**Alternate Methodologies**

Large scale development projects traditionally center around well-organized, structured phases of work. The more complex a project is, the higher its risk, the more rigorous project controls need to be.

The State SDLC waterfall model can generally be characterized as a *plan-driven*software development methodology. The genesis of plan driven models is traditional engineering, which approaches development systematically with well-defined processes. Careful up-front planning, firm requirements, requirements traceability and testability, and clearly defined acceptance criteria are paramount. Plan-driven methodologies are also characterized by strong documentation and detailed traceability of requirements through design, code, testing, and implementation. The strength of these methodologies lies in the comparability and repeatability that stem from standardized processes. Waterfall development is generally considered to be the least risky development model, which makes it popular for large software development projects, particularly in the government sector.

While the traditional SDLC waterfall model is complete, comprehensive, and has broad applicability to most statewide mission-critical systems, it also has drawbacks. When too strictly applied, excessive emphasis may be placed on documentation, and the true objective of the project becomes inadvertently subordinate to the process. Also, because they are linear, waterfall projects are time consuming, and functionality is not delivered until the end of the effort, which may be frustrating to development teams and end users.

Occasionally, an agency may choose to use a different software development methodology. Many alternative methodologies change the standard sequence of SDLC activity execution.  Deviation from the State SDLC standard development methodology is allowable if:

* The nature of the project lends itself to an alternate approach,
* SDLC and project controls are not compromised, and
* The project risk profile is not increased.

A decision to adopt an alternative development methodology should not be taken lightly, and most important, should not be imposed by a potential solutions vendor. **Agencies electing to use alternative methodologies should be comfortable that they have resources that are sufficiently knowledgeable about the methodology’s characteristics, tools, and techniques to manage the effort to successful completion.**

In the sections below, there are brief descriptions of some major alternative development methodologies along with some advice as to when an agency might consider their use. Alternative approaches are categorized as plan driven or agile. The primary differentiator between plan-driven alternatives and agile alternatives is that plan-driven alternatives require that all desired requirements of the end product be developed before beginning construction of the end product. Agile approaches assume that desired requirements of the end product cannot be determined until part of the solution is developed. Agencies seriously considering alternative development methodologies should thoroughly research these methodologies beyond this brief snapshot before proceeding.

**Plan Driven Alternatives**

**Incremental Waterfall –**The incremental waterfall methodology has the same first three to four phases as the traditional waterfall model, but it deviates for phases five through ten **by creating mini releases and segmenting requirements into an incremental series of products, each of which is developed fairly independently from the others**. Incremental waterfall is highly dependent on the development of a complete up front set of requirements, designed and implemented in a series of smaller projects or releases. Each increment adheres to the waterfall sequence.

The incremental waterfall methodology’s benefits include:

* Lower cost and less time required for first release
* Focus on essential requirements, thereby reducing the amount of unnecessary functionality
* Less risk inherent in developing smaller, more manageable systems represented by increments
* Possible reduction in the number of developers
* Possible decrease of user requirement changes because of the faster time to first release
* Possibility for incremental funding
* Phase-level control

The disadvantages of incremental waterfall include:

* Longer-term commitment from stakeholders/business
* Additional effort to implement more rigorous project management controls
* Additional effort and costs associated with increased regression testing

Incremental waterfall may be utilized when:

* Functionality of the application can be broken down into meaningful products
* Stakeholders are available and committed to support the project through all iterations

**Spiral Models –**The spiral model decomposes a large single development cycle of a single phase waterfall into multiple smaller development cycles, each cycle building on the previous one. In spiral development, the end requirements are unknown prior to first release execution. Usually referred to as the “build a little, test a little” approach, it is best suited for projects that have unclear requirements and necessitate only moderate changes. Since unclear requirements are unacceptable on State projects, this model has little applicability.

**Rational Unified Process (RUP) –**The Rational Unified Process is an iterative and flexible software development framework originally developed by the Rational Corporation and now owned by IBM. Methodologies are more prescriptive and detailed, whereas frameworks are more general and allow for much more tailoring. Key aspects of RUP are that it is use case driven, iterative, and architecture centric. The RUP framework can actually accommodate many different development processes, both plan-driven and agile. Although certain RUP practices improve projects, RUP is contradictory to the core SDLC principles of early, detailed, and long-term planning as well as deliverable and phase review and approval. As a result, careful planning is required to incorporate RUP practices while maintaining critical elements of the waterfall approach.

RUP’s four-phase life cycle of Inception, Elaboration, Construction, and Transition are often conducted in multiple iterations. RUP stresses exit criteria for each phase; a phase is exited only after the project team demonstrates that it has met the phase specific criteria.

Within each iteration, tasks are categorized into nine core workflows: six development disciplines (Business Modeling, Requirements, Analysis and Design, Implementation, Test, Deployment) and three support disciplines (Configuration and Change Management, Project Management, Environment). These workflows are executed in parallel throughout a series of iterations. For example, unlike traditional or incremental waterfall, RUP allows for the concurrent execution of requirements definition, design, implementation, and test within a project phase (Elaboration).

The advantages of RUP include:

* Stakeholders are able to view working products much earlier in the life cycle than with traditional waterfall methodologies
* Requirements definition is strengthened by greater stakeholder involvement and review of working products
* Lower cost and less time required for first release
* Incremental work allows higher technical risks to be addressed in an early iteration
* Enhances the possibility that software meets actual stakeholder needs rather than perceived needs

The disadvantages include:

* Work products are not completely finished until the system is released to production
* Full lifecycle costs are unknown early in project
* Effective management and execution are complex and require personnel expertise in RUP
* Flexibility of framework may not provide enough explicit guidance, resulting in project chaos and lack of control

Although State projects require significant up-front planning and requirements definition and do not allow for the concurrent execution of the Requirements Analysis, Design, Development, Testing, and Implementation phases, the following RUP-inspired practices may be utilized to optimize project performance:

* Use case development
* Controlled requirements management
* Iterative system development
* Visual software modeling
* Early use of prototyping in requirements analysis and design
* Continuous quality verification
* Rigorous change control
* Utilization of tools to automate processes
* Modeling business processes prior to Requirements Analysis
* Testing throughout the project
* Robust configuration management controls and tools
* Beta testing

**Agile Alternatives**

Agile principles evolved to address the perceived limitations of waterfall development – mainly that waterfall does not show results until the end, engages stakeholders too late, and unnecessarily delays testing. Agile methodologies are characterized by:

* Lightweight processes (just enough)
* Short iterative development cycles (daily builds to monthly “Sprints”)
* Rapid prototyping and rapid development
* Active, co-located teams including end users
* Heavy reliance on domain knowledge of the project team
* Incremental releases
* Self-organizing teams
* Adaptive rather than predictive mindsets
* Simple design

Agile practices evolved from the **Agile Manifesto**.

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

|  |  |  |
| --- | --- | --- |
| *Individuals and Interactions* | *Over* | *Processes and Tools* |
| *Working Software* | *Over* | *Comprehensive Documentation* |
| *Customer Collaboration* | *Over* | *Contract Negotiation* |
| *Responding to Change* | *Over* | *Following a Plan* |

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

This section provides a short synopsis of some of the most popular Agile methodologies. If agencies are considering using any of these methodologies or other agile methodologies, they should conduct more thorough research with more detailed sources on these methodologies before proceeding.

**Scrum**– Developed by Ken Schwaber and Jeff Sutherland, Scrum is an Agile software development process wherein projects progress through a series of iterations called sprints (typically two to four weeks long each). Scrum is appropriate for projects with rapidly changing or evolving requirements.

Scrum’s distinctive characteristics are:

* Use of self-directed teams
* Daily team measurement
* Avoidance of prescriptive processes
* Client-driven adaptive planning

The overarching technique within a Scrum project is the use of 30-day development Sprints*,*which is essentially a 30-day development cycle, with short term Sprint goals, which are established not by management, not by a prescribed schedule, but by the project team.

Highly-trained and certified Scrum masters or coaches manage the effort by conducting daily Scrum meetings, facilitating the process of establishing Sprint goals, and keeping the team focused on the broader objectives of the effort. The success of Scrum projects requires a co-located project team and highly efficient practitioners to be successful.

It is important to understand that Scrum does not involve:

* Detailed project planning in advance – only the current sprint is planned
* Top down management – Scrum teams decide their own objectives

Although aspects of Scrum may be beneficial for State projects, the Scrum process does not support planning and governance practices required on State projects.

**XP – Extreme Programming, (also eXP) –**XP is an Agile methodology that emphasizes customer satisfaction through the rapid creation of high value software, the use of very skillful and sustainable software development techniques, and flexible response to change. It is based on four key values:

* Communications – stresses communication practices that give developers a shared view of the system which matches the view held by users
* Simplicity – development of the simplest product that meets client needs
* Feedback – stresses frequent feedback from end users, the team, and system
* Courage – team preparation to make hard decisions

XP iterations are no longer than three weeks. Systems code is owned by the team.

XP creators indicate that XP development is best suited for relatively small team projects, with total delivery duration of one year or less. It, like Scrum, is highly dependent on very knowledgeable practitioners and co-located teams. While XP techniques, such as simple design and metaphors can be useful project techniques, XP scales poorly and should not be considered an optional development methodology for State projects.

**Feature–Driven Development** (FDD) – FDD focuses on simple process, architecture planning, efficient modeling, and short development cycles. It depends on highly-skilled people with extensive domain knowledge, design and development experience. FDD uses process and planning in the background to support rather than direct team efforts.

FDD is a five-phase process, which starts roughly at the design phase – that is, it assumes that requirements are already known, documented, and understood. The five phases are:

1. Using class architecture and notes, develop a model of the product to capture the breadth of the domain.
2. Establish a list of features based on the business needs.
3. Create a development plan based on the list of features.
4. Develop design packages and work packages for each iteration.
5. Build the features (implement methods, build, inspect, and integrate code).

In FDD phases 4 and 5 are repeated for multiple iterations until the project is complete.

FDD is a similar technique to XP but is different in a couple important areas:

* XP stresses that system code is owned by the team. FDD advocates that classes of code be assigned to specific owners who are responsible for its overall quality.
* FDD is more applicable to large systems because it is more focused on architecture considerations than XP, which focuses on simple design

FDD advocates strong architecture work and other planning at the beginning of the project, which provides the basis for multiple work packages to be developed in parallel. As a result, FDD methodologies scale more easily than XP.

**Considerations for Using Other Methodologies on State Projects**

While there are different methodologies to choose from which may be appropriate for certain types of Agency projects, picking the wrong one for a particular project could cause it to stall or worse, fail.

**When to Use Plan-Driven Alternatives**

Of the plan-driven types of methodologies, incremental waterfall is an appropriate alternative to single phase waterfall releases. Spiral development, with its general lack of up-front requirements, has little applicability given State funding requirements. Elements of RUP may be utilized, but project teams must ensure that the alternative methodologies do not deter early, detailed, and long-term planning or SDLC deliverable/phase review and approval.

**When to Use Contemporary Alternatives**

Use of Agile methodologies is attractive to project practitioners because the methodologies emphasize development tasks, promise faster results, and are much easier methodologies to adapt to changing requirements. Developers prefer Agile methodologies because they are reputed to be light on documentation, a task that few developers enjoy. Many project teams, however, use Agile methodologies in the wrong circumstances or misapply Agile methodologies, and projects suffer as a result. Agencies must be careful to balance the need for adequate controls when considering an Agile approach.

Barry Boehm and Richard Turner, in *Balancing Agility and Discipline: A Guide for the Perplexed*, provide a good reference for project practitioners attempting to determine whether an Agile methodology is appropriate for a specific project:

| **Factor** | **Agile Characteristic** | **Waterfall Characteristic** |
| --- | --- | --- |
| ***Size*** | Optimal for small projects and teams, reliance on domain knowledge | Tailored for large projects and teams |
| ***Mission Critical Projects*** | Untested, general lack of documentation | Long history of use in such implementations |
| ***Stability and complexity of existing environment*** | Continuous refactoring used, suitable for dynamic and simple environments | Structured baselines used, suitable for more static and complex environments |
| ***Skills*** | Continuous involvement of highly skilled individuals, difficult to cope with many lower skilled resources | Highly skilled resources needed in early phases, designed to cope with many lower skilled resources in later phases |
| ***Organizational Culture*** | Chaotic, dynamic, empowered | Roles well defined, procedures in place |

Agile methodologies work well with small projects (six to eight people) with durations of a year or less and with co-located teams. Agile methodologies, particularly Scrum techniques and rapid prototyping, may be used within a waterfall effort.

**Contractor Considerations**

There are a few considerations before proceeding with a contractor who claims to utilize an Agile development approach:

  The Agile term is broadly misused and is sometimes simply justification for skipping over critical project documentation.

1. Do agency personnel under the proposed methodology sufficiently manage the contractor and contract?
2. How will the contractor apply the methodology correctly? In most cases, Agile methodologies require team co-location and high domain knowledge, so if the contractor is working off-site, it cannot be Agile.
3. While alternative methodologies are acceptable, **agencies are highly encouraged to require contractors to supply a detailed plan**of how they propose to meet SDLC requirements and implement proper project controls within the Agile methodologies. This plan increases the likelihood that the contractor has conducted a detailed compliance review of the proposed methodology and the SDLC and has demonstrated planned compliance.  If an agency elects an Agile approach, routine project status reporting to DoIT will still be vital to the project’s health and success.

**Conclusions**

Agencies are encouraged to use the State standard SDLC methodology because it is the development methodology best suited to the types of projects agencies undertake, and it is the least risky. Other methodologies are acceptable but require some additional up-front planning to ensure that the proposed methodology does not omit any necessary project controls.​​​​​