. Progress is made visible to other team members so they can synchronize and provide feedback to each other. A customer representative is a full time member of the team so that feedback is available as it is needed. Furthermore, development is broken into small increments. This short-cycle approach allows the software to be continuously exposed to the market and offers the opportunity for ongoing feedback, beyond those of the embedded user representative.

Table 3: Extreme programming and control mechanisms

Mechanism	Outcome Controls			Behavior	Clan Controls	
	Traditional Outcome	Emergent Outcome Scope Boundaries	Ongoing Feedback		Clan-Team	Clan-At-titudes
Sit Together			Progress observable by team members.		Team members accessible for advice.	
Whole Team			Feedback including customer representative		Team is self-sufficient and unified. No mavericks.	
Informative Workspace	Visible results		Outcomes observable	Daily Progress visible.	Team can readily see each other's progress.	
Energized Work						When we are at work we will do our best.
Pair Program-ming			New ideas are tested with partner.	Work together.	Activities transparent to team members.	
Stories		Broad statements of intent focus efforts.				
1-3 Week Cycle		Limit amount of change that can occur in each iteration.	Market feedback every 1-3 weeks.	Work to short deadlines. One-day slips result in miss of weekly target.		

Table 3: (Continued)

Mechanism	Outcome Controls			Behavior	Clan Controls	
	Traditional Outcome	Emergent Outcome Scope Boundaries	Ongoing Feedback		Clan-Team	Clan-Attitudes
Quarterly Cycle		Place business constraints as well as market constraints.	Review with management. Guard against feature creep.			
Slack				Meeting weekly cycles more important than features.		
Ten-Minute Build			Make it easy to demonstrate.	Don't break overall system.	Code must work well with others.	
Continuous Integration			Always be ready to demo latest product	Fix bugs as they occur.	Must maintain synchronization with others. No surprises.	
Test-first	*Not really an external contract.*[4]	Develop a detailed goal for each feature.				
Incremental Design		Each iteration focuses on only a few things.	Each iteration ready to use or demonstrate to market.			

[4] At first glance this may seem like a version of outcome control, but it does not fit the conditions. This test is built by the developer before he begins coding. If the developer changes direction, the test cases can be changed. There is no attempt to hold the developer to delivery based on the original test cases.

One interesting insight involves behavioral controls in XP. These behavioral controls all have a focus on immediacy. Daily builds mean that bugs must be fixed as they are found, not added to bug lists. Weekly releases mean that a one day slip can result in a 20% schedule overrun. Ten-minute builds make it quick to compile the system with the latest changes. This sense of immediacy works with the short cycle times to keep the team on course. Although the team has freedom to create, they do not have time for idle wanderings. Compare this to a large, plan-orientated approach where the next milestone may be weeks away. XP epitomizes the adage: 'don't put off until tomorrow what you can do today'.

As the next analysis we will consider the Synchronize and Stabilize approach. The following matrix in Table 4 looks at S&S through the eyes of the control framework. The synchronize and stabilize approach is less free-flowing than XP, but it still gives considerable freedom to its developers. In S&S we some evidence of traditional controls, but there is still a heavy emphasis on the use of emergent outcomes.

The S&S traditional controls include an initial product vision and major milestones. However, these items are too broad to control the daily activities of the developers. Developers may also start with a rough specification that includes up to 70% of the required features. However, this can only act as a crude version of an outcome control. Even if we ignore the unstated 30% of the requirements, the problem is that the items on the list are mutable. Thus, the project could change, replace, or remove elements of the initial specification.

Table 4: Synchronize and stabilize and control theory

Mechanism	Outcome Controls			Behav-ior	Clan Controls	
	Traditional Outcome	Emergent Outcome Scope Boundaries	Ongoing Feedback		Clan-Team	Clan-At-titudes
Start with Vision	Can provide some control – lacks measurable detail.	Carves out area of development.	Provides some basis for feedback.			
Specify goals and non-goals[5] to include up to 70% of the final features before development begins.	Partial specification – however some features may change, so its tough to maintain accountability.	Critical features are fully specified.				
Architecture & Feature team assignments first.		Each team owns specific area.				
Daily Builds			Progress visible	Must keep code in working condi-tion.	Dunce cap if break the build.	
Continuous End-user reviews			Focus on delivered function-ality, not code.			
Major Milestones	Specify due dates for feature bundles.					
Development & Testing done in parallel				Always keep code in working condi-tion.		

[5] A non-goal is something you don't want to do in the current release.

Table 4: (Continued)

Mechanism	Outcome Controls			Behav-ior	Clan Controls	
	Traditional Outcome	Emergent Outcome Scope Boundaries	Ongoing Feedback		Clan-Team	Clan-At-titudes
Code reviews			Independent view of progress.		Code must be acceptable to other team members.	
Check-in and change tracking		Identify who is allowed to make changes to a section of code at any given time.				

The specification list becomes another scope boundary; it provides broad outlines for the direction of development but it does not provide detailed specification. Scope boundaries also are evident in other areas. For example, a feature team must fit their code into the established architecture. This architecture is not an outcome control since it does not dictate what the team must build. However, the architecture does place boundaries on what a team can do. It determines how the team's code interacts with other modules and, thus, dictates what information they can receive and give to other modules. S&S also relies on role definitions to place boundaries on the teams. A feature team only has responsibility for a specific feature-set, and ownership privileges are defined for each piece of code.

As can be seen, S&S's use of scope boundaries is different from those XP. XP relies primarily on short cycle times to rein in development. XP developers don't have time to wander too far afield because the next release is no more than a few days away.

S&S uses a partial specification, defined roles, a fixed architecture, and limits on code ownership to constrain developers. Both methods use daily builds to insure that maverick programmers do not get too far out of step with the rest of the organization. In total, all of these mechanisms can be seen as alternative ways to bound the creativity of the team. Although the team is given permission to innovate, these mechanisms insure that the innovation does not stray too far from the intent of the software.

S&S also includes mechanisms for ongoing feedback (MacCormack et al. 2001). Multiple releases are offered to the market to gather user input. For example, Netscape 3.0 underwent six beta releases to gather user input before it was released to market (Iansiti et al. 1999). Furthermore, the team takes any chance possible to get continuous end-user reviews of the work in-progress. This ongoing feedback is as important as functional/bug testing.

S&S is not quite as extreme as XP. There is still a sense of immediacy derived from daily builds and testing that occurs in parallel with development. However, this sense is somewhat muted. Although software is built daily, an individual piece of code may go several days before being checked into a build. Since there is no user representative on each team, the feedback is somewhat less current. Slightly more structure upfront allows for a greater use of traditional outcome controls. However, this is a relative comparison between the two techniques. In general, the profile of controls is the same. They both rely primarily on emergent outcome controls; they both keep a sense

of immediacy in their behavior expectations, and they both include team-orientated clan controls.

In order to better understand these controls, we next turn to an analysis of the Rational Unified Process (RUP). RUP will provide an interesting contrast because it is a plan-driven approach, but it allows for learning throughout the process.

Before we can more thoroughly analyze the RUP process we must more fully explore the definition of an emergent outcome. In one sense, even the most plan-driven process involves outcomes that emerge from the process. All software development begins with an amorphously-stated need which is eventually solved with the delivery of a finished software product. However, this transformation from need to software is quite different for plan-driven versus controlled-flexible processes.

In a plan-driven process the specification of the software occurs separately from the software development. From the developer's viewpoint a plan-driven method is deterministic with the software program flowing directly from the design specification. Creation of the design either occurs at the front of the process (waterfall) or in spurts between development cycles (RUP). In contrast, in a flexible environment the developer is central to the design creation. Design decisions occur in real time as the developer considers both the user needs and the technical capabilities of the environment.

When we talk about emergent outcomes we are describing a continuous design process that involves the programmer in design decisions. Although RUP does allow for evolving designs, it does not allow for emergent designs. RUP design decisions must be

agreed upon and prioritized by a change committee. The developer is given detailed specifications for implementation, and any creative suggestions must find their way back into the change process.

Table 5 maps RUP techniques onto controls. This matrix understates the sophistication of RUP. From a control viewpoint, much of the planning in RUP is enacted through a design document. RUP includes business modeling plans, architecture plans, and a formal change management board. The results of all of these processes are embedded in the design specification. Therefore, the inclusion of this single document factors in the decisions of all of these processes.

As the table reveals, RUP leans heavily on traditional outcome control. Developers are assigned due dates and detailed specifications for each iteration. Testing helps determine if the specified outcome was delivered. Certain behaviors in terms of tools and standards are also pre-specified. From the developer point of view, RUP is not a flexible project. It is as rigid as a formal waterfall approach.

However, above the developer level RUP does include adaptive aspects. The difference is that the ability to make changes is in the hands of project management, not in the developers' hands. Between each set of iterations the change management board can adjust the controls by changing the specification. The two shaded rows from the table indicate specific aspects of RUP that allow project management to better learn the users' needs.

Table 5: Rational Unified Process and control theory

Mechanism	Outcome Controls			Behavior	Clan Controls	
	Traditional Outcome	**Emergent Outcome** **Scope Ongoing Boundaries** **Feedback**			**Clan-Team**	**Clan-Attitudes**
Design Document	Must deliver to the specification. Proposed changes reflected in subsequent iteration.					
Iteration Due Dates	Measurable outcomes that can be tracked.					
Testing Workflow	Makes outcomes visible.					
Environmental Workflow				Determine tools and standards that the implementer must use.		
Configuration Manager		Define sections of code a developer can access. Rules for sharing code.				
Multiple Iterations[6]		Limit changes for each iteration to specific areas.				

[6] The last two items from the table are not controls enacted on developers. They are project level controls that allow the project to react to change.

Table 5: (Continued)

Mechanism	Outcome Controls			Behavior	Clan Controls	
	Traditional Outcome	Emergent Outcome Scope Ongoing Boundaries Feedback			Clan-Team	Clan-At-titudes
Stakeholder Reviews[6]			Provide feedback through-out the project.			

User feedback in RUP is one step removed from the developer. It is an input to the change management committee who decides on features for each software iteration. The developer does directly receive testing feedback; however, testing and user feedback are fundamentally different. Testing is more of an objective comparison against pre-defined specifications. The need for user feedback in the more flexible methods (i.e. XP) is based on the belief that some user knowledge is tacit and can only be accessed through experience with working software.

In order to complete the picture, the final analysis involves the waterfall method. This analysis is presented in Table 6. We will examine a traditional phase/gate approach wherein the development is divided into phases and 'gates' between each phase are used to control the decision to proceed to the next phase. The decision to proceed is usually only given if the previous phase's work is complete. Furthermore, the development team is strongly discouraged from revisiting decisions relating to previous phases of the project.

The waterfall approach is similar to RUP in the way that it constrains developers. Developers are managed using outcome controls, and some behavioral controls. The difference between the two approaches is not apparent from the developer point of view. It is only when one considers the project point of view that the difference is apparent. RUP uses multiple iterations to allow its project management and stakeholders to adjust the project direction as new learning occurs. On the other hand, the waterfall approach assumes that if enough attention is given to upfront planning the development team will not need to revisit the development plan.

Table 6: Waterfall process and control theory

Mechanism	Outcome Controls			Behavior	Clan Controls	
	Traditional Outcome	Emergent Outcome Scope Boundaries	Ongoing Feedback		Clan-Team	Clan-At-titudes
Design Document	Must deliver to the specification .					
Project Due Dates	Measurable outcomes that can be tracked.					
Testing	Makes outcomes visible. Focused on technical issues, not end-user acceptance.					
Environment				Determine tools and standards that the implementer must use.		
Configura-tion Manager		Define sections of code a developer can access. Rules for sharing code.				

SUMMARY OF METHOD / CONTROL ANALYSIS

The analyses presented in this section reveal a clear difference in control choices for two flexible methods when compared to two plan-driven methods. The plan-driven methods rely heavily on traditional outcome controls to keep on track. They also use some behavioral control in the form of standards that must be used by developers. The

controlled-flexible methods utilize a new variation of outcome control: emergent outcome control. There are two components to emergent outcome control: scope boundaries and ongoing feedback.

Scope boundaries are used to target innovation in the proper area. Developer freedom may be bounded by functional limits. For example, the vision or the architecture may define the area in which the developer is expected to work. They may also be bounded by resource constraints. For example, the developer may only have a fixed amount of time to work on a feature. Developers are allowed to be creative within the limits defined by the scope boundaries.

Ongoing feedback is used to offer corrective action as the outcomes emerge. The feedback that flexible methods provide to developers is different from the feedback provided by plan-driven approaches. The plan-driven methods allow for testing versus specification. This provides feedback on how well the software meets the plan. These plan-driven methods do not pay as much attention to user feedback. In the case of the waterfall method, no user feedback is included. The method assumes that the software is specified correctly in the first place. The RUP process is a plan-driven process that does allow for user feedback and correction during the development process. However, this feedback is much less pervasive and it is filtered through the project management rather than handed directly to developers.

Another difference is in the type of control used to keep projects on time. The flexible approaches use behavioral control to enforce a sense of urgency or immediacy to

the project. Developers have pressure to show continuous progress through daily builds and frequent user reviews. Although one flexible method, the S&S method, does use project milestones, these milestones are not the only form of time pacing that is used. In comparison, the plan-driven approaches rely more heavily on timelines and milestones to keep the project on track.

The final difference is in the area of behavioral control. The flexible approaches rely on peer pressure to help synchronize team activities. Team members may be subject to hazing if their code does not integrate well, or if they depart from team expectations. In contrast, the plan-driven approaches rely more exclusively on formal outcome and behavioral controls.

RESEARCH MODEL

Based on the analyses of the range of software development processes, the following research model is proposed in Figure 5.

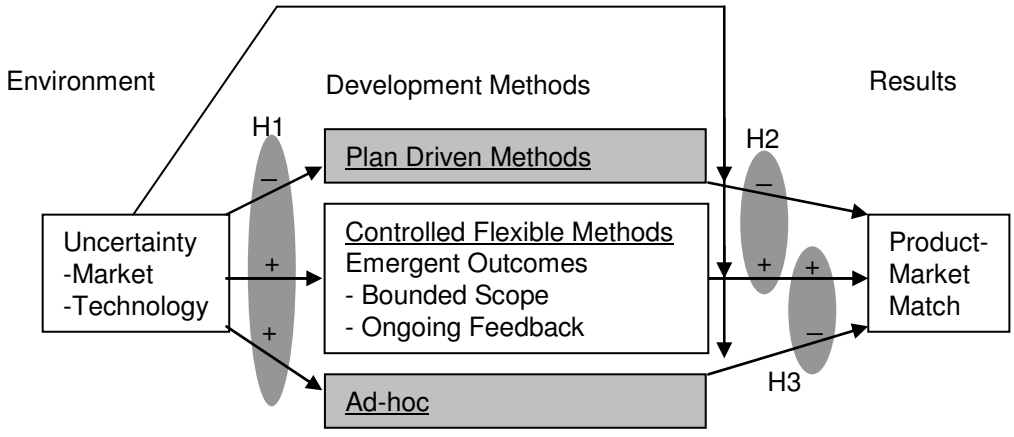


Figure 5: Research model

This model reiterates the previously listed hypotheses, but it also suggests a definition for a controlled-flexible method. Specifically, if techniques to manage emergent outcomes are detected, then the method used is a controlled-flexible method. This results in a revision to the second and third hypotheses as shown below:

H 1: As the market and technology become more uncertain, software development is less likely to be managed using a plan-driven method.

H 2: In an uncertain environment, mechanisms that manage emergent outcomes are more likely to match the market's needs than a plan-driven method.

H 3: In an uncertain environment, mechanisms that manage emergent outcomes are more likely to match the market's needs than an ad hoc approach.

CHAPTER 4: RESEARCH DESIGN

The study uses a theory-based research approach. It has the goal of understanding how flexible development processes are managed, and why flexible processes are chosen. A review of the literature suggests the application of the dynamic capabilities extension to resource-based theory of the firm as an explanation for why flexibility is needed, and the use of control theory to explain how flexibility is managed. These theories are 'trained' in the concerns of flexible development by comparison with existing development processes. From this approach a model describing flexible development is built.

As a confirmatory step in the analysis, the derived model was shared and discussed in a focus group with four experienced developers. The discussion revealed support for the underlying model, but it also suggested that measurement through a survey approach would not be simple. One implication was that development methods-in-practice, might not cleanly break down into three categories (Plan-driven, flexible-controlled, and ad hoc). Instead, any given method could be a local adaptation that borrows elements from all three categories. Furthermore, since these adaptations are localized to a specific instance, there is no pure standard model of what a specific method-in-practice would look like. Finally, it was clear that the context was important in

understanding a specific situation. For example, the differences were expected to exist between the control environment for junior and senior developers on the same team.

STUDY DESIGN

Because of the aforementioned factors, the empirical data for this study is gathered in a field case study. Case studies have frequently been utilized in the study of organizational controls (Appendix 1). Case studies are especially appropriate when "how" and "why" questions are posed (Yin 1994). In this situation the relevant concerns are how flexible methods are controlled and why organizations choose to use flexible methods. In addition, a case study approach is useful because the actual control approach used is expected to be context dependent.

The first suggestion of a context dependent approach comes from the dynamic capabilities literature. Researchers describe dynamic capabilities as being path-dependent. This is supported in the preliminary interviews that were completed for this study. One participant noted that every software project he was involved in included at least some element of flexibility. In fact, it appears that the known methods may represent archetypes that anchor the extreme points on a continuum and that the real world expressions of these methods may represent tradeoffs between the various methods. This can be represented in the following figure (Figure 6) which indicates that three archetype methods exist. In practice, the method utilized by a given entity will represent a tradeoff of features from all three approaches. It is expected that entities

studied will find themselves somewhere inside the triangle and not at any of the three extreme points.

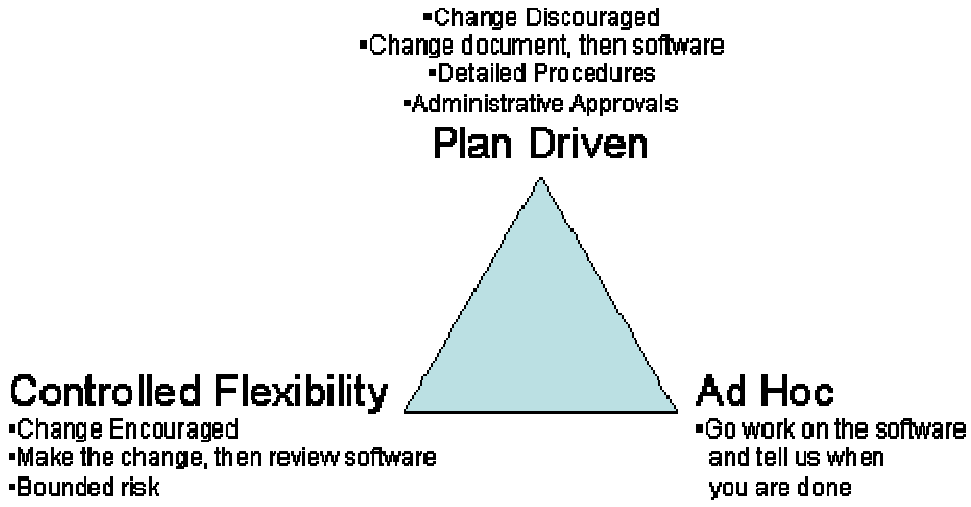


Figure 6: Methods-in-practice represent a tradeoff of archetypes features

UNIT OF ANALYSIS

The unit of analysis for the study is the individual software project. In most cases, the study included multiple informants on a single project. For example, a project leader, a lead developer, and a new developer all offered different viewpoints with regard to controls. The study allowed the researcher to compare differences based on the interviewee's point of view.

SAMPLING FRAME & RECRUITMENT

As recommended by Eisenhardt (1989b), theoretical sampling was used to choose the participants in the study. In other words, cases were selected with purposeful intent to test the predictions of the theory. The sampling frame focused on software development

teams that used a flexible development approach. Furthermore, organizations were sought out that could be used to challenge the hypotheses. For example, the recruitment uncovered one organization that fit the perfect profile for using a flexible approach, but they used a plan-driven approach. An exploratory interview was held with this organization to see why they didn't use a flexible approach.

A development approach is identified as a flexible approach based on the role of the software specification in development and based on what attitude towards change is embodied in the approach. If coding begins without a specification or before the final specification is complete then the project was classified as a flexible approach. In addition to the presence or absence of an a priori specification, the study will consider the development team's attitude toward changes. A flexible method encourages new discoveries during development and encourages developers to explore new solutions. A plan-driven approach discourages changes and/or requires that all suggested changes be incorporated into the formal plan before implementation occurs. Furthermore, a plan-driven approach will use a formal change management process to insure that any proposed changes are fully vetted and approved by relevant stakeholders before they are incorporated into the software.

It should be noted that this sampling approach does not distinguish between controlled-flexible approaches and ad hoc approaches. It is expected that many flexible control structures will be tacit and only the interview process will reveal whether a flexible process is controlled or ad hoc.

Sampling was also biased towards product-orientated software development teams. These teams were expected to be more likely to use a flexible development approach. This expectation is based on three factors. First, the existence of competition results in a moving target. Actions by existing competitors and threats from new entrants can lead to changing goals for the development team. Furthermore, there is more uncertainty in defining the needs of an entire market as opposed to those of a single customer. In a market, different customers not only have different needs, but they can also have conflicting needs. As a result, there can be more uncertainty in pinning down the needs of a market as compared to those of a single customer. Finally, a product is central to the existence of an organization, and product uncertainty may create uncertainty in organizational viability. This can create organizational pressure to prove viability by exposing evolving products to potential customers. These product-related factors increase the likelihood that software product organizations will choose a flexible development style.

Of course, market uncertainty is not the only source of uncertainty in this proposal. Technical or platform uncertainty may also lead to a flexible approach (MacCormack et al. 2003b). This type of uncertainty is likely to be found in the first generation of a new product. Since first generation products also face unproven market conditions they represent an especially attractive environment for flexible methods. As a result, we looked towards first generation product teams in our search for teams utilizing flexible methods.

These selection guidelines were heuristics intended to point the way towards sites that may be using flexible methods. The goal of the site selection is to explore flexible software development, not to find a product orientated company. For example, many custom-oriented software organizations use flexible development approaches. The study will accept custom-orientated software teams or any other team that uses a flexible approach. However the selection guidelines will bias the site selection process towards product-orientated software teams. To the extent that the final sample is biased towards product-orientated teams, there will be a threat to generalizability that will need to be addressed in future studies.

Recruitment began by soliciting executives at development organizations and requesting their permission to include their projects and developers in the study. Based on executive agreement and sponsorship, individual software developers in the organization(s) were approached and interviewed. The data collected covered four organizations as shown in Table 7 below.

Table 7: Description of subjects interviewed

	# Inter-views	
Large ASP		
The company's central servers offer transaction and reporting services to a distributed customer network. The company has about 300 developers, but projects are handled in small teams of three to six developers. Projects are released into the existing shared-resource environment. Programs exchange data through shared data repositories and interact through public API's. When larger initiatives are required, the company breaks them into smaller functional components, and assigns these components to smaller teams. These teams interconnect their components through the common environment in a loosely coupled fashion.		
Custom	3	One interview included two participants, so a total of four participants were interviewed. This team of five developers builds custom software for an international client.
Reporting	3	The interviewed team members were all team leads. In this case they were not all working on the same project. Each team lead managed teams of one to three developers.
Maintenance	4	This team managed enhancement requests to existing products. Therefore, their products begin with a known starting point.
Small Software Company		
Operations Software For Small Businesses	4	This company builds software to support operations for small businesses. They have two existing modules deployed and in use by about one thousand customers. They have been building a new module for two years and they have transitioned through three methods attempting to get the product delivered. Therefore, this one case provides insight into several different ways of doing business. The development team consists of a manager and three developers.
Uncoded Interviews – used for confirmatory evidence		
Focus Group	1x5	5 experienced developers – convenience sample. Not coded.
Consumer Product	4	This group is an additional arm of the ASP mentioned above. The information was redundant, and they were used for confirmation, but they were not coded.
Mobile Device Software	1	Uses off-shore outsourcer to deliver in a flexible process. Data collected through hand written notes. Not coded because there was only one subject, and the notes were hand-written not audio recordings.
Web Development	2	This small company delivers web products using a plan-driven approach. They provided time for a short interview to describe why they didn't use a flexible approach, but it did not follow the full interview protocol and it was not coded.

The focus group listed in the study was part of the pre-study used to validate the findings. The participants in the study were all graduate students at the University of South Florida who had previous experience in development at various companies. Subsequently, target companies were chosen as candidates for case studies to test the various study hypotheses

Many of the interviews were held with one company that has over 300 developers. This company is an Applications Services Provider (ASP) – the software is deployed on centralized servers owned by the ASP and clients access it via a network. The software acts as middleware tying together remote systems of its business customers. Based on the transactions flowing through its middleware, the ASP offers real-time transaction-related services, and reporting capabilities for client's back-office services. Although the ASP does not deal directly with consumers, its transaction-related services enable the ASP's business customers to serve a large consumer market.

Discussions with the company began with a meeting with the CTO and his directors to explain the project concept. From these discussions, projects were selected that provided insight into a contrasting set of project environments. Interviews with this company were held with four different teams. By interviewing teams within the same company, the analysis was able to hold the company environment constant, while considering the differences caused by projects.

> **Custom:** this project was a major new initiative for a client that was just starting its business. The client was located in South America and there was a language barrier between operations personnel at the client and between the development team. Operationally, much of the existing knowledge is tacit in the ASP development team. The ASP team has delivered similar projects in the past whereas the client is new to the application area. However, some of the knowledge is still with the client

since there are some environmental/country factors that are new to the ASP.

Reporting: this team develops reports. Some of these reports are in support of new products; others are developed for specific clients; and others may be standard reports that support all clients that use a specific product.

Consumer Product: Due to regulatory changes, a new consumer product was required. This product was new to the market and had to be delivered on a strict timeline.

Maintenance: The fourth group in the company was selected as a contrast to the other areas. In other areas of the company, needs from customers are often stated at a general level. More detailed design constraints are driven by tacit knowledge, residing either with the developers or embedded in existing systems. As a result, upfront requirements are often incomplete. In the maintenance group, projects are specific enhancements to existing systems, and are often more clearly specified upfront. As a result, this group provides a more plan-driven counter-point to the other areas.

It should be noted that this company does a good job compartmentalizing its initiatives. For example, a project for the reporting team might involve 1-3 team members. The reporting team would consider this a complete project team; however, the

reporting project may actually be part of a larger initiative. Other teams may be working on transaction handling and data storage for the same project, but the reporting team works independently of these affiliated projects.

However, even when projects are truly independent, they are launched into a production environment that is interconnected. The operating environment consists of a set of networked systems that do transaction processing, data storage and manipulation, and reporting. New initiatives are interjected into this internetworked operating environment and they rely to some extent on existing resources and data flows to deliver their new solution. Furthermore, the various software services interact through the use of shared databases and through public API's that allow some programs to offer services to other programs. Because of the interconnected nature of the various software products, regression testing can be extremely important for this company.

In addition to this large company, a smaller company was interviewed. This company develops software that allows small businesses in a specific vertical industry to manage their operations. This company currently has two modules and has been working on a new product for over two years. In their efforts to develop the new product, they have had several problems and had gone through several transitions. It was expected that these changes over time would offer 'natural barriers' that could be used to test hypotheses in a case setting (Lee, 1989). At this company, interviews were held with a product manager, a senior developer, a developer, and a junior developer. This is a small company, and this included their entire development staff.

Two additional organizations were interviewed briefly. These organizations were not available for full interviews. However, each organization had unique characteristics in regard to flexible processes, and they were willing to provide brief exploratory interviews.

One of these companies uses outsourced developers in Russia to develop their software. It was felt that this would provide an interesting case study because outsourced developers are not usually managed using flexible approaches. Seeing how this company adapted its flexible processes to allow remote management was expected to be useful. In this case, the chief technical officer was the only interview subject. The CTO directly managed the outsourced developers, and the developers were not available for interview.

The final small company develops web sites. Web site development is typically believed to use a flexible or even ad hoc development approach; however, this company uses a very structured approach. Typically, they contract with a client to build a detailed upfront plan, and then obtain a new contract to implement that plan. Since this approach runs counter to expectations, it was expected to offer an example that might challenge existing models, or might offer new insights.

INTERVIEW DESIGN & INTERVIEW TECHNIQUES

The interviews were expected to form the basis of an elaborative coding analysis (Auerback, et al. 2003). Therefore, the interview process needed to be open to uncovering new constructs, as well as gathering evidence to prove or disprove the proposed relationships with the existing constructs.

A sample outline of the interview flow is shown in Appendix 2. However, the actual structure of the interview was allowed to vary. For example, at times interviewees would volunteer information that eliminated the need for subsequent questions. In general, the interviews began at a broad level, and narrowed down to more detailed questions related to the hypotheses. For example, interviewers were first asked what factors would lead them to choose a plan-driven versus flexible approach. Next they would be asked to pick two contrasting projects they had experience with and explain why one was more or less flexible than the other. Finally, they were asked what role uncertainty had in making their decision to go plan-driven versus flexible. This approach ensured that the subjects had the opportunity to share insights that were contrary to the study hypotheses, as well as to test the specific hypotheses.

One difficulty in analyzing qualitative statements can be in comparing statements across subjects and seeking common definitions for terms. For example, two subjects who say that their process is flexible may attach different meanings to the term 'flexible'. In fact, in these interviews, even though the subjects were given definitions for flexible-controlled and plan-driven, the use of the terms was not consistent.

The interviews used three techniques to clarify the environments in use: use of specific examples, control mechanisms, and relative comparisons.

Specific Examples: Many times subjects had opinions of the ‘best’ or right way to manage a project, but the evidence of methods-in-use supported different conclusions.

In the interview process, subjects were asked not only their opinions, but also to describe specific projects.

Control Mechanisms: For the subject's current project, information was gathered on the control structures in place. This included mechanisms such as requirements documents, feedback routines, architecture, and standards. As will be described in the Analysis section below, expert raters determined the type of method-in-use based on the control mechanisms in use.

Relative comparisons: Once the categorization of the current project was established, the subject was asked to think of a comparison project. A critical incident approach was used to choose a contrasting project. For example, a subject might be asked to select a comparison project that had requirements more firmly fixed upfront than the current project, or one that was much more loosely structured. Hypothesis testing statements were then based on the relative comparison of the two projects. For example, this approach would allow a rater to compare the amount of uncertainty faced by different projects versus the method used to manage the project. Because of this relative comparison, the exact method used in a given project was less important – the more important factor was the relative difference between two projects.

ANALYSIS AND DATA REDUCTION

The interview sessions generally lasted from one-half hour to one hour. In most cases, only one person was present in the interview, but in two instances two people were present. Most of the interviews were recorded and transcribed. However, one interview

(Mobile Device Software) was only recorded via the interviewer's notes. The raw transcripts were edited to insure confidentiality of the interview subjects and companies. The result was 299 pages of double-spaced, typed text.

The data reduction included three phases: 1) Coding to categorize statements from the transcripts; 2) Rating to analyze the coded excerpts versus the hypotheses; 3) Consolidation to compare and analyze findings between team members.

Phase I of Data Reduction: Coding

The clean transcripts were given to IS graduate students who coded the transcript for data reduction and to highlight key constructs. The coders were blind to the study hypotheses, but they were trained in the important factors to be identified. In other words, the coders identified the presence or absence of constructs. A sample of the form used by a coder is shown in Appendix 5. As this sample reveals, categories were chosen to allow the analysis to test the proposed hypotheses. For example, the coders not only coded instances of uncertainty (which was expected to be associated with selection of a specific type of method), but they also coded all discussions of factors that influenced the choice of a development method. Each coding session took about 3 hours to complete the coding of an 18 page, double spaced transcript.

Each transcript was coded by two independent coders, and the independent work was then compared and reconciled, and an inter-coder agreement was calculated. As Table 8 shows below, the agreement for the coders was above 70% in each instance.

Four sets of interviews were not coded. The original focus group followed a different format than the subsequent interviews. It was only used as a context to design the subsequent interviews and was not used to test or elaborate the hypotheses. The interview with the web development group also was not coded. This interview was fairly short – about 10 minutes – and it did not cover the full range of issues. Instead, this interview focused solely on why they did not use a flexible method, even though they were in an area that is usually managed flexibly. The interview with the mobile device software company was not coded because it was in non-standard format – it only consisted of notes, not an actual transcript. Finally, one of the groups from the large asp (Consumer Product) was not coded due to resource constraints. This last set of interviews included four transcripts. Each of the transcripts was coded at least once, but only two of them were coded the requisite two independent times. An analysis of the preliminary codings did not reveal any surprises, but they were not marked as coded since two interviews were not duplicate coded.

Table 8: Inter-coder agreement

	Inter-coder Agreement
Large ASP	
Custom	
• Manager & Lead	83%
• Sr. Developer	80%
• Developer	72%
Reporting	
• Lead	87%
• Lead II	78%
• Lead III	82%
Maintenance	
• Director	83%
• Sr. Manager	81%
• Sr. Developer	79%
• Developer	78%
Small Software Company	
Operations Product	
• Manager	75%
• Sr. Developer	81%
• Developer	78%
• Jr. Developer	72%

There were three different coders involved in the coding process. For a given transcript there were only two coders, but in order to speed up the coding process three coders were used.

Phase II of Data Reduction: Rating

The coding was used to categorize the interviews by constructs. Once the coding was completed, two independent raters used the coded statements to evaluate the study

hypotheses. The raters consisted of the author of this study, a doctoral candidate, and a Professor at USF. Both of the raters are experienced systems professionals as well as trained academics. The raters' instructions for evaluating the codings are shown in Appendix 6. The ratings were in general agreement; however, the raters met after each rating to insure that the interpretations were in accord and to resolve any differences.

This rating process involved more than just tallying the number of controls used by an organization. The raters needed to understand the characteristics of each type of development method and look for those characteristics in the transcripts. Furthermore, they needed to recognize that an organization could be using a mix of methods. The discussion in Appendix 3 describes the differences in the three development methods. Using these definitions as a guide two independent raters were able to consistently come to agreement on the type of method-in-use. For example, a typical rating is shown in Figure 7 below. The figure displays the two raters' markup of projects from a single transcript. Each diagram shows the current project of the subject (P0) and a project from the subject's prior experience (P1). Although the raters' placement of the projects was not identical, it is reasonably close. As the figure reveals, neither of the projects were identified as 'pure' method types, and in fact this was generally the case in the findings.

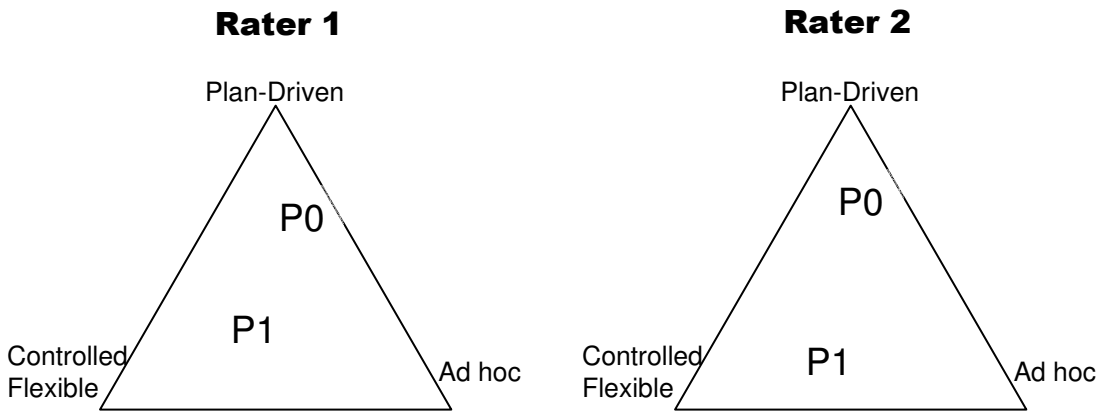


Figure 7: Comparing two raters for the same transcript

I will not attempt to replicate the entire discussion in Appendix 3 in this chapter, but one example may help illustrate the process to determine the method-in-use based on the controls used. One organization had a detailed ticketing system with priorities set by management, and developers could only work on approved tickets. They had a separate quality control organization checking developer's work, and a beta customer providing feedback. Despite these controls, the raters placed this project in an almost completely ad hoc category. The discussion revealed the inadequacies of these controls. Tickets were written at a fairly high level and developers could interpret them as they saw fit. Quality control based its testing on the developer's description of what they had just built – not on a statement of requirements. Integration and regression testing had not been conducted for many months. There were no coding standards and no peer reviews. There was no time schedule on the tickets, and the developers had no pressure to deliver a ticket on time. The product architecture and product vision appeared to be missing. The beta customer was not actually using the product – they just saw regular demos. In fact, the

beta customer had already decided not to buy the product 'even though they did like it'. No one was really monitoring the overall development of the effort. As the senior developer demonstrated the product for the beta customer, they would find themselves surprised by the changes and decisions made by the other developers. In effect, the aforementioned controls (ticketing, QC) were akin to rituals (Ouchi 1980) – they may have helped the organization feel like it was doing something, but they provided little actual control.

As can be seen by this example, determination of the type of method-in-use could not be accomplished simply by tallying the numbers of controls. Instead the raters needed to consider the affect of the controls, and how well the controls appeared to work in guiding the developers.

Once a project's methods were determined, the raters used relative comparisons of projects mentioned by the interview subjects to understand how project differences related to study hypotheses. For example, consider the study shown in Figure 8 below. In this study, the interview subject talked about two projects: P0 (the current project) and P1 (another project from their experience). Contrasting these two projects allowed the raters to understand changes that occurred as projects became less plan-driven and more controlled-flexible. It also provided some hints of the differences between ad hoc and controlled-flexible, but the aforementioned relationship was the primary force at work in this instance. This comparison was not merely inference by the rater; the interview

subjects were asked directly to consider why one project was treated in one way, while the other was treated differently.

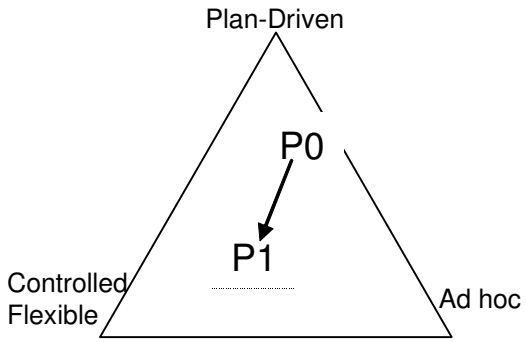


Figure 8: Relative project rating

Based on the project comparison and the codings, the raters then determined whether the interview supported the study hypotheses. This was shown graphically by placing marks on a copy of the study model as shown in Figure 9. In this example, there was strong support for the negative link between uncertainty and plan-driven methods. The lighter check at the bottom indicates weaker support for a positive relationship between uncertainty and ad hoc methods. A red X would have been placed on the model if the transcript revealed disagreement with the model.

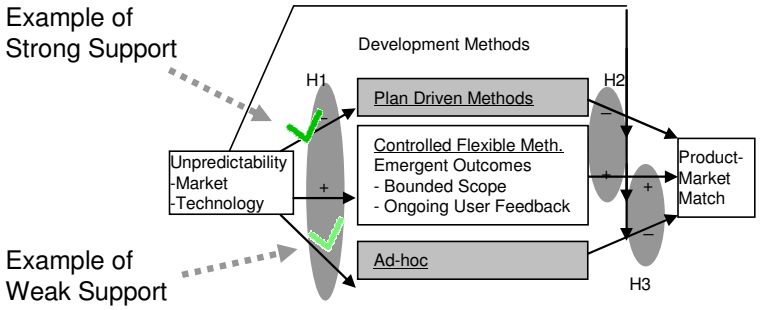


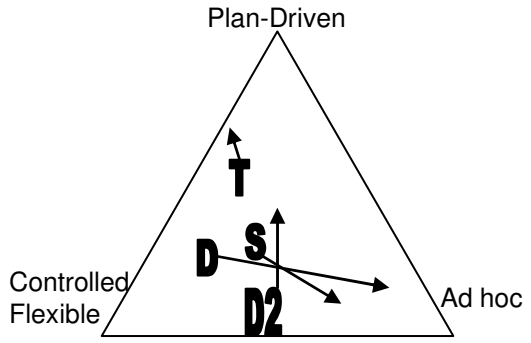
Figure 9: Example of graphical depiction of interview correspondence with model

In addition to the graphical summaries shown above in Figure 8 and in Figure 9, the raters were asked to support their conclusions with notes from the coding and the raw interviews. These notes were also used to note additions to the model or points of interest in the findings. A completed example of a rater's summary of an interview is shown in Appendix 8.

Phase III of Data Reduction: Consolidation

As revealed in Table 8, all of the projects coded had multiple informants (One uncoded project had a single informant, and another had two informants in a single interview). The final step in the analysis involved summarization and comparison of the findings. The individual ratings from team members were consolidated into a summary for the team. Figure 10 shows a graphical summary of the consolidated view of the project. In this example, the Director ("D" on the chart) described a set of controls that were indicative of a controlled-flexible environment; however, the director's perception (as indicated by an arrow from the "D") was that the project was more ad hoc. This chart

allowed comparison of the same project and its controls across team members. As a result the views of various informants could be compared and cross-checked. In addition to the project type graphics, the hypothesis support was summarized in a model similar to Figure 9, above. These two summaries and the contrast between informants highlighted key areas of agreement and reasons for differences. The summaries were reviewed and discussed by a panel of three researchers who have previous development experience. This review process was used to determine the most important trends from the analysis. The complete set of summaries are in Appendix 9.



Letters indicate coders; judgment based on controls
D: Director
T: Tech Manager
S: Sr Developer
D2: Developer
Arrows indicate subject's perception

Figure 10: Summary of a team's projects

USING COMPOSITE MAPS

Another, more abstract way of looking at the summary of projects is through a composite map. For example, consider Figure 11 below. This figure represents the composite codings for the interviewed team members of the operations product team.

Four team members belonging to this team were interviewed: PM – product manager; S – senior developer; D – developer; and Jr – junior developer. Each interview was analyzed independently by two raters to determine the method described by the interviewee. The initials shown on the chart indicate the combined judgment of both raters for a given interviewee. The oval then maps the area covered by all interviewees involved in the project. It should be noted that, even though team members are all from the same project environment, they will not have the exact same rating. Rating differences between team members can come from two sources. First, the two team members may face different control environments. For example, a senior developer might be given more latitude than a junior developer. Next, even if two identical developers are interviewed there will be natural errors in the process will lead to variability in the ratings. These errors may be due to 'real' differences. For example, a chance encounter with a manager can lead to additional controls for one team mate. Or they may be due to the study process wherein a given statement is overlooked in the interview or rating process.

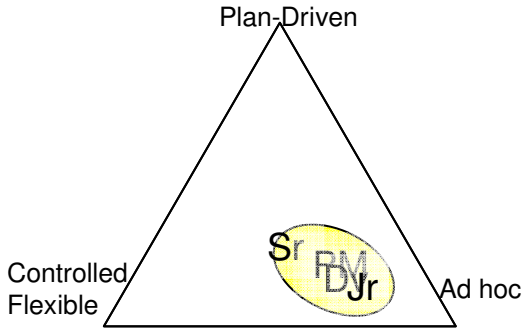


Figure 11: Composite map of method used by custom group

This oval shown in Figure 11 is the composite map that shows the method-in-use described by the team as a whole.

STUDY VALIDITY

There are four types of validity to be concerned about: construct validity, internal validity, external validity, and reliability.

- Construct Validity: Two techniques have been suggested for establishing construct validity. One is to use multiple sources of evidence to allow triangulation. In the current study, this is accomplished in two ways. First, the development of the constructs is derived from two completely separate sources. The first source is the existing process literature. As reported in Chapter 3, the literature provides a source of embedded knowledge on managing flexible processes. It was used to train control theory on the constructs of flexible systems development. This analysis was triangulated through data collected via case study interviews. The case study interviews used a multiple informant approach to also support triangulation. The control viewpoint is likely to differ between project leaders and participants. Additionally, construct validity is supported by having a transparent chain of evidence that clearly documents the source of the findings.
- Internal Validity: The concern of internal validity is establishing a casual relationship. Several steps are used to establish this. First, the study considers alternative explanations to the study hypotheses. Before the interview subjects were asked about specific study independent variables (uncertainty), they were asked to suggest their

own explanation for the observed differences in interview approach. Second, the interview process used a critical incident approach to compare the current project to previous projects that are more representative of different method archetypes. This helps highlight differences that led to the observed patterns. Finally, the multiple informants allow the researcher to uncover possible alternative explanations for study patterns.

- External Validity: This item concerns whether the results can be applied to a larger domain than the original case. In case studies, external validity is achieved through analytical generalization that is based on reasoning through the theory. This is based on considering a theoretical explanation and demonstrating that the proposed theory explains the facts better than alternative explanations. In addition, external validity can be increased through replication. This is achieved in the current study in several ways. First, interviewees were asked to compare the current project to others that they worked on that were managed differently. Many of these other projects were at other organizations. Second, the study includes case data on multiple projects, and data from multiple organizations. Third, the study is also based on data external to the interviews. The embedded logic in existing development processes is one source of data and a separate set of interviews and focus groups (see "Validating Interviews and Focus Groups" below) provide a third set of data.
- Reliability: The data analysis was cross-validated by independent coders (see Table 8: Inter-coder agreement, above).

VALIDATING INTERVIEWS AND FOCUS GROUPS

The study results were also cross-checked using secondary analysis of existing interviews. The secondary interviews come from three sources:

- A set of interviews of three small businesses that develop software products. These companies each had less than 20 developers. Two of the companies built standalone software products and the third built an embedded product that was adjunct to a non-software service. These interviews were previously used in another study (Aebischer et al. 2003).
- The interview of one small business conducted just for this study to focus on only the independent variable and alternative explanations.
- An initial focus group examining study hypothesis. The participants in this focus group were not from any of the interviewed organizations.

TRIANGULATION FOR VALIDATION

The study approach planned carefully to allow cross-checking and triangulation of results. In most cases, there were multiple informants on each project. This allowed the researchers to compare the project's control environment from multiple points of view. Not only did interviewees comment on their own circumstances, but they also reported on work relationships with others on the team. As a result it was possible to consider differences that may have been caused by personal points of view versus those that might have been caused by positional differences of project participants. Furthermore, several

groups came from a single organization. In this way, the researchers were able to hold constant organization-level factors and consider task-related differences.

This triangulation allowed for depth and cross-checking of the situation with the current project of each participant. The study also addressed breadth by asking each participant to compare their current circumstances to a more extreme (more plan-driven, more controlled-flexible, or more ad hoc) project from their previous experience. Since these extreme examples were independently suggested by each participant, they were not directly cross-validated or triangulated. However, they were anchored to the study framework by discussing them in relative terms rather than absolute terms. The alternative projects were all described relative to the base project, and the base project was carefully validated as described above.

CHAPTER 5: ANALYSIS OF RESULTS

This chapter uses the interview data to test the theoretical model. Overall, the study provided strong support for the hypotheses; however it did reveal several boundary conditions and alternative explanations. The study also provided further elucidation of the mechanisms of the proposed constructs and the ways these constructs work in practice. As a result, this work provides lessons to both researchers and practitioners. As a reminder, the model is shown in Figure 12 below.

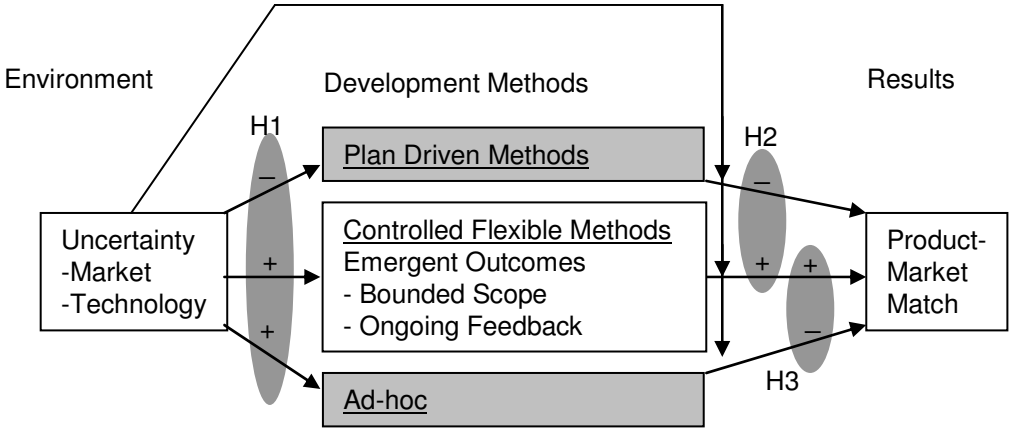


Figure 12: Theoretical model

DEVELOPMENT METHODS IN USE

The original research hypotheses focused on the links between uncertainty and method choice (H1) and between method choice and product-market match (H2 and H3). These hypotheses are defined through a comparison between the method types.

Therefore, before we can make sense of the hypotheses themselves we must consider the types of methods that exist, and what the study has to say about the differences in the method types. This section concentrates on the comparison between plan-driven, controlled-flexible, and ad hoc, and, more specifically, on what the study tells us about the use of emergent outcomes.

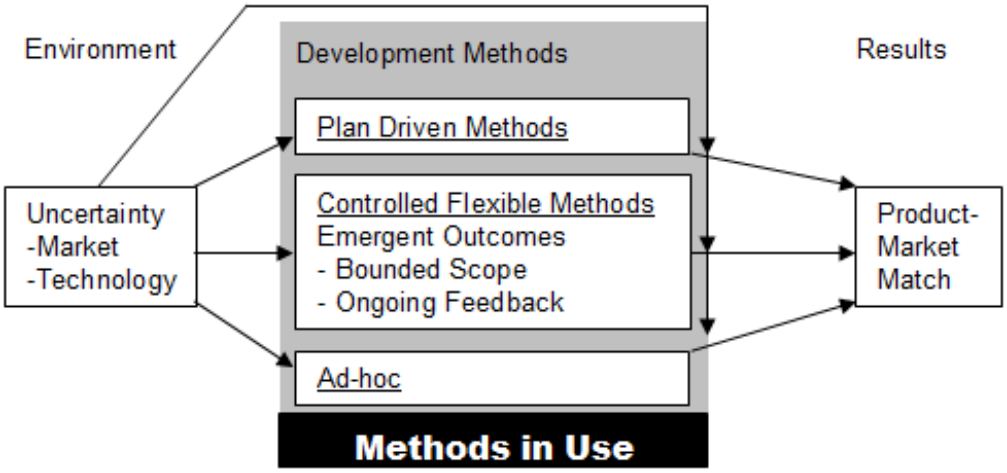


Figure 13: This section focuses on discussion of the development methods-in-use

The study model (Figure 13) shows three types of development methods: plan-driven, controlled-flexible, and ad hoc. In order to classify a given development approach as one of these methods, consideration was given to the control mechanisms in use.

The mechanisms in plan-driven approaches rely on detailed documentation of requirements before development begins. Furthermore, any changes to a plan-driven approach are formally managed through a change approval process. Therefore, changes are not automatically enacted without a formal review and approval.

In contrast, a controlled-flexible method relies on controls that manage emergent outcomes. Emergent outcome control mechanisms consist of two types of controls. The

first type of control places scope boundaries on the allowable solution. Boundary controls include mechanisms such as architectures, formal API's and other controls that bound the allowable solution set without specifying the exact details of the deliverable. The other type of emergent outcome control provides feedback for developers. Feedback mechanisms provide a continuous and ongoing information exchange among stakeholders regarding the progress of the emergent solution. The purpose of this ongoing feedback is to gather input and suggest corrective action to the emergent outcome. This can be contrasted with more traditional cybernetic feedback controls that measure progress against some predetermined goal. In the emergent case, the goal is not only *not* predetermined, it may be tacit, and only be articulated through feedback to the emergent development process.

The third development approach is ad hoc development. It is characterized by a lack of management control. Ad hoc development may include ritual control. Ritual controls have the appearance of controls, but they do little to manage the development process. For example, one company in the interviews used a detailed ticketing system and quality control process to set development priorities. However, this ticketing system did not truly control the developers. The tickets had no tie to a larger business or technical architecture, and the tickets did not contain enough details to tell developers what to build. Developers also were not held to timelines for delivery of the tickets. According to one developer, the quality control process did not tie to the ticketing process. Instead developers were asked to describe what they had built, and quality control tested to insure

that the developer delivered what they thought they delivered. Since there was no a priori agreement on what should be built, developers were never held responsible for delivering against the tickets. Finally, the resulting solutions were only infrequently integrated into a common system build, and were not subject to regression tests. As a result, these 'ritual' controls provide the appearance of control, but the raters judged that this was an ad hoc process since it did not provide any progress towards managing towards goals.

Custom Methods in Use

Although three different development types were identified, even the earliest focus group revealed that it would be difficult to find pure expressions of theses methods. None of the organizations that were interviewed used pure development methods (e.g., no pure plan-driven approaches). The development method-in-use for a given project usually contained a mix of mechanisms, and it was classified as a mixture of different methods. Because of this, it might be appropriate to think of the development method more as a range of choices rather than as a single discrete method-type. Therefore, it became more appropriate to redraw the central box in the study model as shown in Figure 14 below. This illustration indicates that methods-in-use contain mixtures of individual control mechanisms.

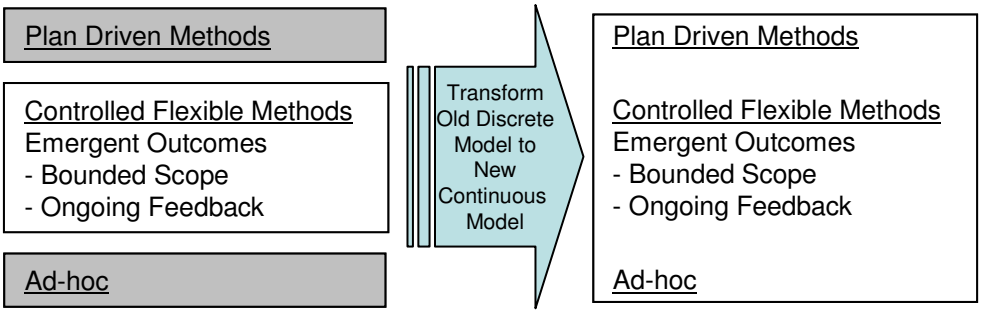
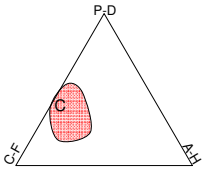
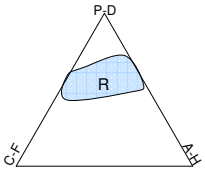
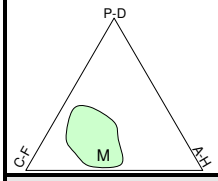
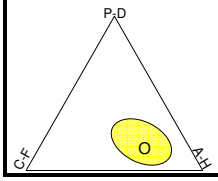


Figure 14: New continuous model of methods

Identifying the Method Used in a Project

The previous discussion indicates that adaptation of methods to individual projects can make it difficult to label the exact method-in-use for a given project. One tool used to analyze the method-in-use for a project was to create a composite map (see Chapter 4, Section titled: Using Composite Maps) that combines information from all team members. The following table describes the methods in use in the various projects and reveals the composite maps for each team.

Table 9: Evidence for methods-in-use

Large ASP	
Custom - Manager & Lead - Sr. Developer - Developer 	**Controlled-Flexible – Plan-Driven Combination** This team is building a custom solution for a business outside the U.S.. The customer doesn't understand the application needs – the developers do understand them. However, the developers don't understand country-specific issues. The customer doesn't always know the country-specific issues either, since this isn't their application expertise area. As a result, the best way to uncover issues is often to try something out and see if it works in the in-country environment. Given this situation, the developers are running as controlled a process as possible. They are pre-documenting requirements and managing to a strict set of controls. However, because there is no source of expertise, the approach can not consist of a strict plan-driven approach. The development team must allow for flexibility as they try various alternatives and learn what works and what doesn't.
Reporting - Lead - Lead II - Lead III 	**Primarily Plan-Driven** This group develops reporting applications. The team members interviewed do not work on the same projects – they are all lead developers who develop specifications for themselves and for more junior developers who work with them. The method used is more plan-driven. Many of the report requirements are known a priori, and little flexibility is required. A few details may emerge in a controlled flexible or ad hoc fashion. However, the group is often dependent on parallel development efforts in other areas. For example, a report may rely on a feed from a system that is still under development. When these parallel groups use more flexible approaches, then the reporting group is also forced to become more flexible since its input feeds and other dependencies may not be fully known until late in the delivery cycle.
Maintenance - Director - Sr. Manager - Sr. Developer - Developer 	**Flexible Development** (between controlled-flexible and ad hoc) In general, this environment is towards the flexible edge of the development methods – between controlled-flexible and ad hoc. Where possible, controls are established. However, rather than plan and control changes, the developers are more likely to try them out, and iterate through multiple market tests; therefore, it is not plan-orientated. The director and manager lean towards the left side of the composite map, and the developers are towards the right side. However all four ratings are still relatively close, and there is not enough separation to distinguish a significant difference between the management and developer levels.
Small Software Company	
Operations Product - Manager - Sr. Developer - Developer - Jr. Developer 	**Ad Hoc Development** This organization leans to an ad hoc approach. Although there are multiple types of controls, these controls do not reinforce each other. This case illustrates that a portfolio of controlled-flexible controls needs to reinforce each other to be effective. If the customer feedback, ticketing process, and QC process were self-reinforcing, then the map would lean more towards the controlled-flexible area. As it is, these mechanisms are isolated – it's is as-if someone tried to stick a single log in the river to control the flow of the river.

Intra-project Differences in Methods

The diagrams in Table 9 reveal that there are differences in the control environment perceived by team members on the same project. The controls used vary based on the seniority level and role of the individual.

> *"if you got something that even in design may still have some questions to answer, is it more likely that those will go to you … and that the [more junior] coding person will get things that are kind of very well boxed …"*

Not only the skill levels, but even the personality of the individual can affect the profile of controls in use.

"[Sometimes] I may have to spend [time] babysitting that person to make sure that .. because sometimes you have people that get distracted and I have to make sure they stay on task."

These findings were quite pervasive, with most projects reporting differences in control mechanisms based on individual being controlled. The differences are not necessarily large. As Table 9 revealed, most methods were similar across individuals. However, the variation in controls across individuals is consistent and is real.

Ad-hoc and Self-Control

Furthermore, even in the case of an ad-hoc approach, there are differences between individual situations. For example, consider this transcript excerpt:

> *We have one ad hoc that's not going to make it now. … the person who was working on it was new. And she was asking me how we do the ad hoc method.*

The implication is that, for some developers, there is substance to the ad hoc approach that must be learned. Traditional control theory states that there needs to be a socialization period in order to achieve self-control. This socialization lets the worker build attitudes and values that are congruent with the organization. However, these interviews reveal not only attitudes, but also concrete skills project skills are needed. For example, one developer discussed his habit of regularly reviewing progress with stakeholders. Interestingly, if this practice had been dictated by management, it would have been an example of a controlled-flexible mechanism. Since it was self-initiated it was not a formal control, but it had the same impact on his work as if it was instituted by management.

These ad-hoc 'skills' are further discussed in the section on Hypothesis 3, below. For now, it is sufficient to note that ad hoc development creates a vacuum regarding control so that individual differences between project team members become more important.

Comparison Projects

In addition to the subjects' reports on their core projects, they were asked to report on comparison projects that they had experienced with throughout their careers. In some cases, these comparison projects were described in detail, but the study relied most heavily on first identifying and cross-checking the position of the subject's current project, and then on a relative position of the comparison project versus the current project. Table 10 below lists the comparison projects that were mentioned. Since many of

the core projects in the study were of a more flexible nature, many of the contrasting comparison projects are more plan-driven.

Table 10: Listing of comparison projects

	Comparison Project*
Large ASP	
Custom (CF/AH*) - Manager & Lead - Sr. Developer - Developer	 - PD Large government project; AH research project - 2 AH with bad requirement definition - 2 AH with some controls
Reporting (PD) - Lead - Lead II - Lead III	 - CF adding excel module to reporting - PD and 3 AH projects - PD reporting project; AH project failure
Maintenance(CF/AH) - Director - Sr. Manager - Sr. Developer - Developer	 - PD transaction processing; - PD enhancement project, CF New Project - PD transaction processing; PD reporting - PD data conversion
Small Software Company	
Operations Product (AH) - Manager - Sr. Developer - Developer - Jr. Developer	 - 2 Large PD projects in utility industry; AH for Day Care - Large PD for state regulated industry - Large PD for deregulation of utility - 1 Person AH reporting projects
	* PD – Plan-Driven; CF – Controlled-Flexible; AH – Ad Hoc

Summary of Methods

Development methods in use may be tailored to specific situations. The needs of the project can drive different control choices. Furthermore, the staffing choices made for a project also affect the control mechanisms that are used. A view of a project as "plan-driven" or "controlled-flexible" is be too simplistic. The actual control method in use usually borrows elements from several methods, and can even vary for each individual in the project. As plan-driven controls are relaxed to allow flexibility, a project will become more ad-hoc unless emergent outcome mechanisms are implemented to achieve

controlled-flexibility. These mechanisms not only work to set bounds on the set of allowable solutions and to provide feedback on the emerging outcome, but they also work by reinforcing each other. As a metaphor, consider a team that is building a dam across a creek by driving wooden posts into the creek bed. If the posts (controls) are placed tight against one another they will work together to divert the flow of the stream. If the posts are randomly scattered throughout the stream bed, the stream will easily flow around the posts one by one.

SELECTING A DEVELOPMENT PROCESS: HYPOTHESIS 1

The first hypothesis (H1), examines how the level of uncertainty affects the control mechanisms that are used. This is illustrated in the shaded area of the model below (Figure 15).

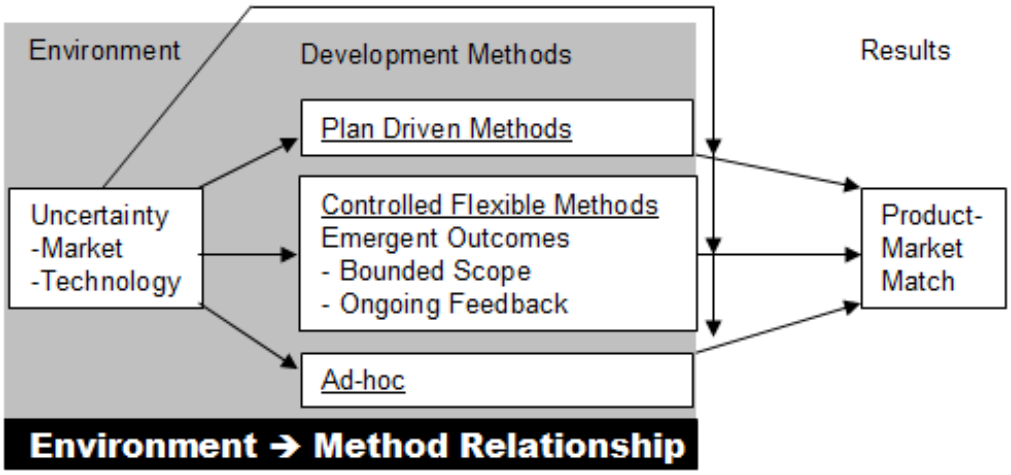


Figure 15: This section concentrates on the early portion of the model

The interview transcripts were analyzed to discover whether the techniques in use supported the proposed study hypothesis. The findings were consistent with the

hypothesis, and they revealed that an increase in uncertainty led to a more flexible approach. This is illustrated in the following table that shows how the independent raters had judged each hypothesis.

Table 11: Support for hypothesis 1

	Rater 1	Rater 2	Comments
Large ASP			
Custom (CF/AH*) - Manager & Lead - Sr. Developer - Developer	✓ ✓ ✓	✓ ✓ ✓	The transcripts included evidence for all hypothesized relationships. One interview placed a higher stress on the role of market uncertainty because the technical side was more known. Also results might differ between individuals: an individual with a more junior role may be given more predictable, plan-driven tasks. There was contra-evidence in that some project methods may not be chosen based on uncertainty, but on time pressures.
Reporting (PD) - Lead - Lead II - Lead III	✓ ✓ ✓	✓ ✓ ✓	Strong support for effect of both technical and market uncertainty on plan choice. Interesting form of technical uncertainty: this team sometimes runs plan-driven projects, but it is tripped up because another team is feeding the data to this team and is running a more flexible approach. As a result, the other team's flexibility creates uncertainty in the technical environment for this team. There was also support for the differing methods for co-workers based on their roles/levels.
Maintenance(CF/AH) - Director - Sr. Manager - Sr. Developer - Developer	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	General support. The light check for Director indicates a less complete example with less redundant evidence. The light developer check indicates a support, but also a competing explanation (time pressure). The transcript also emphasized how dependencies on other areas can affect the methods in the current area: too much flexibility in a dependent area may cause high uncertainty in the current project.
Small Software Company			
Operations Product (AH) - Manager - Sr. Developer - Developer - Jr. Developer	✗ ✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	General support for the findings. Coder one showed a partial miss because of an organization that chose a plan-driven approach even when uncertainty existed. This contra-example occurred because the company based their choice on the existence of a standard methodology at the company instead of on the characteristics of the project
Web Development	Not coded, but it includes clear evidence of a contra-example to the hypothesis. This company uses plan-driven approaches for web development because of a need to minimize costs.		
* A green check mark indicateds support for a hypothesis, and a red X indicates evidence against the hypothesis. When checks or X's are light it indicates that the data is weaker in its support – there may be some ambiguity.			

The support was consistent and strong for the study hypotheses. Following are typical statements.

> *The controlled-flexible … was probably driven by … the inability to get current customer requirements because the whole industry was entering a new facet. The one that was plan-driven, we had a much better idea ... well actually, it was an existing product that we were, you know, about to make a major enhancement to.*

Not only marketing uncertainty decreases the chance of using a plan-driven approach:

> *Interviewer: Compared to the current [plan-driven] project … what was the reason for that [controlled-flexible] project being managed differently?*
>
> *Subject: Yeah, right. It is because [the controlled-flexible one] was dealing with new technology*

This evidence comes from statements about actual projects. Subjects were initially asked to supply their own explanation for the choice of method, and they often supplied an unprompted reference to certainty/uncertainty. In addition, subjects were asked for opinions regarding method to use on some arbitrary (not real) project. Once again, H1 was supported, and there was general support for a more controlled-flexible approach as uncertainty increased.

> *Interviewer: So, if you have uncertainty, would it lead you to use a plan-driven or not use a plan-driven [method]?*
>
> *Response: My personal preference is probably [to] use ad hoc or a controlled flexibility together.*

Although there was general support for the expected hypotheses, the interview process used a critical incident approach to uncover exceptions to the hypotheses. For example, an interviewee might be asked if they had ever used a plan-driven approach when there was uncertainty. The examples uncovered through these questions proved useful in exploring boundary conditions that describe when the proposed model is appropriate. One factor mentioned several times was the role of time pressure in selecting a development method.

The Role of Time Pressure on Selection of Development Approach

A typical example of time-pressure is shown in the following statement:

> *[we couldn't take the time to figure it out up front] because the product was … was going to go live … as mandated by [the government]. … they were supposed to have our product ready to collect data … we couldn't wait until after because we [would have lost data].*

This instance, and others in the interviews, describes an entirely different paradigm than the study hypotheses. The goal is to meet a market entry date with at least some minimum set of features – in this case the ability to capture transaction data. Meeting this date is more important than building a solution with a perfect product-market match. As a result, the team realizes it will need to rework parts of the solution after the initial market launch. Following is an excerpt from another group in this company illustrating this issue.

> *" … [management says] oh by the way, we need this in 35 days. So, you know, we typically then .. I mean, we have methods to try and do that kind of*

procedurally, but if .. at the extreme what you start doing is really just .. is balancing risk with quality then and cutting corners. In other words, you know, well can we cut reviews out? Do we need to do HLD or high-level design for this project. … it increases the risk … but there are certain [times] where you have to break out of the box and kind of just get something done. Especially if your CEO tells you, you will do this in 30 days."

In this case, discussion of uncertainty was not a factor. Given time, the team may have been able to figure out the feature set before development, and thus may have been able to use a plan-driven approach. However, the time required to build the plan would have delayed development until it was not possible to deliver the minimum necessary features.

In fact, a logical consideration of this issue reveals that the effect of time pressures on methodology choice is far from simple. In the examples above, the focus is on initial delivery, and it is acceptable if some features do not come until later. As a contrast, instead of a minimal feature set, pretend the goal is to deliver a complete solution in a minimum possible time. In this full-featured case a plan-driven approach may result in the shortest delivery time. Even if the goal is a partial delivery (minimum features) you can't automatically rule out a plan-driven approach. If uncertainty is low, a plan-driven approach gives the surest chance of delivering the partial feature set. Under low uncertainty, the only time when a plan-driven approach is contra-indicated is if the time pressures are so short that there is not enough time to sequentially develop a plan and to complete the development. Furthermore, if time pressures are this extreme, then the development team will also be unable to use the feedback process that is so important

to a controlled-flexible approach. As a result a team in this situation will find itself abandoning some controls and facing ad hoc processes. As the last interviewee stated, this ad hoc rush team must also accept increased risk of non-delivery or feature mis-match versus a more controlled or planned approach.

In the company quoted here, some of this risk is ameliorated because of the interviewed company's unique position in the industry – the company's developers are the technical experts in this application area. In fact, they are usually more knowledgeable than their customers, and are in fact the leading source of knowledge in the world. Even with this allowance; however, the developers do not have any special insights to customer needs beyond the core technical requirements. Furthermore, controls also help insure integration and quality of the code and these can be damaged by a no-control fast-paced development. The bottom line is that the development organization in this situation must be willing to accept any cost to achieve early market entry, including the cost associated with a risk of a mis-match between the product and the market's needs or of non-delivery.

Given this reasoning, I don't consider this issue of time pressure as an adjustment to the model. Instead, it is more of a boundary condition. The organization will only consider the impact of uncertainty on their planning if they value a high match between the product and the market needs. Also, to consider the impact of uncertainty, the organization must be willing to allow time for either the development of a plan, or the feedback and rework process of controlled development. If the organization will not

accept the time for these controls, they will find their selves using an ad hoc approach, and thus, they may be accepting a less than optimal solution (low uncertainty = plan-driven is better or high uncertainty = controlled-flexible is better).

Size and Choice of Method

Another exception indicated by the study is related to the size of the project. This was indicated by the junior developer on the operations software team. Ad hoc teams were more likely to be used for small projects. In fact, in the case of extremely small projects, an ad hoc approach may be chosen irrespective of uncertainty. For a short-duration, single-developer project it may be less costly to simply build the project and iterate through changes than it is to formally document the project and set up a formal change request process.

Standard Methodologies

A third exception occurs when an organization has a 'standard method' it uses for all development initiatives. The manager for the operations product team described a situation he had at a previous company, and he said:

> *"[the organization was] going through our normal … methodology - develop requirements, you know, a very structure[d] waterfall approach … But the organization lacked a lot of the subject matter expertise"*

Since this exception was only evidenced in this single instance, any rational or generalization must be considered exploratory, at best. However, in this case it appeared the company chose this approach, not because it was the best approach for the project,

but because it was the standard approach for all of the company's projects. Furthermore, the outcome in this example was poor, because the plan-driven approach was not able to handle the uncertainty. This example points out that companies that use 'standard' methods should still include allowance for project-specific characteristics, such as uncertainty, and that the method used should be adjusted based on these individual project characteristics.

Interdependence of Methods in Use

An additional finding was that the method used for an individual projects would be influenced by external factors and controls. For example, one organization tested a highly iterative process designed to gain feedback as the product evolved. However, the quality control group never adjusted to the process. Quality control only began to work with the system when the final release was completed. As a result no one outside the immediate development group saw any of the interim releases. Similarly, another organization described how they had to begin development without a full specification of their interfaces, since the interface systems were in development by a group that was using a flexible approach. As a result, a development team cannot simply implement a new method without the buy-in and active cooperation of all affected stakeholders – from Management to Sales to Operations.

Risk Tolerance and Cost/Risk Tradeoffs

Evidence from one additional (non-coded) company provided support for the proposed hypotheses, and explained how these factors could be traded-off in practice.

This company develops web sites for small businesses. Their interview was not formally coded since it was only about 10 minutes and it did not follow the standard interview flow; however, the interview did provide relevant text. Specifically, the company noted the uncertainty in their chosen environment, and they noted the clients' preference for a flexible approach in this uncertain environment. The problem is that neither the development company, nor the clients are willing to take the cost risk from open-ended development. In order to contain costs, they agree to a fixed contract of deliverables and a plan-driven approach. Generally, this deliverable list is agreed upon after a detailed prototyping period between the development period and the client. In effect, this supports the hypothesis, since this flexible prototype is undertaken as a precursor to the full development.

Summary of Hypothesis One

Remember hypothesis one:

> *H 1: As the market and technology become more uncertain, software development is less likely to be managed using a plan-driven method.*

This hypothesis is fully supported by the analysis. The exceptions and boundary conditions (time and size) mentioned above do not conflict with the link between more uncertainty and less use of plan-driven. Instead, they lead two new propositions:

> *1) In some instances of increasing time pressure a plan driven approach will be less likely to be used.*
>
> *2) In very small projects an ad hoc method will be likely to be used.*

However, especially in the case of the first proposition, the study of time pressures is outside the scope of this research and the affect of time pressure is more complex than this simple proposition would make it appear.

OBTAINING A PRODUCT-MARKET MATCH – PLAN-DRIVEN VERSUS CONTROLLED-FLEXIBLE: HYPOTHESIS 2

At this point we have discussed the findings in terms of the beginning of the model and the middle phase of the model. Moving across the model, the next area of consideration is the influence of the development method on a product-market match (Figure 16). Again, the interviews revealed agreement for the proposed model.

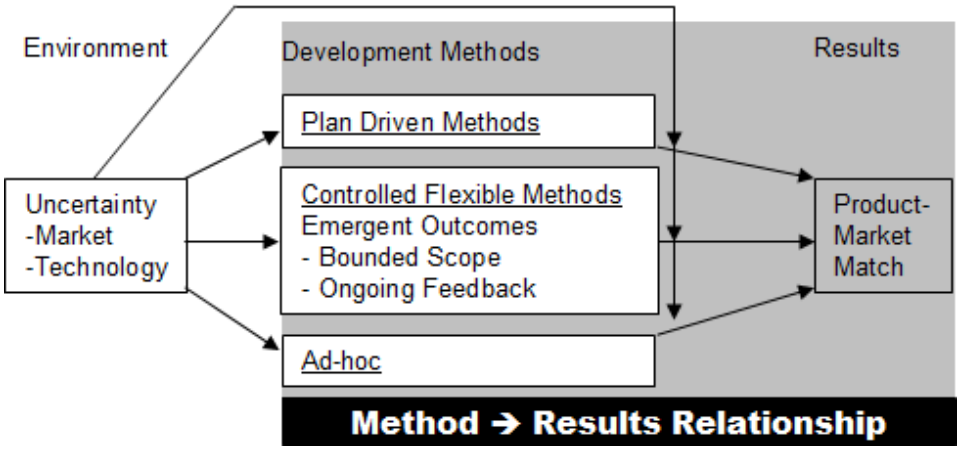
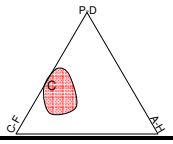
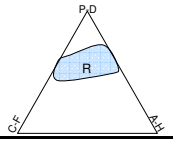
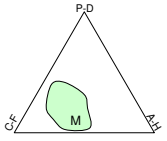
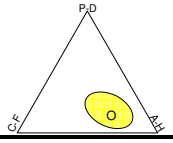


Figure 16: This section and the next section focus on the last stages of the model

The evidence from the study supports the proposed study hypotheses (H2). The findings revealed that the best approach for matching the product to the market's needs in conditions of uncertainty is the controlled-flexible approach. This is illustrated in the following table that shows how the independent coders had judged each hypothesis.

Table 12: Support for hypotheses 2

	Coder 1	Coder 2	Comments
Large ASP			
Custom (CF/AH*) - Manager & Lead - Sr. Developer - Developer 			This group did not support or weaken the argument for H2. There was not enough evidence from this group to make a determination either way. In most cases, the comparison projects were Ad hoc, and in the cases where the comparisons were plan-driven, the evidence was not sufficient to compare the product-market match against a controlled-flexible approach.
Reporting (PD) - Lead - Lead II - Lead III 	✓ ✓ ✓	✓ ✓ ✓	The transcripts for this group revealed solid evidence of the mismatch of plan-driven approaches. Use of a weaker check mark indicates one rater would have preferred more details to double check support.
Maintenance(CF/AH) - Director - Sr. Manager - Sr. Developer - Developer 	✓ ✓ ✓ ✓	✓ ✓ ✓ ✓	These transcripts revealed support for H2. In some cases, however, the significance of the support was lower. The director proved to be less well versed in the details – although the evidence was positive it was a bit light. In regard to the developer, there is some ambiguity. Both raters saw evidence of support, but the description left room for interpretation as to the degree of control of one project, therefore casting some doubt on how success was achieved.
Small Software Company			
Operations Product (AH) - Manager - Sr. Developer - Developer - Jr. Developer 	✓ ✓ ✓	✓ ✓	The group provided support for hypothesis 2. In the case of the Jr. Developer, no support was shown because only ad hoc comparison projects were mentioned. The difference in rating on the Sr. Developer comes from the treatment of intra-project comparisons. Rater 1 compares two different stages of the base project.

The first hypothesis compares a plan-driven approach and a controlled-flexible approach. As the previous section indicated, at some times, organizations do choose to handle uncertain projects with a plan-driven approach. For example, in the following instance the organization had a standard approach that they used, irrespective of the project's uncertainty.

> *… the last project [at my last company] was … a new service offering that was being developed and it was, you know, going through our normal, at the time, our normal methodology - develop requirements, you know, a very structure[d] waterfall approach. But the organization lacked a lot of the subject matter expertise … . It was identified as a risk, … but, we kept marching ahead … they ended up building the wrong thing. … We weren't successful because we had to rewrite it again after we were done.*

In the same transcript, the subject described how a team that was more flexible could work with its customers to handle change during the project instead of waiting until the end, and thus deliver a good solution.

Summary of Hypothesis 2

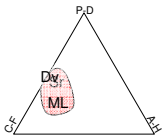
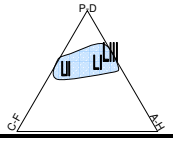
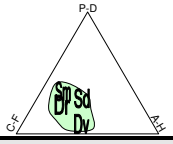
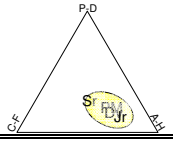
H 2: In an uncertain environment, mechanisms that manage emergent outcomes are more likely to match the market's needs than a plan-driven method.

The evidence from the study supports this hypothesis. Plan-driven approaches lock in on a version of the 'truth' early in their lifecycle. In uncertain situations this early truth is unlikely to be correct, but the plan-driven approach will not uncover the shortfalls until late in the process.

OBTAINING A PRODUCT-MARKET MATCH – AD HOC VERSUS CONTROLLED-FLEXIBLE: HYPOTHESIS 3

The model also predicts that a controlled-flexible approach will do a better job than an ad hoc approach (Figure 16), and this finding is supported by the data, but there are some important caveats to consider. These findings are summarized in Table 13, below.

Table 13: Support for hypothesis 3

	Coder 1	Coder 2	Comments
Large ASP			
Custom (CF/AH*) - Manager & Lead - Sr. Developer - Developer 	✓ ✓ ✓	✓ ✓ ✗ ✓	In general there was strong evidence of support for H3. The one case of non-support involved an ad hoc project that did match the customer's needs. However, match was achieved through multiple iterations and trials. Therefore, it was not a pure ad hoc project. This controlled-flexible type of control was the basis of the success. Furthermore, the transcript revealed that this lone control was inefficient since it led to extra costs due to higher rework.
Reporting (PD) - Lead - Lead II - Lead III 	✓ ✓	✓ ✓	The transcripts for this group revealed solid evidence of the mismatch of an ad hoc approach. The lead developer did not have ad hoc experience, so the first transcript did not show evidence for or against the hypothesis. There was evidence of more or less success in the ad hoc approach based upon the level of training/skill of the developer.
Maintenance(CF/AH) - Director - Sr. Manager - Sr. Developer - Developer 	✓ ✓ ✓ ✓	✓ ✓	These transcripts revealed consistent support for H3. Lower confidence was indicated for instances where the evidence more ambiguous. One interview stressed the importance of training to achieve proper self-control.
Small Software Company			
Operations Product (AH) - Manager - Sr. Developer - Developer - Jr. Developer 	✓ ✓ ✓	✓ ✓	This group provided evidence for the proposed poorer relationship between ad hoc and the product-market match. This interview set clearly showed an experienced developer coping with an ad hoc approach and demonstrated the additional problems that an inexperienced developer may have in the same situation.

One example of an ad hoc failure is illustrated in the following excerpt:

We have one ad hoc that's not going to make it now. … the person who was working on it was new. And she was asking me how we do the ad hoc method. So,… with the amount of delays and the lack of confidence with that deliverable, we had to pull it back. … She missed quite a few things. And based upon that...we had...the release went into production twice but had to be pulled back both times.

In addition to observations of specific projects, the general opinions were that a controlled-flexible approach was superior to an ad hoc approach.

"I go back to an ad hoc as more of an R&D type of approach. It's very unpredictable. That you have to realize that nine out of ten times you're going to throw it away because it won't match anything of what a customer is going to want, but you've had a great time playing. You've learned a lot. But, when you come out … when you go into a controlled-flexible method you have a general idea, more like an original idea of what you're trying to build, but you have approaches to adapt and more of … as the market's changing, as you're going through this time, you know, because the market doesn't stand still and so, you know, it matches quite a bit of what I've seen over the last three years that I've been working in this environment."

This is not to say that an ad hoc approach never works. Consider again the first of the two passages above. It implies that the ad hoc approach might have worked if the person had more training. In fact, throughout most of the transcripts the discussion is not so much on failure of the ad hoc approach, but it is on the special conditions that it takes to make an ad hoc approach work.

The issue of training is mentioned or implied several times. For example, one developer discussed his habit of regularly reviewing his progress with stakeholders. This

wasn't a formal control since his management didn't require it; it was simply something that he chose to do. In contrast, a more junior developer who has problem with an ad hoc approach does not know how to handle various stakeholders:

> *"The person, the management, that told me to do it this way, and QA, who says, 'Well that's not right', I'm kind of caught in the middle as the, as I say is the low man but the actual coder. I'm like, 'Which way do I go?' "*

As a result, this junior developer ends up developing once for one audience and ends up redeveloping it at the end of the process to satisfy all parties. The discussion of experience, and training, in regard to the ad hoc method suggests that the use of these controls can be voluntary and can be learned. When management controls do not exist, a skilled developer may voluntarily implement certain controls. In the language of control theory, these are not truly controls since they are not management-initiated. However from a practical point of view they guide the project just as a management-initiated control would.

Another adjustment to encourage success in an ad hoc project is to limit the size of the project. Small projects, with only one experienced developer can deliver successful outcomes in an ad hoc fashion.

All of these adjustments are forms of control that might be found in a portfolio of controls used in a controlled-flexible environment. Projects that use these adjustments are not 'pure' ad hoc, but they represent a mixture of ad hoc and a controlled-flexible approach. This illustrates an earlier point (Figure 14) that methods in use are often amalgams of different method types. Furthermore, since the success of these ad hoc

methods relies on the extra controls that are added, it is clear that projects will be more suited to the market as the development methodology becomes more controlled-flexible.

Another example of a step towards controlled-flexible was mentioned for several projects that were ad hoc, but utilized a lot of feedback and multiple iterations. A problem is that using only this control without also using other controls can be inefficient:

> *… things [were] constantly changing. [the customer] Didn't really like the placement of certain things. And literally, adding in new functionality all together. And this was on a fixed cost project. So, that didn't work in too well, especially when you're doing it for a fixed cost.*

The previous example illustrates that controls not only impact developers, but they also shape interactions with stakeholders. In this case, there were not any scope boundaries on the project. The only control was a feedback process, there was no limit on customer requests and the project was allowed to run up extra costs. This illustrates the importance of controls that work together and reinforce each other to build to a common goal.

The problem with controls that do not work together was also illustrated in the case of the operations software company that was rated as an ad hoc approach. On the surface, this company had a number of controls including customer feedback, a ticketing/priority system, and a quality control process. However, these controls were incomplete and they did not reinforce each other. For example, the quality control team did not tie its testing to the customer feedback. The work of individual developers was

not coordinated or integrated, and there was no clear technical architecture or business vision to tie the pieces together. As a result of these gaps, the organization operated as if it didn't have any controls at all.

Summary of Hypothesis Three

H 3: In an uncertain environment, mechanisms that manage emergent outcomes are more likely to match the market's needs than an ad hoc approach.

The evidence supports the expected hypothesis. The evidence is that the chance of success in an ad hoc approach can be improved if it is managed by an experienced lead developer who is capable of implementing their own controls. The evidence also points out the danger of incomplete sets of controls. One or two isolated control mechanisms may increase the product-market match at the cost of excessive cost overruns or other problems. Or isolated controls that do not reinforce each other sufficiently can result in an out-of-control project.

FEEDBACK ON MODEL

During most of the interview, subjects were blind to the study hypotheses, yet their unprompted support for the hypotheses was clear. This evidence was gathered by testing the subject's reports of experience with actual projects. At the end of each interview, subjects were shown the full research model (Figure 12). After revealing the model, the interviewer encouraged the subject to reveal any disagreements with the model. In most cases, the transcripts support for the model, as revealed in the following excerpt:

"Yeah. Unpredictable market and technology, because if you can predict the market and technology, then you have plan driven which is exactly what I said. But, if it's unpredictable and I just want to make sure I understand that. If it's unpredictable controlled-flexible, yeah, I agree with that."

However, a few subjects initially picked a plan-driven approach as the most likely to achieve a product-market match. These statements were interesting, because they invariably came at the end of a long (45 minute) interview wherein the subjects had been describing the shortcomings of a plan-driven approach in meeting uncertainty. One possible explanation is that the subjects based these statements on the need for 'social desirability', and the perception that a plan-driven approach was the proper thing to do. However, in each instance, the interviewer would clarify the response with a question similar to this:

'What about in an uncertain environment, is plan-driven still the best?'

In these cases, the subjects responded that plan-driven was best if uncertainty didn't exist, but in the case of uncertainty a flexible approach would be preferred.

These discussions of the proposed model allowed the subjects to offer their own opinions. With the caveat that these statements only applied when uncertainty existed (a necessary condition of the model) the interviewed subjects agreed with the model.

SUMMARY OF ANALYSIS FOR ENTIRE MODEL

In general, the analysis supports the full set of proposed hypotheses. One of the biggest differences is recognizing that the choice of development methods does not represent a set of three discrete choices, but instead represents a range of possibilities that

consist of tradeoffs between various control mechanisms. As such, the hypothesis wording should be adjusted as shown in Table 14.

Table 14: Revised hypotheses based on findings

Original Hypothesis	Revised Hypothesis
H1: As the market and technology become more uncertain, software development is less likely to be managed using a plan-driven method.	H1': As the market and technology become more uncertain, software development is likely to rely on fewer plan-driven control restrictions.
H2: In an uncertain environment, mechanisms that manage emergent outcomes are more likely to match the market's needs than a plan-driven method.	H2': As uncertainty in market needs and technology increases, increasing the use of mechanisms that manage emergent outcomes and decreasing the use of plan-driven mechanisms is more likely to result in product-market matches.
H3: In an uncertain environment, mechanisms that manage emergent outcomes are more likely to match the market's needs than an ad hoc approach.	H3': As uncertainty in market needs and technology increases, increasing the use of mechanisms that manage emergent outcomes are more likely to result in product-market matches.

Furthermore, the study model should be adapted as shown in the Figure 17 below. This model changes the picture for development methods from three discrete boxes to one continuous box to account for the more continuous nature of tradeoffs between the types of methods.

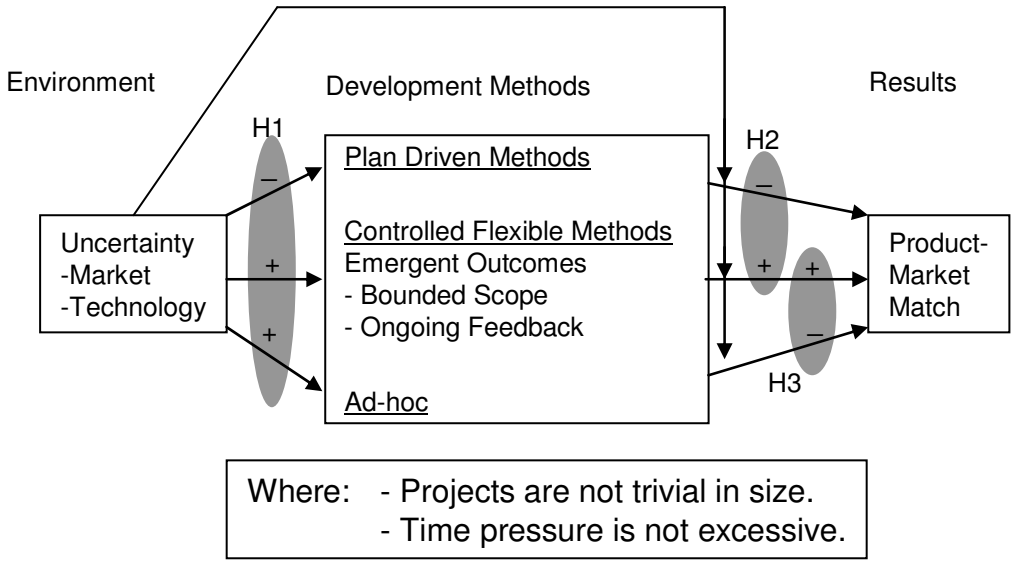


Figure 17: Revised theoretical model

The revised model also shows that two boundary conditions govern the model:

1) In some cases projects are extremely small: only one developer and the scope of effort can be measured in hours or a handful of days. In these cases, the project may be run using an ad hoc approach, irrespective of the uncertainty.

2) Some projects have extremely short timeframes, and management is willing to tradeoff controls and product features as long as the product gets to market on time. In this case, management may choose a more ad hoc approach and less of a plan-driven or a controlled-flexible approach, irregardless of the amount of uncertainty that exists. As a caveat, though, this tradeoff also brings a risk of a poor market-product match. The study findings clearly show the superior nature of a

controlled-flexible approach in an uncertain situation. Furthermore, although it was not a study hypothesis, the findings suggest that a plan-driven approach would be superior in an uncertain situation. Likewise, it is expected that a more ad hoc approach will carry more risk of technical failure whether technical uncertainty is high (controlled-flexible recommended) or low (plan-driven recommended). Management should be educated as to the presence of these risks in these circumstances.

CHAPTER 6: CONTRIBUTIONS, LIMITATIONS, AND SUMMARY

This study builds from three research streams: the literature on management control, the literature on systems development methods, and the emerging work on the dynamic capabilities extension to the resource based theory of the firm. Although this literature is not new to the study of information systems, this is the first time the three areas have been merged into a common picture focused on systems development methods. The goal of this study is to provide a language or framework for understanding when and how to apply various control mechanisms for managing dynamic development environments.

The literature on systems development methods includes many alternatives for managing systems. These approaches can be categorized into three general types: plan-driven, controlled-flexible, and ad hoc. However, this study has revealed that these 'types' may not be distinct, separate approaches. Instead they are extreme points that define a continuous space, and that any specific method-in-use can include a combination of characteristics from any one the three anchor points. This space is illustrated in Figure 18 below.

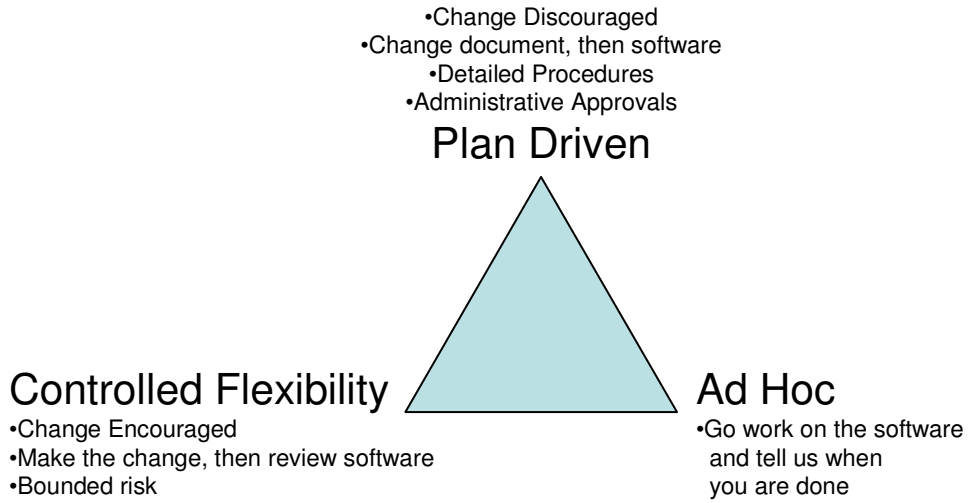


Figure 18: Methods-in-use may combine characteristics of multiple method types

The dynamic capabilities literature suggests that organizations can develop a strategic advantage by building a capability to react to a fast changing, or uncertain, environment. Organizations can develop routines that enable them to deal with these situations (Eisenhardt et al. 2000). However, this literature does not describe how these routines work. For this we must turn to the management control literature (Ouchi, 1979). The control literature lists three types of controls that can be used by management to manage projects: behavioral control, outcome control, and clan control. This study has extended the concept of outcome controls to include emergent outcome control.

EMERGENT OUTCOME CONTROLS

The first step in the analysis was to 'train' control theory on flexible development. The applicability of controls to a dynamic environment was discussed in Chapter 3. In this chapter, I compared control theory to existing development processes. The findings

indicated that control theory was applicable, but that some adjustments were needed. Specifically, control theory was not sensitive enough to the special needs of managing a dynamic environment. As a result, this study recommends the expansion of outcome control to include management of emergent outcomes. Emergent outcome controls constrain (scope limitations) and guide (feedback) activities without dictating the exact outcomes or behaviors that must be used to develop the system.

THEORETICAL MODEL

Based on these theories and the encompassing literature, a model was built to describe management of development in dynamic situations. Based on the dynamic capabilities literature, the independent variable was shown to be the unpredictability or uncertainty of either the markets needs or the technical solution. This was predicted to influence the type of method chosen, which would be either plan-driven, controlled-flexible, or ad hoc. These methods could be distinguished by the type of control used. Plan-driven would use traditional outcome controls that specify the outcome, and any changes to that specification would be required to go through a formal change management procedure that would result in modification of the control document. A controlled-flexible approach could be recognized because it would use emergent outcome controls. These are controls that constrain the scope of the solution without dictating the precise details of the solution. Emergent outcome controls also include forms of feedback that act to correct and guide the development throughout the development process. Finally, the model ties the choice of method to the product-market match. This last link

returns to the dynamic capabilities approach. The goal of developing dynamic capabilities, such as the controlled-flexible approach is to allow the organization to better react to changing situations so that the organization can meet changing needs. Based on this approach, the following model was developed (Figure 19).

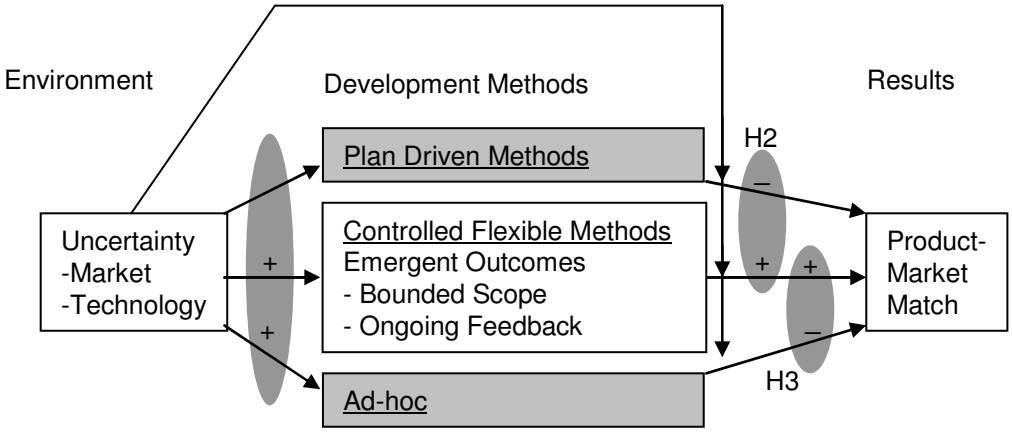


Figure 19: Original theoretical model

THE STUDY FINDINGS

Table 15: Summary of findings

Condition	Finding
When uncertainty exists	• More flexible methods are preferred. • A controlled-flexible approach is more likely to match the software to the market/user than either a plan-driven or ad hoc approach.

Characteristics of a controlled-flexible approach	• Emergent outcome controls
	  ○ Scope boundaries place limits on the amount of flexibility that exists (e.g. Software Archictecture)
	  ○ Ongoing feedback insures sure that developers receive continuous input from all stakeholders.

Other findings	• A control portfolio is needed to constrain a flexible approach. In contrast, isolated controls leave too many degrees of freedom. • When uncertainty is not the central issue (e.g. time pressure), other approaches may be recommended.

In general, the study provided strong support for the proposed model. It indicated that increased uncertainty of market needs and technology capability would lead an organization to choose a more flexible (plan-driven or ad hoc) approach. Further, the study indicated that, when uncertainty was present, a controlled-flexible approach would be more likely to provide a product that matched the needs of its users.

In defining the nature of a controlled-flexible approach, the study relied on the use of emergent outcome controls and used the presence of emergent controls as an indicator of a controlled-flexible approach. Organizations that implement only one or two controls (e.g. feedback only) are likely to see inefficiencies in terms of higher costs, or to have ineffective controls that can be side-stepped by development teams. Instead, an

organization needs to craft a portfolio of controls that carefully bound the activities and resources of the team to make sure that the resultant flexibility is contained within the desirable solution set.

However, there were two specific changes to the model suggested by the study. First, it suggested that the idea of three discrete types of methods (plan-driven, controlled-flexible, ad hoc) may be misleading. Any specific method-in-use can borrow from any of these methods, and the resulting method may be a melding of the three possibilities. Secondly, the study identified two boundary conditions. Both of these boundary conditions favor the choice of an ad hoc method more than the original model reveals. The first boundary condition relates to very small projects. If the project is very small, in size and duration it may be simpler to use an ad hoc approach and build it than it is to plan for it – even if there is no uncertainty. The second boundary condition applies to critical projects that have an extremely short timeline. In these cases, the overhead used to manage controls would cause the project to miss the need. Management may choose to use a more ad hoc approach and deliver a partial solution in the first iteration. However, this latter boundary condition does not negate the advice of the model in regard to matching the product to the market. This also means that management must tradeoff the quicker deliver with the risk of a poorer match. At the extreme one might expect a complete mismatch with the intended goals. Given these adjustments, the final model is as shown (Figure 20).

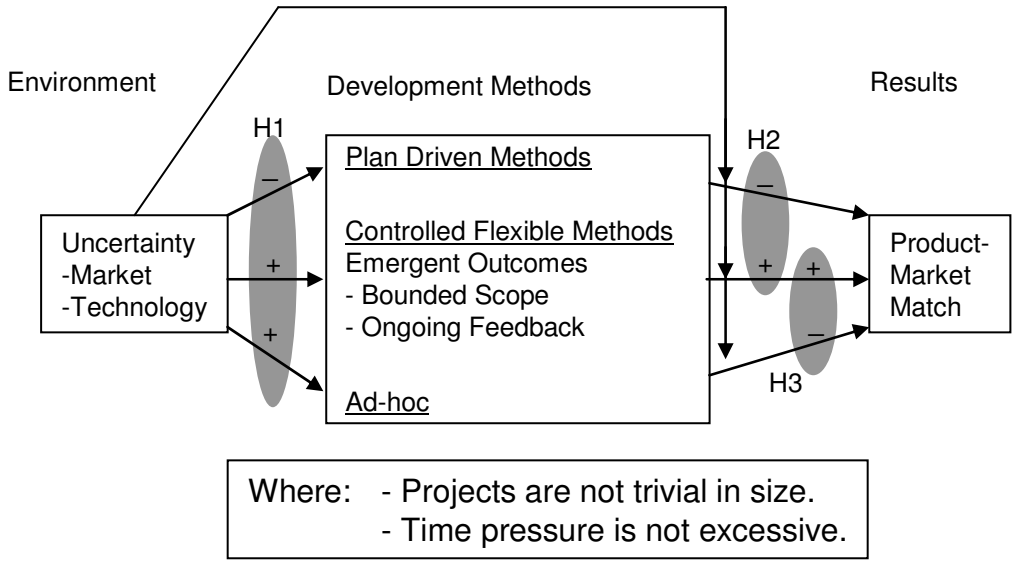


Figure 20: New theoretical model

OTHER FINDINGS

In addition, the study transcripts led to other findings regarding the development process. In general, these additional facts should be given less weight when compared to the core findings. In comparison, the core findings were the subject of a priori hypotheses. Since these core items were pre-known they were subject to careful validation techniques. Triangulation and replication were used to double check and verify the core findings. The additional findings, suggested below, were more exploratory in nature. As such, they could easily be due to spurious correlations or could be related to other factors outside the controls of the study. Therefore, these should be considered areas for future investigation, not as core findings of this study.

Other Findings: Structure of Ad Hoc

Except in the most trivial projects, the ad hoc approaches used in this sample of organizations were more structured than the literature would suggest. In the literature, ad hoc is similar to the concept of self-control. With self-control the individual is expected to hold the correct values and motivations to promote the organization's interests. However, successful ad hoc developers used techniques similar to the controlled-flexible approach. The difference is that these controls were imposed by the developer– they were not dictated by management. For example, one experienced developer chose to hold regular review sessions with their stakeholders, even though there was not a management requirement for these meetings. In contrast, a less experienced developer developed in isolation from others, and then they were surprised by at the end of the project when the software did not meet the users needs. This need for ad hoc skills is exemplified by the following comment.

> *"… the person who was working on it was new. And she was asking me how we do the ad hoc method …"*

The implication is that a 'good' ad hoc approach will use many of the controls as a controlled-flexible approach. The difference is that the decision to implement these controls is up to the developer instead of being dictated by management.

Other Findings: Ritual Controls

Ouchi (1979) talks about 'ritual' controls, and they are in evidence in this study. This occurs when management uses arbitrary controls that do not actually improve the

chance of achieving the desired outcome. Ouchi explains that this happens when management does not know what to control. Establishing rituals may help everyone feel better about the situation without actually providing positive guidance.

This is especially an important issue when relying on a portfolio of controls, similar to the controlled-flexible approach. Understand that, by design, a controlled-flexible control does not specify a solution. Instead, it maps out a space describing where the acceptable solution should sit. For example, a software architecture does not specify what to build, but it does provide guidance as to the structure of the solution. Each additional controlled-flexible mechanism further defines the space of possible solutions and further constrains the space. If the set of controls do not reinforce each other, or if they work against each other then the environment degenerates to an ad hoc approach.

As an analogy consider a project to build a series of levees to 'control' a lake in case of flooding. It would take too long to build the levee one piece of a time, so we assign ten teams and give each responsibility for a different section of the levee. Each team carefully surveys the land and places their portion in the area that they think best, but since the teams don't work together, the sections never match up and the water simply floods around the spaces between the sections.

Similarly, the controls must be linked and work together or you will end up similar to one group in our study who had numerous controls (ticketing, customer feedback, management priority setting), but who were missing a product vision,

architecture, project due dates and a plan for overall project management and integration. In total they had a set of ritual controls and an ad hoc solution.

Other Findings: Inter-dependence of Controls

Another issue is the treatment of a controlled body as a separate entity from its environment. There were many cases in the study where the team's control environment was impacted by outside areas. In one example, the team was delayed waiting on another team that controlled the incoming data feed. In two other examples, the quality control organization was not willing to adapt to a faster-moving iterative development approach. In many instances, there was no link to the sales force and to customers to provide access to feedback during the development process. Control decisions may be ineffective if they do not take into account the inter-dependencies with external entities.

Other Findings: Incorrectly Using Uniform Controls for All Projects

Several of the interviewees pointed out problems seen at previous companies. In these instances, a development initiative was managed using a standard process for the company. The problem is that a standard process does not account for differences in individual projects. Problems can occur if a standard plan-driven process is imposed on an uncertain situation.

Other Findings: Incorrectly Using Uniform Controls for All People

The issue in the previous paragraph points to a problem that can occur if all projects are treated the same. Similarly, it may not be appropriate to control all people the

same. Some developers may not work well in a flexible environment, but that does not mean they can not be on a flexible project team. The transcripts included many examples of more junior developers on flexible teams. These junior developers received pieces of the project that were better understood, and that could be managed in a more plan-driven manner within the overall project.

Other Findings: Structure of Feedback

There were some brief examples that called into question the nature of effective feedback. In one instance, a project was delivered 'live' over a web interface so customers could try it out as it evolved. This interactive feedback did a good job of engaging customers and resulted in a system that met their needs. In another case, a ongoing feedback was delivered through reports on the system progress, and in this case the team did not find out about a potential system mismatch until the system delivery was complete. Since, there was not a very large sample relating to differences in feedback, we must reiterate the exploratory nature of this observation. However, this does highlight an area for future investigations.

LIMITATIONS

This is only one study, and it faces the same threats to generalizability that are faced by any single study approach. The study design did allow for analytic generalization to test for a broader applicability of its hypotheses. For further generalizability it should be replicated, or built upon by a broader-based survey approach. Furthermore, the definition of success in the study was somewhat narrowly focused on

the product-market match. The study findings suggested that cost of development and total development time were other measures that could be used to compare outcomes in the case of development under uncertainty.

Another limitation relates to the size of the development teams. All of the teams in the study used small teams. Even in the case of larger projects at the ASP company, the initiatives were sub-divided into smaller team efforts. There is nothing in the findings to suggest that these results would not be upheld for larger groups, but there could be additional boundary conditions in the case of larger development teams.

CONTRIBUTIONS

This study presents significant contributions to both theory and practice. First, it offers recommendations to management researchers in both control theory and to the dynamic capabilities approach.

Management Researchers

In terms of control theory, the study finds that controls can be adaptable. In the past, the literature has recognized that management can use controls dynamically, in that they can change which controls they use as the controlled environment changes. In contrast, these adaptable controls are those that guide the change process without needing to be replaced with new controls. In this study I have labeled these adaptable controls 'emergent outcome controls'. Emergent outcome controls are composed of mechanisns that place boundaries on the allowable flexibility and of controls that offer ongoing feedback regarding the progress of the project.

A more tenuous, but potential contribution to control theory is in the area of self-control. The findings suggest that specific control mechanisms can be taught and can be self-administered to increase the chance of success in situations that are governed by self-control.

Finally, this research offers a contribution to the dynamic capabilities (DC) extension to resource-based theory of the firm. DC states that organizations can build the capability to manage in dynamic situations, but it does not describe what those capabilities are, nor does it describe how they should be built. This research establishes emergent outcome controls as a candidate for managing and achieving dynamic capabilities.

Information Systems Researchers

The previous contributions are in the area of management and organizational research, but this study also offers specific suggestions for information systems researchers. The findings reveal that uncertainty or unpredictability can be a motivator for choosing between system development methods. Further the study shows that emergent outcome controls can be used to guide development in these situations.

In building and evaluating processes for flexible development, this offers a framework for analysis. Questions that can be asked include: 1) How well does the process handle uncertainty? 2) Does it give the team flexibility to learn and adjust to emerging information? 3) At the same time, does it constrain the scope of possible changes or map out the allowable space for changes? 4) Finally, does it allow for

ongoing, real-time feedback from peers, customers and other stakeholders throughout the life of the project?

However, the biggest contribution to information systems researchers may not be in the specific findings offered in this study, but it may be in how the findings were generated. This model describes factors to consider when developing in uncertain environments. The study also reveals that this model is not sufficient in every case – specifically, it calls for a different approach when, instead of uncertainty, the goal is to support quick entry to the market. In fact, one could envision other situations with other key drivers (e.g., mission critical – high quality). This points out the need to consider the different drivers for making choices among development methods. In fact, different systems approaches may be 'best' depending on the need that is being addressed. Therefore, this study opens a new door in considering development methods. It calls for series of theory-based studies to consider the conditions under which various methods should be used. Each of these theory-based studies would build a model similar to that used in the current study to show how various needs lead to various method choices.

Practitioners

The study also offers a contribution to practitioners. First, the study points out that organizations should consider the unique characteristics of individual projects before choosing a method, and that they should not choose a one-size-fits-all management approach. It also points out that controlled-flexible methods are preferred methods in the case of uncertainty in market needs, or in case of technical uncertainty. When designing

controls, practitioners should consider portfolios of overlapping controls that define the proper areas for flexibility, and should allow for continuous and ongoing feedback from a broad range of stakeholders. As part of this process of defining controls, teams should recognize that the entire project does not need to be managed the same way. Some parts of the project can be plan-driven and assigned to one set of developers, while other developers handle portions of the project with more flexible needs. Finally, the study points out that an organization can forgo the use of controls in the case of extreme urgency; however, the tradeoff in these cases is that there is a higher risk of a mismatch between the product and the market. Furthermore, although it is outside the scope of this study, there is nothing about a non-controlled ad hoc process that inherently minimizes development time. Unless the team is well trained in development skills there is a risk of increased development time.

FUTURE RESEARCH OPPORTUNITIES

This study not only establishes an interesting contribution to the field, but it also points the way to several opportunities for additional research. There are several possibilities related to the core hypotheses of the current study.

- The basic study could be replicated and expanded to allow for greater generalizability.
- The boundary conditions of the new theoretical model provide opportunity for further research. When does a small project become non-trivial and, thus, require a more

formal methodology? Also, are there alternatives to an ad hoc approach under time pressure?

- The previous item points to the possibility of a range of similar research studies that examine alternative methods based on different starting assumptions: ie, instead of uncertainty as the independent variable, they would be concerned with factors such as speed of market entry, or quality-orientated development. The goal of these studies would be to come up with a descriptive theoretical model similar to the one used in this analysis. This would not be an attempt to build a new process, but rather it would attempt to understand the key drivers in existing processes.

The future possibilities above are natural extensions from the core research findings of this study. In addition, there were several exploratory findings of this study that could become the subject of new analyses in their own right.

- Study the nature of ad hoc and the meaning of self-control. This study suggested that certain skills could be taught, so that a self-control approach is not simply based on attitude, but also on the use of self-imposed controls.
- Examine the interconnectedness and structure of a portfolio of controls: what rules govern the integration of controls, and what are the characteristics of effective controls as opposed to ritual controls.
- Research the concept of utilizing different methods based on the projects needs. Some certification programs may encourage organizations to develop standard development methods. Certainly, too much customization carries risks. Every time a new process is

used there is lost efficiency due to training/learning costs. Furthermore, constant change does not allow an organization to learn from its experiences. This suggests that organizations need a portfolio of set methods, and that the specific method used can be selected based on the project's needs. In this case, there needs to be guidance regarding the shape of that portfolio. What are the different conditions that trigger selection of a different method? How many alternatives or variations are required?

- Another exploratory finding concerns the different skills and management needs of different developers. More generally, this could be expanded to address mixed-method projects where some elements are managed in a plan-driven fashion and others are managed in a more flexible fashion. Furthermore, this study would address how these different approaches are melded into an overall management approach for the entire project.
- Finally, a study could examine the nature of feedback. What constitutes effective feedback, and what is ineffective?

As this list reveals this is a rich area with significant further room for investigation. These studies offer the same advantage of the current analysis: They combine an important area of practice, with the ability to use and expand theory, and a rich base of process/method literature.

CONCLUSION

This study examined the need to manage development methods in conditions of uncertainty. It suggested that more flexible methods would be chosen when uncertainty

was the key factor driving decision making in a development project. It also suggests that the most effective methods would offset that flexibility with a portfolio of integrated controls that map out the areas of potential flexibility. If an organization interprets 'a flexible approach' to mean a 'no control' approach, then they may be making a mistake. The study offers contribution to management and information systems researchers in the form of expansion and integration of existing theories and it introduces a new concept of emergent outcome control as a technique for controlling in dynamic conditions. It also offers insight to practitioners in the form of an explanation of how flexible systems operate. This explanation is intended to help them understand the goal of flexible approaches and to help them insure they don't mis-apply those approaches in the wrong circumstances. These contributions are summarized in Table 16 below. Finally, it offers a rich set of concepts for further research.

Table 16: Summary of contributions

Audience	Contribution to Audience
Control Theory Researchers	Introduces emergent outcome control, so that control theory can handle dynamic environments more effectively.
Dynamic Capabilities Researchers	Offers control theory with emergent outcome control as an explanation of the structure underpinning dynamic capabilities.
Software Development Process Researchers	Provides a theoretical model that explains the actions of flexible processes. Points out that different models apply in various circumstances, and offers new opportunities for research through investigation of the portfolio of models that may apply in various circumstances.
Practitioners	Provides an explanation for the goals of existing flexible processes: arriving at the best market match in conditions of uncertainty. This offers two advantages. 1) It provides a better suggestion of when a flexible approach is appropriate (*why*). 2) It provides an understanding of *how* flexible mechanisms work, so that practitioners can better judge the impact of adaptations.

As a final summary, I answer the original research questions in a brief conclusion.

Why do organizations choose flexible methods instead of plan-driven approaches?

They choose flexible methods in order to deliver a good product-market match when conditions of uncertainty exist in the market and the technology.

How do the control mechanisms used in flexible methodologies influence the dynamic capabilities of a software development organization?

Emergent outcome controls are mechanisms that are used to manage flexible development. Emergent outcome controls consist of two classes of mechanisms. Scope mechanisms set boundaries that place limits on the allowed flexibility to keep it focused. Ongoing feedback is used to keep

the teams synchronized with each other and to keep them aligned with the emerging needs of stakeholders.

REFERENCES

Aebischer, K., and Harris, M. "Software Development in Small IT Firms: The Whitewater Method," AMCIS 2003, Tampa, Fl, 2003.

Auerback, C.F., and Silverstein, L.B. "Qualitative Data: An Introduction to Coding and Analysis," New York University Press, New York, 2003

Andres, H.P., and Zmud, R.W. "A contingency approach to software project coordination," *Journal of Management Information Systems* (18:3), Winter 2001, pp 41-70.

Barney, J.B. "The Resource-based Theory of the Firm," *Organization Science* (7:5) 1996, p 469.

Beck, K., and Andres, C. *Extreme Programming Explained: Embrace Change*, (Second ed.) Addison-Wesley, Boston, 2005, p. 189.

Beck, K., Beedle, M., Bennekum, A.v., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., and Marick, B. "The Agile Manifesto," 2001, http://www.agilemanifesto.org, Accessed on: 2/9/2005

Bhattacharya, S., Krishnan, V., and Mahajan, V. "Managing new product definition in highly dynamic environments," *Management Science* (44:11), November 1998, p S50.

Bhattacherjee, A. "Managerial influences on intraorganizational information technology use: A principal-agent model," *Decision Sciences* (29:1), Winter 1998, pp 139-162.

Birnberg, J.G., and Snodgrass, C. "Culture and Control: A Field Study," *Accounting, Organizations and Society* (13:5) 1988, pp 447-464.

Boehm, B., and Turner, R. *Balancing Agility and Discipline: A Guide for the Perplexed* Addison-Wesley, Boston, 2004.

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

Bourgeois, L.J., III, and Eisenhardt, K.M. "Strategic Decision Processes in High Velocity Environments: Four Cases in the Microcomputer Industry," *Management Science* (34:7), July 1988, pp 816-835.

Brown, S.L., and Eisenhardt, K.M. "The art of continuous change: Linking complexity theory and time-paced evolution in relentlessly shifting organizations," *Administrative Science Quarterly* (42:1), March 1997, p 1.

Cardinal, L.B., Sitkin, S.B., and Long, C.P. "Balancing and rebalancing in the creation and evolution of organizational control," *Organization Science* (15:4), July-August 2004, pp 411-431.

Carlile, P.R. "A pragmatic view of knowledge and boundaries: Boundary objects in new product development," *Organization Science* (13:4), Jul/Aug 2002, p 442.
Choudhury, V., and Sabherwal, R. "Portfolios of control in outsourced software development projects," *Information Systems Research* (14:3), September 2003, pp 291-314.
Crowston, K. "A Coordination Theory Approach to Organizational Process Design," *Organization Science* (8:2), March-April 1997, pp 157-175.
Crowston, K., and Kammerer, E.E. "Coordination and Collective Mind in Software Requirements Development," *IBM Systems Journal* (37:2) 1998, pp 227-245.
Cusumano, M.A., and Selby, R.W. "How Microsoft Builds Software," *Communications of the ACM* (40:6), June, 1997, pp 53-61.
Cusumano, M.A., and Yoffie, D.B. "Software Development on Internet Time," *IEEE Computer* (32:10), October 1999, pp 60-69.
Dahan, E., and Mendelson, H. "An extreme-value model of concept testing," *Management Science* (47:1), January 2001, p 102.
DeSanctis, G., and Poole, M.S. "Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory," *Organization Science* (5:2), May 1994, pp 121-147.
Diefendorrff, J.M., and Gosserand, R.H. "Understanding the emotional labor process: a control theory perspective," *Journal of Organizational Behavior* (24:8), December 2003, pp 945-959.
Eisenhardt, K.M. "Control: Organizational and Economic Approaches," *Management Science* (31:2), February 1985, pp 131-149.
Eisenhardt, K.M. "Making Fast Strategic Decisions in High-Velocity Environments," *Academy of Management Journal* (32:3), September, 1989a, pp 543-576.
Eisenhardt, K.M. "Building Theories from Case Study Research," *The Academy of Management Review* (14:4), October, 1989b, pp 532-550
Eisenhardt, K.M., and Martin, J.A. "Dynamic capabilities: What are they?," *Strategic Management Journal* (21:10-11), October-November 2000, pp 1105-1121.
Eisenhardt, K.M., and Tabrizi, B.N. "Accelerating Adaptive Proceses: Product Innovation in the Global Computer Industry," *Administrative Science Quarterly* (40), March 1995, pp 84-110.
Faraj, S., and Sproull, L. "Coordinating expertise in software development teams," *Management Science* (46:12), Dec 2000, pp 1554-1568.
Harris, M. L., Collins, R.W., and Hevner, A.R., "Controls in Flexible Software Development," *Hawaii International Conference On System Sciences,* January, 2006
Hayes, F. "$170 Million Lesson," Computerworld, March 14, 2005, http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,100335,00.html?SKC=itgovernment-100335, Accessed on: March 31, 2005
Henderson, J.C., and Soonchul, L. "Managing I/S Design Teams: A Control Theories Perspective," *Management Science* (38:6), June 1992, pp 757-777.

Hofstede, G. "The Poverty of Management Control Philosophy," *The Academy of Management Review* (3:3), July 1978, pp 450-461.
Iansiti, M. "Shooting the rapids: Managing product development in turbulent environments," *California Management Review* (38:1), Fall 1995 1995, p 37.
Iansiti, M., and MacCormack, A. "Living on Internet Time: Product Development at Netscape, Yahoo!, NetDynamics, and Microsoft," *Harvard Business School Case Study* (9-697-052), June 30 1999, p 12.
Jasperson, J., Carte, T.A., Saunders, C.S., Butler, B.S., Croes, H.J.P., and Zheng, W.J. "Review: Power and information technology research: A metatriangulation review," *Mis Quarterly* (26:4), Dec 2002, pp 397-459.
Jaworski, B.J. "Toward a Theory of Marketing Control: Environmental Context, Control Types and Consequences," *Journal of Marketing* (52:3), July 1988, pp 23-44.
Jex, S.M. *Organizational Psychology* John Wiley & Sons, inc., New York, 2002, p. 540.
Kao, E., and Schott, C. "Oracle Introduces Oracle Database 10g - The First Database Designed to Power Enterprise Grids," in: *Oracle Press Release*, Oracle, Redwood Shores, CA, September 8, 2003, http://www.oracle.com/corporate/press/2287166.html, Accessed on: April 12 2005
Karagozoglu, N., and Brown, W.B. "Time-Based Management of the New Product Development Process," *Journal of Product Innovation Management* (10:3), Jun 1993, pp 204-215.
Kirsch, L. "The Management of Complex Tasks in Organizations: Controlling the Systems Development Process," *Organization Science* (7:1), January-February 1996, pp 1-21.
Kirsch, L. "Portfolios of Control Modes and IS Project Management?," *Information Systems Research* (8:3), September 1997, pp 215-239.
Kirsch, L.J. "Deploying Common Systems Globally: The Dynamics of Control," *Information Systems Research* (15:4), December 2004, pp 374-395.
Kirsch, L.J., Sambamurthy, V., Ko, D.-G., and Purvis, R.L. "Controlling Information Systems Development Projects: The View from the Client," *Management Science* (48:4), April 2002, pp 484-498.
Kruchten, P. *The Rational Unified Process An Introduction*, (Second ed.) Addison-Wesley, Reading, MA, 2000, p. 298.
Lee, A. “A Scientific Methodology for MIS Case Studies,” *MIS Quarterly* (13:1) March 1989, pp 33-50
MacCormack, A., Kemerer, C.F., Cusumano, M., and Crandall, B. "Trade-offs between Productivity and Quality in Selecting Software Development Practices," *IEEE Software* (20:5), September-October 2003a, pp 78-85.
MacCormack, A., and Verganti, R. "Managing the sources of uncertainty: Matching process and context in software development," *The Journal of Product Innovation Management* (20:3), May 2003b, p 217.

MacCormack, A., Verganti, R., and Iansiti, M. "Developing Products on "Internet Time": The Anatomy of a Flexible Development Process," *Management Science* (47:1), January 2001, pp 133-150.
Malone, T.W. "Modeling Coordination in Organizations and Markets," *Management Science* (33:10), October 1987, pp 1317-1332.
McConnell, S. *Rapid Development* Microsoft Press, Redmond, Washington, 1996.
McDonough III, E.F., and Leifer, R.P. "Effective Control of New Product Projects: The Interaction of Organization Culture and Project Leadership," *Journal of Product Innovation Management* (3:3), September 1986, pp 149-157.
Merchant, K.A. "Progressing Toward a Theory of Marketing Control: A Comment," *Journal of Marketing* (52:3), July 1988, pp 40-44.
Neill, C.J., and Laplante, P.A. "Requirements Engineering: The State of the Practice," *IEEE Software* (20:6), November/December 2003, pp 40-45.
Nidumolu, S.R., and Subramani, M.R. "The matrix of control: Combining process and structure approaches to managing software development," *Journal of Management Information Systems* (20:3), Winter 2003-2004, pp 159-196.
Orlikowski, W.J. "Integrated Information Environment Or Matrix of Control? The Contradictory Implications of Information Technology," *Accounting, Management & Information Technology* (1:1) 1991, pp 9-42.
Ouchi, W.G. "The Relationship Between Organizational Structure and Organizational Control," *Administrative Science Quarterly* (22:1), March 1977, pp 95-113.
Ouchi, W.G. "A Conceptual Framework for the Design of Organizational Control Mechanisms," *Management Science* (25:9), September 1979, pp 833-848.
Ouchi, W.G. "Markets, Bureaucracies & Clans," *Administrative Science Quarterly* (25:1), March 1980, p 129.
Ouchi, W.G., and Johnson, J.B. "Types of Organizational Control and Their Relationship to Emotional Well Being," *Administrative Science Quarterly* (23:2), June 1978, pp 293-317.
Peters, T.J., and Waterman, R.H.J. *In Search of Excellence: Lessons from America's Best-Run Companies*, (1st ed.) Harper & Row, New York, 1982, p. 360.
Piccoli, G., and Ives, B. "Trust and the unintended effects of behavior control in virtual teams," *Mis Quarterly* (27:3), Sep 2003, pp 365-395.
Piccoli, G., Powell, A., and Ives, B. "Virtual teams: team control structure, work processes, and team effectiveness," *Information Technology & People* (17:4) 2004, pp 359-379.
Porter, M.E. *Competitive Strategy* The Free Press, New York, NY, 1980, p. 396.
Ricciuti, M. "The long march to Longhorn," in: *Perspectives*, CNET, September 10, 2004, http://news.com.com/The+long+march+to+Longhorn/2010-1016_3-5360695.html, Accessed on: April 12 2005
Robey, D. *Designing Organizations* Irwin, Homewood, Illinois, 1996.

Sambamurthy, V., Bharadwaj, A., and Grover, V. "Shaping Agility Through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms," *MIS Quarterly* (27:2), June 2003, pp 237-263.

Schumpeter, J.A. *The Theory of Economic Development*, (7th edition ed.) Harvard University Press, Cambridge, MA, 1934.

Schwaber, K., and Beedle, M. *Agile Software Development with Scrum* Prentice Hall, Upper Saddle River, NJ, 2002, p. 158.

Sheasley, W.D. "Leading technology development process," *Research Technology Management* (42:3), May/June 1999, p 49.

Takeuchi, H., and Nonaka, I. "The New New Product Development Game," *Harvard Business Review* (64:1), January-February 1986, pp 137-146.

Teece, D.J., Pasano, G., and Shuen, A. "Dynamic Capabilities and Strategic Management," *Strategic Management Journal* (18:7), August 1997, pp 509-533.

Van de Ven, A.H., Delbecq, A.L., and Koenig Jr., R. "Determinants of Coordination Modes within Organizations," *American Sociological Review* (41:2), April 1976, pp 322-338.

Veitch, M. "Oracle 10i Overtakes Users," VNU Net, July 21 2003, Accessed on: March 31, 2005

Ware, L.C. "The Benefits of Agile IT," in: *CIO Reports*, CIO Magazine 2004, http://www2.cio.com/research/surveyreport.cfm?id=74d, Accessed on: March 31 2005

Yin, R.K. *Case study research: design and methods*, (2nd ed.) Sage Publications, Inc., Thousand Oaks, CA, 1994, p. 171.

Zelkowitz, M.V., and Wallace, D.R. "Experimental Models for Validating Technology," *IEEE Computer* (31:5), May 1998, p 9.

APPENDICES

Appendix 1: Comparison of Methods Used in Control Theory Studies

The following chart shows the studies and methods that have been used to study control theory. It reveals that research on control theory relies about equally on qualitative and quantitative analysis.

Table 17: Summary of research methodologies used to study control theory

Ex*	Co*	Ql*	Qt*	Journal	Comment	Study
x				JMIS		(Andres et al. 2001)
x				Des Sci		(Bhattacherjee 1998)
			x	AcOSo		(Birnberg et al. 1988)
		x		OrgSci	Case study	(Cardinal et al. 2004)
		x		ISR		(Choudhury et al. 2003)
		x		OrgSci	16 interviews with 12 individuals. A negative case approach that involves model building simultaneous with data collection so that disconfirming models can be discovered and checked	(Crowston 1997)

* Ex= Experiment; Co=Concept; Ql=Qualitative field study; Qt=Quantitative field study

Appendix 1: (Continued)

Table 17: (Continued)

Ex*	Co*	Ql*	Qt*	Journal	Comment	Study
		x		IBM	Given that that the tasks in requirements analysis are information based, they adopted the view that organizations are information processors. Based on this and the recommendations of March and Simon they used three methods for data collections to uncover issues 1) interviewing individuals, 2) examining documents, and 3) observing individuals at work. They relied most heavily on semi-structured interviews.	(Crowston et al. 1998)
			x	Mgmt Sci	Survey of 64 stores – some with multiple informants	(Eisenhardt 1985)
			x	Mgmt Sci	Survey	(Faraj et al. 2000)
			x	Mgmt Sci	Survey approach using key informants to gather information on project teams.	(Henderson et al. 1992)
	x			AMR		(Hofstede 1978)
		x		MISQ	Metatriangulation and crucial warp thread.	(Jasperson et al. 2002)
	x			J.Mktg		(Jaworski 1988)
			x	OrgSci		(Kirsch 1996)

* Ex= Experiment; Co=Concept; Ql=Qualitative field study; Qt=Quantitative field study

Appendix 1: (Continued)

Table 17: (Continued)

Ex*	Co*	Ql*	Qt*	Journal	Comment	Study
		x		ISR	Since the research examined "how" and "why" a case study was appropriate. To strengthen the generalizability of the study, to produce enough data to suggest modifications to control theories and to provide empirical grounding four case studies were conducted	(Kirsch 1997)
		x		ISR	A qualitative study guided by Eisenhardt (89); Miles & Huberman (94); Strauss and Corbin (90); Yin (94). Scientific realism or soft positivism (Madill et al 2000).	(Kirsch 2004)
			x	MgSci		(Kirsch et al. 2002)
	x			MgSci	Analytical – Mathematical Proof	(Malone 1987)
		x		J Prod Innov Mgmt	used a 'critical incident methodlogy' to study control and coordination of 12 new product development projects at three organizations.	(McDonough III et al. 1986)
	x			JMrkt	Conceptual, a comment on Jaworski	(Merchant 1988)
			x	JMIS	Survey	(Nidumolu et al. 2003-2004)
		x		Acctng Mgmt& Tech	A field study. My research methodology was a contextualized, interpretive one, employing the techniques of organizational ethnography.	(Orlikowski 1991)

* Ex= Experiment; Co=Concept; Ql=Qualitative field study; Qt=Quantitative field study

Appendix 1: (Continued)

Table 17: (Continued)

Ex*	Co*	Ql*	Qt*	Journal	Comment	Study
		x	x	ASQ	Interviews & Questionnaires	(Ouchi 1977)
	x			MgtSci	Conceptual	(Ouchi 1979)
	x			ASQ	Markets, Bureaucracies, & Clans	(Ouchi 1980)
		x		ASQ	industry survey to identify the top two companies in an industry, and then used a qualitative interview approach to explore those	(Ouchi et al. 1978)
x				MISQ	Longitudinal experiment	(Piccoli et al. 2003)
x				IT&P	Experiment involving 3,4 person teams	(Piccoli et al. 2004)
			x	AmSoc Review		(Van de Ven et al. 1976)
4	6	11	9		**Total Studies**	
13%	20%	37%	30%			

* Ex= Experiment; Co=Concept; Ql=Qualitative field study; Qt=Quantitative field study

Appendix 2: Sample Interview Questions

The actual questions asked were dependent on the interview flow. This is not a fixed questionnaire. However, it is representative of the type of questions that were covered.

EXPLORATORY

Ask about their job function and current project.

Explain the following diagram. Ask for a description of the software development method in the current project based on the diagram.

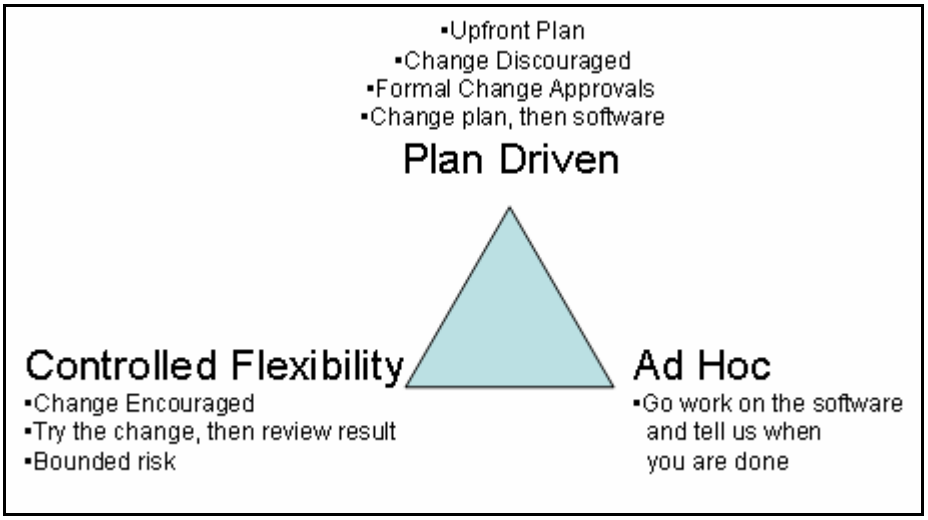


Figure 21: Type of development method

Appendix 2: (Continued)

WHY CHOOSE A DEVELOPMENT METHOD

Relative to the method used on the current project, ask about another project that used a more extreme approach (ie, more plan-driven, or more of a controlled-flexible approach)?

Why was this more extreme project managed differently?

Which project (current one or the extreme project from previous question) faced a more uncertain market condition? Uncertain technology condition? How does uncertainty affect your choice of process? What else affects your choice?

Which product provided a better match with market needs? Why?

ON THE CURRENT PROJECT AND THE ALTERNATIVE:

How well did you understand the market's needs when the project began? Has/will the market changed during development?

How well did you understand the technology and architecture when the project began? Has/will the technology and architecture changed?

How do developers know what to build?

- Who provides requirements? How complete? Specifications?
- Can developers to try new ideas?
- What keeps the developers from wandering off from the objectives?

Where do change requests come from? How are they managed?

How much are developers constrained by:

- Time limits? Architecture? Roles and responsibilities? Feature definitions? Upfront vision? Code ownership? Code Reviews? Other?

How often are builds (integration) conducted? Releases? Incremental Releases?

What are the sources of feedback for the developers?

- How do developers find out what (users/stakeholders/managers) think of the product?

How do the developers interact with one another?

Who conducts testing? When? When/who writes test cases?

How are milestones managed?

How well does the product match the needs of the users?

- In terms of changes over time are you improving features or adding features?

Appendix 2: (Continued)

OTHER

Which approach(es) do you prefer to use and why (plan-driven/ad hoc/ controlled-flexible)? Which give you more satisfaction?

Consider the following diagram. Let's discuss how well the diagram describes controlled-flexible development.

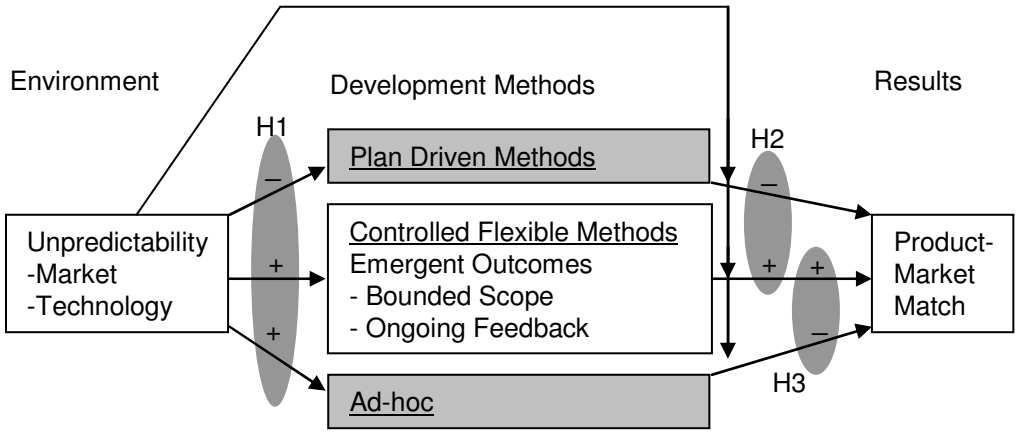


Figure 22: Development methods that can be used

Appendix 3: Distinguishing Methods Based on Controls in Use

In the interviews of developers, many types of controls and other properties of development approaches have been discussed. In order to understand whether a particular control is representative of a given type of development approach, we must have a detailed understanding of the differences in the types of development.

The following conceptual analysis is intended to dissect the differences and similarities between plan driven, controlled-flexible, and ad hoc approaches to development. For this conceptual analysis, these three terms will be treated as unique and separable concepts. In reality, most methods use some mixture of styles.

TERMS (GENERAL DEFINITIONS):

Plan Driven (PD): The plan drives all development. Changes are discouraged, and they must be approved and added to the plan before development can begin.

Controlled-flexible (CF): The 'plan' is not detailed, and ongoing change is expected. The bias is towards trying out new ideas or prototyping them and then getting feedback, rather than on trying to figure them out perfectly upfront. The developer is expected to actively lead the evolution of the product; however, the developer's freedom is bounded by controls that are used to channel their creativity.

Ad Hoc (AH): The plan is not detailed. Developer(s) are given latitude to develop as they deem appropriate, and the organization relies on the developer's desire to deliver a good product to insure that proper results are achieved.

Appendix 3: (Continued)

These categories (PD, CF, & AH) represent separate concepts, but each of them also shares characteristics with the others. If we think of these categories on a triangle (Figure 23), there are ways that PD and CF are alike and different from AH. Similarly, there are common features of PD and AH that are not features of CF. Finally, CF and AH share features that are not found in PD.

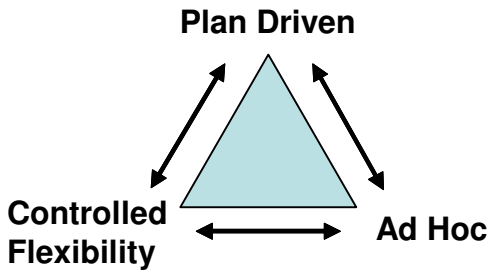


Figure 23: Differences and commonalities in methods

Controls for Plan-Driven and Controlled-Flexible Approaches

A shared characteristic of PD and CF is that they both include controls that guide and constrain developers. On the other hand AH development is generally uncontrolled. AH relies on the developer's desire to deliver good work in order to achieve acceptable results.

In general, CF controls allow flexibility for the developers, but will bound the scope of that flexibility and channel it into appropriate areas. In contrast to this bounded flexibility, the PD approach will specify the development details as thoroughly as possible in order to tightly control the outcome.

Appendix 3: (Continued)

Because of this difference (bounded flexibility versus tight specification) the way that these controls appear may be different. In PD, some controls may not be explicitly stated because they may be inherent in the specifications contained in the plan. In CF, all controls must be more explicit since the upfront design may not be detailed enough to reflect these controls.

For example, consider the architecture of the solution.

> In a PD approach the detailed specifications will be based on a planned architecture, and following the specification will result in adherence to the planned architecture. Therefore it is not necessary for the architecture to be explicitly spelled out.
>
> On the other hand, any plan in a CF approach is not likely to be detailed enough to ensure adherence to the architecture. Furthermore, CF developers are expected to create new modules not shown in the plan. Therefore, any architecture must be communicated explicitly to the developers so that they can insure their changes remain consistent with the overall technical vision.

Appendix 3: (Continued)

> In contrast, the AH approach is not constrained at all by an architecture. Developers are expected to create their own structure as they create the solution.

In addition to code-oriented controls, such as the architecture, a special concern of the CF approach is the integration of team members. Since team members are allowed to creatively explore new directions, there is a risk that their code will evolve in incompatible ways. As a result, CF methods include controls to encourage collaboration. These include approaches such as daily stand-up meetings and daily code builds. It also includes environmental changes, such as the use of a 'bullpen' that places all developers in close proximity, and the use of techniques/charts to display progress across team members. Once again, these controls may be present for a PD approach, but they are redundant if the plan is thorough.

In contrast to the PD and CF approaches, the ad hoc approach will be relatively unencumbered by controls. This is not to say that there will be no controls at all. In a true 'no control' situation, developers could go golfing every day and still collect their pay check. In fact, you could envision a situation wherein the developers were quite rigidly 'controlled', but the development was still ad hoc. For illustration, let's envision the following environment:

Appendix 3: (Continued)

> There is a strict attendance policy with developers expected to be on-time, have a specified 30 minute lunch break, and stay until a precise quitting time. Developers have a strict dress code. The organization is very hierarchical, and developers have no say in their assignments. Personal phone conversations are not allowed during working hours, and internet browsing is not allowed. Likewise, there is no personal use of email during working hours. For security reasons, no work may be taken home, and no software or content from outside the workplace may be loaded on work machines. Workers must submit a weekly written report of their work progress.

This hypothetical example describes some elements that might be used to rigidly control workers lives; however, none of these elements control the actual development of the work product. The last element (the weekly progress report) is especially telling. If workers receive feedback on their weekly progress, this could help shape the work product. However, a written report that is divorced from the actual work product, and that does not result in feedback will not impact the work product.

The key point is that a development environment that has controls can still be classified as ad hoc if those controls do not affect, or shape the work product.

Appendix 3: (Continued)

Furthermore, even in a 'pure' ad hoc environment *some* work-product related controls are likely to exist. The programmers must know what to develop. Are they developing a CRM system, a card game, or something else? Even if no specifications exist, there must be some vision, however incomplete, of the eventual goal.

Given this discussion, it is possible to use the presence or absence of controls to help classify the type of approach used. The presence of a variety of controls that shape the work product signals a controlled-flexible approach, or it indicates a plan driven approach that uses redundant controls in addition to the detailed specifications. If there are no obvious controls focused on the work product, it may be a plan-driven approach if the controls are implicit in a detailed specification. However, lacking work product controls and lacking a detailed specification, the approach is an ad hoc approach.

Communication Patterns of Plan-Driven and Ad Hoc

The PD/AH approaches are similar in the way they communicate with the customer. PD and AH are not concerned with customer feedback until the development is completed. Although communication with the customer may occur, it will be orientated towards the plan, or the customer's abstract needs, not towards the emerging software. On the other hand the hallmark of the CF approach is continuous customer communication regarding the emerging software.

PD development assumes that the customer's needs are faithfully represented by the plan, and that the team can faithfully follow the plan. During development,

developers might query the plan owners to clarify specifications. The focus is on clarifying the plan, not on demonstrating work in-progress. Similarly, AH development does not encourage interim reviews. It is assumed that the developers understand the need and can figure out the details as they proceed.

Whereas PD and AH *allow* ad hoc communications between developers and users/owners, the CF approach *requires* ongoing communication. Furthermore, the focus of that communication is quite different. PD/AH communications clarify abstract needs, whereas CF, gathers feedback on the emerging work product. Stated differently, the PD/AH focus is on feedback to the plan (a sketchy plan in the AH case), while the CF focus is on feedback on the evolving product.

In some cases, a retrospective analysis of a *failed* PD/AH project may appear to be a successful CF product. Consider the case where a PD/AH product fails to match the market's needs. During beta testing problems with product's design are uncovered, and it must be redeveloped before it is introduced. This beta test feedback may look like a CF approach, but consider the difference in the two beta approaches.

> **PD/AH Beta Test:** the beta test comes at the end of the development process. The expectation of the team is that the beta is the preliminary introduction of the product to the market. It is primarily intended to identify any problems (bugs) that are only visible in real world operating

conditions. Since the development budget is essentially spent, any feature changes will lead to cost and time over runs.

CF Beta Test: the beta test begins early in the development process and continues throughout the life of development. It is intended to gather feedback on product usability. The list of features is still open to negotiation based on the beta results. The team has planned for ongoing development cost and time based on beta feedback.

Therefore, the failed PD/AH case involves cost overruns and delays in introductions. In this case the feedback is a sign of problems. The CF feedback is an expected part of the process and does not mean that problems have occurred. In differentiating between the two situations look for either:

PD/AH: Significant development time followed by the beta, and resulting in consternation as the solution proves to have the wrong features. Cost and time overruns.

CF: Ongoing customer review beginning early in the development process, with the expectation that feedback will occur, and with plans to utilize that feedback.

Appendix 3: (Continued)

Of course, this confusion between the PD/AH cases and the CF case only occurs when PD/AH fails and if retrospective reports describe customer feedback in the middle of the project. If the PD/AH delivers as expected and on-time, then there will not be a problem in this regard.

Therefore, communication can be used to diagnosis the type of approach being used. If communication centers around clarification of the plan the approach is more PD/AH based. If communication is absent, it is more PD/AH. If there is continuous feedback focused on the emerging software, it is controlled-flexible. If the feedback comes as a surprise upon preliminary introduction, then it is more PD/AH based.

Common Features of Controlled-Flexible and Ad Hoc

The difference between the CF/AH approach and the plan driven approach is the way they treat fuzzy requirements. Both CF and AH expect requirements to be somewhat loosely defined. Developers are encouraged to work with these loose requirements to build software, whereas PD developers expect to get final detailed requirements before development begins.

Any of the development methodologies (CF/AH/PD) may have unclear requirements. However, in CF and AH it is expected that many requirements will be unable to be clarified at the planning stage, and the approaches are tailored to deal with uncertain and changing requirements. In contrast, fuzzy requirements are considered

errors in a plan driven approach. One goal of the PD approach is to start development with a detailed plan that clearly explains the program to be developed.

When faced with an unclear requirement a CF or AH developer is free to ask the user/owner for clarification. However, the approaches assume that users will not always be able to articulate all the details of the best solution. Furthermore, the user may not fully understand the capabilities of the tool used to deliver the solution. The assumption is that the developer is uniquely positioned to understand the user needs and the tool's capabilities. In the end the CF/AH developer is expected to create a solution, and to demonstrate it via working software.

In contrast, a plan driven approach assumes that the creator of the plan will deliver a complete, clear plan. It is assumed that unclear requirements will be anomalies, and when they occur, the proper course is to return to the owner of the plan for clarification. Whereas a CF/AH approach allows the option of asking for clarification, the plan driven approach requires the developer to seek clarification. Furthermore, a PD developer can not accept a partial answer. The PD developer must get a clear specification before they can begin to develop the module.

Although both CF and AH allow the developer to explore creative solutions to open issues, they are not identical in how they approach this process. The difference is in how the solution is demonstrated to the user. A controlled-flexible approach continuously reviews progress with the user; it gathers feedback incrementally throughout

development. This allows the developer to further adjust any changes that are introduced into the system. The ad hoc approach assumes that the developer's decision will be correct, and no attempt is made to gather feedback before the final product is completed.

Therefore, if developers are given loose requirements with the expectation that the developers will suggest detailed solutions, it is more CD/AH based. If specifications are detailed, and all questions are referred back to the plan owners, then the approach is more PD.

SUMMARY OF CONTROL DIFFERENCES

In conclusion, the following statements can be used to understand whether a control instance is plan driven, controlled-flexible, or ad hoc.

Plan-driven: is indicated when development is based on a detailed plan.

Controlled-flexible: is indicated when there are continuous reviews and feedback based on the evolving software throughout the development lifecycle.

Ad hoc: Is driven by a vision, and there are little other controls. There may be a set of very loosely defined requirements, and a final due date for the deliverable.

Furthermore, some control instances are characteristics of two approaches.

Plan driven & Controlled-flexible: These approaches may use explicit controls. A lack of controls indicates an ad hoc approach.

Appendix 3: (Continued)

Plan driven & Ad Hoc: Communication usually involves querying the customer for clarification. Actual demonstration of the evolving software and feedback based on the emergent software is a controlled-flexible approach.

Controlled-flexible & Ad Hoc: Developers are encouraged to creatively solve problems and develop solutions based on their knowledge of the customers and development tools. In contrast, in a plan driven approach, change is discouraged, and issues are surfaced to the plan owners who make any necessary adjustments.

Appendix 4: Notes for Coders

Each coder was given a form to fill out for each transcript. The form listed eight categories, and the coders were instructed to code the text into the appropriate category. A sample coding form is shown in a later appendix (Appendix 5). The coders determined the appropriate categorization based on the definition of the category. For example, a passage that talked about uncertainty would be placed in the category about uncertainty. There were no pre-determined categorizations – any given passage could be left uncoded if it did not fit any category, or it could be coded multiple times if it matched more than one category. In order to guide the coders in regarding to the definitions of the categories, the following definitions were provided.

JOB/ROLE

Describe the interviewee's project role. Are they involved in design? Programming? A senior developer?

PROJECTS:

The interview will focus on the developer's current duties. The current project should be described using the top row, labeled "0". In addition there will be comparison projects discussed. Each comparison project should be labeled sequentially, starting with the number "1".

Appendix 4: (Continued)

SUCCESS/FAILURE

Discuss indications of success or failure on the project. This might include how well the software meets the needs of the customers as well as the ability to meet milestones, bugs, customer satisfaction, or any other measure of success.

CONTROLS/CONSTRAINTS:

Factors that control, guide, or constrain developer. Include anything that affects what or how a developer programs. Also include specific examples that show that there is no control. Some examples of controls include:

Written plans/docs	Time limits	Project due dates	Specific API's
Architecture	Roles/responsibilities	Feature definitions	Training
Upfront vision	Code ownership	Code Reviews	Fixed DB Structure
Frequent builds	Continuous Q/A	User feedback	User Interface Rules
Manager feedback	Demonstrations for users	Peer feedback	User Requirements

METHODS:

Include both specific methods, (eg, extreme programming) and general categories, including:

- *Plan Driven Development:* relies on an upfront plan. Change is discouraged. Changes must be formally approved, and the plan must be updated before the software is changed.
- *Controlled Flexibility:* Change is encouraged. Try the change, then review the result. Bounded risk. The project is controlled or kept on track using other factors than written plans. For example, user feedback, fixed API's, fixed architecture, etcetera.

Appendix 4: (Continued)

- *Ad hoc:* The developer is given a general goal and then is asked to go away and work on the software and return with a finished product. Other than the initial, broad instructions there are no means of guidance for the developer.

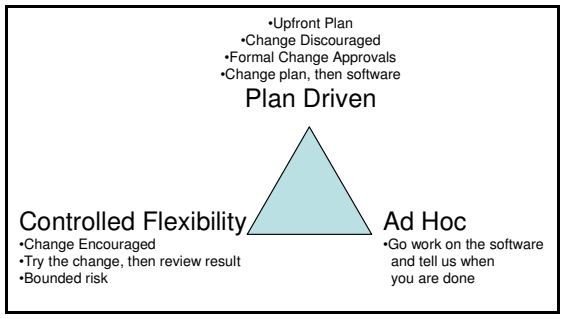


Figure 24: Methods in use

FACTORS THAT INFLUENCE WHICH METHOD IS CHOSEN:

Look for statements like this: We used a flexible approach because …

UNCERTAINTY/UNPREDICTABILITY - MARKET OR TECHNICAL:

This category is somewhat overlapping with the previous category (factors that influence which method is chosen). Oftentimes, uncertainty or unpredictability will be mentioned as an influencing factor. This category is separated to call special attention to uncertainty factors, but it is simply a specific example of a factor that might influence the method chosen.

- Market uncertainty - when the best features are unclear. There is no authoritative expert that can answer design questions. What does the market want or need?

Appendix 1: (Continued)

- Evidence includes changing requirements, customers that don't know what they want. The need to see a system to figure out exactly what they need.
- Technical uncertainty - When there are technical questions regarding development. Can I build it? Tools/software related, not process. I've never used this language before. I don't understand this DB technology. Difficulty hitting performance standards. The best technical approach and architecture may be unknown. Or, a technology may have been chosen, but the team's ability to deliver may be in doubt due to factors such as a lack of experience with the technology.
- Process uncertainty would not be uncertainty – it is a case of lack of controls.

REQUIREMENTS: A SPECIAL TRIGGER WORD

The word "Requirements" is a trigger word. When you see it, the relevant passage should probably be coded somewhere, and it may belong in more than one category. Following are some examples of the use of the word requirements.

- **Success/Failure:** If the discussion talks about what is delivered versus the original requirements than it could be an indication of success or failure. For example, 'the requirements we were using were wrong. In the end the product did not meet the needs of the customer because of this mis-match.'
- **Controls:** A requirements document is a control. Likewise the lack of any requirements from the customers could be a case of a non-control. Both controls and non-controls should be categorized in the controls section.

Appendix 4: (Continued)

- **Methods:** Discussion of requirements could highlight whether or not a plan exists. If there is a detailed requirements document and it is strictly enforced, then it is a plan-driven approach. If there are no requirements and the developers are free to developer whatever they feel is best, then the transcript may be discussing an ad hoc approach.

FACTORS THAT INFLUENCE WHICH METHOD WAS CHOSEN AND UNCERTAINTY:

The discussion may indicate the changing nature of the requirements – this demonstrates uncertainty since the team can not lock down the requirements. Likewise, it could talk about the difficulty of getting requirements and the lack of knowledge on the customer side. This is also uncertainty – since no one knows and can authoritatively state the requirements there is uncertainty regarding the true requirements.

Appendix 5: Sample Completed Coding Form

Transcript Name: ASP Reporting lead Coder: Combined/Reconciled

1) Overall Job/Role. Involved in design? Junior/Senior?

Line #	*Factor*
5-6 9-14 34-44	Associate Systems Engineer- Lead role Review requirements docs for any questions and assign the modules to code for individuals.

2) Describe projects mentioned in interview

	Line #	*Factor*
0	34-44 48-60	Reporting. Reporting arm of the company. Co-ordination between different groups involved. Customize reporting engines accordingly Enable customized/adhoc as well as automated reports
1	224-235	Adding excel reporting to their existing reporting.

3) Project Success/Failure – including product/market match

*Ref**	*Line #*	*Factor*
0	487-495	Changing requirements caused it to be pulled out of system test phase. However it is now getting signed off for production support(a month later), hence a decent product market match.

I. Controls/Constraints on Development

*Ref**	*Line #*	*Factor*
0	17-19	Business Industry Analysts' produce the requirements documents. They have a good understanding of the business and get on with the customer.
0	27-28	The whole development cycle of the type of projects he does takes 6 months on an avg. Requirements review
	48-60 48-60	Co-ordination between different groups involved is required so that they give compatible outputs. Customer may request ad hoc funstionality as well as regular automated reports, the reporting engine needs to be configured to handle those.
0	72-80	For large projects, requirements docs go well over 100 pages, and verification and validation ls worth a week's work. Revision of the specs and design its impacts on the DB; Each take a week to be done Database design /systems test would take another 3 weeks .
0	95-96	Generate data feeds from loaded in the tables.
0	101-105	Coders' teams are contracted. They translate the requirements into actual specs.
0	111-127	Requirements may be broken into and given to 2-3 contractors depending on its size.
0	135-150	The engineer has peers who may have a different idea about filling the requirements. He runs everything by his boss who steers it and does goal

	156-165	management through the specs. He has a say in the design decisions. Some output is predictable, some is not. Programmers do not have much leeway because by then the formats and outputs are already spec'd out.
0	267	Firm and specific requirements are based on the needs of one /group of lead customers who really specify .
1	224-235	They were using a COTS s/w to introduce excel reportint in the existing portfolio which meant they had constraints on how to design the system. Also, the output would be defined by that tool.
0	276-282	Reporting side needs less sticking to the standards than other groups who have more standards constraints.
0	301-313 322-324 348	There is follow up and controls to keep developers on track. However, as long as they deliver the specific deliverables in time, drifting is ok. Weekly follow ups. Formal weekly builds . (developers build several times per day…) Common build every night.
0	333-337	Sometimes the meetings are one-on-one, but there is flexibility in deciding if they'l be weekly/daily.
0	353-370	Code management team kicks off the build everyday. If something causes an error, developers are answerable to that. Developers go back the most recent build to check out the erroneous part.
0	377-379 384	Milestones for design, coding, systems tests and production. For developers there will be sub-milestones too…
0	393-396	Small coding assignments but Introducing new technology increases development time
0	401-418	Collaborative effort, since a lot of developers are contractors with no internal knowledge, and a lot of the reporting engine code is custom, a lot of communication is required. Informal communication and training between the developers
0	418-425	Contractors start working on something within a week of initial orientation. Its is at least 2-3 weeks before the become productive.
0	433-446	Individual capability and length of time they spent there defines what they will get assigned to. Assign fuzzy spec'd work to a more flexible person.
0	487-510	Sometimes clients do not figure what exactly what they want right till the end.
0	516-525	Developers do not do reviews with customers, BIAs do. They have a sample report attached to the development specs. More upfront.
	572-616	Developers can be a little ad-hoc and explorative before starting to develop. If the idea is too far expensive, then some steering down is needed. Ongoing feedback is a way to keep them reined in. A controlled-flexible approach would have limits from architecture to vision to coding standards. The ongoing feedback facilitates the management of

	591-593	boundaries. Let choices of methods to the developer because of the unpredictable characteristic of the market In case of unpredictable market technology, try to get options before making the choice. (early on), later, the constraints become stronger.

Note if constraint pertained to a specific project stage, or was specific to certain individuals.

II. Choices of methods

*Ref**	*Line #*	*Factor*
0	196-210 215	90 % plan driven , 10 % controlled flex –ad hoc.
1	224-235	More controlled-flexible to ad-hoc

Include factors that lead to specific methods and factors that inhibit use of methods

III. Factors that influence which method is chosen

*Ref**	*Line #*	*Factor*
0	199-202	Plan driven because these are well defined day-to-day activities.
0	205-210 261-262	More toward controlled flex because they're looking at new ways to solve existing problems/new ways to bring technology in. This would need a lot of prototyping initially, but would fall to being plan driven once that is over. Firm and specific requirements.
1	224-235 257-258 261-262	More controlled-flexible to ad-hoc because they were using COTS s/w which they had to figure out which one would work best for them , integrate it into their system. Once they choose a package, it defined the output of their system. They have broad latitudes in controlled flex. They're bringing something new to the market.

IV. Uncertainty in market or technology

*Ref**	*Line #*	*Factor*
1	249-250	new technology and technology that was out in the market place
0	393-396	Introducing new technology increases development time
	453-458	Specific market needs leads to a plan driven approach
0	487-510	Client not figuring requirements right till the end causes a poor prod-market match and delays in production.
	550-562	Unpredictable market and technology leads to a plan driven approach. Unpredictable market and technology leads to controlled-flexible
	591-593	Unpredictable characteristic of the market. Need to explore and investigate options before making choices.

V. Overall, describe development culture for this project and organization.

*Ref**	*Line #*	*Factor*
	453-469	Better chance of getting a good market match where requirements are well

		defined. If they're not, one needs to be more flex and have the liberty to investigate different avenues.
	453-458	Specific market needs leads to a plan driven approach and then to a better match to customer needs
	463-469	He seems to prefer and promote the plan driven approach that can create a bias in his reflection. With a little bit of flexibility if needed.

Type of Coding	Total Coded	Errors
Disagreed	7	7
Agreed	45	
Total	52	7
Percent Agreement	87%	

Appendix 6: Notes for Raters

Roman numerals in the instructions refer to sections of the combined coded transcript (See Appendix 5 for a sample coding form).

1) Development Methods
    a. Use IV to place P0,P1… and add any notes to justify
    b. Verify your conclusions with section V (perception)
2) Environment
    c. Use VII, and VI for market/tech uncertainty
3) Results
    d. Use III for product/market match

For 2) and 3) above, use a check to show support for the model and a cross to show problems with support. The check and cross come in both solid and light versions. The solid indicates strong support for the hypotheses, and the light versions indicate weak support. See the example in Figure 25, below. Explain your reasoning and supporting evidence in notes. Also describe any exceptions or addendums to the model from the interview.

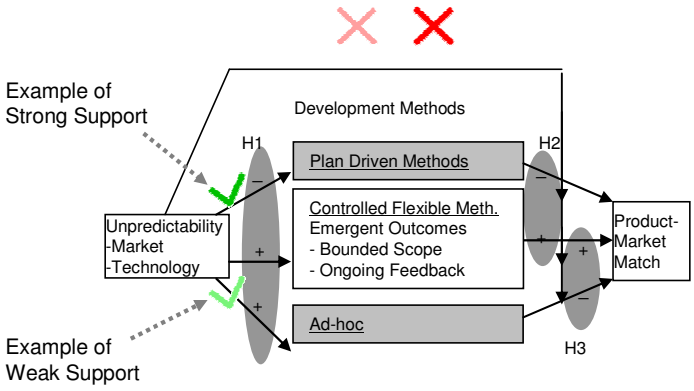


Figure 25: Sample rating summary

Appendix 7: Chain of Evidence Example

This appendix will demonstrate the data reduction effort worked by showing a few excerpts from a transcript, and discussing how the excerpts were transformed through the data reduction and analysis steps.

The steps in this analysis and data reduction were as follows:

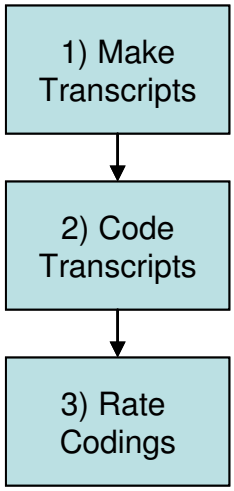


Figure 26: Steps in data reduction and analysis

Initially, the interviews were voice recorded and they were transcribed to typed text. These transcribed texts were then edited slightly to ensure confidentiality. This involved removing specific company names, product names, and the names of people, and replacing them with generic equivalents. As an illustration, "Tide Laundry Detergent" would be replaced with [product 1]. Each transcript was then given to two independent coders for categorization of the text. The coders worked with the forms, such as the one shown in Appendix 5. They were asked to categorize the passages in the text according to the categories I-VII shown in the form, and guided by the definitions shown

in Appendix 4. The form also had an eighth category, but this category was purely for informational purposes and was not formally part of the data reduction effort. This coding was completed by two independent coders and then was reconciled to arrive at a combined, coded summary for each interview. Next two independent raters used the codings to analyze the evidence versus the proposed model hypotheses. These ratings were then compared for all interviewed members in the work group and used to generate a composite rating for the entire group.

In this appendix, we will walk through an example of one section of coding for one transcript. In order to set the context, this example is extracted from the transcript of a subject in the maintenance development group at the large ASP. The total transcript was 18 pages, and it included discussion of the current focal project of the subject, and two comparison projects from the subject's previous experience. This example focuses on the analysis of the level of uncertainty that existed for one of the comparison projects.

First, let's examine the raw text of the transcript. Rather than including the entire 18 pages, we will use of foreknowledge of the passages the coders focused on to select only relevant excerpts. However, I must emphasize that the coders were given free reign in the passages to select. Based on the definitions they had to work from, they could have chosen any passages as relevant to uncertainty.

The first excerpt shows that it was not always a simple matter of looking for a key word, such as uncertainty. In the early portion of the interview, the subjects were allowed

Appendix 7: (Continued)

to suggest their own independent variables. In this example, the coders recognized that the subject was describing an uncertain situation with unknown and evolving requirements.

Excerpt One: Lines 214-234

Note: I: is the interviewer speaking. R: is the respondent.

I: *What was the difference that might have driven one of those examples to a controlled flexible and the other example to a more plan-driven?*

R: *The controlled flexible, I think, the one I'm thinking of anyway, was probably driven by, I'm going to say, the inability to get current customer requirements because the whole industry was entering a new facet, basically. What I'm talking about is the whole [P1 project], you know, that we went through, the movement that we went through a couple of years ago. That was difficult, and when I say it was difficult to get requirements, we developed requirements, but even in customer calls … this was a new [product] … it really is very difficult for [the customer] to even anticipate, you know, what kind of activity, what kind of reporting, that kind of thing. So, that was, I kind of think of that as more toward this type of development because we had to...we started off with something but we had to be pretty flexible as went through and had to, kind of, make some of our own decisions, not only from a development side, but even from a business side.*

Later in the same interview, the subjects were asked directly to compare the uncertainty of the project they were working on currently (the plan-driven project) with the comparison project (the controlled-flexible project). Although this seems a bit redundant given the previous answer, my intent as an interviewer was to make sure that the subjects interpretation of uncertainty was the same as mine.

Excerpt Two: Lines 275-279

I: *Which project faced a more uncertain market situation? The plan-driven one or the controlled flexible one?*

Appendix 7: (Continued)

R: *I'd have to say the controlled flexible one, again, only because I think, as a business and as a development shop, we had to make some decisions along the way without, you know, a lot of blessing by our customer, …*

Excerpt Three: Lines 311-314

I: *Can you think of a project you had that had some significant technology uncertainty?*
R: *Not really.*
I: *… so I think that the reason you haven't had technology uncertainty is because that technology decision's already been made by the time it gets in your hands. Would you agree with that?*
R: *That's true, and we do have a, kind of, a standard tool kit, basically that we, you know, that we tend to fit things into unless we really have a reason that we need to go outside the tool kit. So, I would agree with you there.*

Again, at a later point in the same transcript, the coders recognized a discussion of uncertainty. This section begins looking like it is a discussion of success/failure, and in fact, it was also coded under that category. But as the passage continues, it reveals a basic lack of clarity that began at the start of the project, when the customers didn't know what they needed, and resurfaced as early as a month later at the first feedback point.

Excerpt Four: Lines 627-670

I: *Have you ever had the project that just delivered the wrong thing? You got done and it went out and just, like, this is not really what we wanted in the first place. You missed it?*
R: *Well, you know what's funny is that our...the number of [P1] projects, again, let me just give you a little explanation of what...we really delivered [part of the product], like I said, we had like, basically nothing to go on. While, what little feedback we were getting … was that, they wanted more of an analysis kind of tool. … Well, probably a month into it, when we actually started [delivering], what we really found out from the customers, what they really wanted was an operational kind of report. They wanted to know, day-to-day, [information] …*
I: *Why do you think that happened?*

Appendix 7: (Continued)

R: *Again, I think it had to do with the fact that we had to make some decisions way upfront on, well, you know, "What do we think our customers need?" Because they couldn't really tell us, and they were kind of telling us this … Never once did they say, "No, that's okay, but what I'm really gonna need to know is this", and they didn't know until we really got into it…*

These transcript passages excerpted above were chosen by the coders by including them on the coding form. As was shown in Appendix 5, the full form included many different sections. For this example, we will excerpt the relevant section on uncertainty for project P1 to show how the passages were marked up by the coders.

Table 18: Excerpt of coding table

*Ref**	*Line #*	*Factor*
1	214-234	Market uncertainty because of the beginning of a new facet of the industry.
1	275-279	Market uncertainty because they were designing a new product that had very dynamic requirement specs and sometimes they had to make critical design decisions without any formal sign-offs by customers. Not much of tech uncertainty because they were familiar with the s/w-h/w, OS platform they were using pretty well.
	311-314	Reason why tech uncertainty is not a problem is that projects that come to them are filtered , and they usually have an idea of what approach to use.
1	627-670	Evolution of customers' demands after a release. The customer was unable to explain its needs at first, so the developers had to take upfront decisions.

This excerpt is from the reconciled coding. This means that it combines the initial codings of two independent coders. The excerpt includes all of the statements in one specific transcript that pertain to the level of uncertainty that existed for project P1. The column "Ref*" is used to indicate the project for these coded elements. Since this excerpt only includes statements for (P1), all of the statements have a one in the first column. The line numbers refer to the specific portion of the transcript that the coder is categorizing.

Appendix 7: (Continued)

The factor is the coder's explanation of why they are including this passage in this category. When this was handed to the raters, they first refer to the factor description to understand the type of uncertainty that is demonstrated. In case clarity is needed, or in case evidence is conflicting, the raters may also use the line numbers to refer back to the relevant portions of the transcript.

These coded transcript segments indicated the presence of uncertainty in project P1. Other potions of the analysis were used to verify that P1 was a controlled flexible approach. This information was combined with similar information on project P0, including a direct comparison between P0 and P1, in order to determine that this transcript supported the initial hypothesis that a plan-driven approach would become less likely as uncertainty increased. The raters summarized their reasoning with a brief written explanation, and by marking up a copy of the theoretical model with to show where the transcript supported (or disagreed) with the proposed hypotheses.

In this example, rater one describes the individual pieces of evidence and then concludes with the following statement:

> *"The statements fully support that, where uncertainty is low, they went with the Plan Driven approach and they had a good Product Market Match. Where uncertainty was higher they went with a controlled flexible approach and they had a good product market match."*

The support for hypothesis one was shown explicitly by marking the theoretical model with green checks, as shown in this example.

Appendix 7: (Continued)

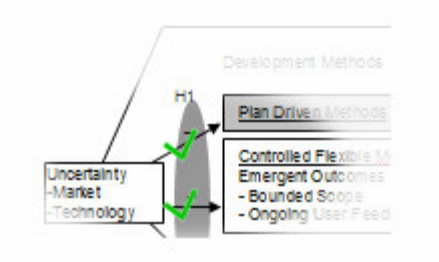


Figure 27: Excerpted model from rater one showing support for H1

Of course, the full rating process included the full coding document and resulted in markup of the entire model. In the figure above, we only focus on the portion of the model that was addressed in this example. Rater two arrived at a similar conclusion; however, this person highlighted the word "market" in their markup of the model. This is shown in Figure 28 below. The purpose of the highlighting was to emphasize that, in this case, the support for the hypothesis was based on evidence of market uncertainty, and the technical uncertainty was not a factor.

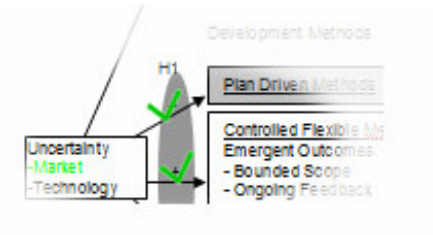


Figure 28: Excerpted model from rater two showing support for H1

CONCLUSION ON CHAIN OF EVIDENCE

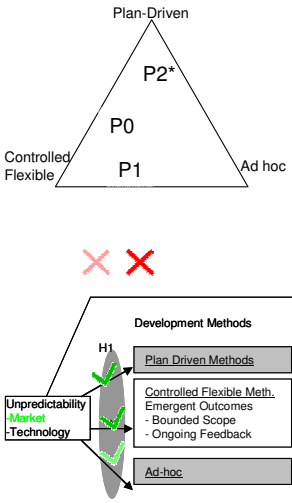
As this appendix reveals, the process working from the interview through to the resultant conclusions was transparent, and every attempt was made to reducer biases.

Appendix 7: (Continued)

During the interview, the subjects were first asked to compare projects that used different methods, and were asked why the projects needed different approaches. This was unprompted in that subjects were not offered any preconceived explanations. Subsequently, subjects were asked specifically about the role of uncertainty in their decision process. Even if the previous answer suggested uncertainty as a decision factor, the subjects were asked again to insure that interpretations were consisted. At the end of the discussion, subjects were shown the full model and encouraged to directly challenge the assumptions of the model. At the next step, coders were told what to look for (ie, uncertainty), but they were not told where to find the discussion of uncertainty. Furthermore, the coders were blind to the study hypotheses so that they would not be tempted to pick and choose which examples of uncertainty to highlight. Finally, two independent raters who had full knowledge of the hypotheses and were asked to look for evidence of support or non-support of the hypotheses. Although the raters were given full access to the transcripts, they worked first, and primarily, from the reconciled coding sheets. These coding sheets generally reduced an 18 to 20 page transcript to a two to three page summary that highlighted relevant items. This data reduction allowed the raters to focus on relationships in the data rather than on the larger mass of data. Furthermore, the summaries were designed to provide reveal both evidence in support of the hypotheses as well as evidence that might refute the hypotheses, or that might suggest boundary conditions.

Appendix 8: Sample Rating Sheet for One Rater and One Interview

ASP Maintenance Tech/RWC



Notes

Situation specific moderators? – (1) Interaction across projects also a driver of changes; shared technical environment across projects; each project has a manager but overall PM to coordinate project PMs (?). (2) nature of product – visual, difficult to do IRD

P0 – CF because of systematic customer FB; goal may be preconditioning the market; when methodological confusion, cooperative approach to resolutions; notion of fit of methods to situation; control to prevent deviation from standards via continuous interaction (not pre-set plans); task specialization by high interaction and mentoring relationships between developers, mainly informal (no code reviews); informal code reuse; formal controls on on-track to time, but plan for releases is done to prevent rush to completion; flexible toward client change requests

P1 - prototype development not plan (written documents); input from lots of sources internally, incremental development & testing; control to prevent deviation from standards via continuous interaction (not pre-set plans); initial IRD from clients incorrect (and perhaps conflicting between separate stakeholders?), so need to responsive, which the DB environment facilitated

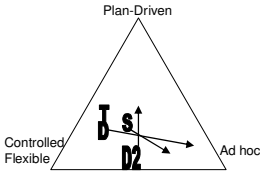
Overall on environment: 815: “most of the time we employ most of the methods”

*P2 – H1 and H2 links – good p/m match because requirements known (in contrast to P0 and P1) and plan driven (although not much detail here, only higher level comments 256-267)

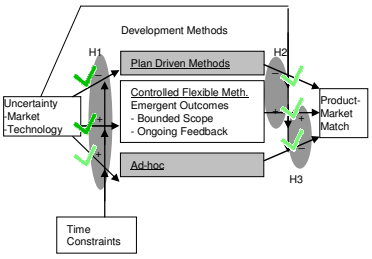
Light arrows for Ad-hoc path because of initial P1 experiences with inability to get customer requirements right at the beginning, but then able to recover without much failure because of DB environment

Note – I was a bit confused in the coding about which project CNAM belonged to – P0 or P1; used P0

Appendix 9: Summary of Interviewed Groups



Letters indicate coders; judgment based on controls
D: Director
T: Tech Manager
S: Sr Developer
D2: Developer
Arrows indicate subject's perception



Combination of All Interviews For Maintenance Interview Group

There were differences in the control environment described by the four people involved.

The director is something of an anomaly in the pattern, but during the interview it was clear that the director was not close to the operations that they controlled.

Focusing on the other three employees, we see that controls seem to be more flexible as we move down the hierarchy. There is no evidence of this in the text, but one explanation could be communication ambiguity. In other words, the Superior thinks they are being clear and definitive in directions, but the implementer finds they need to interpret them to implement them.

The interviews generally supported the hypotheses. There was strongest support for the choice of methods, and weakest support for the link to product-market match. Also these interviews did not provide as clear evidence on the issue of adhoc development, although there was implied support for the hypotheses.

Going beyond the model, four interesting areas were mentioned in the interviews.

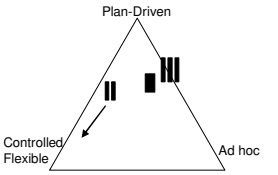
Interaction with other products: It was mentioned several times that products could affect or be affected by other products. There are two implications of this. First, the decision of development method can be impacted by things in the environment. For example, if an associated product is built in an ad hoc manner, then it could cause uncertainty in the current product, thus forcing either a controlled flexible or ad hoc approach. Second, the presence of inter-related products acts as a constraint or control on developers. Developers must take into account the impacts on these products as they shape their own initiatives.

Time Constraints: Several times it was mentioned that a more flexible approach was needed because they didn't 'have time' to figure out the requirements ahead of time. This is interesting because the flexible approaches are not explicitly focused on saving time, and their were no special schedule-orientated controls detected. The inherent agile assumption is that the exact requirements are literally unknowable as long as the software is an abstract concept, and it is only through interaction with working software that they can be finalized. With this agile view, you could take all the time you needed upfront, and still not nail down the final requirements. It may be that 'time constraints' is another way of expressing this same concept, but if not, it may be that there is a mismatch between goals and methods.

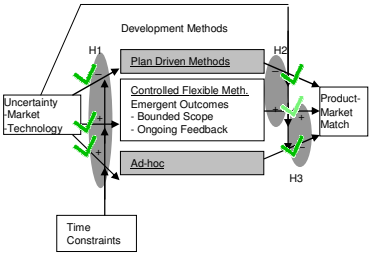
Ad hoc causes uncertainty: The model indicates that uncertainty may lead to choice of an ad hoc method, but one subject noted that an ad hoc approach causes uncertainty. This viewpoint certainly makes sense. An ad hoc approach leads to unbounded, unplanned changes. This 'surprise' changes could result in uncertainty in those who depend on the output of the ad hoc process.

Voluntary "Controls": There was evidence of the voluntary adoption of controls by the developer. For example, their were not scheduled reviews with colleagues, but the developer would regularly meet with others to discuss the emerging outcome. It would not be proper to call these controls, because controls are enacted by management to achieve desired results, whereas these are voluntarily enacted by employees. However, this is an interesting insight into self-control. The literature does not speak of activities or methods used to achieve successful self-control in a team environment. The literature implies that self-control occurs purely internal to the employee and does not describe how users can adopt techniques of their own volition to achieve acceptable outcomes.

Appendix 9: (Continued)



Letters indicate subjects' control environment
I: Lead I
II: Lead II
III: Lead III
Arrows indicate subject's perception. Note that I and III have fairly accurate perceptions



Combination of All Interviews For Reporting Interview Group

These three individuals work on the same project types, but not on exactly the same project, so the comparison between individuals is somewhat less directly comparable.

In general these are straight forward, more plan driven projects. However, the reports are often dependent on the outputs of other groups, and these outputs may be under development (uncertainty) when the report development is completed.

The overall project might take six months elapsed. It can involve three to five weeks of effort. Most of that time is spent waiting on the output data to be produced.

There is less direct evidence of the controlled flexible approach support for product market match. They tend to either plan driven, or in a few cases, ad hoc. Therefore, the light check on controlled-flexible to PM Match is not a problem, it is simply weaker evidence in the case.

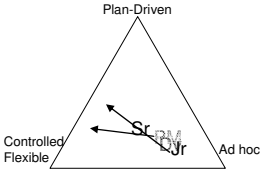
The interviews show how development can be impacted by what happens in other parts of the company and/or other areas in the project team. In this case, the uncertainty comes because they are dependent on development in other areas to give them inputs to their work.

When schedules are tight they work in parallel with the work on the systems that they are dependent on. As a result, uncertainty arises because they don't have working systems to query during development. In general; however, this is fairly plan driven.

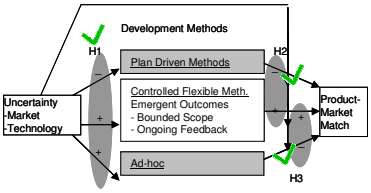
They may go more flexible because of technical unpredictability. Less likely to assume market unpredictability. They assume that requirements (from BIA?) are correct. Often they deliver and its not what the customer wanted. It may be that they assume that the BIA's are correct when they are not. This is interesting because from the requirements point of view in this company, generally the company is the expert and can dictate the details – except in this reporting area. They may not recognize that different parts of the system should be differently – ie reports should rely more on flexible interaction with end users. (see lead II)

Lead III emphasized the need for experience for success; especially in ad hoc.

Appendix 9: (Continued)



Letters indicate subjects' control environment
PM – Product Manager
Sr – Senior Developer
Dv – Developer
Jr – Junior Developer
Arrows indicate subject's perception.
Note that Sr Developer and Developer have fairly accurate perception.



Combination of all Interviews for Operations Software for Small Businesses

This group is the development arm of a small company providing operations software to nursing homes. An existing product provides leads tracking and customer care. For two years they have been working on a new billing system module. The billing initiative began with a plan-driven approach, managed by outside consultants. It failed to meet the markets needs when it was introduced. They took it in house to fix things – this phase ended inconclusively. Recently they began work with a beta customer. They received a lot of feedback from this customer, and the customer 'is happy with the current solution, but has decided not to install it'. They have identified a second beta customer and are beginning to work with them.

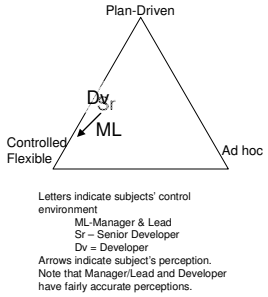
This group has the forms of control, but their actual structure is quite ad hoc. They have a ticketing/priority system that determines jobs, but don't appear to have a overall vision – they are fixing individual nits, but don't appear to be making progress towards a goal. They have a QC process, but they don't have any spec's to test against, and they don't regularly conduct integration or regression tests. There is no accountability for time schedules. There is some lesson about the differences between going through the motions and actually using controls. There is no integration and reinforcement between controls.

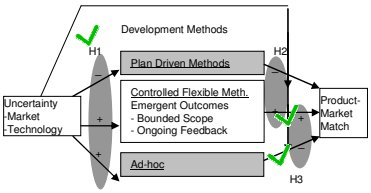
The experiences of the developer and jr dev. also point to self control lessons. The literature talks about self-control in regard to establishing values and common frameworks for decision making. This case points out that there are actual skills that need to be applied to self-control. Without these skills, well-intentioned, value-ridden approaches may lead only to frustration.

The manager thinks his group is better controlled than it is. The Sr. developer and the dev. are both experienced dev's doing the work and they are realistic about their own control environment. However, the sr. dev is in a leadership role to the others and she thinks the other's are better controlled than they are. The Jr. dev is not as realistic as the others about his own environment, but indications are that he does not have the experience to make these judgments. Taken together this suggests that experienced workers may be able to judge their own environment, but those who are removed may not understand how attempted controls are received.

The case provides good support for the model in general, but lacks evidence on the controlled-flexible path. It does indicate some boundary conditions for the model. In one instance, ad-hoc is used, not because of unpredicability, but because of simplicity. A single person project can be small enough that it is simpler to do the project and try it out than it is to use any kind of documentation or controls. In another instance, a company uses a plan-driven approach despite existing unpredictability. It may be that this is 'the way the company operates all of its projects', and no individual decision was made for this project.

Appendix 9: (Continued)





Combination of all Interviews for Custom Group

This group is developing a custom solution for a client in Brazil. The system requirements situation on this system is somewhat unusual. From a general perspective, the knowledge of the application rests with this group who is doing development. The customer does not really know what they need, they just know that they need inter-connectivity with others in the industry, and that this is the group to provide those features. On the other hand, there are some specifics to the Brazil environment that this group does not understand. Some differences in standards and some other unspecified differences. Furthermore, there are barriers for information exchange: remote settings (Brazil versus Tampa), and language (Portugese versus English) were two that were mentioned.

In general the group leans towards plan-driven as much as possible. However, they realistically know that there are things that they don't understand about the application upfront. They take the approach that the best way to understand these items is to dig in to the application and begin to develop it – this process will uncover the issues. They have a fairly good understanding of what they are doing, and a good understanding of their process. Their people are fairly senior.

The case provides good support the model. However it does not have much evidence on the link from plan-driven to poor product-market match. It does include an instance of a fairly ad hoc project that delivers a good product-market match. However, this is not pure ad hoc – it contained a strong feedback loop that was ongoing throughout the project, therefore the good match was likely based on the controlled-flexible characteristics of this ad hoc project.

This project emphasizes the method differences that may exist based on role/level of the person. The higher level people are more likely to be involved in requirements generation. As jobs are handed out, those with fuzzier requirements are given to these higher level people. Lower level people are more likely to get clear-cut requirements that are more completely spelled out.

Accepting change brings cost in terms of regression testing. May be over 2 times as much testing time as development time. May be related to ASP/Service bureau approach and embedded code

Time may drive the decision to go plan-driven versus c-f. Plenty of time, then go p-d, short schedule then go flexible.

ABOUT THE AUTHOR

Michael Harris is completing his Ph.D. in Business Administration from the Information Systems and Decision Sciences Department of the College of Business in the University of South Florida. He will graduate in August, 2006. Mr. Harris received an MBA from Emory University in 1991, and he received a BS in Industrial Engineering from the University of Missouri-Columbia in 1982. Mr. Harris has been accepted for publication in the Communications of the ACM, and he has been published in eight conferences, including the ICIS and AMCIS conferences. Mr. Harris has over 20 years of executive and systems experience in the workforce.