Higher Creativity for Virtual Teams: Developing Platforms for Co-Creation

Steven P. MacGregor
IESE Business School, Spain

Teresa Torres-Coronas
University Rovira i Virgili, Spain

Information Science REFERENCE

INFORMATION SCIENCE REFERENCE
Hershey · New York

Acquisitions Editor:	Kristin Klinger
Development Editor:	Kristin Roth
Senior Managing Editor:	Jennifer Neidig
Managing Editor:	Sara Reed
Assistant Managing Editor:	Sharon Berger
Copy Editor:	Joy Langel
Typesetter:	Jamie Snavely
Cover Design:	Lisa Tosheff
Printed at:	Yurchak Printing Inc.

Published in the United States of America by
Informaion Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue, Suite 200
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-pub.com
Web site: http://www.igi-pub.com

and in the United Kingdom by
Information Science Reference (an imprint of IGI Global)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanonline.com

Library of Congress Cataloging-in-Publication Data

Higher creativity for virtual teams : developing platforms for co-creation / Steven P. MacGregor and Teresa Torres-Coronas, editors.

p. cm.

Summary: “This book presents advanced research on the concept of creativity using virtual teams, demonstrating a specific focus and application for virtual teams. It presents tools, processes, and frameworks to advance the overall concept that leveraging ideas from different locations in an organization and within extended networks is based on creativity, which can deliver innovation”--Provided by publisher.

Includes bibliographical references and index.

ISBN 978-1-59904-129-2 (hardcover : alk. paper) -- ISBN 978-1-59904-131-5 (ebook : alk. paper)

1. Virtual work teams. 2. Teams in the workplace. 3. Creative ability in business. I. MacGregor, Steven P. II. Torres-Coronas, Teresa, 1966-

HD66H545 2007

658.4’022--dc22

2007007265

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Chapter XII
Enhancing Flexibility in Dispersed Product Development Teams

Preston G. Smith
New Product Dynamics, USA

ABSTRACT

Highly creative product development teams are exploring the unknown. Initial plans are likely to change as they understand better how the customer will use the product they are developing, as competitive products appear, and as new technologies evolve. Thus, a creative team must remain open to change as its plans shift. If the team is dispersed (virtual), the complications of dealing with changes in plans magnify. This chapter provides tools and approaches for being flexible to such changes as creative teams proceed. These include ways of lowering the cost of change, anticipating change, isolating change, and maintaining options as late as possible. Such tools and approaches will help teams working on highly creative projects to take advantage of their creativity, even when they are dispersed over time and distance.

INTRODUCTION

Dispersed[1] product development teams have become increasingly popular over the past decade, especially with large multinational companies. In many cases, these teams span multiple continents and time zones. In order to maintain control over such a far-flung organization, management generally imposes procedures and plans so that all parts of the team remain focused on a common objective.

While they clearly have their strengths, such procedures and plans can undercut creativity. They encourage heavy upfront planning and reward sticking to plan. In contrast, creativity requires experimenting, trying things out, and adjusting as better solutions appear. In short, dispersed teams are easiest to manage when they can execute their original plans without change, but creativity requires change.

This chapter addresses this paradox by introducing the notion of flexibility in dispersed teams and by showing how one can enhance the flexibility of a team to deal effectively with change.

Creative product development teams need the flexibility to be able to explore options and make changes, even late in the development cycle. Unfortunately, such flexibility is difficult to achieve, especially for dispersed teams. This chapter will explore flexibility and offer flexibility-enhancing tools aimed at teams spread across various locations.

What Flexibility is and Why it is Important

Flexibility is the ability to make changes relatively late in a project without being too disruptive. The later one can make changes or the less disruptive they are, the more flexible the process is. One usually measures disruption in terms of the money, labor, or time lost in making the change. See Figure 1, in which, after the initial planning period, the restricted flexibility level locks too much down too early, but the completely flexible level leads to chaos at the end of the project. Thus, the rate of convergence must be managed consciously throughout, as shown in the moderately flexible level.

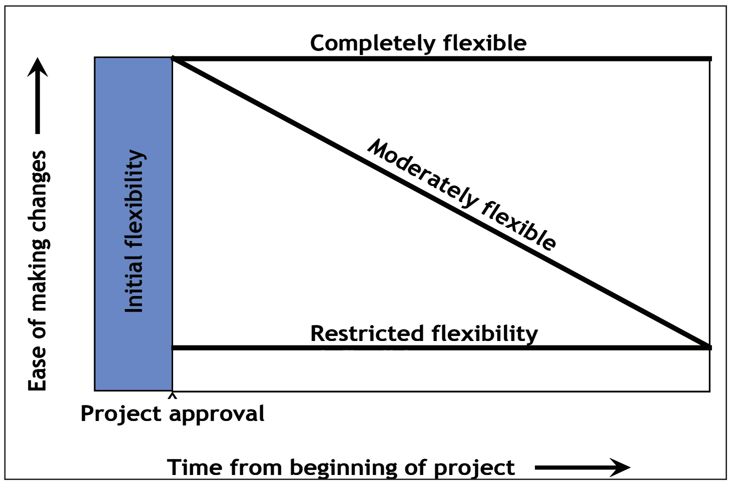
Change can appear in many forms. A common one in product development is a change in product requirements, which may occur because the developers neglected to identify a requirement earlier, because feedback from prototypes or market research has uncovered a new requirement, or because a competitor has just offered new functionality. Technical change is another source, and it can occur when a new technology appears, when the capabilities of a technology expand, or when developers discover weaknesses or limitations in a technology that they are planning to use.

Flexibility is important because the essence of product innovation is change, as discussed below. Productive innovation benefits from change, and inhibiting change stymies innovation.

Flexibility and Creativity

Product development is the creation of something that has not existed before, and as one pursues this creative act, unplanned changes will occur. Creativity involves generating, assessing, and choosing among options. Creative professionals are trained to generate many options without

Figure 1. Three levels of managing flexibility in a development project (Source: 2007 by John Wiley & Sons; used with permission)



judging them, then to narrow them gradually toward a final solution.

In effective product development, the customer drives these choices and the team often implements them using some type of technology. Customers often clarify their thoughts as to what they really want only after they have tried a model or prototype, and as engineers get into a design and start testing it, they often find that a technology does not work as originally envisioned. Thus, the need for a change arises, and it can arise at any time during the project.

The more creative the project is in terms of satisfying new customer desires and the more adventuresome engineers are in applying new technology to unmet needs, the more change is likely to occur. Effective innovation encourages such change, and resisting change inhibits innovation.

James Adams, author of the classic on creativity, *Conceptual Blockbusting* (Adams, 1974), connects creativity and change thus: "Creativity and change are two sides of the same coin. They are often linked, in that creativity is needed to respond successfully to change and creativity, in turn, results in change" (Adams, 1986, p. 3).

There are alternatives for dealing with changes. One is to discourage them after a certain initial point in the project ("freezing" the design), but this will clearly diminish creativity and result in inferior products as better information arrives later in the project. Another is attempting to predict change, but this is likely to be frustrating and result in a rigid process that impedes creativity. Perhaps one could hope that change will not occur and then deal with it when it does, but such behavior will erode project performance, because developers will not be prepared to deal with change when it occurs.

Consequently, this chapter takes the position that change will occur and applies practices that will diminish the impact of changes, even when they occur relatively late in the project. In short, the goal is to put practices in place that will allow the team to accommodate and even to embrace change. For a dispersed team, these practices must work in a dispersed environment.

BACKGROUND

Change—even disruptive change—is becoming more common and more frequent in business today (Brown & Eisenhardt, 1998). This is especially true in new-product innovation (Christensen & Raynor, 2003; Christensen, Anthony, & Roth, 2004). On the other hand, business managers, hard-pressed to perform under competitive stress, are moving toward more sure-fire methodologies that focus on minimizing variation and eliminating mistakes, waste, and rework. These include phased development, such as Stage-Gate$^{®2}$ (Cooper, 2001), Six Sigma (Eckes, 2003), and lean product development (Mascitelli, 2004).

Phased development organizes the product development process so that important steps, especially in the front end, are not skipped. Six Sigma, as its name suggests, continually refines business processes to minimize variation. And lean development, which has grown out of lean manufacturing in several different directions, centers on strengthening processes to eliminate waste. This can either be waste in the design phase or downstream in the manufacturing phase. As different as these approaches are, however, they all have one characteristic in common: they all attempt to improve the business by strengthening processes.

Strengthening processes has been beneficial in general—but it carries with it a side effect. It tends to make the process rigid so that it must be followed and cannot be changed easily. Consequently, as management moves toward stronger business processes, they tend to move away from flexibility.

As discussed, flexibility is connected with change and innovation. Predictably, as firms move away from flexibility, they are having more difficulty being really innovative with their new products. This is reflected in recent data, which show that from 1990 to 2004, more innovative products (new-to-the-firm products and new-to-the-world products) have declined substantially in product portfolios while safer ones (additions, improvements, and modifications to existing products) requiring less flexibility have increased, as shown in Figure 2.

Although flexibility is a new topic in nonsoftware product development, the software development community has practiced it for several years under the name of agile development. Larman (2004) provides an overview of agile development, including descriptions of several popular agile methodologies. Boehm and Turner (2004) show how to balance the need for agility and the need for discipline on a specific project and, in the process, illustrate the factors that determine whether a given project should follow a more fluid or a more structured process.

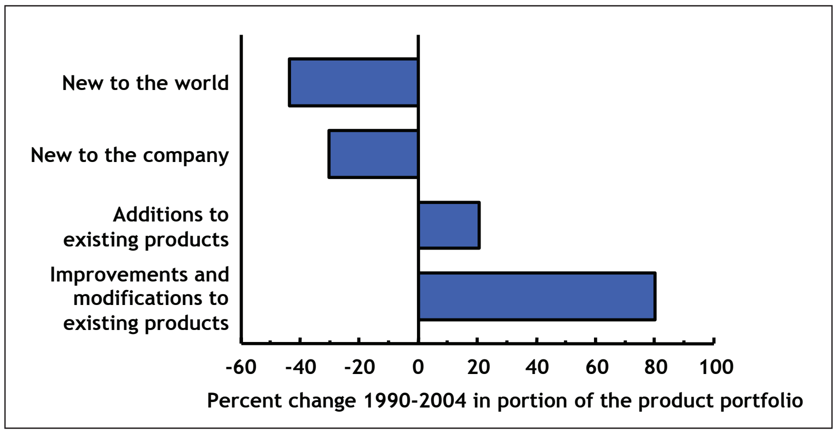
Lessons from the agile software arena point the way for flexible development and establish the guiding principles and values. Yet it is not possible to apply the techniques of agile software development directly to nonsoftware projects. Several characteristics unique to the software medium allow the agile tools to work there, for example, object technologies and the ability to automate the build process so that the team can build an update of the product cheaply and daily. In general, these characteristics do not apply to other types of products, so for nonsoftware products the need exists for solutions other than the agile development tools in order to enhance flexibility.

USING FLEXIBILITY TOOLS AND APPROACHES

The tools and approaches described in this chapter work in various ways to improve flexibility:

- They may isolate or encapsulate change so that a change does not ripple through the whole product causing massive redesign.
- They may allow one to move ahead iteratively with lots of feedback when it is not possible to see very far ahead.
- They may expose new options through experimentation and intentional expansion of the design space.
- They may keep options open longer by delaying decisions (but still without affecting the project's overall schedule).

Figure 2. Product innovation has decreased dramatically since 1990 (Source: Cooper, 2005. Figure copyright 2007 by John Wiley & Sons; used with permission)



- They may reduce the cost of change by maintaining backup positions or understanding the consequences of a choice.

These tools must be used selectively. Each tool has types of projects where it fits well and others where it fits poorly. Just as with a set of mechanic's tools, it is not a matter of using all of the tools for every job but one of selecting the tools that fit each job and using an appropriate combination. In general, each project will require a different combination.

However, one lesson that carries over from agile software development is that the tools tend to fit together in a mutually supportive way (Beck, 2000, Chapter 11). There are synergistic effects of combining the tools. Consequently, do not focus on just one tool, but try to apply a group of them that will support each other.

In most cases, a tool focuses selectively on anticipated types of uncertainty. Usually, it is not possible, or economical to encompass all types of uncertainty with one application of a tool or approach, and providing flexibility in one area may limit your flexibility in another area. Thus, it is usually necessary to make choices as one proceeds as to where change is most likely to occur and focus the tools on areas where flexibility might have the greatest payoff in allowing change with little disruption.

As Boehm and Turner (2004) illustrate, these tools and approaches can have undesirable side effects. Flexibility and stability need to be balanced, and the more dispersed a team is, the more the balance is likely to shift toward stability—to the detriment of creativity.

THE TOOLS AND APPROACHES

This section covers eight types of tools and approaches that enhance flexibility:

- Customer understanding
- Product architecture
- Experimentation
- Set-based design
- Product development teams
- Decision making
- Project management
- Development process

After describing each tool, the chapter closes with further discussion on how they can be combined.

Customer Understanding

It is fundamental that the needs of customers drive the development of successful products. Good practice normally is to assess the needs of customers, capture the essence of these needs in a document often called a product specification, and design according to this specification. Seldom does this work well in actual projects:

- Writing is an inadequate medium to describe the complexities of customer use or customer desires.
- Often time pressure forces developers to start designing before they have all customer requirements.
- Customer usage patterns are complex and change over time.
- What is essential to one customer is unimportant to another.
- Customers change their minds after they see how a product works.
- Customers use products in ways never considered by the designer.

This means that the specifications will change over the course of development. Depending on how innovative the product is, they could change a little or a great deal.

The first tool for dealing with such changes is to build an early warning system, that is, a system to alert you to changes in the customer environment and allow you to check out your designs early. There are no standard prescriptions for doing this, because experience shows that the best solutions invariably are the ones created by a company to meet its specific needs. But here are some guidelines:

- To get advance notice of potential changes, get in touch with lead users, as popularized by von Hippel (1994). These are the people who are leading change and are likely to modify your product to suit their leading requirements.
- Get the designers themselves in direct contact with users of the product. They see different things than marketing or sales staff, which inadvertently filter out valuable clues.
- Get in touch early in the project, and—most importantly—keep in touch throughout the project. You never know when change will occur!
- To balance exceptional or noncharacteristic incidents that designers might see in isolated cases, have marketing people survey the customer arena regularly to provide balance and interpretation of incidents.

In most cases, a dispersed team will have additional challenges in putting its designers in ongoing contact with customers. Because economics often drives dispersion, your designers are likely to be in a low-wage region of the world, such as India or China, while your intended customers are located in a wealthier region of the world, such as North America or Europe. Thus, putting designers in contact with customers may be a challenge.

Another approach is to emphasize product descriptions at a level that is less likely to change. For instance, most developers work from a product specification that is a detailed list of features or requirements. Such details are almost bound to change as you learn more about your customers and the design space. Instead, place primary emphasis on a product vision (Clark & Fujimoto, 1990), which is a short statement (100-200 words) that describes the distinctive characteristics of this product relative to other products within the company's portfolio, or indeed the competition. The vision is far less likely to change.

Related approaches that center on aspects less likely to change are ones that attempt to capture the customer. One is personas (Cooper, 1999), which are descriptions of archetypes of predominant classes of users, each carefully created from methodical customer research. Suppose you were designing a waterproof digital camera, and your primary persona were Jeslyn, a white-water kayaker. Then if you were considering a change in camera operation that would require two hands on it, someone would immediately object, "We can't do that! Jeslyn will have the paddle in her other hand." A similar tool is use cases, which software engineers use to describe how a user would interact with a product to perform a certain task (Cockburn, 2000).

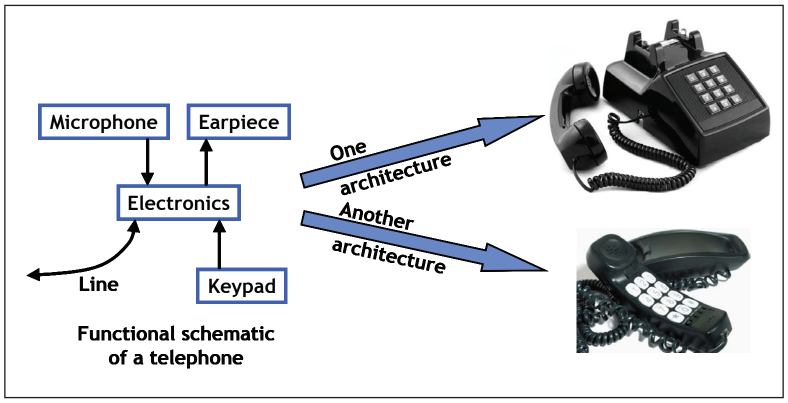
Product Architecture

Just as one may put fences around pastures to avoid chasing livestock across the countryside, one places "fences" around chunks of a product to contain design changes to relatively small parts of the product.

There is lots of talk about product architecture, but it is a rather abstract subject addressed from many different perspectives. It is therefore useful to start with an example of an architectural choice. Figure 3 shows two different architectures of a corded telephone. Both of them share the same functional schematic, but the architect chose to put the keypad function in different chunks of the physical product.

From this figure follows a useful definition of product architecture: Architecture is the way in which the functional elements of a product are

Figure 3. Two different architectures for a corded telephone (Source: Copyright ©2007 by John Wiley & Sons; used with permission)



assigned to its physical chunks and the way in which those physical chunks interact to achieve the product's overall function.

One reason this is a confusing subject is that designers can use architecture to achieve many different ends, so each person discusses it in terms of what he or she wants to achieve. Some possible objectives include:

- Product development flexibility (our objective)
- Manufacturing flexibility
- Product distribution flexibility
- Time to market
- Product serviceability

Each objective will result in a different architecture. It follows that architecture is a strategic decision, and you must choose your business objective before you can create an appropriate architecture.

Even narrowing to product development flexibility, there are still architectural choices to be made. Usually, it is expensive or impossible to find one architecture that will facilitate any kind of design change. Thus, one must make some assumptions as to where change is most likely to appear and design the architecture accordingly.

An important consequence is that architectural choices should not be technical decisions but instead business decisions. This may seem obvious, but too many companies turn such decisions over to their engineers and thus forfeit the business benefits.

Product architectures span the range from modular to integral, as illustrated in Table 1. Each approach has its place, and most implementations are somewhere between these two extremes. Modular architectures are advantageous for flexibility, because they allow us to place "fences" around portions of the product most likely to change so that the change is limited to that portion of the design. The fences are actually called interfaces, and interface design and location is thus a critical part of organizing a product for flexible development. Like fences, developers must maintain interfaces consistently over time or they will decay and lose their "fencing" power.

What is special here for dispersed teams? First, recognize that architectural choices are important, and they are made quite early in the project (often even in preceding projects!). Because these are business decisions, you will somehow have to assemble your dispersed business team early in the project to plan the architecture for flexibility—or whatever other business objective you choose. Do not assign this task to the engineering team, even if it is all in the same location.

Second, you will need to designate someone to maintain the architecture. Although an engineer should not be the primary creator of the architecture, an engineer might be the ideal candidate to be responsible for maintaining it, because most of the violations are likely to arise in engineering as designers make design compromises.

Experimentation

In a project with little change—and thus little creativity—traditional methods of project planning, management, and control work well and are efficient. When change is commonplace, planning requires a shorter horizon, and management and control take on more of a cut-and-try style. Cut-and-try is just another name for experimentation, which could encompass formal or quick experiments, simulations or analysis, prototypes or mock-ups, models, tests, and tryouts.

Not only does experimentation assume a central role in the flexible approach, but recent developments in experimentation technology have made many types of experiments ten to 100 times faster, cheaper, and more effective (Thomke, 2003). Many managers shift to these new tech-

Table 1. Comparison of modular and integral architectures (Source: Copyright ©2007 by John Wiley & Sons; used with permission)

Type of Architecture	Modular	Integral
Characteristics	Chunks are decoupled, operate independently	All portions are interdependent
Example	Desktop Personal Computer	Walkman®*
Advantages	Can change design easier, test independently, reuse portions	Cheaper to make, lighter, more compact
Limitations	Planning time, performance weaknesses, integration burden	Difficult to change, late testing

*Note: * Walkman is a registered trademark of Sony Corporation.*

Table 2. Observe the great differences between traditional and front-loaded prototyping, which open up possibilities for more iterative processes that fit with changing environments (Source: Copyright ©2007 by John Wiley & Sons; used with permission)

	Traditional prototyping	Front-loaded Prototyping
Number of prototypes	Few	Many
When used in development	Late	Throughout
Prototype's objective	Verify	Learn
Prototype cost	High	Low
Prototype build time	Slow	Quick
Prototype attractiveness	Refined	Perhaps crude
Prototype's scope	Broad, vague	Narrow, specific
Departmental orientation	Primarily engineering	Any and all departments

niques and pocket the savings. Others recognize that such great improvements open new process possibilities, as Thomke and others (Smith, 2001) have shown. Specifically, they allow you to run many more experiments, run them much earlier in the development process, and use them for learning and direction rather than their traditional role of verification. Table 2 contrasts a traditional with a so-called front-loaded process. Although this table is specifically for prototyping, it applies similarly for other types of experimentation.

Projects with little change benefit from an established process and known steps to reach a predetermined goal. With lots of change, the development team has little or none of this benefit. It must operate in a more iterative, cut-and-try mode. Experimentation fits this mode perfectly, but it requires a new mode of operating, as shown in Figure 4. This loop starts with a formulated hypothesis for the outcome of the initial experiment and repeats throughout the project. In fact there are likely to be multiple loops (experiments) proceeding simultaneously.

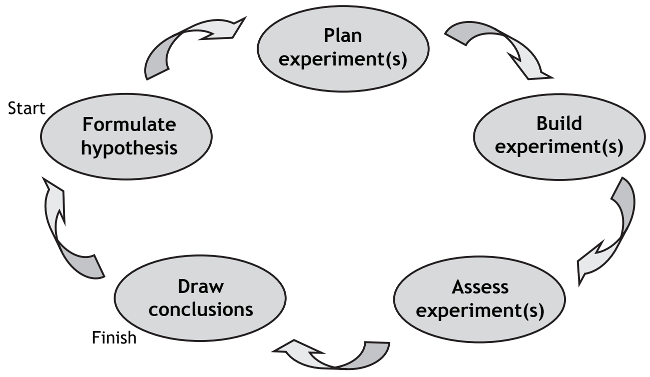
The key part of this loop is the hypothesis, which serves to focus the experiment and enable drawing actionable conclusions. If you wish to test two hypotheses, it is usually best to run two experiments.

Failure plays a critical role in such hypothesis-based experiments. Corporate cultures usually discourage failures while paying lip service to accepting them. But there is something much more fundamental at stake. If your hypothesis is that the experiment will succeed and it does succeed, you haven't learned much (the purpose of an experiment is to learn so that you can move forward). Consequently, if you plan a sequence of experiments with success as the expectation and success as the outcome, progress will be slow (little learning). In contrast, the experiments from which you learn the most are those where a priori expectation is a 50-50 split between success and failure. This is how you should plan each loop for maximum rate of progress.

Thomke makes another important point about failure. He distinguishes between failures and mistakes. Mistakes are poorly planned experiments or ones with uncontrolled extraneous variables. With these, when you reach the end of the loop in Figure 4, you cannot reach clear conclusions, and the experiment is wasted. Failures are valuable, but avoid mistakes.

Regarding the aversion to failure, corporate cultures often discourage the early, quick-and-dirty prototypes that might expose one's ignorance. Although designers are taught in university to make lots of simple prototypes early to explore options, many corporate cultures actually reward refined prototypes made late in the process when most uncertainties are resolved. For instance, see

Figure 4. An iterative experimentation process (Source: Copyright ©2007 by John Wiley & Sons; used with permission)



Kelley and Littman (2001), which is the story of the renowned product development firm, IDEO. Although this book touts quick, early prototypes, the books photos reveal only beautiful, late-stage prototypes.

The conclusion: although supporting failure and quick-and-dirty prototypes are well-known means of facilitating innovation and are given a great deal of lip service, fitting these styles into a corporate environment will require ongoing effort and executive support.

For dispersed teams, experiments present special challenges, because many experiments, by nature, exist in only one location. A test is run in a specific laboratory, and the broken parts that may result exist only there. A prototype is built in only one model shop. A simulation is run on one engineer's computer. Thus, with a dispersed team, there is the added challenge of dispersing experimental artifacts and results. Some experimental tools work well for this. For instance, an especially fast and inexpensive type of rapid prototyping system is called a conceptual modeler (Smith, 2001), or more colorfully, a 3D printer, because when connected to a desktop computer, it "prints" three-dimensional plastic parts. If connected to a remote computer over the Internet, it thus becomes a so-called 3D fax machine that can provide prototypes to remote members of the team in real time.

Set-Based Design

Set-based design comes from Toyota's thoroughly studied "lean" product development process. Because Toyota is generally regarded to have the best automotive development system in the world (Sobek, Ward, & Liker, 1999), this unusual and somewhat counterintuitive system has attracted much attention.

Consider a simple nondesign example to convey the concept. Suppose that Emery (leader), Susan, and Walter need to meet. Emery suggests Tuesday at 10:00, to which Walter immediately objects (out of town). So Emery proposes Thursday at 3:30, but Susan has a conflict then. This continues for several more iterations—and it would be even more difficult if the participants were in different time zones. Such a process corresponds to a conventional so-called point-based methodology. The parallel in set-based operation would be for Emery first to ask Susan and Walter for their calendars for the week. Then he picks a clear time for all of them. Not only are they finished quickly, but Emery has some back-up meeting times in case the primary one fails. Observe that contemporary information technology, such as Microsoft® Outlook®[3], facilitates set-based scheduling, and this works equally well for a dispersed team.

Conventional point-based design technique is based on making *choices*. The designer keeps making choices at forks in the road to improve the design until it is good enough. In contrast, set-based design operates on *constraints*. The designer explores the constraints that limit the design, for instance:

- Which types of solutions won't work?
- What would be too expensive or take too much time?
- What would have reliability or safety problems?
- What would be difficult to manufacture?

The objective is to see how much of the design space is open and where it is open rather than to arrive at a design immediately.

Toyota follows the constraints approach for different reasons than those enhancing flexibility do. For Toyota, exploring the design space results in better, more robust solutions because the point-based approach may proceed into an area that is suboptimal, and they would not know this because they would have no visibility into other areas; they remain unexplored. More importantly, with point-based processes, developers may go out on a branch that does not work out and have to retreat. Thus, although progress may be slower in set-based, they are much less likely to have to back out and find a new route. That is, design convergence is far more likely.

The advantage of set-based design for flexibility is that it defers decisions. Deferred decisions are discussed shortly, but the idea is that developers will not have to reverse a decision that they have not made yet. This maintains our options and our flexibility as long as possible. The team spends its early time not on making decisions that might have to be changed but on assembling information on their constraints and options so that they can make decisions quickly and confidently when the time comes.

Figure 5 illustrates how a set-based design progresses. As the team discovers additional constraints, the design space shrinks at a controlled rate (not too fast, not too slow), by adding constraints progressively to shrink the design space. This continues to leave space to maneuver, although the maneuvering space shrinks to the best solution over time. This convergence rate is Toyota engineering management's primary lever for controlling the set-based process. They want the space to shrink continuously but not too quickly.

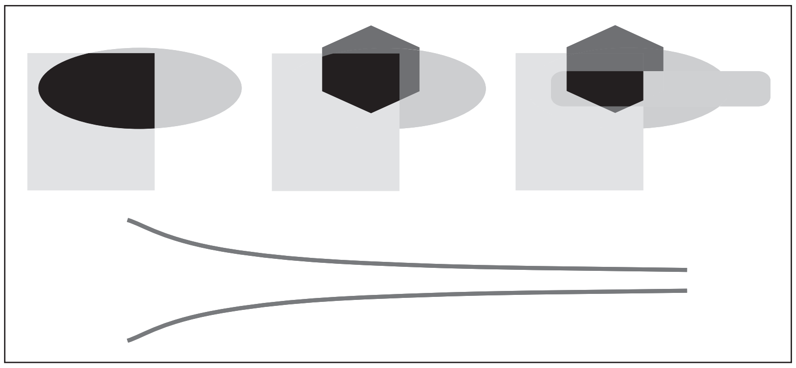
Toyota's application to automobile design is a mature product area. Set-based design is effective for modest uncertainty, as one would encounter in automobile design. When uncertainly is great, however, one could converge into a solution space that turned out to be inadequate. In such cases, one faces so-called unknown unknowns (unk unks), where assumptions of convergence and nearby solutions are often unfounded (Loch, deMeyer, & Pich, 2006). Experimentation is a better approach here.

Dispersed teams pose no major challenges for set-based design. Toyota uses it in their functional organization, in which the members of one development project are dispersed throughout the facility, although generally all in the same city. It is mostly document-based, as practiced at Toyota. The process is not simple, however. They apply it with lots of training and mentoring, which is the company culture.

Product Development Teams

As discussed earlier in the "Background" section, the principles and values of flexible development stem from agile software development. One value that is central to agile development and embedded in every agile methodology is that people are more important than process. This comes directly from the Agile Manifesto (Beck et al., 2001), which compares four values that set agile projects apart from traditional ones. The first of

Figure 5. The team manages set-based design to converge at a desirable rate (Source: Copyright ©2007 by John Wiley & Sons; used with permission)



the four comparisons states that people and interactions are more important than processes and tools. Thus, this "Product Development Teams" section should be placed at the top of this discussion—but you will notice that the Process section falls last for this reason.

This said, this is an entire book about teams, and you will find countless dispersed team insights and tools throughout it. Consequently, this section will be short to cover only two tools that you might overlook otherwise.

The first tool is to establish the team's authority clearly. If you think about the environment in which teams operate, you might notice that the team needs certain types of authority to operate effectively. For example, the team may find that in order to be most creative, it needs to be able to:

- Remove a cynic from the team
- Have an on-demand account at a local model shop
- Have a 3D fax machine (see above) in its Singapore office
- Set its work hours in each facility

Thinking more broadly, you will find that someone in every company needs the authority to make dozens of types of decisions regarding a project, such as a decision to hire staff, a decision to proceed to the next phase of the project, and a decision to invest in buildings and equipment. Try creating such a list specifically for your organization. It is likely to have about 50 types of authority on it.

This list is useful for two purposes. First, you may observe that the team and management are unclear on just what kinds of authority the team has. Often, management assumes that the team has a certain type of authority, but the team is reluctant to move ahead because management has not granted this authority explicitly. Thus, the team is hamstrung by an unclear set of operating rules. The solution is for the team to discuss the aforementioned list of authorities with management and agree explicitly both on the team's areas of authority and on where the team will need to obtain management approval.

Also, you can perform a triage on your list. You will find that some kinds of authority the team clearly already has, so they are not issues. Other types the team does not need or want (they are more work than they are worth, for instance), such as obtaining a new building. But in the middle there are likely to be a few types of authority that the team does not have now but it could operate more effectively if it did have this authority. Then you can discuss this short list with management.

The second tool is partial colocation. By definition, a dispersed team is not colocated. But there are clear advantages to colocation for creative teams. Fortunately, once you appreciate what it is that makes colocation valuable, you can find ways of approximating these characteristics. For instance, if the team has one chance to come together as a team, ensure that this happens at the beginning of the project, when it is most valuable. Try to colocate clusters of team members in a location. If your members are split among three locations, for example, make sure that all team members in each location are colocated. And make sure that your communication media work for you; for example, if delays in e-mail response are slowing the team down, establish team protocols on how quickly an e-mail will receive a reply.

Decision Making

If you dissect the product development process down to its core, you will find that the core process is decision making. The literature has emphasized the few major decisions that come at the end of phases (Deck, 2002), but more important are the thousands of daily decisions made by the team and by individuals as they work their way forward in the design. These decisions cumulatively determine not only the quality and attractiveness of the resulting product but also the performance level of the development team.

How you approach these many decisions should be determined by what you wish to emphasize in your development. If speed of development is your priority, you should find a way to make such decisions quickly. If productivity (products developed per unit of resources) is your objective, accurate decisions (do it right the first time) should drive your process. And if your first priority is flexibility, you should make decisions in a way that facilitates flexibility.

The key to making flexible decisions is not to make them until you have to, because you have more flexibility before you commit to a decision. This leads to the concept of the *last responsible moment*. The last responsible moment is the earliest time when:

- An important option expires
- The decision goes onto the critical path
- The expense of carrying the decision rises dramatically

It is important to recognize that this is not procrastination. On the contrary, one actually works quite hard on the decision from the time that one sees that a decision will be needed until actually making it by collecting information that will support making a better decision when its time comes. That is, you defer the decision itself, but you do not defer the data collection and analysis needed to make the decision. This results in not only more flexibility but also in better decisions, because they are based on fresher, more complete information.

The last responsible moment should not be applied to all decisions. Sometimes the decision is clear (only one reasonable choice) or it can be reversed easily later if necessary. Then the decision can be made early so that it is not a threat to the critical path.

Many analytical and computer-aided tools are available to help make decisions. Savage (2003) provides several, complete with supporting software. One is decision trees, which is a graphical technique that lays out a sequence of linked decisions together with the uncertainties involved so that one can see the complete picture before committing to the first decision. Another tool is Monte Carlo simulation, which allows the decision to be "played out" involving uncertainty to understand what the probability distribution of the outcome will be.

Consensus is an important part of group decision making, especially for a dispersed team. Many decisions require the consensus of several parties with interests in the decision. This is more than just being nice. If you do not take the dissenters' opinions into account when you make the decision, dissenters are likely to undermine future activity related to the decision. Thus, consensus means full agreement to move forward together.

Obtaining such agreement can be difficult for dispersed teams. One tool that is helpful here is a consensus gradient. Everyone involved votes on the proposition under discussion using a carefully arranged scale that goes from full agreement to veto, such as:

1. Completely agree and commit
2. Agree (and commit)
3. Don't disagree (but commit)
4. Some reservations (but commit)
5. Veto

You tally the results first by addressing the vetoes. Any vetoes must be resolved to have a consensus. Then see if there is a preponderance of votes in categories 3 and 4. If so, there is little energy behind the proposal and it is likely to die.

Note that the consensus gradient is easy to use in a dispersed environment, once team members understand how to use it. Beyond this, in a dispersed environment, you will need to ensure that the communication channels are wide open to facilitate good decision-making. For example, ensure that delays in responding to e-mails (mentioned earlier) are acceptable.

Project Management

The contrast between a flexible project and a traditional one is perhaps greater in the project management area than in any other. The *Agile Manifesto* (Beck et al., 2001) illustrates this for software projects, but the contrast also carries over to nonsoftware projects.

Consider project planning. The traditional way of doing this is to plan the whole project in uniform detail from beginning to end. However, agile software developers produce only an overview plan of the whole project initially and then plan the details of each iteration as they enter it. In fact, the team often does the final level of planning within iteration as it proceeds. The process they use is similar to the rolling-wave project planning approach that is applicable to nonsoftware projects (Githens, 1998).

Another contrast with traditional project management is in how the team views corrective action. Traditionally, corrective action is "Documented direction for executing the project work to bring expected future performance of the project work in line with the project management plan" (PMI, 2004). In contrast, because agile and flexible developers place less credence in the overall project plan, they are just as likely to suspect the plan as the execution when execution does not match the plan and correct whichever one they find to be wrong.

More broadly, the two approaches view the project objective—and thus what constitutes project completion—differently. In general, traditional project managers work to complete a list of deliverables that they established at the project outset. When they deliver all of these, the project is complete. Again, agile and flexible development managers place less emphasis on the original list of deliverables, because it, or whatever was influencing it, may have changed. Thus, they must look more fundamentally at delivering value to the customer. This is clearest to see in software information technology (IT) projects, where an actual customer is likely to be on the development team, and the team delivers features in iterations. At the end of each iteration, the team and the customer jointly decide if they have delivered enough value to call the project complete. Notice that this could include more or fewer product features than originally planned, and there is often little commitment to complete the original list. In nonsoftware projects, the product is not so easily divisible into features and value may not be so easy to assess, but the emphasis is still more on delivering value than on predefined deliverables.

The development process is different. One normally views a traditional project as being sequential, with one task leading to the next in a progressive manner. But a flexible process proceeds in a more iterative manner (iteration is actually a part of a traditional innovation project too, but it is often ignored in planning). Observe that Microsoft Project is a popular tool for planning and scheduling traditional projects. However, Project will not allow iteration: if you try to make a task feed back into an earlier task, you will receive an error message in Project. Project has no way to escape from iteration, so it does not allow it.

Finally, project risk management is fundamentally different. In the traditional approach, risk management is an identifiable set of activities that, when done well, begin at project planning stage and are well integrated with other project activities (Smith & Merritt, 2002). In a fast-changing project, the risks change often and initial risk identification is of little benefit, because most risks are unknown at this point. Consequently, the entire development process *is* risk management—the iterations; prototyping, testing, and experimentation; parallel development paths, and project staffing (Loch et al., 2006).

Data management presents a special dilemma for flexible teams. In general, agile and flexible teams eschew detailed documentation and find simple ways to document things. Often, this means using wall charts and sticky notes, together with digital cameras to record them. This is quick and easy, but it has a couple of problems. One is that for complex data and projects that require traceability, such as for product requirements in some regulated industries, such records are difficult to change frequently. It is usually better to invest up-front in building a database for project data.

The other problem is that for dispersed teams, clearly wall charts and sticky notes are not very portable, so team dispersion must shift the documentation balance to more formal means than are needed for a colocated team.

Development Process

The contrast between traditional and flexible approaches carries over into the development process as well. If you ask someone following a traditional approach how they develop products, they are likely to respond with the process they use: Stage-Gate, PACE®[4] (Product And Cycle-time Excellence), or their own proprietary one. That is, the process is the centerpiece of their product development. Those following a more flexible approach might mention a methodology (Extreme Programming or Scrum in agile software development for example), but such methodologies are not centered on the process.

In lack of a process to point to, flexible developers point to a set of values that guide them, such as the *Agile Manifesto* (Beck et al., 2001), or to a set of tools, such as the ones described previously.

It is important to note that the picture is not as black and white as presented. Most developers use a process somewhere between flexible and structured, as Boehm and Turner (2004) describes, and the balance tends to shift during a project from more flexible in the beginning to more structured at the end. This occurs because uncertainty decreases during the project while the amount invested rises, both of which suggest a more structured approach later.

As we move to a more flexible process, the type and caliber of people on the team will also shift. In a flexible environment, people will have to be comfortable with more ambiguity, and at least some of them will also need skills to adapt and create processes as they go, as Cockburn (2002, pp. 14-18) describes.

A flexible process is likely to be heavily dependent on experimentation. If so, particular attention should be paid to the capacity to experiment: testing laboratories, model shops, rapid prototyping machines, and analysis software. This is especially challenging for a dispersed team, because replicating such facilities in multiple locations is costly. Experimentation capacity is critical because if it is not sufficient, experiments will wait in queue for completion. Studies of queues show that time in queue increases dramatically long before one reaches the rated capacity. This understanding is vital for using experimentation effectively. The learning type of experimentation discussed here loses its value if people have to wait to receive the learning. They will just have to proceed in making decisions without it, and then the learning will not support their decisions.

Finally, if you are building a flexible development process, build it up, as Boehm and Turner (2004) recommends, rather than starting with a process and trying to remove items. Although the latter seems attractive and easier, what happens is that, to be safe, people will be conservative in removing something that has had value in the past. Also, it takes a seasoned practitioner to be capable of judging that an item will not be needed. The beauty of the flexible approach is that you can always add later what you didn't notice you needed today!

ASSEMBLING A KIT OF FLEXIBILITY TOOLS

Please reread the section Using Flexibility Tools. You are likely to gain additional insights from it now that you understand each of the tools, and it will be helpful as you assemble the tools to use on a specific project.

Remember that each project is different and thus will require a different combination of the tools. In some cases, a tool may not apply to your project. For instance, project architecture generally is not useful with homogeneous products like paints or plastics. On the other hand, be careful about excluding a tool just because it may be difficult to apply. For example, members of a global product development team located remotely from their customers could easily dismiss customer visits as impractical. However, Morgan and Liker (2006, p. 30) report that a Toyota chief engineer (in Japan) found it so important to experience his customer situation in North America that he explored 50,000 miles of highways in the United States, Canada, and Mexico, and this gave him a great deal of understanding to make trade-offs and changes as development of a new Toyota model proceeded.

These tools generally have costs or other undesirable side effects associated with them, so they must be applied selectively. Use them more on projects where change is likely and the benefits of change will pay off. Identify the portions of a product that are most likely to change and apply them there rather than broadly across the product. As a project moves from its beginning toward market introduction, project complexity and investment increase while uncertainty should decrease, all of which suggest that you should reduce the amount of flexibility as the project progresses.

Be forewarned that simply putting these tools in place is the easy part. More difficult and more critical to long-term success is cultivating the underlying values and culture that support flexibility.

People naturally gravitate to what is comfortable, and uncertainty is uncomfortable. Managers in particular, like to know what is going to happen, even if they have to make up a story to satisfy their need for certainty (Smith, 2005). Consequently, as much as you might wish to enjoy the benefits of flexibility quickly, you are more likely to be effective by starting with a manageable pilot project using some of your most capable, flexible people and expanding slowly as your people and management gain experience with the flexible approach (Smith & Reinertsen, 1998, Chap. 15).

By nurturing the adoption of these tools and approaches, you will develop a greater ability to make changes during development, and this will, in turn, provide a supportive environment for the types of iteration, trials, and exploration that are necessary for creativity to flourish.

REFERENCES

Adams, J. L. (1974). *Conceptual blockbusting*. Stanford, CA: Stanford Alumni Association.

Adams, J. L. (1986). *The care and feeding of ideas*. Reading, MA: Addison-Wesley.

Beck, K. (2000). *Extreme programming explained*. Boston: Addison-Wesley.

Beck, K., et al. (2001). *Manifesto for agile software development*. Retrieved April 16, 2006, from http://agilemanifesto.org

Boehm, B., & Turner, R. (2004). *Balancing agility and discipline*. Boston: Addison-Wesley.

Brown, S. L., & Eisenhardt, K. M. (1998). *Competing on the edge*. Boston: Harvard Business School Press.

Christensen, C. M., & Raynor, M. E. (2003). *The innovator's solution*. Boston: Harvard Business School Press.

Christensen, C. M., Anthony, S. D., & Roth, E. A. (2004). *Seeing what's next*. Boston: Harvard Business School Press.

Clark, K. B., & Fujimoto, T. (1990). The power of product integrity. *Harvard Business Review*, *68*(6), 107-118.

Cockburn, A. (2000). *Writing effective use cases*. Boston: Addison-Wesley.

Cockburn, A. (2002). *Agile software development*. Boston: Addison-Wesley.

Cooper, A. (1999). *The inmates are running the asylum*. Indianapolis, IN: SAMS/Macmillan.

Cooper, R. G. (2001). *Winning at new products*. Cambridge, MA: Perseus.

Cooper, R. G. (2005). Your NPD portfolio may be harmful to your business's health. *Visions*, *29*(2), 22-26.

Deck, M. J. (2002). Decision making: The overlooked competency in product development. In P. Belliveau, A. Griffin, & S. Somermeyer (Eds.), *The PDMA toolbook for new product development* (pp. 165-185). Hoboken, NJ: John Wiley.

Eckes, G. (2003). *Six sigma for everyone*. Hoboken, NJ: John Wiley.

Githens, G. D. (1998, October). *Rolling wave project planning*. The 29th Annual PMI Seminars & Symposium, Long Beach, CA.

Kelley, T., & Littman, J. (2001). *The art of innovation*. New York: Doubleday.

Larman, C. (2004). *Agile and iterative development*. Boston: Addison-Wesley.

Loch, C. K., DeMeyer, A., & Pich, M. T. (2006). *Managing the unknown*. Hoboken, NJ: John Wiley.

Mascitelli, R. (2004). *The lean design guidebook*. Northridge, CA: Technology Perspectives.

Merriam-Webster. (2000). *Merriam-Webster's Collegiate Dictionary* (software ed., Version 2.5). Springfield, MA: Merriam-Webster, Inc.

Morgan, J. M., & Liker, J. K. (2006). *The Toyota product development system*. New York: Productivity Press.

PMI (2004). *A guide to the project management body of knowledge* (PMBOK® Guide, 3rd ed.). Newtown Square, PA: Project Management Institute.

Savage, S. L. (2003). *Decision making with insight*. Belmont, CA: Brooks/Cole.

Smith, P. G. (2001). Using conceptual modelers for business advantage. *Time-Compression Technologies*, *6*(3), 18-24.

Smith, P. G. (2005). Why is agile development so scary? *Agile Project Management Advisory Service* (Cutter Consortium), *6*(9), 1-3.

Smith, P. G., & Merritt, G. M. (2002). *Proactive risk management*. New York: Productivity Press.

Smith, P. G., & Reinertsen, D. G. (1998). *Developing products in half the time*. Hoboken, NJ: John Wiley.

Sobek, D. K., II, Ward, A. C., & Liker, J. K. (1999). Toyota's principles of set-based concurrent engineering. *Sloan Management Review*, *40*(2), 67-83.

Thomke, S. H. (2003). *Experimentation matters*. Boston: Harvard Business School Press.

von Hippel, E. (1994). *The sources of innovation*. New York: Oxford University Press.

ENDNOTES

1 This author believes that *virtual,* in the context of teams, is poor terminology, because

virtual means, "being such in essence or effect though not formally recognized or admitted." (Merriam-Webster, 2000) Teams are all about performance, and such cloudy terminology weakens teams' performance orientation. Therefore, *dispersed* is used in place of *virtual* in this chapter.

2 Stage-Gate is a registered trademark of the Product Development Institute.

3 Microsoft and Outlook are registered trademarks of Microsoft Corporation.

4 PACE is a registered trademark of PRTM.