Blending Traditional and Agile Project Documentation

A project Portfolio Perspective

Fergal McGovern, Founder, VisibleThread

Audience: IT Directors, Program Managers, Project Managers, Business Analyst Team Leads

INTRODUCTION

As the merits of agile and iterative approaches are recognized, for many IT directors, project/program managers and Business Analyst leads, the question becomes; how to adapt current process documentation to facilitate both traditional and agile approaches across a portfolio of projects, often as part of a wider program effort.

This paper will show how it is possible to apply agile documentation practices side by side with traditional documentation practices. We present practical guidance for project documentation where a blend of traditional and agile projects may exist.

This paper is applicable to larger enterprise with a variety of project styles and will have particular relevance for enterprises looking to adopt more agile approaches for certain projects.

SIZE OF PROJECTS; RABBIT, HORSE OR ELEPHANT.

First, it is useful to consider the types of projects that may exist in a typical project portfolio.

James and Suzanne Robertson, authors of 'Mastering the Requirements Process'; categorize projects either as Rabbit, Horses or Elephants. This is a useful delineation.

- **Rabbit** Projects: Very nimble, short duration projects, small co-located teams, non-mission critical, less than $500k in fully burdened cost. These are non-strategic. Examples include small migration projects, static Web site creation, or simple upgrades to existing systems.

- **Horse** Projects: These are longer in duration, typically exceeding 6 months & tend to have non co-located teams. Project cost will range from $500k to $10m as a rule of thumb. At the high end of the dollar amount, the project(s) may be strategic in nature. Third parties such as external integrators and/or vendors will likely be involved. Examples of these projects are deploying infrastructure to support a new channel to market, a business intelligence project or new capability mandated by regulatory drivers.

- **Elephant** Projects: These are multi-year projects / programs including multiple related projects. They tend to be distributed and will have resources of various skill levels & business/technology backgrounds participating. Many stakeholders involved at the start will not be expected to be involved for the project duration. The dollar cost for these are above $10m, ranging to hundreds of millions. Elephant projects are strategic to the business. External integrators and/or third party vendors will be involved in these projects as a rule. Examples of such projects are the development and deployment of a new billing system, data warehousing projects, or system integration projects as a result of a merger or acquisition.

Smaller rabbit style projects, defined as less than $500k in total cost, tend to be co-located and can often work without need for comprehensive documentation. We focus here on horse and elephant projects.

TRADITIONAL/WATERFALL PROJECT DOCUMENTATION

Let's revisit what we know about traditional / waterfall projects.

Traditional projects are driven by phases, each phase yields an outcome. The projects are described as waterfall because in a strict view, you will not commence one phase until the previous phase is complete and has passed through a phase gate or approval process. When issues are identified requiring change, a change control is issued thereby adjusting one or all of; the timeline, resourcing needs and cost estimates.

At a high level, a set of typical phases and associated document outputs may include:

Initiation Phase	yielding a **Project Charter** and/or **Vision** document
Analysis phase	yielding a **BRD** (Business Requirements Document) document, a **Functional Requirements** document and a **Non-Functional Requirements** document
Design phase	yielding an **Architecture and Design** and possibly a **wireframe** document in the case of UI oriented initiatives
Implementation phase	yielding a **Help & User** documents
Test / Validate phase	Yielding a **Test Plan** document
Deployment phase	yielding a **Deployment Plan** document

Visually, traditional (waterfall) projects are represented below.
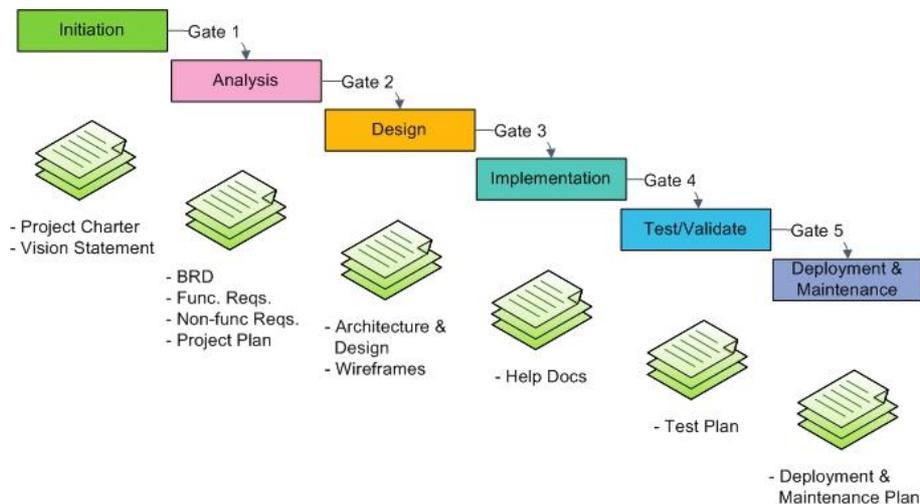

Figure 1 - A waterfall project with associated documentation artifacts

AGILE PROJECT DOCUMENTATION

Agile processes emphasize iterative delivery. Rather than assuming all detailed requirements and specifications are defined up front, agile projects take a 'just enough' approach. Agile projects anticipate change and bake that into the process.

From a documentation stand-point, this means agile has some considerations & challenges that are quite different from waterfall:

1. In Agile projects, documentation evolves and stakeholders look to keep documentation as light as possible, but not lighter. This latter point requires non-trivial judgment calls from experienced project

stakeholders including business analysts and project managers. Too little documentation and you are in trouble, too much and you are overloaded.

2. One of the core tenets of the agile approach is: 'working software over comprehensive documentation'. Many agile proponents misinterpret this to mean 'no documentation' whatsoever. This is very dangerous behavior, especially for horse and elephant projects. The objective is to arrive at 'just enough' documentation for the type of project.

3. Transient communication techniques are strongly emphasized in agile projects; examples include team discussions and physical artifacts such as index cards on a wall. These work well for the 'here and now' communication process where verbal communication augments story content and where teams are co-located. Transient communication, on its own however is insufficient for horse and elephant projects. You also need persistent communication.

4. Documents serve as the best form of 'persistent' communication; that is, the communication of record for agile projects. Persistent communication is required for larger projects in an agile context; to help communicate to non co-located project stakeholders, to help coordinate teams, to archive key business decisions and changes, to satisfy audit requirements, to help communication to the business and executive teams, to foster 'corporate memory', and finally; to help maintenance teams after deployment.

Let's consider a typical agile project from a documentation stand point.

The starting point; project inception, is similar to traditional projects. It involves establishing business justification and results in a **charter** and/or **vision** document at a high level.

Agile proponents refer to the stage after inception as iteration-0[1]. This is a special iteration where initial scoping and requirements analysis takes place. Other 'preparation' type activities also occur here; assembling the technical infrastructure, setting up test harness for test driven development, assembling the team etc.

Following iteration-0 comes a collection of iterations, each iteration delivering a slice of working functionality. Iterations are time-bound and will tend to be between 1 and 6 weeks. A backlog of prioritized activities is used to manage an iteration, with top priority items delivered at each iteration. The backlog is re-prioritized at the end of each iteration. Any created documentation is in parallel re-validated and adjusted accordingly.

Common to all of the agile methods (SCRUM, Crystal, XP etc.) is the notion that at conclusion of each iteration, a deployment should be possible. In practice, for larger horse and elephant initiatives, a collection of iterations will typically comprise a deployment.

So, the initiation phase / iterations and associated document outputs may include:

Initiation Phase	yielding a Project Charter and/or Vision document
Iteration-0	yielding a **lightweight BRD** doc and/or 'just enough' **Functional Requirements** and **Non-Functional Requirements** in documented format to begin iteration 1.

---

[1] There is some debate in the agile community as to whether an iteration-0 is a distinct phase, however up-front planning and requirements analysis activities to an appropriate level must be completed for agile projects to be successful.

|	
Iteration 1..n	each iteration yielding a revised version of the **Backlog,** modified **Use Cases** or **Stories**, **Non-Func Reqs.**, **Tests**, **Design Models**, possibly **wireframe** documents in the case of UI oriented initiatives and other ancillary documented models to allow the iteration to proceed.
Deployment	yielding **Help** and **Maintenance plan**

Visually, agile projects can be represented as:
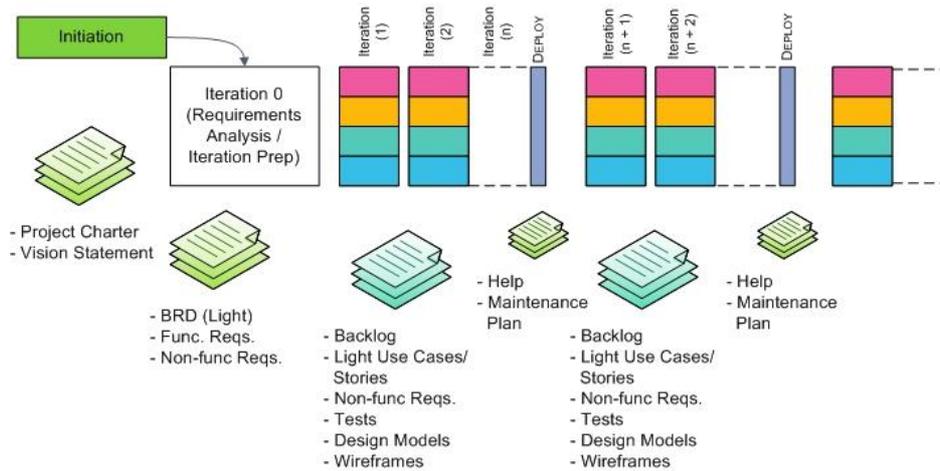
Figure 2 - Agile/Iterative Projects with possible documentation artifacts

AGILE DOCUMENTATION – RECOMMENDATIONS.

The following are some specific recommendations for agile project documentation for larger horse and elephant projects:

- **Use 'right-weight' Document Structures**. Agile projects require different structure and templates compared with their traditional project counterparts. In an agile scenario, you do not need to include the same amount of upfront detail. Documentation guidelines or templates should have a structure that represents the key aspects that are appropriate for the style of project. Throwing a full traditional template at an agile project makes little sense so aim to trim it down as appropriate.

Figure 3: A formal Use Case Template showing detailed levels of content. (VisibleThread structure reference)

Figure 4: Simple Use Case Template for more agile initiatives (VisibleThread structure reference)

- **Track Activity patterns & avoid stale documents**. Check editing activity levels for document sets. Agile project documentation will likely have changes occurring in the corpus of project documents on a very frequent basis. Either new documents will be created or existing documents edited on a just enough basis. For instance, a new detailed use case may be needed for a specific iteration, this may be added as a new doc or edited within an existing doc. Expect changes at least as frequently as every iteration.

Agile projects show flattened levels of activity spread evenly throughout the lifecycle. For traditional projects you expect to see changes concentrated in early analysis phases. If you are not seeing frequent updates for documents in an agile scenario you need to understand why.

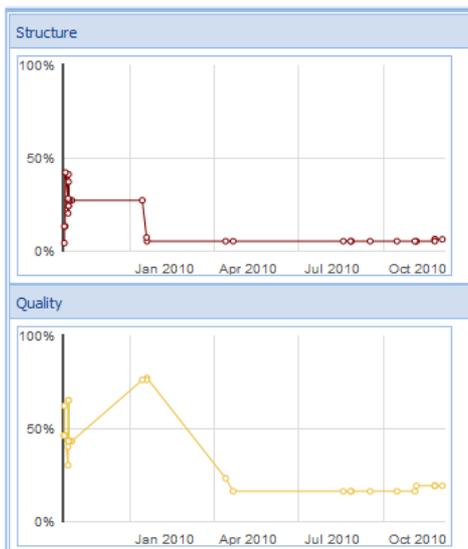*Figure 5: Activity levels for a BRD in a traditional project. Each plot point denotes an edit. The main concentration of edits occurs at the starting phase. (VisibleThread displaying activity levels for a traditional document)*

Figure 6: Activity levels for a document in an agile project. Edits are spread evenly across the lifecycle.(VisibleThread displaying activity levels for an agile document)

- **Track Document Quality**. Regardless of whether you are using waterfall or agile approaches, the content within the documents should adhere to 'SMART' principles, that is; they should be: Specific, Measurable, Achievable, Realistic and Time-bound. Check for weak and non-specific language in the specifications. Use tooling to scan documents in bulk for weak and ambiguous language.

Structure Analysis	**Quality Analysis**	Discovery	Concept Tracking	Activity	
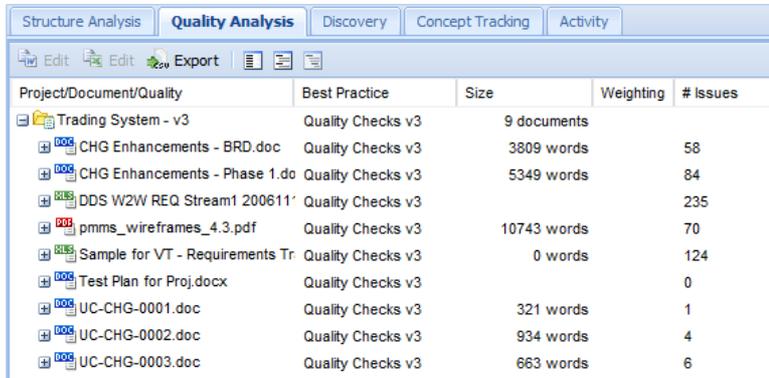
Project/Document/Quality	Best Practice	Size	Weighting	# Issues
☐ Trading System – v3	Quality Checks v3	9 documents		
⊞ CHG Enhancements - BRD.doc	Quality Checks v3	3809 words		58
⊞ CHG Enhancements - Phase 1.do	Quality Checks v3	5349 words		84
⊞ DDS W2W REQ Stream1 200611	Quality Checks v3			235
⊞ pmms_wireframes_4.3.pdf	Quality Checks v3	10743 words		70
⊞ Sample for VT - Requirements Tr	Quality Checks v3	0 words		124
⊞ Test Plan for Proj.docx	Quality Checks v3			0
⊞ UC-CHG-0001.doc	Quality Checks v3	321 words		1
⊞ UC-CHG-0002.doc	Quality Checks v3	934 words		4
⊞ UC-CHG-0003.doc	Quality Checks v3	663 words		6

Figure 6: A set of documents showing possible language issues. (VisibleThread displaying quality analysis)

- **Look for Dependencies**. Larger agile projects are rarely developed in isolation. When you have traditional and agile projects running side by side, identifying dependencies with existing systems components within and across projects are crucial. Due to the nature of the 'just enough' approach where teams are self-organizing, a risk is that not enough coverage of key concepts is outlined in the documentation and therefore the traditional project stakeholders are unaware of impact. Business rules and technical decisions affecting core domain entities need to be explicitly stated in documentation and tracked to facilitate impact analysis by adjacent teams.

- **Capture & validate NFRs explicitly, not as stories**. For both traditional and agile projects, missing the right non-functional requirements, scaling, performance etc. is high risk. It is a skilled activity and requires diligent analysis & validation techniques. This is doubly important in agile projects. Non-Functional Requirements are not functional entities; however they are often treated as stories in agile projects. This is risky. Treat them as distinct entities. Use tabular representations if possible in doc or spreadsheet form. Having testable NFRs foster more complete designs and promotes better test driven development as test cases (& test data) are easier to identify.

- **Conduct Pair Inspections**.Documents should be reviewed actively & informally as part of the authoring & document production cycle. Conduct these informal reviews once a week, with project stakeholders wearing different hats, e.g. BAs pairing with Testers etc. Because of the agile 'just enough' philosophy, judging how much content is necessary becomes an important skill. Pair inspections help this process.

- **Use Automation**. Larger agile projects within a portfolio require a higher degree of tooling & content monitoring than traditional projects. Document content needs to be analyzed in real time to avoid staleness, otherwise you run the risk of not enough up to date documentation. Solutions such as VisibleThread have been explicitly designed to automate the metrics you need for distributed agile teams. For larger horse and elephant projects, choose a solution that directly supports MS Office (MS Word and Excel) and can track document content metrics for both waterfall and agile projects across the portfolio.

CONCLUSION

Applying agile documentation practices side by side with traditional documentation practices is possible. When considering how to manage this, take a portfolio viewpoint. Understand that agile projects place a different

emphasis on documentation, with agile favoring a 'just enough' approach. Ensure that enough documentation is present, particularly for larger horse and elephant projects.

Adopt the recommendations outlined in this paper to safeguard project success and consider an automation approach to ensure integrity of your portfolio's documentation.

ABOUT VISIBLETHREAD

VisibleThread helps corporate IT departments create superior program / project documentation leading to successful project delivery for traditional and agile projects. Our document structure and quality analysis tools, combined with the ability to create tailor-made best practices documents, provide customers with the insight and metrics they need to make better decisions throughout the IT project lifecycle. VisibleThread ensures a uniform approach to IT documentation resulting in consistency across the project portfolio yielding higher quality outputs and lowered implementation cost.