August 2012

# Solution Development Approach Guidance

Protecting America's Pensions

Pension Benefit Guaranty Corporation

Version 1.0

## Table of Contents

Introduction ..... 1
What is the Purpose and Scope of this Document? ..... 1
How Do I Use this Document? ..... 1
Solution Development Approaches and the ITSLCM ..... 2
What is a Solution Development Approach? ..... 2
How do Solution Development Approaches Relate to the ITSLCM? ..... 2
How is a Solution Development Approach Selected? ..... 2
Hints and Tips For Selecting a Solution Development Approach ..... 3
Solution Development Approach Overviews ..... 5
Waterfall ..... 6
What are the Strengths of a Waterfall Development Approach? ..... 6
What are the Weaknesses of a Waterfall Development Approach? ..... 7
When is Waterfall an Appropriate Solution Development Approach to Select? ..... 7
When is Waterfall Not an Appropriate Solution Development Approach to Select? ..... 7
Incremental ..... 8
What are the Strengths of an Incremental Development Approach? ..... 9
What are the Weaknesses of an Incremental Development Approach? ..... 9
When is Incremental an Appropriate Solution Development Approach to Select? ..... 10
When is Incremental Not an Appropriate Solution Development Approach to Select? ..... 10
Iterative ..... 11
What are the Strengths of an Iterative Development Approach? ..... 11
What are the Weaknesses of an Iterative Development Approach? ..... 12
When is Iterative an Appropriate Solution Development Approach to Select? ..... 12
When is Iterative Not an Appropriate Solution Development Approach to Select? ..... 12
Agile ..... 14
What are the Strengths of an Agile Development Approach? ..... 14
What are the Weaknesses of an Agile Development Approach? ..... 15
When is Agile an Appropriate Solution Development Approach to Select? ..... 15
When is Agile Not an Appropriate Solution Development Approach to Select? ..... 16
Prototyping ..... 17
What are the Strengths of a Prototyping Development Approach? ..... 18
What are the Weaknesses of a Prototyping Development Approach? ..... 18
When is Prototyping an Appropriate Solution Development Approach to Select? ..... 18
When is Prototyping Not an Appropriate Solution Development Approach to Select? ..... 19
Spiral ..... 20
What are the Strengths of a Spiral Development Approach? ..... 21
What are the Weaknesses of a Spiral Development Approach? ..... 21
When is Spiral an Appropriate Solution Development Approach to Select? ..... 21
When is Spiral Not an Appropriate Solution Development Approach to Select? ..... 22
Conclusion ..... 23
Appendix A ..... 24
Further Reading ..... 24

## Revision History

| Version | Date | Change Description | Author |
|---|---|---|---|
| 1.0 | August 2012 | | Project Management Division |

# Introduction

## What is the Purpose and Scope of this Document?

The Solution Development Approach Guidance document is a resource for assisting Pension Benefit Guaranty Corporation (PBGC) federal staff and contractors in selecting an approach for developing and implementing IT solutions that will meet business needs, fulfill business requirements, and close performance gaps. IT solutions include, but are not limited to, Commercial Off-the-Shelf (COTS) solutions, Government-Off-the-Shelf (GOTS) solutions, customized development solutions, enhancements to existing IT solutions, hosted solutions, infrastructure solutions, or any hybrid of the previously mentioned. Also, note that for the purpose of this document, the term "development" refers to custom development, COTS/GOTS configuration, and/or a hybrid.

Although determined in the Planning Phase, the selected development approach will be applied to the tasks in the Solution Implementation Phase of the Information Technology Solutions Life Cycle Management (ITSLCM) framework for the purpose of successfully implementing an IT solution.

The solution development approaches outlined in this document include some of the most widely used within the IT Industry; however, this document does not contain an exhaustive list of all of the approaches that teams[1] can use at PBGC.

## How Do I Use this Document?

The Solution Development Approach Guidance is organized into four sections:

- **Section 1: Introduction** - provides a description of the document's purpose, scope, and how to use the document.
- **Section 2: Solution Development Approaches and the ITSLCM** - provides an overview of what a solution development approach is, how a development approach is used in conjunction with the ITSLCM, and what should be considered before selecting a specific development approach.
- **Section 3: Solution Development Approaches** - provides an overview of some of the solution development approaches used within the IT industry, known strengths and weaknesses of each approach, and when each approach may or may not be an appropriate selection.

  Look for "stars" to indicate the known strengths of a development approach.

  Look for "lightning" to indiciate the known weaknesses of a development approach.

- **Section 4: Conclusion**

[1] For the purpose of this document, "teams" refers to the collective group of individuals that have a role in implementing an IT solution and may be involved in the selection of the solution development approach.

## Solution Development Approaches and the ITSLCM

### What is a Solution Development Approach?

A solution development approach refers to the relationship between tasks, processes, and cycles that are cultivated and specifically ordered for the purpose of delivering an IT solution. Development approaches provide a recognized method of structuring the necessary components of IT development and, when used appropriately, can assist in managing constraints, delivering required functionality, and meeting business needs.

A wide variety of development approaches have evolved over the years in response to the growing complexity of IT software and systems. Each of the established approaches have recognized strengths and weaknesses that can contribute to the success or challenges while implementing IT solutions. As each IT solution is unique, it is critical that the selected development approach correlates with the solution's constraints and variables. Choosing an approach that is inappropriate for the solution or the expectations of stakeholders can cause a delay in schedule, increase in cost, and may ultimately fail to solve the business problem. Comparatively, choosing the correct approach can assist in delivering a solution that closes performance gaps and meets the needs of the user community, on time and within budget.

### How do Solution Development Approaches Relate to the ITSLCM?

PBGC's ITSLCM framework is designed to accommodate different approaches of developing and implementing IT solutions. While the framework identifies the essential tasks, such as requirements analysis, design specification, and testing, the selected development approach will help teams define the relationship between required tasks, team member roles and responsibilities, and deployment cycles.

### How is a Solution Development Approach Selected?

During the development of the IT Program Release Plan[2], the Integrated Program Team may make an initial determination of which solution development approach will be applied to upcoming releases and projects. Although the IT Program Release Plan is developed [and updated annually and/or as needed] at the program planning level, the associated IT and Business Project Managers should be consulted and involved in the initial discussions and planning. Involving more members of the IPgT, such as the IT and Business Project Managers, will enhance the connection between each project and release and the goals, strategic alignment, management strategies, and artifacts of the overall program.

If the appropriate solution development approach cannot be determined during the development of the IT Program Release Plan, the selection can be postponed until more in-depth project planning tasks begin.

If an initial selection of the solution development approach was made during the development of the IT Program Release Plan, the selection should be reviewed by the Integrated Project Team (IPT) during the project's planning. The federal IT and Business Project Managers need to understand the particular nuances of the selected approach and confirm the resources required to make the implementation a

[2] The IT Program Release Plan details the releases and projects that are scheduled to be implemented over the period of time that covers the current year through the following Budget Year (as indicated in OMB Exhibit 53 and 300).

success are available. For instance, if an iterative or agile approach has been selected, resources from the business will need to be heavily involved (in comparison to a linear, waterfall approach) throughout the solution's development to provide feedback and participate in testing activities.

If outside resources will be or have been contracted to support the solution's implementation, additional steps may need to be taken to ensure all appropriate stakeholders agree to and understand the selected development approach. If the development approach selection has been finalized prior to procuring outside resources, language may need to be included in the appropriate contract documents to help ensure vendors have an accurate understanding of expectations and resources required. If the development approach selection has not been finalized prior to procuring outside resources, the contractor project manager should be involved in any relevant conversations and decisions.

Contractor staff may also recommend a particular development approach in their proposal or can be asked to recommend a development approach once on site. While contractor staff may bring skills and expertise in executing a particular development approach, it is imperative that the federal staff responsible for a solution's implementation understand and have the resources to implement the approach as they are ultimately accountable for ensuring requirements are met and performance gaps are closed.

There are a variety of reasons as to why the initial selection of a solution development approach, which is documented in the IT Program Release Plan, may be changed. Regardless of the reason for the change, the program's IT Program Release Plan needs to be updated to reflect the agreed upon change.

## *Hints and Tips For Selecting a Solution Development Approach*

Before selecting a solution development approach, consider the following -

- Determine if there is a business need to immediately (or as quickly as possible) implement a portion of the solution's requirements, which have been identified as high priority, safety or security critical, etc. An approach that supports staged deployments will allow a team to deliver requirements based on their priority.
- Ensure that all team members and appropriate stakeholders understand the approach and the correlating responsibilities. User involvement, development milestones, and certain tasks vary by approach.
- Think about the team that will be involved with developing and implementing the solution. Consider individual team members' experience, interpersonal skills, and work load, as well as the size of the team and work location. A team that is small, that works in relative physical proximity, and is led by an experienced Project Manager may be more apt at successfully utilizing an agile approach. In comparison, a large team comprised of individuals with less experience or have competing responsibilities may be better suited to a more traditional, linear approach, such as waterfall.
- Consult the Release Manager and the Application & Infrastructure Impact Meeting (AIIM) infrastructure calendar prior to selecting a development approach for any planned outages and to ensure testing resources will be available. The number of projects/releases and high priority or risk solutions may affect testing and release schedules. Communicating the proposed

approach and/or presenting the solution's goals at AIIM may also help decrease the possibility of unidentified impacts.

- If the development approach is selected after acquiring contractor support, the Project Manager should ensure there are not any contract constraints, such as mandatory signoffs and acceptance dates that would affect the selection of a specific development approach. If the IT or Business Project Manager is not the Contracting Officer's Representative (COR), he or she may need to be consulted as well.
- The user community's familiarity with the proposed solution must be considered. If users are unfamiliar or the solution requires new technology, their involvement or feedback throughout development may increase the likelihood the solution solves the problem and meets the expectations and needs of the users.
- Estimate the level of effort that will be required to define and analyze the solution's requirements. If the requirements are relatively straightforward and understood by the appropriate stakeholders, multiple cycles of analysis and refinement may not be necessary. For example, a planned maintenance release for a solution already in production may be relatively straightforward for stakeholders to understand. In comparison, if the solution is new, supports a large number of users, and/or has complex interrelationships with data, security, etc., a team may need to utilize an approach that supports continual requirements analysis and allows users to provide ongoing feedback during design and development.
- Review the solution's scope, budget, and schedule. If a small budget and limited timeframe has been allocated to the solution's implementation, a team may want to avoid an approach or technique that is traditionally associated with high costs, such as Agile.
- Certain *type*s of solutions, such as COTS, hosted, maintenance, etc., may be better managed with some development approaches over others. For example, a COTS product may support a successful iterative approach, while a maintenance release may be best implemented with a linear approach.
- Variables external to the solution itself should be taken into account. For example, changes to or new government mandates, political climate, changing technology, vendor dependencies, etc. may affect a team's ability to successfully adhere to a particular development approach.

## Solution Development Approach Overviews

The following section provides an outline of various solution development approaches and will provide readers with a general overview of each approach, some of the known strengths and weaknesses of each of the approaches, and variables or characteristics of a solution that may suggest a particular approach is or is not an appropriate selection. While the solution development approaches outlined in this section are among the most commonly used to develop IT solutions, they do not represent an exhausted list of all approaches that may be used during the Solution Implementation Phase of the ITSLCM.

The following solution development approaches are outlined in this section:

- Waterfall
- Incremental
- Iterative
- Agile
- Prototyping
- Spiral

## Waterfall

Waterfall is a linear solution development approach. The waterfall approach is based on teams adhering to a sequential series of tasks, reviews and gates, and deliverables that must each be completed and approved before moving to the next. The waterfall approach emphasizes defining and finalizing all of a solution's requirements and design specifications prior to beginning development tasks.

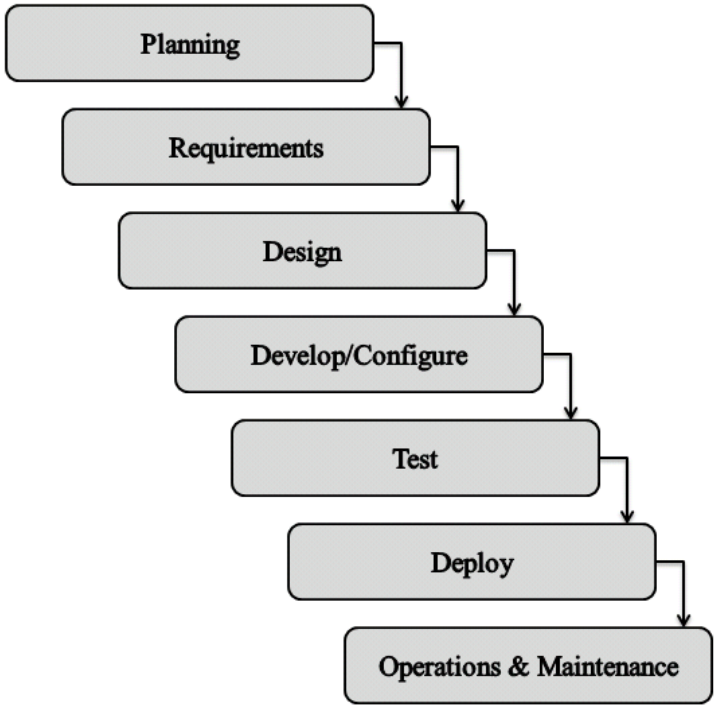


**Figure 1: Waterfall**

*In waterfall development, tasks occur in a top-to-bottom linear line, like a waterfall.*

### *What are the Strengths of a Waterfall Development Approach?*

- After the requirements and design specifications are finalized, resources from the business and user community can be conserved while the development team develops/configures the solution.
- The emphasis on documentation helps to ensure traceability and constancy throughout a solution's lifecycle.
- Comprehensive documentation and progress reviews between tasks may help facilitate communication among stakeholders and can be useful if team members are expected to fluctuate.
- The systematic transition between tasks, deliverables, and reviews assists in managing and maintaining constraints, resources, and stakeholder expectations.
- As defined tasks and deliverables need to be completed and approved before teams proceed to the subsequent one, the solution's progress can be measured with definitive milestones and accomplishments.

### *What are the Weaknesses of a Waterfall Development Approach?*

- The solution is not usually tested by users until it is almost fully developed. As a result, unidentified or misunderstood requirements may not be detected until the solution is almost fully developed, which can result in expensive or time consuming corrective actions.
- Stakeholders do not usually "see" the solutions features or tangible progress throughout the development process.
- The structured sequence of tasks makes it difficult to respond to changes. As the waterfall approach discourages changes once a task has been completed or a document has been approved, it is often a time-consuming and expensive exercise to revert back to a previous phase.
- Clear division of responsibilities and separation of work between the business and developers may lead to communication gaps or misunderstandings.
- Inflexible and tight controls prevent teams from moving to the next task before the previous one or documentation is approved, which can cause team members to become idle while waiting for approval.

### *When is Waterfall an Appropriate Solution Development Approach to Select?*

The following variables indicate that waterfall may be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The requirements can be comprehensively articulated and are stable at the onset of the project. The user community is able to understand the design specifications.
- The IT Project Manager or project team is less experienced and would benefit from set milestones and a structured approach that would assist in managing the schedule, cost, resources, etc.
- The solution requires comprehensive, complete documentation.
- Resources and testers from the user community are unable to provide continual feedback and input during the development of the solution.
- The development and/or business team members are expected to fluctuate during the development of the solution.

### *When is Waterfall Not an Appropriate Solution Development Approach to Select?*

The following variables indicate that waterfall may not be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- There is a business need to immediately implement the solution's high priority requirements, while other requirements can be incrementally added to the solution in later releases.
- Requirements cannot be wholly articulated or will require multiple cycles of analysis and refinement.
- Some requirements may fluctuate or are based on unsteady variables, such as budget changes or technology updates.
- The solution has lots of impacts and/or dependencies and needs to be tested throughout development.
- The solution involves new technologies or real-time systems.

## Incremental

Incremental is a combination of a linear and an iterative solution development approach. Incremental is based on the need to reduce risks and incorporate user feedback by breaking development into several small increments. In each increment, new functionality is integrated into the overall solution in order to meet the prioritized requirements based on the needs of the business and user community. Each increment encompasses a planned set of requirements, which allows the solution to be tested and for users to provide feedback throughout the development process.

There are various methods that can be used to incrementally develop a solution. Teams can select to make each increment a mini-waterfall, meaning that each individual increment contains an abbreviated version of the tasks included in the waterfall development approach. Teams may also select to define and obtain approval of the solution's requirements and design specifications before beginning a series of develop-and-test increments.

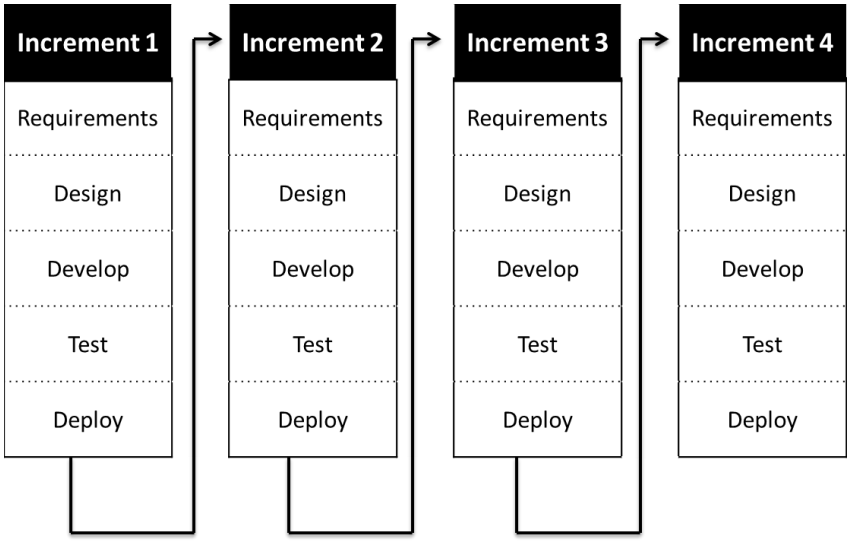


**Figure 2: Incremental**

*In this version of incremental development, teams should complete an increment, which contains abbreviated versions of the tasks included in the waterfall development approach, before moving on to the following increment.*

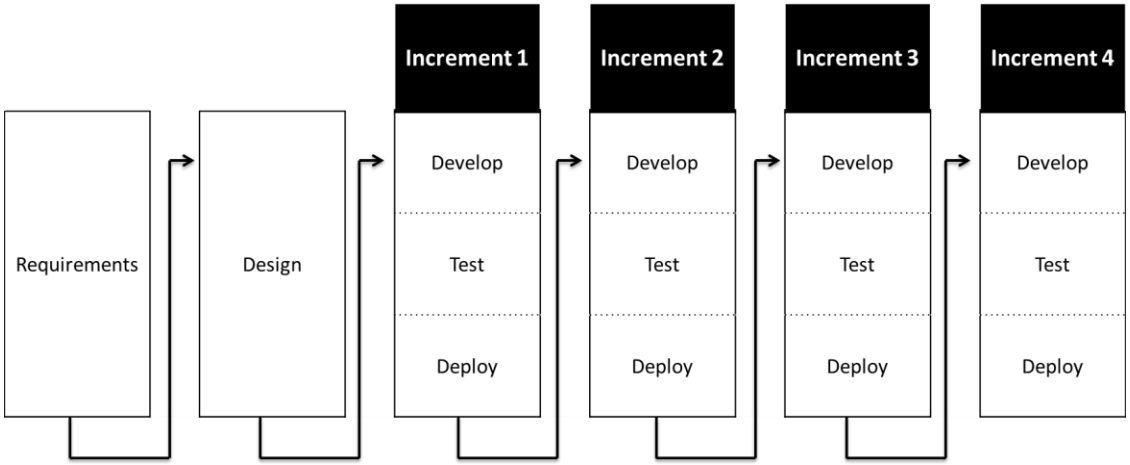


**Figure 3: Incremental**

*In this version of incremental development, a solution's requirements and design specifications should be defined before teams begin individual develop-and-test increments.*

## *What are the Strengths of an Incremental Development Approach?*

- Incremental development and testing provides additional opportunities to identify internal and external impacts or dependencies.
- Frequent testing may increase the developers' ability to isolate and resolve technical issues.
- Lessons learned and knowledge gained can be incorporated in later increments.
- Stakeholders can see tangible progress of the solution's development with each increment.
- Continuous interaction and collaboration between the developers and user community helps ensure the solution solves the business problem and fosters a sense of ownership among stakeholders.
- Incremental testing and user feedback assists in identifying overlooked or misunderstood requirements.
- Business needs can be met and [prioritized] requirements can be fulfilled with each increment so users can begin utilizing the solution's functionality while later increments are developed.

## *What are the Weaknesses of an Incremental Development Approach?*

- Consistent resources from the user community need to be available throughout the development to participate in testing and provide feedback on the solution's design or user experience.
- Schedule delays in one increment can cause delays in the overall solution's development.
- Unforeseen inner-dependencies within the solution may require adjustments to the order in which functions are developed and requirements are met. Interfaces within the system need to be understood and well defined by the team during the upfront planning.
- Issues discovered early in development tend to be shifted to later increments, which may cause a false sense of early project success or running ahead of schedule. Issues or problems should be addressed as they arise.
- Teams may become too focused on the requirements, milestones, or issues and successes of a single increment and can lose focus on integrating into the overall solution. As with any

development approach, scope, schedule, and issues need to managed throughout a solution's lifecycle.

### *When is Incremental an Appropriate Solution Development Approach to Select?*

The following variables indicate that incremental may be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The solution's requirements or design specifications cannot be fully articulated at the start of development or could be impacted by external variables, such as budget changes or new technologies.
- The solution has intricate, complex features that need to be independently tested prior to being integrated with the rest of the solution.
- The team is experienced and the user community is able to commit resources to test and provide feedback throughout the development.
- The solution requires new or complex technology.
- Stakeholders need to see tangible progress throughout the solution development process.

### *When is Incremental Not an Appropriate Solution Development Approach to Select?*

The following variables indicate that incremental may not be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The overall scope of the solution is relatively small, the solution does not require new or complex technology, or stakeholders are already familiar with the general solution, such as an enhancement to an existing system or software.
- The solution's requirements and design specifications can be articulated prior to beginning development.
- Resources need to be conserved or testers will not be available throughout the solution's development.
- Team members are expected to fluctuate during the development of the solution.

## Iterative

Iterative development promotes flexible, dynamic development, opposed to the rigid, sequential method emphasized in linear approaches. Although there are many specific types of iterative development approaches, such as Agile, Spiral, and Prototyping, that have unique characteristics (i.e. emphasizing particular tasks, risk management strategies, etc) common aspects of iterative development include:

- Solution functionality is developed and implemented in multiple iterations based on the business needs and prioritized requirements.
- Emphasis is placed on communication between the development teams and the user community.
- Requirements and design are analyzed, refined, and tested throughout development to ensure the solution solves the business problem(s) and closes performance gaps.
- Each iteration improves on existing or delivers new functionality.

<table>
<tr><th></th><th>Requirements</th><th>Design</th><th>Develop</th><th>Test</th><th>Deploy</th></tr>
<tr><td>1</td><td>Iteration 1</td><td>Iteration 1</td><td>Iteration 1</td><td rowspan="2">Iterations 1 & 2</td><td rowspan="5">Iterations<br>1, 2, 3, 4, 5</td></tr>
<tr><td>2</td><td>Iteration 2</td><td>Iteration 2</td><td>Iteration 2</td></tr>
<tr><td>3</td><td>Iteration 3</td><td>Iteration 3</td><td>Iteration 3</td><td rowspan="2">Iterations 3 & 4</td></tr>
<tr><td>4</td><td>Iteration 4</td><td>Iteration 4</td><td>Iteration 4</td></tr>
<tr><td>5</td><td>Iteration 5</td><td>Iteration 5</td><td>Iteration 5</td><td>Iterations 5</td></tr>
</table>

**Figure 4: Iterative**

*In iterative development, teams plan cyclic iterations of IT development tasks. Teams may select to plan iterations based on the ability to combine and merge various development tasks, such as testing multiple iterations simultaneously, in order to deliver requirements and meet business needs.*

### *What are the Strengths of an Iterative Development Approach?*

- Communication and feedback between the development team and the business/user community minimizes the possibility that requirements are misunderstood or the design does not meet the business need. Continual interaction encourages commitment and a sense of ownership among the solution's stakeholders.
- The solution's functionality is tested throughout development, which encourages issues and unanticipated impacts to be discovered and corrected as they arise.
- Stakeholders can see tangible progress throughout the solution's development.
- Continual review of requirements and frequent testing promotes requirements tracking and bi-directional traceability.

- Small iterations allow teams to focus on risk management, complex portions of the solution, and/or other components based on the variables of a specific solution.

### *What are the Weaknesses of an Iterative Development Approach?*

- Early planning and understanding of the solution's general requirements are imperative for the team to map requirements to a specific iteration to ensure the solution's constraints can be managed and requirements are implemented based on priority and/or dependency.
- Resources from the business/user community need to be available throughout the solution's development to provide feedback and participate in testing activities.
- The IT Project Manager (ITPM) must be experienced in Change Management. Modified or new requirements must be carefully analyzed and communicated to the appropriate stakeholders to ensure possible impacts or risks are identified.
- The flexible, dynamic nature of iterative development does not lead to as many inherit controls or obvious milestones as a linear approach. Teams should baseline some assumptions regarding how tasks will be performed, lines of communication, and how earned value will be calculated prior to beginning development.
- Challenges and issues need to be communicated and addressed throughout the solution's development. If possible challenges are reserved for later iterations, stakeholders may be given a false sense of early success.

### *When is Iterative an Appropriate Solution Development Approach to Select?*

The following variables indicate that iterative may be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- There is an immediate business need to implement a portion of the solution's requirements.
- The solution is technologically complex or requires feedback from a large number of stakeholders.
- The solution requires new technology or technology users are not familiar with.
- The implemented solution will need to handle a large volume of users, users will need the solution for a multitude of purposes, or the users will have varying levels of experience with the solution.
- The team is experienced and the business is able to commit resources to test and provide feedback throughout the development.

### *When is Iterative Not an Appropriate Solution Development Approach to Select?*

The following variables indicate that iterative may not be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The user community is already knowledgeable about the solution (i.e. the project or release is a maintenance or minor enhancement effort).
- Team members are expected to fluctuate during development or the business cannot commit resources to be available throughout development.
- The solution's requirements and design specifications can be fully articulated and understood by stakeholders prior to beginning development tasks.

- The solution is not very technically complex or there is a low risk of impacting features or other solutions already in production.
- The effort has a short schedule, low budget, or resources need to be conserved.

## Agile

Agile is an iterative development approach and, like other iterative approaches, emphasizes business and IT collaboration and breaking a solution's development into small segments in order to meet requirements, collect feedback, and test frequently. Agile tends to distinguish itself from other iterative approaches by emphasizing short development sprints (usually no more than a few weeks each) and the principles documented in the "Agile Manifesto," including:

- Customer satisfaction is the highest priority and is maintained through continuous and frequent delivery of requirements.
- Changes should be incorporated quickly.
- Business and IT representatives should have daily, face-to-face interaction.
- Daily communication and collaboration between developers and customers.
- A working solution is the primary measure of progress.
- Individual team members must be motivated and committed to excellence.
- Development cycles are time-boxed into short sprints in order to frequently deliver prioritized and planned requirements and functionality.

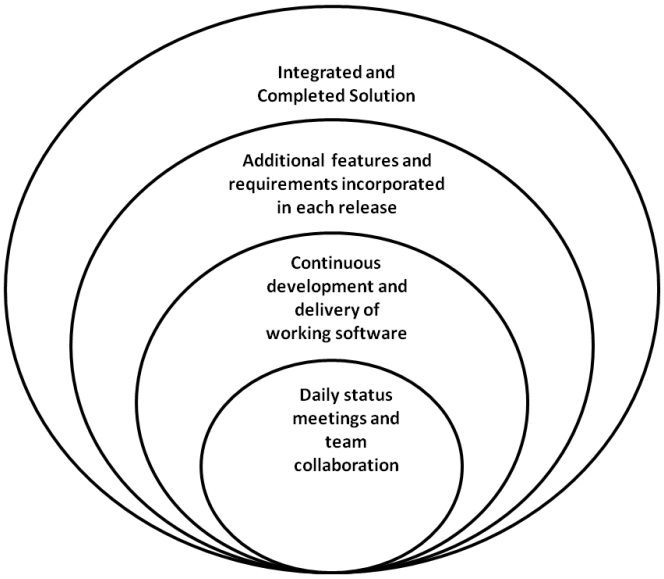


**Figure 5: Agile**

*In agile development, teams develop IT solutions using the principles of the "Agile Manifesto," such as daily team collaboration, continuous delivery of requirements, incorporating changes quickly, and individual motivation, in order to ensure customer satisfaction.*

### *What are the Strengths of an Agile Development Approach?*

- The emphasis on customer satisfaction helps ensure requirements are met, the solution closes performance gaps, and quality standards are maintained throughout the project or release.
- Allows for flexibility to incorporate changes to the requirements or design specifications throughout the development process. Quick response to change and continuous development are encouraged.

- Daily, face-to-face communication between IT and business representatives can encourage team camaraderie and a sense of ownership among stakeholders. Continuous interaction promotes a greater understanding of the overall solution and decreases the possibility of the solution not meeting the business need(s).
- Working products are delivered quickly and frequently, which allows stakeholders to see tangible progress throughout the solution's development.
- Each iteration or development sprint is tested, which allows for corrective actions and lessons learned to be incorporated throughout development.

### *What are the Weaknesses of an Agile Development Approach?*

- Daily, face-to-face interaction between IT and business representatives can encourage continual requests for change. A change management process needs to be carefully followed in order to prevent scope creep or unanticipated impacts on the solution.
- Short, time-boxed development sprints and informal, face-to-face communication can lead to documentation gaps. Decisions and changes need to be documented and communicated to all appropriate stakeholders and all required documentation needs to be maintained [as necessary] and complete at the end of the project or release.
- The rapid pace of agile may cause teams to appear unorganized or developing using an "ad-hoc" approach. Each iteration or development sprint should have clearly defined goals and deliver requirements based on the prioritized needs of the business.
- Individual team members need to be highly motivated, experienced, and dedicated to the solution. Unstable resources can lead to schedule delays, conflicting feedback, or knowledge gaps regarding the solution's development.
- Although the "Agile Manifesto" calls for progress to be measured by delivered functionality, teams may have to agree on additional milestones or a means of tracking and reporting progress. Milestones and progress reporting expectations should be defined prior to beginning development.

### *When is Agile an Appropriate Solution Development Approach to Select?*

The following variables indicate that agile may be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The solution's scope, requirements, or design specifications may fluctuate based on user feedback, budget changes, new technology, etc.
- Business or end-user stakeholders are not familiar with the solution's design or a new technology is being used. Continuous interaction may increase understanding and ensure the solution fulfills to business need(s).
- Individual team members are motivated, experienced, and dedicated to the solution. The developers assigned to the effort are highly skilled and will be able to maintain the rapid pace and quality standards required in agile development. Similarly, team members responsible for documentation must be able to produce quality deliverables in a time-frame that coincides with an agile schedule.
- The team is centrally located, relatively small, and is expected to remain stable throughout the solution's development.

### *When is Agile Not an Appropriate Solution Development Approach to Select?*

The following variables indicate that agile may not be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- Team members are not able to commit to daily, face-to-face interactions due to completing priorities, lack of resources, or varying work locations.
- Team members are inexperienced or expected to fluctuate during development.
- The solution has a small scope, budget, schedule, and/or risk level.
- The solution's requirements and design specifications are stable, approved, and understood by appropriate stakeholders prior to beginning development tasks.

## Prototyping

Prototyping is a unique technique that can be used for a multitude of purposes throughout a solution's development. Although prototyping is not generally considered a development approach unto itself, it is a strong tool that can be integrated with a variety of iterative approaches. A prototype is a mock-up of specific components or portions of a solution that stakeholders can experience and interact with. Prototyping assists in refining requirements and design specifications, collecting feedback, testing specific conditions, and mapping requirements to specific functions. Prototyping may reduce uncertainty by allowing stakeholders to see tangible options prior to making final decisions or approvals.

Regardless of the prototype's purpose, complexity, or featured functions, most prototypes can be generally categorized as "throwaways" or "evolutionary." Throwaway prototypes are generally used to assist in developing and analyzing requirements and are rapidly built. Development teams do not spend time testing or creating extensive code behind an interface because as the name suggests, the prototype will be thrown away after it has served its purpose. Evolutionary prototypes are developed with the purpose of [eventually] being integrated with the rest of the solution. While development teams need to be able to incrementally modify or add functions based on stakeholder feedback, additional time will be needed to develop quality code and integration components. In order to effectively manage expectations, it is important that all appropriate stakeholders understand which type of prototype is being built and for what purpose.

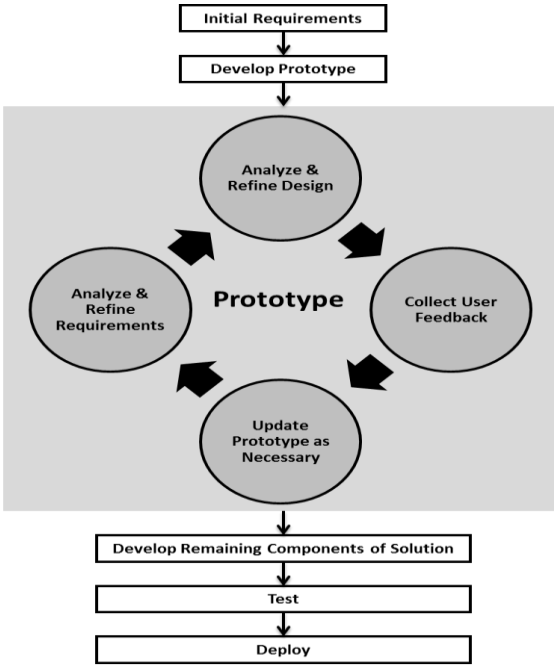


**Figure 6: Prototyping**

*Prototypes may be used to assist in collecting, analyzing, defining, and refining requirements and design specifications from solution stakeholders.*

### *What are the Strengths of a Prototyping Development Approach?*

- Prototypes serve as a communication tool among users and developers. Instead of merely reading a requirements or design document, stakeholders can actually interact with portions of the solution and see interfaces and graphics.
- Early interactions with a tangible product may enhance stakeholder buy-in. Similarly, the opportunity to participate in and provide feedback early in a solution's development increases a sense of ownership among stakeholders.
- Helps ensure the implemented solution's interface, design, and functionality meets the customers' expectations.
- The "hands-on" approach provides an enhanced understanding of the solution to stakeholders, supports early training efforts, and may ease the transition to a new or enhanced solution.
- Developers can create multiple "scenarios" or "paths" for users to explore, which may assist in finding defects or issues early in development.

### *What are the Weaknesses of a Prototyping Development Approach?*

- The approval and change control process is not inherently strict. While users and developers work closely to refine the solution's requirements and design, a change and analysis process needs to be in place to prevent scope creep and identify possible impacts.
- Prototypes may appear to be fully functional or near complete solution and users may want to deploy ahead of schedule. Expectations of the purpose and scope of the prototype need to be carefully managed to avoid frustrations and ensure the solution is correctly designed and fully functional before "going live."
- Prototypes tend to be associated with development efforts that are high risk, have an extended schedule, and/or have a large budget. Although prototypes can be scaled to almost any solution's needs, including simple paper images of the design, project and release teams should ensure the time and money spent on a prototype can be justified as enhancing the final solution's quality and better meeting the business needs.
- Although developers may be encouraged to quickly incorporate a stakeholder's feedback into the prototype, teams may need to conduct an in-depth "walk-through" of the solution prior to deployment to record all changes, functions, and ensure documentation is comprehensive and accurate.
- The prototype's purpose and scope need to be well understood by the development team to avoid creating overly complex or detailed models. Prototypes that include unnecessary back-end code or unnecessary features may lead to an overrun in budget and schedule.

### *When is Prototyping an Appropriate Solution Development Approach to Select?*

The following variables indicate that prototyping may be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The solution will have a large number of users, has complex interrelationships and functions, or will be used for a variety of purposes and stakeholders need to understand/visualize how the solution will operate in different scenarios and conditions.
- The team is experienced and will be able to manage scope and change effectively, while quickly responding the stakeholder feedback, throughout the solution's development.

- The solution requires extensive user interface or is utilizing a new technology that stakeholders are not familiar with.
- There is a business need to provide visual and tangibles examples to stakeholders early in the development process or to support early training and transition to the new solution.
- The project/release has an adequate budget, schedule, and dedicated resources to support the initial development of a prototype and several iterations of refinement.

### *When is Prototyping Not an Appropriate Solution Development Approach to Select?*

The following variables indicate that prototyping may not be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- Team members are expected to fluctuate or resources will not be continuously available to provide feedback on the prototype.
- The solution has a small budget, minimal risk, short schedule, and/or only incorporates basic enhancements or modifications to a solution already in production.
- Stakeholders cannot agree on the purpose, scope, or need for a prototype. If a consensus or general understanding cannot be reached prior to developing a prototype, expectations of schedule, quality, design, and even scope may be challenging to manage afterwards.
- The solution's requirements and design specifications can be articulated and understood by the majority of stakeholders prior to beginning development tasks.

## Spiral

Spiral is a type of iterative development that emphasizes risk assessment and management throughout a solution's development. The approach is based on the theory that breaking development into small increments [or spirals] enhances a team's ability to focus on developing specific functions or risk management strategies. A spiral approach allows teams to incorporate and blend aspects of other development approaches based on the needs and risks of individual spirals or the solution as a whole. For example, teams may use a prototype model to gather user input while each spiral adds functionality under the control used in a traditional waterfall approach.

As with other iterative development approaches, each spiral includes requirements analysis, design review, development/configuration, and testing; however, the spiral approach calls for teams to incorporate these tasks into four basic quadrants that promote keeping risk management at the forefront of a solution's progress.

1. *Determine the spiral's objectives, alternatives, and constraints.* Teams should clearly understand the objectives and goals of the spiral and how it contributes to the IT solution as a whole. In addition, teams should identify possible design alternatives and resource, schedule, etc. constraints that could impact the ongoing effort.
2. *Risk assessment and analysis.* The design alternatives are analyzed and various development approaches are examined to address the identified constraints and risks.
3. *Develop and verify.* Teams develop/configure specified portions of the solution and perform testing.
4. *Review and strategize.* The business community reviews the outcome of the spiral and overall progress of the solution's development. Stakeholders are involved in identifying and assessing new and outstanding risks, issues, constraints, and lessons learned. Risk management strategies, and the development approach(es) for upcoming spirals should be reviewed.

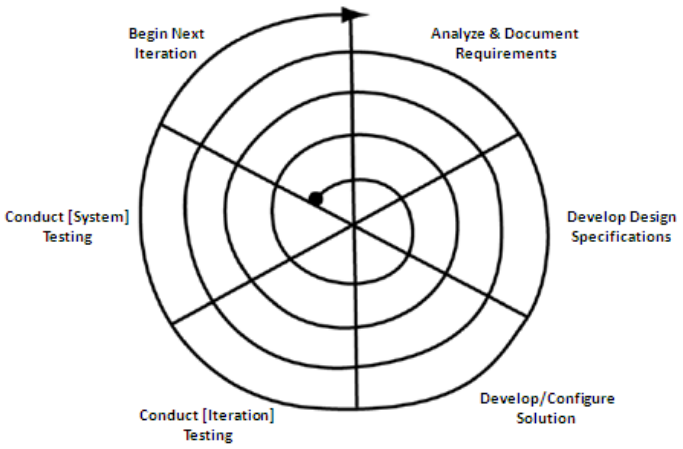


**Figure 7: Spiral**

*In spiral development, IT solutions are developed in "spirals" that incorporate tasks and processes designed to ensure risks are managed and reviewed throughout development.*

### *What are the Strengths of a Spiral Development Approach?*

- Spiral provides teams with flexibility to incorporate elements of other development approaches and determine the length of each spiral. There is no set pattern teams must follow – each spiral should be based on the needs of the user community, prioritized requirements, or risk management strategies.
- Modifications or alternatives to requirements and design can be analyzed and, if necessary, incorporated throughout development in order to manage risks.
- Breaking development into small spirals allows teams to utilize different approaches and focus on highly complex or sensitive portions of a solution.
- Continual review and analysis of progress, risks, constraints, etc. naturally facilitates communication and transparency between IT and business stakeholders.
- Stakeholders can see tangible progress throughout development as each new spiral builds on the previous and fulfills specified requirements.

### *What are the Weaknesses of a Spiral Development Approach?*

- Expertise in risk analysis and management are critical. A team's ability to identify risks, assign accurate risk levels, develop appropriate risk management strategies, and track escalating risks and impacts is often critical to the success of a high risk solution using a spiral development approach.
- As the approach can be customized to the needs of the overall solution or individual spirals, it may be very challenging to reuse the exact pattern on a different development effort.
- Schedule delays in one spiral can cause the entire effort to fall behind schedule.
- A strong change control process needs to be carefully management throughout a solution's development. As requirements and risks are reviewed frequently, a spiral approach may encourage stakeholders to suggest modifications. Any changes to a solution's requirements, design, risk management strategy, budget, schedule, etc. need to be analyzed, documented, and communicated to all appropriate stakeholders.
- Although the flexibility of customizing the approach to each spiral allows teams to mitigate specific risks or focus on highly complex development portions, it is often challenging to define specific milestones or for stakeholders to understand the solution's progress. Stakeholders should agree on how progress will be measured prior to beginning development tasks.
- Resources from the development team, business, and other appropriate stakeholders are needed throughout the solution's development to review the solution's progress, risks, and risk management approach(es).

### *When is Spiral an Appropriate Solution Development Approach to Select?*

The following variables indicate that spiral may be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The solution has multiple known risks and avoidance or mitigation is a high priority. Overall accuracy is essential to the solution's success.
- Stakeholder resources are stable and able to commit to being involved throughout the solution's development.
- Portions of the solution's requirements or design are based on maintaining risks below a certain level or changes to the solution's risks may impact the solution's requirements or design.

- Team members have demonstrated an ability to effectively manage change, scope, budget, schedule, and risks simultaneously.
- The solution requires complex or new technology and may require different approaches for various portions of development.
- There is an immediate need to quickly development and implement a portion of the solution's requirements, while others can be delivered in later spirals.

### *When is Spiral Not an Appropriate Solution Development Approach to Select?*

The following variables indicate that spiral may not be an appropriate development approach for implementing a particular IT solution; however, the needs of the business and any variables unique to the solution should be considered before deciding on a solution development approach:

- The solution is low risk or stakeholders have decided to accept the risks.
- Consistent resources will not be available throughout the solution's development.
- The solution's requirements and design specifications are not overly complex or are well understood by stakeholders prior to beginning development tasks.
- The solution has a small budget or short development schedule.

## Conclusion

The variety of solution development approaches that can be used at PBGC enables flexibility in developing and implementing IT solutions. No one development approach is appropriate for all IT solutions and teams should select an approach that will help facilitate effectively closing performance gaps, meeting the needs of the business and user community, and fulfilling requirements.

Choosing an approach that is inappropriate for the solution or the expectations of stakeholders can cause a delay in schedule, increase in cost, and may ultimately fail to solve the business problem. Comparatively, choosing the correct approach can assist in delivering a successful solution, on time and within budget. Some of the variables that should be considered before determining the solution development approach include:

- The experience, skills, and availability of the team assigned to development and implement the IT solution.
- The type of solution being implemented (i.e. COTS, Hosted, maintenance, etc.).
- The solution's scope, budget, schedule, and relevant resource availability.
- The estimated level of effort needed to elicit, analyze, and document a solution's requirements.
- The stability of the solution's requirements.
- Other development efforts within the organization that could affect testing and/or release schedules.
- The user community's familiarity with the proposed solution.
- Risks associated with the solution.

Regardless of which solution development approach is selected, all team members and appropriate stakeholders should understand any responsibilities, milestones, or processes associated with the selected approach. Team members' commitment to and understanding of the selected approach contributes to the ultimate successful delivery of an IT solution.

## Appendix A

### Further Reading

Center for Disease Control. (2008). Iterative Development.
http://www2.cdc.gov/cdcup/library/examples-job_aids/CDC_UP_Iterative_Development_Job_Aid.pdf

Center for Disease Control. (2008). Guidance for Iterative Development.
http://www2.cdc.gov/cdcup/library/examples-job_aids/CDC_UP_Guidance_for_Iterative_Development_Using_EPLC_Job_Aid.pdf

Farrell, A. (2007). Selecting a Software Development Methodology based on Organizational Characteristics. (An essay submitted in partial fulfillment of the requirements for the degree of Master of Science in Information Systems). Athabasca University.
http://library.athabascau.ca/drr/download.php?filename=scis-7/open/AdrienneFarrell_Final.pdf

Fu, Jicheng, Farokh Bastani, and I-Ling Yen. (2008). Model-Driven Prototyping Based Requirements Elicitation. University of Texas at Dallas. http://cs2.uco.edu/~fu/papers/MDPRE.pdf

Manifesto for Agile Software Development. Principles behind the Agile Manifesto. (2001).
http://agilemanifesto.org/principles.html

Pan, J. (1999). Software Testing. Carnegie Mellon University.
http://www.ece.cmu.edu/~koopman/des_s99/sw_testing/