BEST PRACTICES WHITE PAPER

Introducing Agility into a Phase Gate Process

Jenny Stuart, Vice President of Consulting, Construx Software

Version 1.1, June 2011

Contributors

Earl Beede, Senior Fellow

Jerry Deville, Senior Fellow

Tom Landon, Senior Fellow

Phase Gate processes are common in mature software development organizations that want to support continuous evaluation of products and projects to ensure that it makes sense for the business to continue its investment in them. Some organizations want to introduce additional agility into their gating process while maintaining its oversight and governance. This white paper outlines the major considerations and keys to success to introducing agility into a well defined Phase Gate process.

Contents

Introduction

Phase Gate processes are common in mature software development organizations, in organizations that have regulatory compliance requirements, when the software is just part of a larger product that must be managed within a phase gate process, or when there are strong governance processes. Phase Gate refers to the practice variously called Phase Gate, Stage Gate, or other similar names.

Phase gate is a practice in which products move through distinct phases or stages separated by key decision points referred to as *gates*. Each gate occurs at a point in time where the management team needs to make a decision about whether to proceed with the product. Approval to proceed past many of the gates involves releasing funds for work on the product—often in the form of a project—to continue. Other gates enable control over the end-to-end life cycle of the product. For example, there is typically a gate that supports making the decision to retire the product or specific versions of the product.

A Phase Gate process provides a high-level structure for monitoring project performance, and it provides multiple opportunities to assess the project and ensure it is meeting the needs of the business. The intent is to support continuous evaluation of the project's risk, cost, and return on investment to ensure that it makes sense for the business to continue its investment.

While some people believe that a Phase Gate process requires the organization to use a waterfall life-cycle model, in which all work occurs sequentially, this is not the case. Some organizations find that introducing a Phase Gate process accidentally results in a waterfall software development approach and decide to return to more agile approaches. Other organizations that have been using a waterfall life cycle become interested in introducing a more Agile software development process to achieve a business goal, such as improving the organization's ability to respond to a changing marketplace.

This white paper outlines the major considerations and keys to success when introducing a more Agile software development life cycle within a well-defined Phase Gate process. It provides several examples of how other organizations have mapped in Agile approaches in this situation.

Considerations

Identify the Agile Methodologies

Before introducing Agile into a Phase Gate process, it is important to understand which software development approach (or set of approaches) need to be mapped into the process. Numerous software development approaches fit under the umbrella of Agile—from Evolutionary Delivery to Extreme Programming (XP) to Scrum to Lean—and it can be difficult to understand which approach is best for any specific organization.

Construx's *Optimizing Agile for Your Organization* white paper can be a good place to begin when an organization has not yet selected its Agile approach. Organizations that are already piloting or using Agile approaches should determine which ones are in use and whether they want to support all the approaches, a subset of approaches, or a single approach.

Identifying the current or candidate Agile approaches will help the organization see that understanding the processes, practices, and overall flow is necessary to map the approach into a Phase Gate process.

Determine Compliance Constraints

Many organizations find that there are a set of constraints in its Phase Gate process that must be adhered to. These are necessary for the business to operate successfully and include items such as capitalization requirements, SOX compliance requirements, FDA requirements, and so on. Beyond these, there are often perceptions within the organization of constraints that would impede an Agile approach but that do not actually exist.

While mapping Agile into an existing Phase Gate process, it is important to understand what aspects of the process are hard constraints and where there are perceived constraints and misperceptions. As part of this attempt to understand the constraints, it is useful to ask multiple people throughout the organization questions such as the following:

- What compliance constraints must we abide by while using more Agile practices and lifecycles?
- Are there compliance obstacles that you had to overcome/address to introduce an Agile approach on your project? If so, what are they and how did you address them?
- What compliance obstacles are you still facing?

Examples of constraints and compliance obstacles that have been expressed by some of the organizations that Construx has worked with are described in Table 1.

Table 1 *Agile Compliance Obstacles*

Expressed Obstacle	Outcome
Compliance is a "one-size fits all" model.	Compliance did not require a single model. The organization required that there be a common language and framework that was flexible enough to meet the needs of its various development teams while clearly supporting compliance.
There is a set of required templates.	The content was required; the format was not. There was a hard compliance constraint that stated specific template documents were available, but use of the templates was not required. The templates were preferred by the compliance organization because they made compliance audits easier to perform, but their use was not mandated. However, the teams had to be able to demonstrate compliance whether or not the templates were used. A set of "Agile" project templates for the required compliance documentation was created, but again, the use of the templates was not mandated by the compliance organization.
Capitalization requirements prohibit frequent releases.	Frequent releases could be supported, but it had to be done in a way that did not trigger an official release from a capitalization perspective. It was possible to have a project with incremental releases to beta users within the context of a single project. The teams could conduct a series of internal releases that were not released to the customers. Instead of being released to the customers, the product was brought to a releasable state to support iterative development.
FDA compliance regulations impede agility.	It was possible to incrementally build the materials that were needed to ensure compliance. The FDA compliance regulations were concerned with patient safety and focused on ensuring that all the necessary documentation was in place to validate that the product was doing what it was supposed to do and not doing what it should not do. The teams were able to incrementally build the documentation—including requirements, test cases, traceability matrixes, unit test logs, etc.—that were necessary to ensure compliance.

A first step that an organization should take when seeking to adopt a more Agile approach within a Phase Gate process is to understand the real and perceived compliance constraints and determine how it will address each.

In some cases, the organization will simply need to address misperceptions. In other cases, the organization will need to modify the selected Agile approach to incorporate or address hard constraints. On occasion, the compliance constraints will limit the organization to agility in only part of the product development process. Most organizations have found that they can at a minimum introduce a more incremental approach during the design and construction phase of the life cycle.

Understand the Objectives of Each Gate

The gates within a well-defined Phase Gate process typically occur at points in time when a business wants to make a decision about whether to proceed with the project or not. They occur as the organization has more and more detail about the cost, timeframe, and return on investment for the project.

When introducing an Agile approach into a formal Phase Gate process, it is important to understand the objectives of each gate. Some keys items to understand are

- What is the spirit of each gate?
- What business decision does the organization want to make?
- What are mandatory controls, and what controls are optional?
- What deliverables are required at each gate and to what level of completeness?
- What quality assurance activities are required at each gate?
- Is it possible to pass through gates multiple times?
- Is the gate used to create confidence that the project is going well or for legal compliance?

By understanding the objectives for each gate, it is possible to begin discussions of the hard compliance constraints and the ways that Agile may need to be tailored or tuned to support the business objectives. This detailed understanding can clarify which elements are nice to have and which ones will not be required for Agile projects. It can enable discussions to occur of ways in which Agile projects are passed through gates multiple times with an increasing level of completeness that supports governance and enables agility.

Example Approaches

This section outlines two example approaches to mapping Agile approaches into a Phase Gate process. Each example maps into the same existing waterfall-based Phase Gate process. The examples outlined here show an increasing level of agility, with the first example iterating only during the main development phase.

Phase Gate Process Overview

Before discussing the examples, a brief summary of the Phase Gate process is provided. The process includes five phases, with a gate between each phase. Each gate is a decision point where executive management and major stakeholders make a go/no-go decision about proceeding with the project. The process is shown in Figure 1.

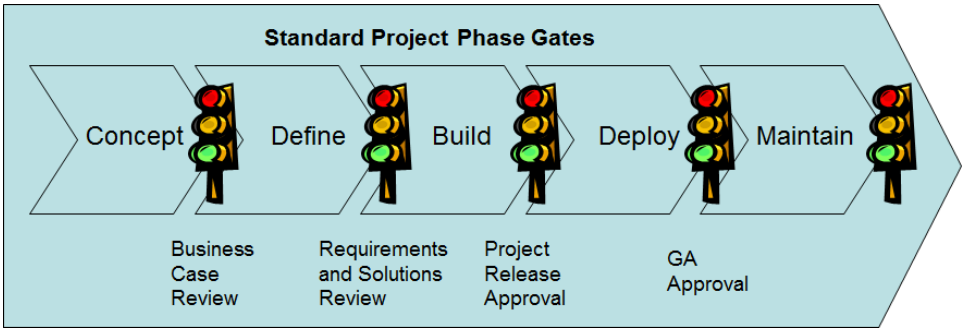

Figure 1 *Product Development Phase Gate Process.*

The major activities that occur in each phase are as follows:

- **Concept.** Determine if the organization should pursue the proposed concept or abandon it in favor of another concept that would provide greater value. Typically, this phase involves developing and analyzing the project's business case, which includes the business objectives as well as the rough-order-of-magnitude budget and return-on-investment calculations. At the conclusion of this phase, the organization decides if it is worth investing further time and resources in analyzing the idea.
- **Define.** Refine and expand the business case to determine if the organization should or should not build the product/system. Typically, this phase involves refining the requirements, technical constraints, and architecture and creating an estimate. The work is focused on gaining a shared understanding of what needs to be built, how it will be built, the resources that will be required, and the overall timeline. At this time, the business case is also reevaluated to see if it is still valid. At the conclusion of this phase, the organization decides to commit to the project plan, cancel the project, or defer the decision until additional work is accomplished.

- **Build.** Develop the product/system and associated support material (e.g., training, support documentation, market collateral). Typically, this involves more detailed analysis, design, construction, testing, and documentation. This phase often includes user acceptance testing or other validation activities. At the conclusion of this phase, the organization decides whether to release the system/product.
- **Deploy.** Release the product/system to the users, and turn over responsibility to the support team. This phase often includes beta-testing and field trials to validate the system works in a broader set of configurations. At the conclusion of this phase, the organization decides whether to make the product widely available to its clients.
- **Maintain.** Perform the ongoing maintenance and support of the released software, including items such as hot fixes and patches. The Maintain phase is typically not included as part of the software project; instead, it is part of daily operations. Typical activities include help desk support, issuing hot fixes and patches, and minor enhancements. At the conclusion of the phase, the version (or in some cases, the entire product) will no longer be sold and supported.

The following two examples outline how a more Agile software development approach can be mapped into this product development oversight process.

Example 1: Iteration During the Development Phase

This example describes how an organization can introduce agility during the Build phase. It is most closely aligned to the Stage-Delivery life-cycle model, in which the organization defines the software concept, analyzes the requirements, and creates a solution/architecture design of the whole program before beginning construction. The Build phase is then broken into increments that include detailed design, coding, and testing. The approach and how it maps into the Phase Gate process is shown in Figure 2.

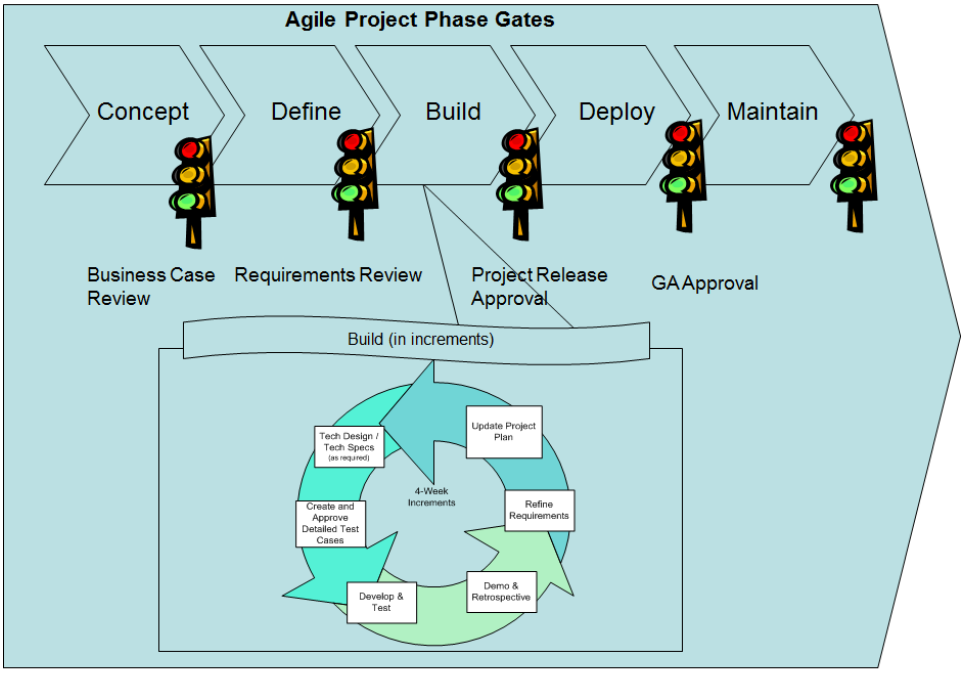

Figure 2 Agility throughout the main development phase.

This approach does not change the timing or decisions made at any of the gates described at the beginning of this section. It does provide more visibility into the progress during the Build phase.

An approach like the one described here is a good fit for organizations that have

- Extremely rigorous compliance constraints
- A high need for long-range predictability of the combination of cost, schedule, and features
- Projects with well-understood and highly stable requirements

Other organizations might want more agility than what is described here. The next example outlines a mapping where agility is introduced earlier in the software development process.

Example 2: Iteration from Requirements Development

This example describes how an organization can introduce agility after the business case has been approved. It is most closely aligned to an Evolutionary Delivery life-cycle model, in which preliminary requirements and architecture work is done but construction begins while there is still significant uncertainty. The approach and how it maps into the Phase Gate process is shown in Figure 3.

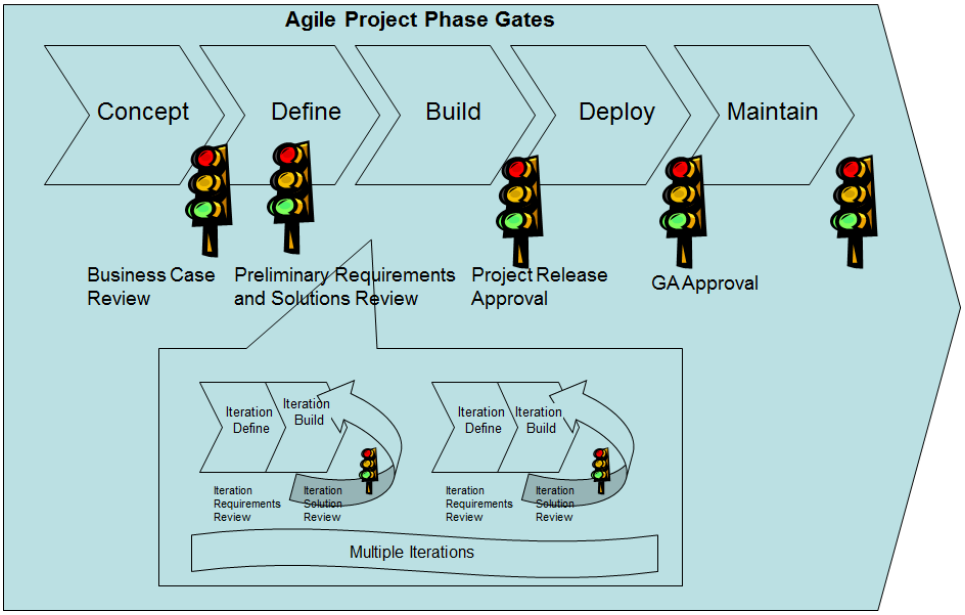

Figure 3 *Agility throughout the project.*

This approach does change the timing and types of decisions made at the Define gate described at the beginning of this section. Rather than created complete functional and nonfunctional requirements and a solutions document, Agile projects refine the design and requirements to the point where the development team has a reasonably clear idea of the overall requirements and high-level design before beginning development.

Before proceeding past the Define gate, the Agile project team conducts a "Preliminary Requirements and Solution Review." The intent of this gate is to reduce the risk that the high-level requirements are incorrect and that the current design decisions might lead the project team down dead-ends. The project team and management should agree that there is sufficient information so that sensible business decisions about whether to proceed can be made.

During each of the increments, the team and customer representative conduct "Requirements Reviews" of the requirements as they are defined and refined. The architecture is refined, updated, and reviewed as needed. These reviews occur internally within the team. However, if the project is long, risky, or mission critical, the team may need to pass through the Define gate again as the process progresses.

Each increment also includes all of the construction, unit testing, and initial system testing of the features in that increment. It may also include more complete system testing such as load testing, performance testing, stability testing, and so on. However, many organizations find that these activities cannot occur in each increment, and instead they are completed in increments later in the development process or when the project is getting ready to pass through the Define gate. Construx's *Optimizing Agile For Your Organization* white paper discusses these considerations in more detail.

An approach like the one described here is a good fit for organizations that have

- A need to respond to changing market needs and are willing to change the final feature set throughout the project
- Unknown or uncertain requirements and need input from internal or external customers throughout the project
- A fixed delivery date and are willing to modify the feature set to achieve it

Keys to Success

Many organizations are interested in either modifying an existing Phase Gate process to support Agile or mapping an Agile software development approach into an existing process. In both of these cases, Construx offers the following keys to success:

- **Understand the objectives of using an Agile approach.** From increasing an organization's ability to respond to a changing market to improving internal progress visibility, there are many good reasons for introducing more Agile software development approaches. Understanding the business objectives that each individual organization has for increasing agility helps management and individual project teams map their Agile approach into the process. When the individuals on project teams, in management, and in the compliance organization are all aligned with regard to their understanding of the objectives, they can find ways to balance hard compliance constraints with those objectives.

- **Adapt Agile to fit the needs of the organization.** Most organizations with an existing Phase Gate process have requirements for project oversight and governance that are not addressed in by-the-book Agile approaches. There is typically a set of required documentation that must be developed for legal or regulatory requirements. When adopting Agile and mapping it into a Phase Gate process, it is very appropriate to extend and expand the by-the-book Agile approach to include activities, deliverables, and roles that are needed by the adopting organization to be successful in its business.

- **Engage the compliance organization early.** Some organizations have compliance departments or compliance officers within each department. In either case, it is important to engage representatives from the compliance function as early as possible. The compliance organization can provide guidance on which elements of the Phase Gate process are hard constraints that must be met, where there is flexibility, and if it may be possible to make minor modifications to the process to support the desired level of agility.

  In most cases, it is helpful to provide the compliance organization with the business motivations and objectives for introducing a more Agile approach. In some situations, it is useful to have them participate on an Agile pilot project to help them see how Agile will be used in the organization, determine changes that are necessary to the selected Agile approach to support compliance, and identify if there are opportunities for modifications in the Phase Gate process that will support compliance objectives and better support Agile development.

- **Pilot before deploying widely.** Pilot projects are an opportunity to try a Phase Gate Agile software development project to determine what works well, what needs to be modified, and what additional changes are necessary to support the compliance and governance requirements. Pilots are a chance for an organization to learn in a small deployment and tailor the approach before a wider deployment occurs.

- **Provide guidance.** Many organizations find that there are numerous misperceptions about the rigidity and constraints imposed by a Phase Gate model. To help teams understand how to conduct Agile projects under the governance of a Phase Gate process, it can be useful to provide guidance. Some organizations create lightweight documentation that shows how Agile maps into the Phase Gate model, describes when gates occur, and explains whether they are different from the waterfall gates, required deliverables, and so on.

- **Build a feedback loop.** As an organization begins to run Agile projects within a Phase Gate process, there are often modifications based on lessons learned that need to be made to the Agile approach or to the preliminary mapping into the Phase Gate process. It generally takes the experience of applying Agile to a few projects to tailor and tune the approach so that it provides the desired agility and meets the visibility and governance requirements of the Phase Gate process. It is important to establish feedback loops so that there is a way to capture lessons learned and modify the Agile approach and mapping as they are refined.
- **Focus on business benefits.** In some cases, the desire to introduce agility comes from the development organization. When this occurs, it is important that the suggested changes be communicated in a way that clearly supports the overall business objectives for the organization and fits within the existing Phase Gate process. The reason for being more agile must be tied to business benefits that people outside the development organization can understand. It is also helpful to use terminology that they are familiar with, rather than introducing new technical language that others outside the development organization may have a harder time understanding.

Combining Agile software development and a Phase Gate governance model is possible and is currently occurring in many organizations. These keys to success are offered to help organizations implement this capability.

Contributors

Jenny Stuart, VP Consulting

jenny.stuart@construx.com
+1(425) 636-0108

Earl Beede, Senior Fellow

earl.beede@construx.com
+1(425) 636-0114

Jerry Deville, Senior Fellow

jerry.deville@construx.com
+1(425) 636-0118

Tom Landon, Senior Fellow

tom.landon@construx.com
+1(425) 636-0113

About Construx

Construx Software is the market leader in software development best practices training and consulting. Construx was founded in 1996 by Steve McConnell, respected author and thought leader on software development best practices. Steve's books *Code Complete*, *Rapid Development*, and other titles are some of the most accessible books on software development with more than a million copies in print in 20 languages. Steve's passion for advancing the art and science of software engineering is shared by Construx's team of seasoned consultants. Their depth of knowledge and expertise has helped hundreds of companies solve their software challenges by identifying and adopting practices that have been proven to produce high quality software—faster, and with greater predictability. For more information about Construx's support for software development best practices, contact us at consulting@construx.com, or call us at +1(866) 296-6300.

Construx®

SOFTWARE DEVELOPMENT BEST PRACTICES

Construx Software Builders, Inc.
10900 NE 8th Street, Suite 1350
Bellevue, WA 98004
U.S.A.