CHAPTER 3

Planning for Dynamic Environments

Introduction

Traditional project management for static environments is focused on the idea that a project can and should be managed primarily with a detailed project plan, but in dynamic environments over-reliance on a detail plan is considered a counterproductive, undermining performance. This chapter will explain a flavor of planning more suited to rapid change, including: emergent planning, framework plans, recursive design cycles, prototypes, pilots, and staged releases. Dynamic planning is about planning at the most appropriate times in the most appropriate ways, and then complementing the plan with appropriate control, culture, and leadership, which are discussed in later chapters.

The Traditional Approach

Project management as defined by the various bodies of knowledge is focused on a "management-as-planning" view of control (Koskela & Howell, 2002; Williams, 2004, p. 6; Johnston & Brennan, 1996), which is fine for projects with clear goals and where the methods used are well established and known (Turner & Cochrane, 1993). These projects lend themselves to detailed up-front planning and the use of the traditional *waterfall* life cycle. The *waterfall* life cycle has limited overlap between phases, meaning only minor planning revisions are done during execution. The waterfall approach involves building a detailed plan, and then using the plan as the primary control tool to manage execution. Once the plan is built and agreed upon, the original business objectives built into the plan are much less important.

For speedy projects, however, researchers like Koskela and Howell (2002) argue that traditional project management undermines performance. The problem is that circumstances change at a faster rate than it's practical

to re-plan (Sugden, 2001; Sachs & Meditz, 1979, p. 1081; Williams, 2004; Ashton, Johnson, & Cook, 1990). For instance, the business environment may change, requiring a change in objectives, or technology may leap forward, or both as happened with the Iridium satellite project. The detailed planning approach inhibits a project's ability to adjust to a changing environment (Payne & Turner, 1998). Some speculation and planning for the future is important, but excessively detailed long-term planning in these environments can waste time and resources and lead to false expectations. For dynamic environments, an emergent or learning flavor of planning and execution is more appropriate (Mintzberg & Lampel, 1999).

Dynamic Planning

Dynamic project environments are more suited to an emergent project management approach (Lewis, Welsh, Dehler, & Green, 2002) or as the *PMBOK® Guide* describes it, "progressive elaboration," where the planning detail is progressively developed as more is learned (PMI, 2013). The emergent approach does not mean you don't do proper planning. Dynamic planning works as follows:

- Initially, a high-level plan is created, with:
  a) a clearly articulated vision;
  b) project broken into stages;
  c) each stage ending with a decision gate (requiring re-planning, revised business cases, and formal review); and
  d) each stage including progressively lower levels of detail according to how far ahead one is looking.

- As the project progresses, the project is constantly reviewed (especially at stage gates) to:
  a) add feedback from one stage, along with environment changes, and changes to business objectives, to the plan for the next stage; and
  b) decide whether to proceed or kill the project.

So for instance, if your plan had three stages, each stage would have different levels of detail according to how imminent its execution was. For instance, the plan for next two weeks would have great detail, and the following stage might have less detail ("a look ahead plan" (Laufer, 1997)), and the last stage much less detail. The lower levels of detail for the future stages, as shown in Figure 3.1, allows those stages to be more flexibly adapted to changes. With this approach it is essential you learn from each stage to fill out the detail in the following stage.

Traditional projects rely on a directive style of control where a plan is developed and execution is controlled using the plan to go down a well-defined

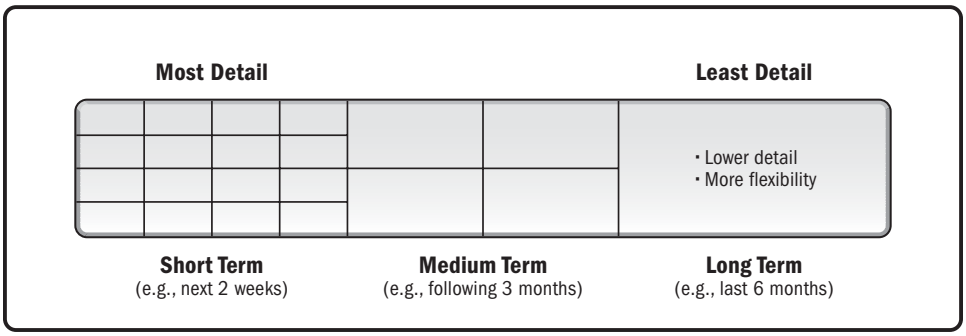

Figure 3.1 Dynamic planning detail levels.

path to predetermined goals. Dynamic environments, however, require a learning approach to cope with ambiguity and changing objectives (Molin, 2003; Mintzberg & Lampel, 1999). Pich, Loch and De Meyer (2002) describe the "learning" as being less dependent on critical path planning and task scheduling, and more dependent on a vision with detail filled out much closer to execution. The expectation that constant re-planning will be required is essential. Monitoring in these environments is less about comparing work to the plan, and more about tracking achievements and scanning for new events.

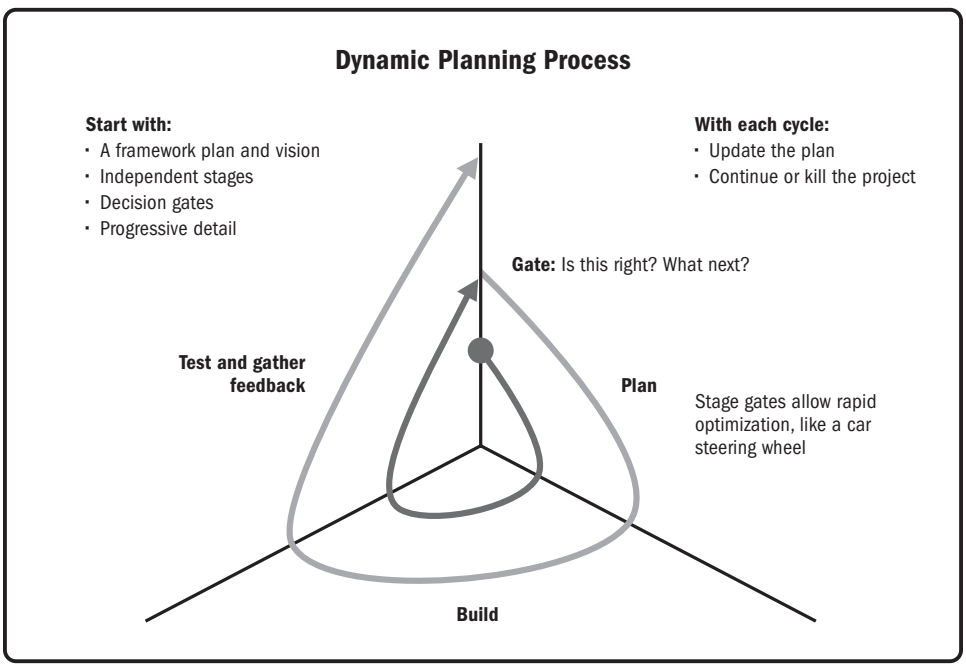

Figure 3.2 Dynamic planning process.

The head of Intel, Andy Grove, advises that "the biggest failures that you may encounter is not that your plan fails but you fail to depart from that plan" (Grove & Ellis, 2001). While useful as a guide, excessive detail in the early stages of a project may be problematic and misleading in a dynamic environment (Collyer & Warren, 2009) and counter-productive to maintain. Grove and Ellis (2001) had previously advised that "plans are highly overrated" and that "plans are a baseline, in my opinion; a model of a life that you depart from as you go on."

There are some overheads involved in adopting the iterative approach. These overheads may include those related to the process of releasing a deliverable, collecting feedback, adjusting plans, and re-work where a deliverable's material value is discarded. These overheads will vary according to industry. A consideration, therefore, in selecting the level of iteration employed on a project is the relationship between overheads and the benefits gained from redirecting the project based on real world feedback. In some environments, iteration overheads may be low (e.g., software development) and the benefits high. In these cases, an iterative approach would have obvious overall benefits. For environments with high iteration overheads it might be more prudent to use other techniques such as prototyping, late design freeze, or staging. Three alternate approaches to iteration are described in Table 3.1.

Table 3.1 Project iteration options according to overhead costs.

Iterative Overheads (Cost of re-releasing)	**Iteration Approach**	**Example**
Low	**Multi-Version Project:** Re-cycle through all project stages – plan/build/release and feedback. Discard previous versions of deliverable.	Build test software and collect feedback, then replace completely with improved software.
High	**Single Version with Prototyping:** Re-cycle planning and prototyping stages with a late design freeze. The build phase is only executed once.	Build a number of small-scale prototypes before building the main structure all at once.
High	**Single Version-Multi portions:** Re-cycle though completely independently useful portions of a deliverable without revisiting previous portions.	Build a usable warehouse to one third of the required size. Use. Review. Build next third, and so on.

Single Version Projects – Iterative planning only

Traditionally, single-version projects employ waterfall planning where each phase of the project is carried out in sequence (e.g., planning then execution). Where testing is cheap, but execution is expensive (e.g., for instance, high technology projects) there are typically long design and testing phases, sometimes with two or three design cycles, before a freeze in the second or even the third quarter of the project's duration. This is partly an iterative approach where only the design, prototype, or pilot cycles are repeated, followed by a *design freeze*, and the main execution phase (to build the production product) is carried out only once. The temptation here is go on designing at infinitum, so this approach requires discipline (Reinertsen, 1992). Striving too long for a perfect plan when you have one that is good enough, and the world is changing around you, can lead the project into failure (Rubens, Kaplan, & Okamoto, 2011; von Clausewitz, 1873). The approach for single-version projects is as follows:

- To develop faster, simply start earlier. Start planning as early as possible (Reinertsen, 1992; Laufer, 1997).
- Avoid a detailed plan at the start. Avoid "micro-scheduling" in favor of broad milestones. The initial plan should only consist of: business needs, technology options, broad scope, risks, preliminary costs, funding strategies, and milestones (Laufer, 1997).
- Break the deliverable into independently useful portions where possible and defer unmet requirements to later stages (Shenhar, 2001b). Rather than aiming for a perfect design, quickly implement a partial solution; be willing to test a flawed product.
- For the components least subject to change, build them first, based on a design that allows maximum flexibility for later adaption (Gray & Larson, 2003, p. 548; Shenhar & Wideman, 2000; Shenhar, 2001b; Simons, 1995; Thomke, 1997).
- For the most variable components, start investigating them early, but freeze their design last (Laufer, 1997). Have the discipline to freeze the design in time to meet the overall objective.
- Use fast and repeated design/development cycles, allowing the project to adapt at a higher speed than environmental changes (Shenhar, 2001b).
- Identify default solutions early to be sure of a solution within the time frames and then investigate other options in parallel to seek further optimization.

Case Study 3
Intel Planning Adapts

Andy Grove, the CEO of Intel, gave a keynote in 2001 revealing some of his views on planning and culture during rapid change. Despite being a details man, Grove was a strong advocate for adaptability:

> We operate in a life that is rich in technology, and technology will continue to experience rapid change...The biggest failures that you may encounter is not that your plan fails but you fail to depart from that plan...plans are a baseline, in my opinion; a model of a life that you depart from as you go on. (Grove & Ellis, 2001)

Multi-Version Projects – Fully Iterative

On projects where multi-version manufacture is not cost-prohibitive (e.g., software development), you can try successive-release versions to generate real world feedback to design better and better versions of the product with each release. This fully iterative approach is especially useful where the product is relatively new and therefore risky (Brooks, 1975; Boehm, 1988). The approach allows you to discover unknowns and adapt quickly to a changing environment. The iterative approach is also known as *spiral*, or *incremental*. When there is limited knowledge about how a product might interact with its environment, an iterative approach is an effective way to test and collect that information, while minimizing the risk of going down the wrong path. Some versions of the iterative approaches even use feedback as the primary control mechanism, rather than planning (Fowler & Highsmith, 2001). Feedback drives tests and re-releases of the evolving software, without involving long-term planning. Agile development employs this approach with the initial scope remaining very basic, followed by regular and early delivery, focusing more on strong communication than a complex process (Fowler & Highsmith, 2001).

Dynamic Planning Examples

The dynamic planning principles emerging from research were strongly supported by the practitioners I interviewed. All but one participant gave examples of their own use of these approaches. The emergent planning concept attracted the greatest consensus across participants as a means to manage dynamism. For example, an IT project manager reported "I like to lay out the major phases / deliverables / milestones at the outset, but only plan

the detail for the phase I'm about to start." A venture capitalist related how "while an overall plan was in place to start with, the individual stages are often revised."

Contrasting one of the construction participants with the defense participants regarding safety may illuminate an interesting consideration for deciding whether to embrace or resist change. For each one of these leaders (construction and defense), the "embrace change" approach carried very high risks, but for the defense participant, the risks of resisting change were even higher. The defense participant reported that embracing and adapting to change during a military campaign actually reduced overall risk despite its problems. The defense participant, therefore, employed rapid adaptation principles, such as delegated control and management by objective.

For the construction participants, embracing change increased financial and physical security risk while providing little advantage of any kind. This led them to adopt principles that resisted change, such as strict centralized control implemented against detailed static plans. The road tunnel construction planner described how he strongly resisted change unless it was necessary to bring work back in line with the plan. The construction planner actually suggested if an order is slightly wrong it's sometimes better to follow that order to avoid chaos. It may be that the construction industry achieves its overall safety and financial imperatives more effectively through strict management of and resistance to change. Indeed, this may also be possible in an industry where there are relatively slow rates of change in tools and techniques, offering little advantage to those who embrace them in the course of a project. Where the benefits of embracing change do not outweigh the benefits of making static, the preference in some industries may be to maintain order and make static in order to obtain other benefits such as financial predictability and safety.

A green-power-generation start-up revealed some of the challenges with emergent planning when they said:

> Earlier stages do inform later stages but in more of an informal, unplanned way...Running a pilot is fundamental to the business plan. It's a proof of concept. The business plan is set up to deal with this uncertainty. Some people would like to reduce overlap between stages and do things more sequentially to reduce the variability in the planning. For instance it's hard to finalize the design of the power station without well outputs, which depend on the results of the subsurface work.

> The solution we are trying to work with is to design scalability/adjustability in subsequent stages (e.g., power generator) to allow them to adapt to the results of the early stages as they become known.

An IT project manager described how she used emergent planning because sometimes there is no alternative:

> A global rollout of infrastructure for a major global investment bank involved replacing a legacy infrastructure. Prior to deployment a significant amount of testing was completed and it was believed that a full understanding of the full impact on equipment and applications was obtained. However, during implementation it became clear that there were many regional based applications and environments that were impacted differently. As a result the rollout was completed country by country and data was gathered after every implementation in order to prepare for the next.

A film producer described how the developers of *Who Wants to Be a Millionaire*, syndicated in 100 countries, was piloted seven times before being released. Even one of the construction participants provided examples of how the results of the first tunnel construction project significantly altered plans for subsequent tunnels. In fact, all but one of the research participants gave examples of emergent planning techniques, including prototypes, pilots, and experiments.

Case Study 4
The Bourne Ultimatum

Film production is a good example of a dynamic environment. Matt Damon (2007) related the issues he faced filming The Bourne Ultimatum on the streets of Tangiers. Traditional crowd control was not possible, so the goal was defined in broad terms and the means were left adaptable. The director made a basic plan that could easily be changed, anticipating there would be problems that could be rectified either during execution or in later iterations (e.g., in post-production editing). So, a high-level plan was developed and then high levels of communication were used during execution, in multiple iterations, with expectations of unpredictability.

Staging with Gates

Breaking the project into multiple stages, with stage gates, is important to give "steerability." At the end of each stage there is a "gate" where progress is discussed and the plan can be adjusted or the project killed. Davila, Epstein, and Shelton (2006) reported how Exxon has used a gate process since the 1980s with great success, and "according to Exxon it's been the best initiative it has undertaken in a decade; it has shaped the way it does business" (p. 276). Stage gates are akin to "time-boxing" in the software development industry (Barker, Fiedler, & Johnson, 2008). To use a military analogy, stage gating is "aim, fire, aim," not "ready, aim, fire." Fast and repeated design/development cycles allow the project to adapt at a higher speed than the environmental changes. Build in maximum flexibility so the product can be further adapted in later stages. Accept there will be later stages of development and adaptation. Have the discipline to freeze the design and accept that it will slowly desynchronize with the environment. Staging also allows different parts of the project to be run in different ways. Less-variable components can be run using a more detailed planning approach, while components subject to higher change can use the learning approach where exploration is started early and the design is frozen as late as possible.

Simplification – Smallest Possible Stage 1

Failure rates are known to increase with project size (Jones 2003; StandishGroup; 1994), and size is compounded by dynamic environments. For a dynamic environment, the smallest possible scope in the first stage mitigates the negative impacts of environmental change. In dynamic environments you don't know anything until you've tested in the real world. A simple fast first stage is important to give fast feedback. Getting feedback quickly prevents wasted effort on an inappropriate design. A simple and quickly released first stage tests the concept (e.g., *prototype*, or *proof of concept*). For example, consider how a budding author might practice with magazine articles to gather feedback instead of wasting years writing a novel only to discover that he can't write. In the software industry they call this the "fail-early" approach (Van de Vord & Pogue, 2012). There will usually be significant pressure to add functionality to the first release, but this must be resisted. A useful way to reduce this pressure is to receive ideas enthusiastically, but ruthlessly mark them down for subsequent releases.

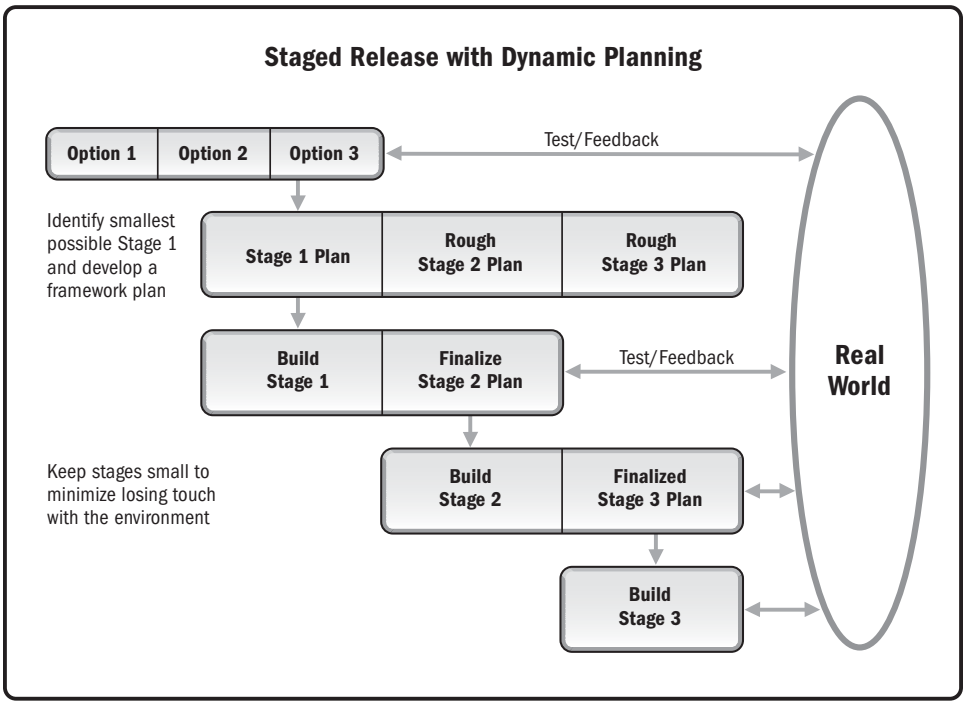

Figure 3.3 Experiments, staged release, and emergent planning.

Simplification and Staging Examples

This section highlights some relevant observations from the study participants, along with some non-participant examples. This book also includes a full chapter on case studies at the end.

The following are some comments made by the participant project managers relevant to the use of scope reduction and staged releases. The manager of a program of drug-development projects reported how she initially brought drugs to market with only "lead indicators" developed, and later investigated the full range of possible applications in order to develop the drug to its full potential. The manager of a project to develop a new energy storage technology reported how he was initially developing it for the industrial market, with a view to expanding applications once it was proven. The manager of a program to develop new power generation technologies planned to test the new power-generation process on a very small scale, initially providing power for a small town, before exploring the potential to power a state. An anecdotal example provided by a pharmacy industry project manager was that of Rituximab, developed by Biogen Idec and Genentech, and how it was initially developed to treat one type of cancer patient

group. When that proved successful, it was expanded to treat others, and later arthritis.

When the Defense Advanced Research Agency (DARPA) began work on the High Altitude Endurance Unmanned Aerial Vehicle Program, their master plan restricted them to combining mature technologies to avoid the time and risks associated with technology development, reporting the "emphasis on the use of COTS [commercial off the shelf] was important; the contractors believe that COTS is a useful tool for reducing costs and cycle time" (Drezner, Sommer, & Leonard, 1999, p. 236). Paul Trott (2005) reported how the chemical industry "is increasingly developing smaller, more flexible plants rather than the large, single purpose plants" to be more responsive to a more rapidly changing market (p. 261). Skybox Imaging changed the nature of the satellite industry by building satellites with off-the-shelf electronics (Truong, 2014). It launched its first satellite in November 2013 and within seven months was acquired by Google for US$500 million.

Across all of the focus groups, the staged release approach was considered to be effective in the management of dynamism. One project manager reported:

> It provides points throughout the process for consolidation and reflection, making sure that the direction and development is true to intent, and all controls (i.e., budget, time, quality) are maintained…and more importantly the destination is in line with the strategic objectives of the exercise.

An IT project manager opined, "sometimes on slow projects the technology changes significantly so staged releases [are] necessary." The same participant identified that staged releases had been adopted and refined in the software development industry in the form of "time-boxing," which is where "you have to FINISH things in that time-box…so you are delivering SOMETHING and remove risk that you work for years and deliver nothing."

The staged releases approach was also considered to be "important to build confidence by getting things out there quickly...it helps build confidence and helps engage." A phone app developer reported, "I use this for the mass market…mobile consumer market…and get feedback from the minimum…and then use the feedback…and we just drop the product if it's not working."

Where the project managers were unsure about the suitability of the project deliverables, they were keen to test them in the real world, and use the results to either confirm the plan, optimize the plan, or end ("kill")

the project entirely. The bravado with which participants appeared willing to kill projects suggested that perhaps they had learned this from bitter experience.

Staged releases was identified as a fail-early approach, highlighting one of its key advantages. Fail-early is software-development industry terminology for the technique whereby systems are designed (programmed) to generate highly visible failures at the slightest hint of a problem. The increased visibility of failures makes it easier to identify and correct mistakes early before they cascade into higher consequence failures later in the project. In software coding, this approach is contrasted with an approach that absorbs errors and potentially results in unstable or non-deterministic states. A space launch industry project manager related how:

> we often get accused in my business of killing flies with sledge hammers, because we've learned that real quickly they turn into Godzilla.

With that comment another participant retorted, "and you've had enough disasters to justify that," and so it emerged from the focus groups that experienced practitioners embraced the staged releases approach out of fear built up from past failures. They were keen to find out as early as possible whether they had any problems that could potentially "bring down" the project.

Another idea emerging from study participants was that staged releases could be used to assist with change management. It was proposed, "you can start doing your change management early…because you have something tangible…it builds confidence." Another reported, "when the result of the first stage is considered the enterprise readiness for change is evaluated." The multi-stage release approach allowed the project manager to "kick-off" the change management earlier and more effectively than a single-stage approach. This approach was considered important for a rapidly changing environment where there is little time.

Participant project managers agreed that the staged-releases approach was an effective way to manage dynamism, but project managers should be aware that it does involve an execution cost trade-off because of the cycle repetition. However, the cost trade-off was more than offset by cost savings by: (a) having an early working product delivering benefits before the window of opportunity is lost, and (b) gathering early feedback on performance and relevance, therefore allowing more appropriate adaptation to the changing environment in subsequent released. The staged

releases approach was revealed to be also be known as fail-early and time-boxing (FG).

Contract Terms

In dynamic environments it is important not to lock down parts of a project that might change. It is sensible, therefore, to contract the parts of a project that won't change first (Laufer, 1997), providing you are ready to go to contract at all. Consider for instance the Collins Submarine project, which might have fared better by delaying the weapons system contract until later in the project. Beyond that, there is ongoing debate about the relative benefits of fixed-price versus cost-plus contracts. On the one hand, the rigidity of the fixed-price Collins Submarine contract was believed to have contributed to the installation of obsolete computer systems that required immediate replacement at great cost (McIntosh & Prescott, 1999). On the other hand, there is a belief that cost-reimbursement contracts create inducements to inflate costs, or avoid cost-reduction measures. As McIntosh and Prescott (1999) asserted when they reviewed the project:

> For a relatively routine product or one where the specifications are clear and unambiguous and where payment is made mostly on delivery, (a fixed-price contract)…can work well. However, for a large, complex and new project, for which a design does not exist in detail and for which generous up-front payments are made, its effect can be deleterious. Particularly in the later stages, it can encourage the supplier to contest the specifications, and their interpretation, and to avoid responsibility wherever possible to protect profit… (p. 5)

For U.S. defense projects, where technological advances are an increasing challenge, design competitions have often been used, leading to a single source contract for the winner (Ergas & Menezes, 2004). Unfortunately, once the production phase begins, there is little incentive to pass on savings or innovations until the next design competition, which is usually years away (Rogerson, 1994). The innovation cycle is, therefore, slowed down significantly overall.

To maintain innovation and timeliness in a dynamic environment, one option is to reduce the spacing between successive generations of the product, creating a kind of parallelism where the new version is pitted against its current iteration (Ergas & Menezes, 2004). This seems to be the approach

used by technology manufacturers such as Apple where there are often multiple generations competing. Parallel contracts require a willingness to develop new systems before current versions reach end of life. For complex innovation projects there is a trend toward hybrid contracts (Drezner & Leonard, 2002; Ergas & Menezes, 2004; Ingols & Brem, 1998). This approach involves defining desired outputs and giving suppliers greater control over how those outputs are achieved. Systems development is subject to cost reimbursement, and suppliers are made aware of a "must cost" cap above which the project is canceled. The cap is defined by the perceived value of the likely end product.

Fixed-price contracts can cause "disputes, adversarial practices, and protracted legal battles between clients and contractors" (Davies, Gann, & Douglas, 2009, p. 108). To give an example of fixed-price problems after the Collins Submarine project, the investigators reported:

> Far more significant to the course of the project was the belief underlying the fixed-price model that a contractor's signature would see it taking responsibility for the risks and the obligations required to discharge the contract. The assumption proved to be by no means guaranteed, as developments during the life of the project proved capable of defeating the intent of any legal obligation, regardless of earlier agreement between the participants. (Woolner, 2009, p. 65)

While the intent of fixed-price is to transfer risk and increase the chances of the project being on time and on budget, this is not necessarily so, and the model does not easily allow for adaptation to changing needs and applications. Cost-plus incentive contracts are more suited to dynamic environments for their adaptability. When the British Airports Authority (BAA) started their US$8.5 billion Terminal 5 project, they chose a cost-plus incentive contract in which the client paid the constructor the costs incurred plus a profit margin, and the BAA assumed full responsibility for the risk and worked collaboratively in integrated project teams with first-tier suppliers to create innovative solutions. The new Australian approach is to make better use of performance-based contracts (PBCs) with a view to gaining better value for money while maintaining reliable profits for suppliers (Department of Defence, 2010). In the end, however, all contract types, including fixed-price, cost reimbursement, or target cost incentive, can be structured as PBCs by adding performance payments linked to key performance indicator. The benefits and issues of three common contract types are shown in Table 3.2.

Table 3.2 Contract types for dynamic environments.

	Fixed-Price	Cost Reimbursement	Target Cost Incentive
Description	Contractor paid a fixed amount regardless of actual cost.	Contractor paid costs plus a fee for profit/overhead.	Contractor paid costs plus a fee for profit/overhead, which is adjusted accordingly to provide an incentive to save money.
Benefits for Dynamic Environments	Most efficient and effective approach for components that should remain static.	Adaptable; only pay for costs incurred.	Provides incentive to minimize costs; aligns goals of sponsor and contractor.
Issues for Dynamic Environments	Not suitable for the significant variability of dynamic environments. This can be mitigated to some extent with performance-based incentives.	Risk moves from contractor to sponsor; less incentive to control cost; higher administration.	Requires negotiation of reasonable bonus/penalty; competition required to incentivize contractor to agree to penalties.

Calculating the Cost of Lost Opportunity

Research studies have well established the importance of project speed in dynamic environments. In fact, a delay of 10% in the technology industry was found to have a more significant impact on total revenue than 10% overrun in production cost (Dumaine, 1989). Every day that the project's finished service or product is available for use is the "opportunity." Every day the product is not available is the "lost opportunity." The importance of delivering project goals within a limited window of opportunity featured prominently in the research participants' accounts. Delivering within this window was considered to require a delicate balance between quality and expedience. The relationships between scope, quality, duration, and cost in dynamic environments are shown in Figure 3.4, Figure 3.5, and Figure 3.6. Figure 3.4 shows how opportunity can be lost as a consequence of an excessive focus on quality. In dynamic environments, product life spans are typically much shorter. In a dynamic environment, therefore, the traditional concept of quality must be qualified by the much shorter usability window. Conversely, one must also be aware that the cost of insufficient quality at some point will outweigh the lost opportunity cost, as depicted in Figure 3.5. A critical consideration in managing dynamism is balancing the risks of insufficient quality against the cost of lost opportunity.

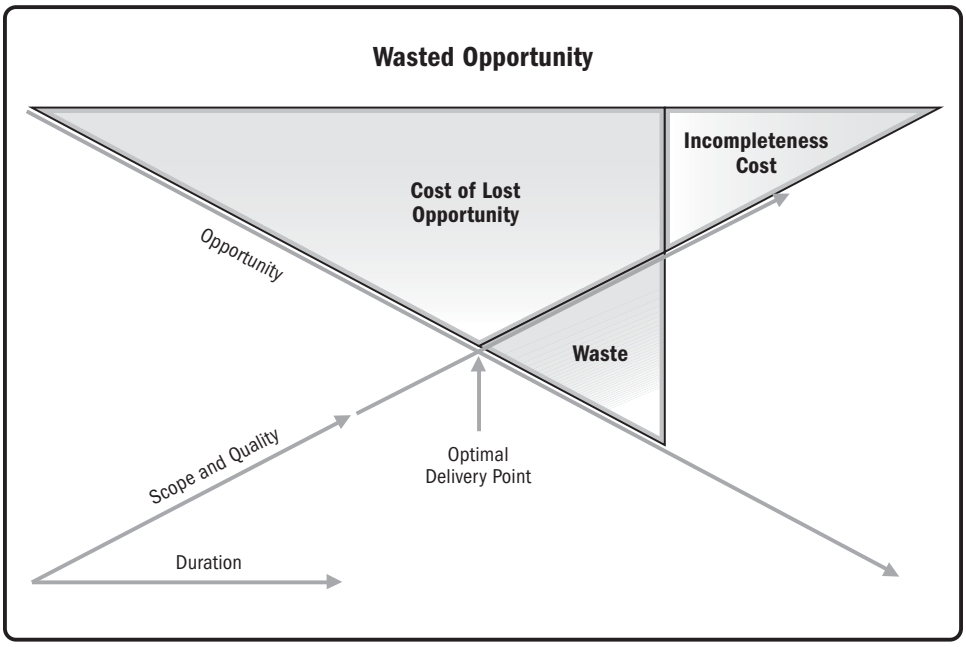

Figure 3.4 Cost of sub-optimized opportunity.

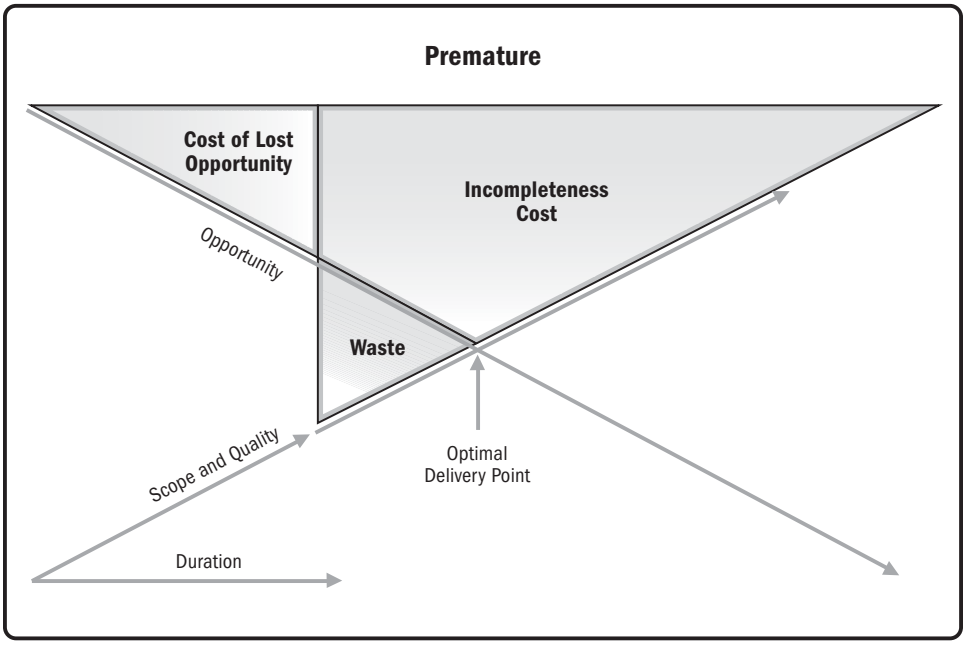

Figure 3.5 Cost of sub-optimized completeness.

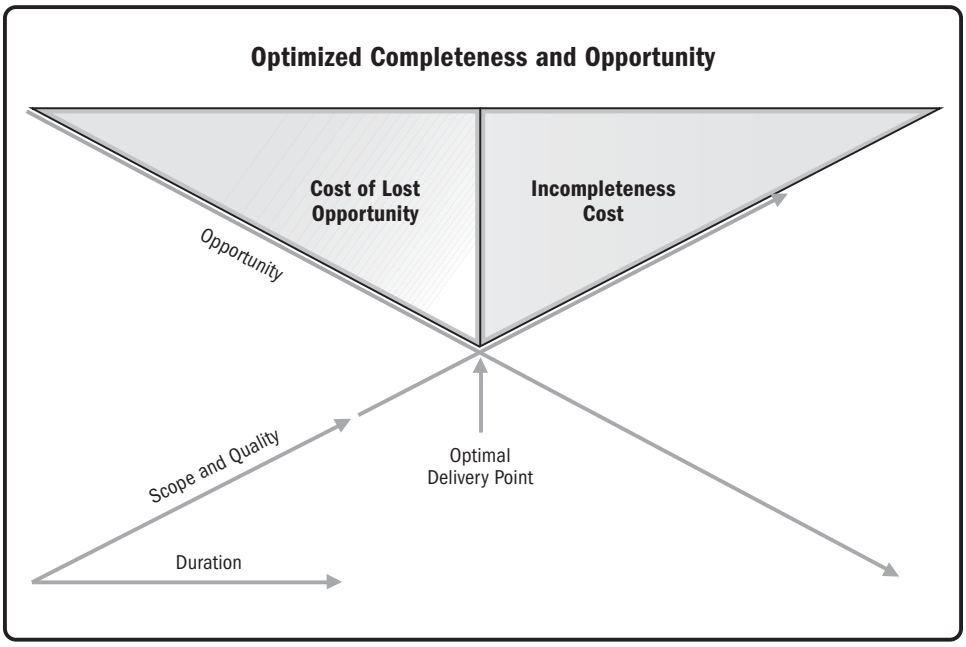

Figure 3.6 Optimized balance of completeness and opportunity.

A project manager explained how he establishes the relative urgency of a project:

> To justify the urgency I work out how much it costs the organization not to have the project deliverables. If there is not much cost I back off the urgency—but if there is a high cost I explain this in a slide to justify the speed. Some places don't work out how much the delay costs…but that's how you justify short-cutting—which everyone hates—but some projects need it. If you are coming into somewhere that hates short cutting, maybe they've been burnt with rework, you have to explain that.

In dynamic environments the expected lost opportunity should be used to justify high-speed focused delivery approaches employed, such as framework planning, parallel experiments, or the staged approach. In the words of one project manager, "I think this goes to the heart of the business case of the project. [There is] no point delivering something in 12 months' time if you've missed the opportunity."

Summary

Plan initially with the desired benefits, objectives, and milestones. Calculate and be aware of the cost of lost opportunity. Commit initially to the smallest possible scope first stage in order to obtain real world feedback early with the smallest level of commitment or cost. The objective is to minimize effort on unsuitable approaches and to reduce the amount of time the environment has to diverge from the plan. Break the project into stages, with each stage ending with a decision gate (requiring re-planning, revised business cases, and formal review). Each stage should include progressively lower levels of detail according to how far head one is looking. Identify areas subject to uncertainty or change and start investigating those early, but freeze their design as close as possible to the start of execution. As the project progresses, use progressive elaboration to firm up plans for imminent stages and to review the plan against objectives and business benefits, potentially killing the project if required.