Software Engineering Institute

Parallel Worlds: Agile and Waterfall Differences and Similarities

M. Steven Palmquist
Mary Ann Lapham
Suzanne Miller
Timothy Chick
Ipek Ozkaya

October 2013

TECHNICAL NOTE
CMU/SEI-2013-TN-021

Software Solutions Division

http://www.sei.cmu.edu

Carnegie Mellon

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the

SEI Administrative Agent
AFLCMC/PZE
20 Schilling Circle, Bldg 1305, 3rd floor
Hanscom AFB, MA 01731-2125

DM-0000488

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to thank John Hawrylak of the SEI for his insight reviewing this document.

In addition, the authors would like to thank the following members of the Agile Collaboration Group for their suggestions, comments, and reviews:

- Jennifer Walker (Raytheon Company)
- Curtis Hibbs (The Boeing Company)
- Richard Carlson (The Boeing Company)

Executive Summary

The purpose of this technical note is to help people who are familiar with the DoD's "Traditional World" of waterfall-based software development understand the terms, tasks and phases that are used in the "Agile World" of Agile software development methods. The technical note should also assist those readers who are more familiar with Agile software development methods better understand the DoD environment.

The technical note does not champion either development approach, but rather provides a Rosetta Stone[1] to help practitioners familiar with either development approach better understand the language used by the other. Nor does the technical note claim that the approaches are equivalent—only similar in that they use the same building blocks (but use them differently).

The first section of the technical note provides background material. It provides an overview of the waterfall software development method as well as a discussion of how this came to be the foundation of DoD's "traditional" approach to developing software systems.[2] This section goes on to provide an overview of Agile software development methods.

The second section of the technical note discusses some of the similarities between the Traditional World and the Agile World. The third section of the report is devoted to a single discussion of a philosophical distinction between the Traditional World and the Agile World.

The fourth section of the technical note is a series of tables describing 25 select Traditional World and Agile terms. Each table defines the term or concept, describes where it might be used, and identifies associated terms or concepts.

The technical note concludes with a summary and an appendix which provide greater detail about the origins of the waterfall software development method and its history in DoD.

The authors hope that this technical note stimulates discussions among practitioners in both the Agile community and the waterfall community so that terms and definitions can be added, updated, or removed as needed.

[1] The Rosetta Stone (Egypt, Ptolemaic Period, 196 BC) is a decree inscribed in a stone written in three scripts (hieroglyphic, demotic, and Greek); because it was the same decree written in multiple languages, it was an invaluable key to deciphering the hieroglyphs.

[2] We understand that the waterfall paradigm has morphed since it was first conceived in the 1970s, with the system engineering V diagram emerging as one of the most widely-used variants. For the purposes of this paper, however, we will use the term waterfall to characterize this approach.

Abstract

This technical note (TN) is part of the Software Engineering Institute's series on Agile in the Department of Defense (DoD). It primarily addresses what at first seems a small issue on the road to Agile adoption—the confusion of terms. However, this is a much larger issue, as ineffective communications among and between stakeholders is often cited as a significant stumbling block on any project.[3] Confusion over simple terms is a needless hurdle.

Many terms and concepts used by Agile practitioners seem to confound those working in the DoD's Traditional World of waterfall-based environment, and vice versa. The goal of this paper is to assemble terms and concepts from both environments to show both the similarities (of which there are many) and differences (of which there are also many).

A comprehensive cross dictionary was beyond the scope of this work; the authors strove to select from those terms most commonly encountered when considering Agile adoption. Therefore, the authors selected terms based on suggestions from both inside and outside the SEI, but deliberately limited themselves to 25 terms from each environment.

[3] Poor Communications, Unrealistic Scheduling Lead To IT Project Failure; K.C. Jones, Information Week; http://www.informationweek.com/poor-communications-unrealistic-scheduli/198000251

Introduction

Developing software using the waterfall paradigm or one of its derivatives has become so entwined with the Department of Defense (DoD) acquisition system (at least in perception) that it is often difficult to pry them apart. In light of that, it is the combination of these two spheres that we call the "Traditional World" in this technical note.

This means that the Traditional World is not just the waterfall software development methodology itself but is the entire environment, laws, and regulations that have grown up around it. This includes the acquisition community, the requirements community, the test community, the management community, the development community, the oversight community, and the like.

These stakeholders are the target audience for this report. In light of the DoD's recent emphasis on incorporating Agile software development methods into this environment, our goal is to help people familiar with the Traditional World understand the terms and concepts of the "Agile World." We also include a short description of the Traditional World to help Agile practitioners who may be unfamiliar with its history and concepts.

We will point out similarities and differences between the two methods (Traditional and Agile). The similarities in terms are not exact in most cases but only likenesses in each world. The parallels we draw we hope can be used to help dispel the fear of the unknown which could lead to rejection.

Section 1 begins with a brief overview of the waterfall design methodology as well as some context as to how it became to be the Department of Defense's tradition. We describe the waterfall development methodology separately from the DoD acquisition process, for they are indeed completely separate. However, for the dictionary tables that follow we included a number of DoD acquisition-related and systems engineering-related terms as we feel they are critical to our goal for this work. Some examples of these DoD acquisition or system engineering-related terms are "earned value management" and "critical path" as well as other terms and jargon such as "ball park estimate".

Section 1 continues with a brief overview of the Agile development methodologies, with an emphasis on eXtreme programming (XP) and Scrum expressions as they represent two of the most prevalent Agile methods.

Section 2 is a high-level discussion of some of the similarities between the Traditional World and the Agile World. It primarily focuses on the observation that both the Traditional World and the Agile World—as with all software development methods—use the same basic building blocks, such as requirements, test, design, etc.

Section 3 goes on to observe that while the basic building blocks are the same, the two worlds are separated by appreciably different perspectives on how these building blocks are used.

Section 4 is a Traditional-to-Agile and an Agile-to-Traditional dictionary to assist practitioners of both methodologies understand the terms used by the other.

Section 5 is a summary of the technical note.

In creating this document, the SEI drew from a number of sources including but not limited to these documents and websites:

1. *DAU Glossary of Defense Acquisition Acronyms and Terms, 14th Edition*, July 2011
2. http://www.aspe-sdlc.com; Agile Glossary: Words and Terms Common to Agile Methods
3. ISA/IEC/IEEE 24765 Systems and Software engineering – Vocabulary; December 15, 2010
4. http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx
5. *PMBOK Guide® – Third Edition*
6. http://www.accurev.com/wiki/agile-glosssary
7. http://www.develop.com/agiledemystified
8. http://xprogramming.com/book/whatisxp/

1 The Two Worlds—Traditional and Agile

1.1 Background

The DoD has a long history with software development, including a number of techniques that in today's parlance would be considered "Agile" or even "extreme." As with all emerging technologies however, software engineering has had its share of issues, in part due to the heavy influence of hardware-oriented development approaches on software development. In response to the "software crisis" of the late 1960s DoD took steps designed to control the development of complex software-intensive systems[4].

As one example, in the 1970's DoD created Ada to serve as a department-wide standard programming language to satisfy the Department's requirements for embedded and mission-critical software. DoD also hoped that Ada would encourage good software engineering.[5]

For a variety of reasons that seemed sound at the time, DoD began issuing a series of standards and policies that discounted or moved away from its more-Agile experiences, instead pushing a rigidly sequential, big design up front (BDUF), big test at the end, "document everything" approach.

This approach came to be called waterfall due to several common graphical representations of the approach. Even at its inception, however, many experts cautioned that the notion that DoD's large, complex software development efforts could be controlled with a rigidly-controlled, document-intensive and review-intensive approach would not work. There is some irony in that a similar message was very visible in popular culture at the time:

> "*The more you tighten your grip, Tarkin, the more star systems will slip through your fingers.*[6]"

History has unfortunately proven these cautions to be correct. It is true that projects and programs using the DoD acquisition paradigm and waterfall software development methods (i.e., the Traditional World as we have defined it) have delivered solid and even sometimes spectacular results. However, it is also true that the word "spectacular" was often used to describe a waterfall failure, not a success.

DoD did in fact begin backing away from waterfall almost as soon as it issued the initial mandates.[7] Even the term waterfall has not been officially used in DoD for a number of years.[8] Even

---

[4] For a brief description of this, please see Appendix A.

[5] Committee on the Past and Present Contexts for the Use of Ada in the Department of Defense, National Research Council. "The Changing Context for DOD Software Development." Ada and Beyond: Software Policies for the Department of Defense. Washington, DC: The National Academies Press, 1997

[6] Princess Leia to Grand Moff Tarkin; *Star Wars: Episode IV – A New Hope* (1977)

[7] In 1986, a draft copy of Revision A to MIL-STD 2167 appeared which removed the emphasis on top-down design and called out rapid prototyping as an alternative to the waterfall.

[8] Acquisition Strategy Considerations, 2000 Department of Defense Instruction Number 5000.2; October 23, 2000; the term waterfall is not used but is called a "single step to full capability."

with this, however, waterfall software development methodology continued to be a major—if not the dominant—influence on DoD software acquisition and development.

It was in part a reaction to the "analysis paralysis" and other waterfall issues that gave rise to the movement we now call Agile. In writing about the Agile Manifesto's origins, Jim Highsmith says the group was driven by "the need for an alternative to documentation driven, heavyweight software development processes" —which is how the waterfall methodology was (and is) frequently characterized [Highsmith 2001]. The Agile Manifesto is a short philosophical summary of the group's values regarding software development [Beck 2001]:

- Individuals and interactions are valued more than processes and tools.
- Working software is valued more than comprehensive documentation.
- Customer collaboration is valued more than contract negotiation.
- Responding to change is valued more than following a plan.

There is no hiding that if you over-emphasize wrong aspects of *process and tools*, *comprehensive documentation*, *contract negotiation*, and *following a plan* you will have the worst of waterfall. On the other hand, if you over-emphasize wrong aspects of *individuals and interactions*, *working software*, *customer coloration*, and *responding to change* you will have "cowboy coding"—a frequent but erroneous characterization of Agile. An incorrect emphasis on the left-hand side of each statement would also not be successful in the DoD environment. The right side is still of value and if completely ignored the development effort will fail. This is central to the debate about Agile scalability—a balance between the two must be maintained.[9]

Highsmith went on to say:

> *The Agile movement is not anti-methodology; in fact, many of us want to restore credibility to the word methodology.*
>
> *We want to restore a balance.*
>
> *We embrace modeling, but not in order to file some diagram in a dusty corporate repository.*
>
> *We embrace documentation, but not hundreds of pages of never-maintained and rarely-used tomes.*
>
> *We plan, but recognize the limits of planning in a turbulent environment.*
>
> *Those who would brand proponents of XP or Scrum or any of the other Agile Methodologies as "hackers" are ignorant of both the methodologies and the original definition of the term hacker.*

As Highsmith implied and we agree, the new world of Agile methods and the Traditional World's waterfall-based methods are not opposites; they are just different perspectives which place differ-

[9] Agile scalability is a topic of much interest, research, and debate and it beyond the scope of this report; however it is an important enough issue that we felt the need to acknowledge it.

ent emphasis on similar parts. Depending on the project's needs, a combination of both methods may be most appropriate. It does not have to be an either-or proposition.

Surprisingly, we did not discover much material exploring the concept that Agile methods and Traditional waterfall methods are simply different perspectives of and emphasis on the same fundamental activities. In many ways, we feel that this could have been one of the seminal discussions in the early stages of the Agile movement as it would have eased the misunderstandings and mistrust that plagued the early attempts to incorporate Agile principles into the Traditional waterfall world.

The remainder of this report will discuss the different perspectives that represent the Agile and Traditional views of software development. As we will argue, the *what* is the same (requirements, design, code, test, integrate, deploy), but the *how* can be quite different.[10] To make it more confusing, some of the same terms are used in both environment but have different meanings. Thus, the context is important. We will identify and define the more common terms used within both Agile and Traditional, provide definitions, and provide an explanation of how they do—or don't— relate.

1.2 Waterfall Overview

Note: this brief overview is intended for the reader who is not familiar with the waterfall software development model, and it is deliberately shorter than the Agile Overview as most readers are assumed to be from the Traditional World. For a more in-depth discussion, please see Appendix A.

Before we begin our discussion of the Traditional World, we want to emphasize that the people credited with the creation of the "waterfall method" apparently never envisioned it as a solution to DoD's complex software development problems. Dr. Winston Royce, the man who is often but mistakenly called the "father of waterfall" and the author of the seminal 1970 paper *Managing the Development of Large Software Systems*, apparently never intended for the waterfall caricature of his model to be anything but part of his paper's academic discussion leading to another, more iterative version [Royce 1970].

In his paper Royce argued for a more-iterative version of his "waterfall" model, and went so far as to say that even his more-iterative model was "risky and invites failure." He further goes on to discuss the additional steps he felt were needed even to allow even the more-iterative models to be successful. These included his recommendation that this model be run at least twice (iteratively), with the first time being a significant prototyping phase that was used to better understand the requirements, better understand the technologies involved, and ensure it was providing what the customers actually needed.

It is especially prophetic that Royce stated that "*one could expect up to a 100-percent overrun in schedule and/or costs*" if the additional steps were not incorporated.

Walker Royce, Royce's son, said this of his father:

[10] But the reader must be cautioned we are not talking about coding *per se*; coding is still coding in both worlds but there is a great difference in how requirements are prioritized and managed, work is planned, testing is conceived and implemented, etc.

"He was always a proponent of iterative, incremental, evolutionary development. His paper described the waterfall as the simplest description, but that it would not work for all but the most straightforward projects. The rest of his paper describes [iterative practices] within the context of the 60s/70s government contracting models (a serious set of constraints) [Larman 2003]."

Royce also emphasized a main element of what would be called "Agile" nearly three decades later when he recommended that the customer be involved well before testing as "for some reason what a software design is going to do is subject to wide interpretation even after previous agreement."

However—and for a variety of reasons the discussion of which is outside the scope of this paper—Royce's cautions were not incorporated into DoD software directives. For example, DOD-STD-2167 (1985) Section 4.8 Development methodologies states that "The contractor shall use a top-down approach to design, code, integrate, and test all CSCIs …" Also, the lower half of Figure 1 in DOD-STD 2167—the graphic used to illustrate software (CSCI) development—used a graphic from the early part of Royce's paper—not the graphic he used later in his paper, which at least indicated iterations.

In its most basic representation, the waterfall model has these main blocks[11] as shown in Figure 1:

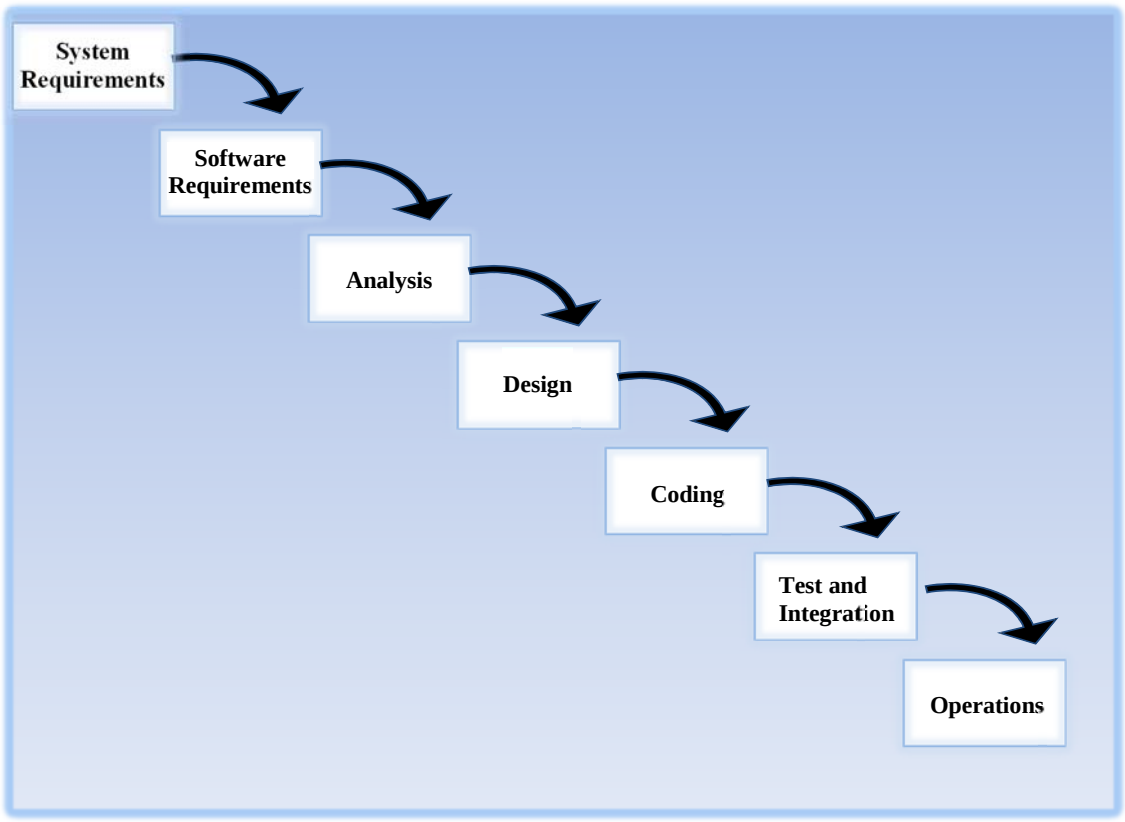


Figure 1: Basic Representation of Waterfall Model

[11] In this representation we have added Software Requirements as the follow-on step to System Requirements as many systems are not software only.

From this representation, we can see the main points that—rightly or wrongly—came to be the pillars of the Traditional World's waterfall-based software development approach:

- Systems are developed in a *sequential process* (at times even in a single pass).
- All requirements are determined *up front*, with the dual assumptions that they can all be *known up front*, and that they are and will remain *unchanging.*
- Analysis is done *once*, and precedes *design.*
- Design is done *once*, and precedes *coding.*
- All coding is done *once*, and precedes *testing and integration.*
- All testing is done once followed by or in conjunction with integration activities, and precedes operational use.

Not included in this depiction (except by a generous interpretation of the arrow linking each box) are these points:

- *Formal* review *and approval* is required to proceed from any one step to the next.
- The *customer is most involved setting the requirements*, mostly disappears during analysis design, and coding,[12] and re-emerges during test and acceptance.
- Extensive *documentation* is required for each and every step.

There were a number of reasons why this approach seemed sound, and we cannot overstate that this approach was a response to risk and uncertainty. This approach was seen as a risk management technique, and was born in a time when system development itself was both hardware-centric[13] and happened at a much slower pace than we see today.

Some of the key risk management points and their rationale that were incorporated into the traditional, waterfall-based world included

- early identification of all the requirements was needed to support planning and budgeting
- identifying and locking down the requirements early in the program would prevent scope creep
- documenting all aspects of design and decision-making was needed for future reference
- extensive reviews would not only ensure all stakeholders were aware of progress, but they would also allow issues to be uncovered earlier
- all requirements needed to be documented before any design work began to ensure the architecture was adequate
- all coding needed to be complete before test to ensure the entire system's capabilities could be evaluated
- all requirements had to be met before the system could be fielded

---

[12] The customer is involved in design reviews during these stages, but these have generally devolved into capstone meetings preceded by numerous technical interchange meetings

[13] "*The main goal of software development was to exploit the limited hardware resources (storage and processing power) in an optimal way*."; A Synopsis of Software Engineering History: The Industrial Perspective; Albert Endres; Position Papers for Dagstuhl Seminar 9635 on History of Software Engineering, August 26-30, 1996

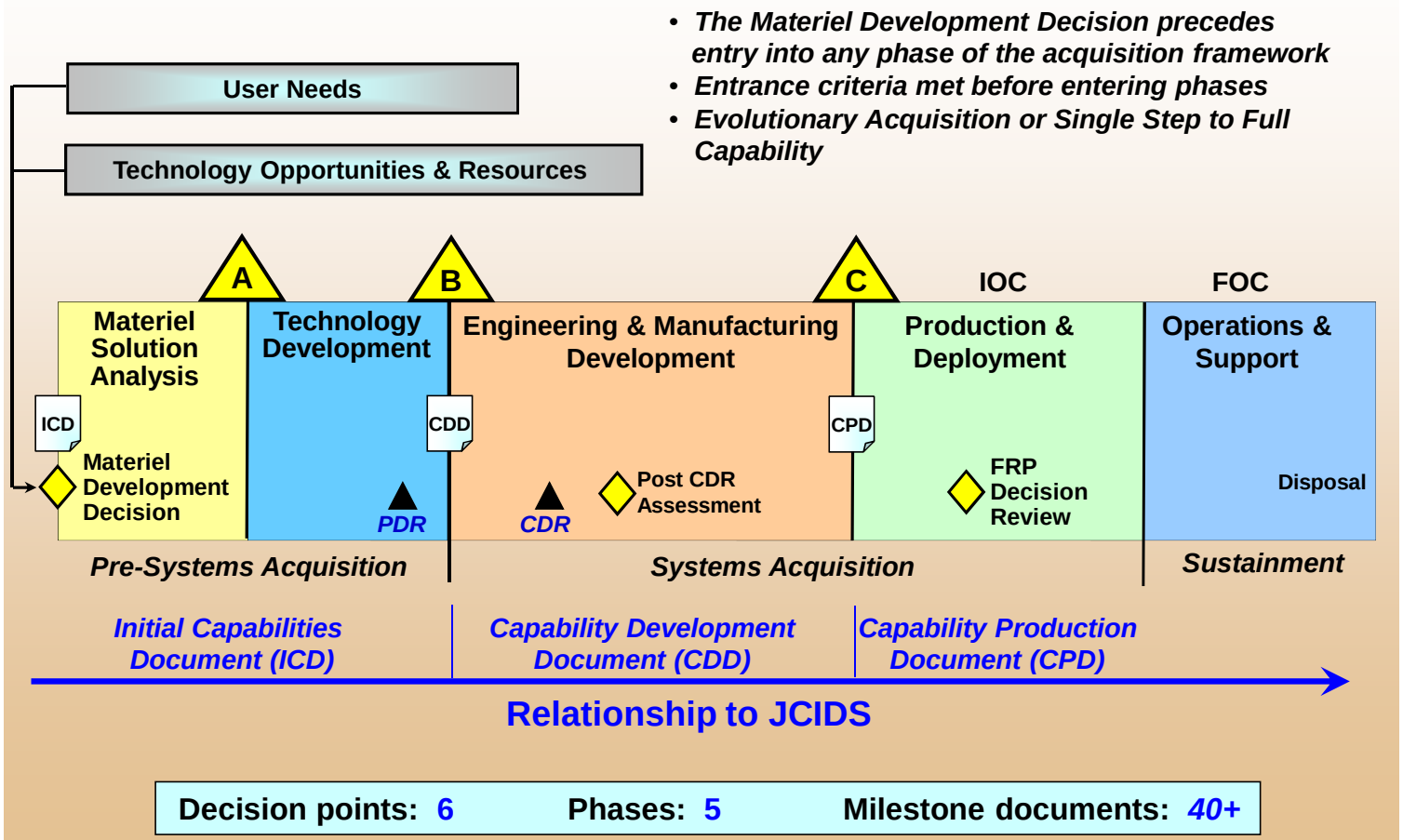
Taking all of these together creates the framework for the Traditional World. It has undergone many evolutions over the last several decades, including recent efforts to make it more "agile." For Example, Section 804 of the 2010 National Defense Authorization Act (NDAA) cites many principles that parallel Agile methods:

- early and continual involvement of the user;
- multiple, rapidly executed increments or releases of capability;
- early, successive prototyping to support an evolutionary approach; and
- a modular, open-systems approach.

As another example, Figure 2 is a depiction of the process from a Defense Acquisition University (DAU) presentation *The Defense Acquisition System;* note that the guidance indicates a program may proceed using either Evolutionary Acquisition or Single Step to Full Capability (i.e., waterfall) [Wills 2010].

While Figure 2 doesn't use the classic waterfall steps as it moves from left to right, it still maintains the basic concepts: system requirements are determined up-front, system analysis precedes design, system design precedes construction, and system construction precedes deployment.

The Defense Acquisition Management System



6 Aug 2010

7

Figure 2: DAU Representation of Software Life Cycle

However, while this report uses the waterfall methodology as our primary framework for the Traditional World, we don't mean to say that waterfall is the only software development paradigm used by the Traditional World. For example, Figure 3 is from the Federal Highway Administration's Systems Engineering for Intelligent Transportation Systems.[14]

As the reader can see, the yellow-boxed phases across the top effectively match the phases from the DAU document. The blue-boxed system engineering V also effectively matches the waterfall process: development is still sequential, requirements are determined up-front, analysis is done once and precedes design, etc.

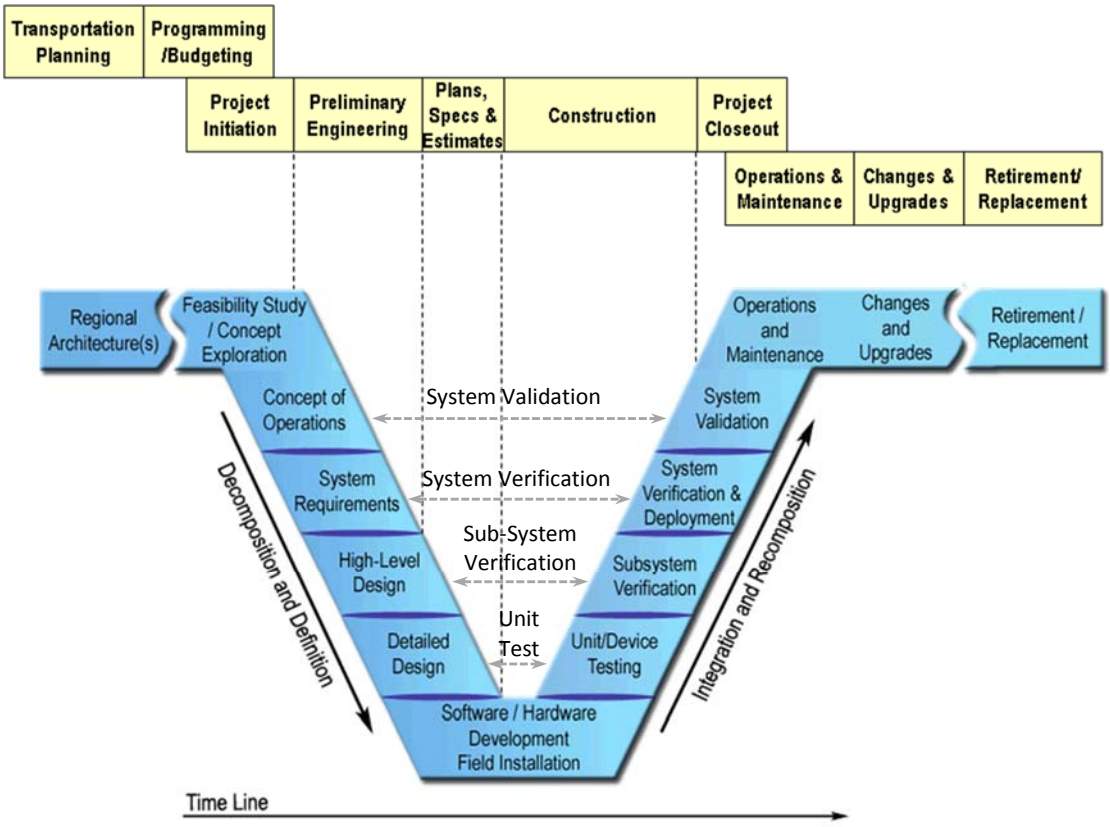


Figure 3: The System Development V

It is true that most major development programs can or will use the waterfall or the system engineering V process many times as a system moves through the different phases in the program life cycle. It is also true that the V model also incorporates testing at each phase (continuous or near-continuous test and integration is emphasized in Agile methods). However, it still retains the same basic issues that plague the waterfall paradigm:[15]

[14] Adapted from http://ops.fhwa.dot.gov/publications/seitsguide/section6.htm

[15] Adapted http://www.waterfall-model.com/v-model-waterfall-model/

- It assumes that the requirements and their order and priority do not change.
- The design is not authenticated until later in the program, and only then as part of system and sub-system performance.
- At each stage there is a potential of errors; the first testing is done after the design of modules which is very late and costs a lot.

1.3 Agile Overview

Agile is not one specific method; Agile is both a philosophy and an umbrella term for a collection of methods or approaches that share certain common characteristics. One definition for Agile is [Lapham 2010]:

An iterative and incremental (evolutionary) approach to software development which is performed in a highly collaborative manner by self-organizing teams within an effective governance framework with "just enough" ceremony that produces high quality software in a cost effective and timely manner which meets the changing needs of its stakeholders.[16]

This definition is rather long but it covers our purposes. If a shorter definition is desired, Alistair Cockburn has said that Agile is the "… early delivery of business value. That involves early and regular delivery of working software, a focus on team communications, and close interaction with the users."[17]

To mimic our waterfall characterization, Agile development approaches have these main points:

- Requirements are *characterized* up-front, and assumed to be *changing.*
- Systems *evolve during a series of short iterations,* where *analyze, design, code, test, and potentially shippable code happens each iteration.*[18]
- *Customer participation and approval is required during each iteration and to make the decision to proceed to the next iteration.*
- *Documentation is developed only as needed and is often tailored for the project.*

As a first foray into how Agile is used, we will present an example of how Agile methods might be used on a government software development project. We must be very clear—this example is for discussion only. It is not meant to be all-inclusive, it includes steps or processes that are not used by all Agile practitioners, and it incorporates elements of multiple Agile methods.[19]

Our purpose for this description is to help someone not familiar with Agile methods see how they might be used in a government acquisition program. Again—this should not be interpreted as a recommended approach but as an illustration.

---

16 http://www.agilemodeling.com/essays/agileSoftwareDevelopment.htm

17 http://bradapp.blogspot.com/2006/05/nutshell-definitions-of-agile.html

18 This is a nominal representation as sometimes iterations are collected into a release before going to operations.

19 Principally Scrum and eXtreme programming, or XP; sources include *eXtreme Programming and SCRUM: A Comparative Analysis of Agile Methods* by Nicholas R. Zuiderveld (Portland State University), http://www.scrumalliance.org/learn_about_scrum, and http://www.jera.com/techinfo/xpfaq.html

- The government describes its needs by creating a vision for its system.
- The government creates a list of relatively coarse-grained, high-level requirements, and indicates which are the most important — i.e., the government creates a *prioritized backlog* of product requirements.
- Several development contractors describe their proposed approach to meeting the government's needs, and this includes a high-level cost estimate.
- After the government selects one development contractor, the government and the contractor agree on the overall scope of the project—the system goals, the budget, and the schedule.
  - The government and the development contractor jointly create an overarching, high-level plan or *roadmap.*
  - Both parties agree that the government's description of its needs—however thorough—is incomplete and will evolve (i.e., requirements and their priority will change).
  - Both parties agree that the development contractor's software development process – however efficient—can be managed but not fully planned (i.e., risks happen).
- The high-level requirements are fleshed out by putting each into a context of environment, inputs, products, etc. to make them easier to understand and communicate; i.e., a small *story* is written about each.
- The development contractor refines its estimates of how long and how much it will take to build the capability or feature set needed to implement a requirement/story; in addition it assesses the risks associated with each.
- The product backlog is then *groomed* to balance the government's priorities (what the users need first) and the development contractor's risks (what risks and issues must be explored and resolved in order for system construction , deployment, and operation to be successful).
- Using the prioritized product backlog, the Roadmap is then broken down into a series of *releases.*
  - The requirements are allocated in light of a cross-cutting view of the overarching high priority architectural requirements that must take place to achieve the success of the iterations.
  - The highest priority requirements/stories are allocated to the first release to create its release *backlog*, with the next highest priority requirements allocated to the second release to create its release backlog, and so on.
  - Each release in the roadmap has a *release plan*, with earlier releases having greater detail than later releases.
- The releases are broken down into one or more *iterations* (or *sprints*) which can range in duration from two weeks to two months—but two to four weeks is typical.
  - The iterations are deliberately kept to short, manageable time blocks to better manage planning and maintain team focus as well as to allow for frequent product demonstrations and team process improvement.[20]

[20] In a typically IT-oriented project, this would be the rate of consumption of features, but in a project such as a safety-critical aircraft project not all of these releases are customer-facing but can be internal releases.

  - The refined cost and schedule estimates are used to ensure the work allocated to the iteration is realistic.
  - The highest priority requirements/stories from the release backlog are allocated to the first iteration to create its *iteration backlog*, the next highest priority requirements are allocated to the second iteration to create its iteration backlog, and so on.
  - Each iteration in a release has an *iteration plan*, with earlier iterations in the release planned in greater detail and later iterations planned in less detail.
- The iterations are broken down into a series of *daily work* assignments (sometimes called *tasks*) for the development team.
- During each day's work
  - The team holds a *daily stand-up* meeting where each team member states what work they completed, what they will work on next, and identifies any issues or roadblocks they are facing.[21]
  - The code base is *continuously integrated* and often tested at least once a day,[22] and the code base must pass all tests after integration.
- During each iteration
  - The government (i.e., the customer) cannot add more requirements to the iteration, but it can clarify the requirements already allotted to the iteration.
  - The team maintains large, visual displays of progress and problems to keep all team members informed; these displays (sometimes called *information radiators*) can be electronic and may be accessed via a tool as opposed to just displayed.
  - The release backlog is groomed and the plan for the next iteration is refined to allow for uninterrupted work (i.e., rolling wave planning).
- At the end of each iteration
  - The development contractor will demonstrate or deliver some working, useful capability to the government.[23]
  - Once the government is satisfied, the capability is prepared for release or carried forward into the next iteration (depending on the release plan).
  - If appropriate, training materials and documentation are completed.
  - If there are any unfinished capabilities, they are placed back in the backlog. Reprioritization may occur as their priorities may have changed and depending on the "new" priority they could be included in the next iteration.
  - The government and the contractor hold a *retrospective* to review what worked and what didn't work.
- At the end of each release
  - The development contractor will deliver at least one useful capability.

---

[21] Solutions to the issues or roadblocks are not reached during the stand-up. Follow-up meetings or discussions are scheduled to accomplish that effort.

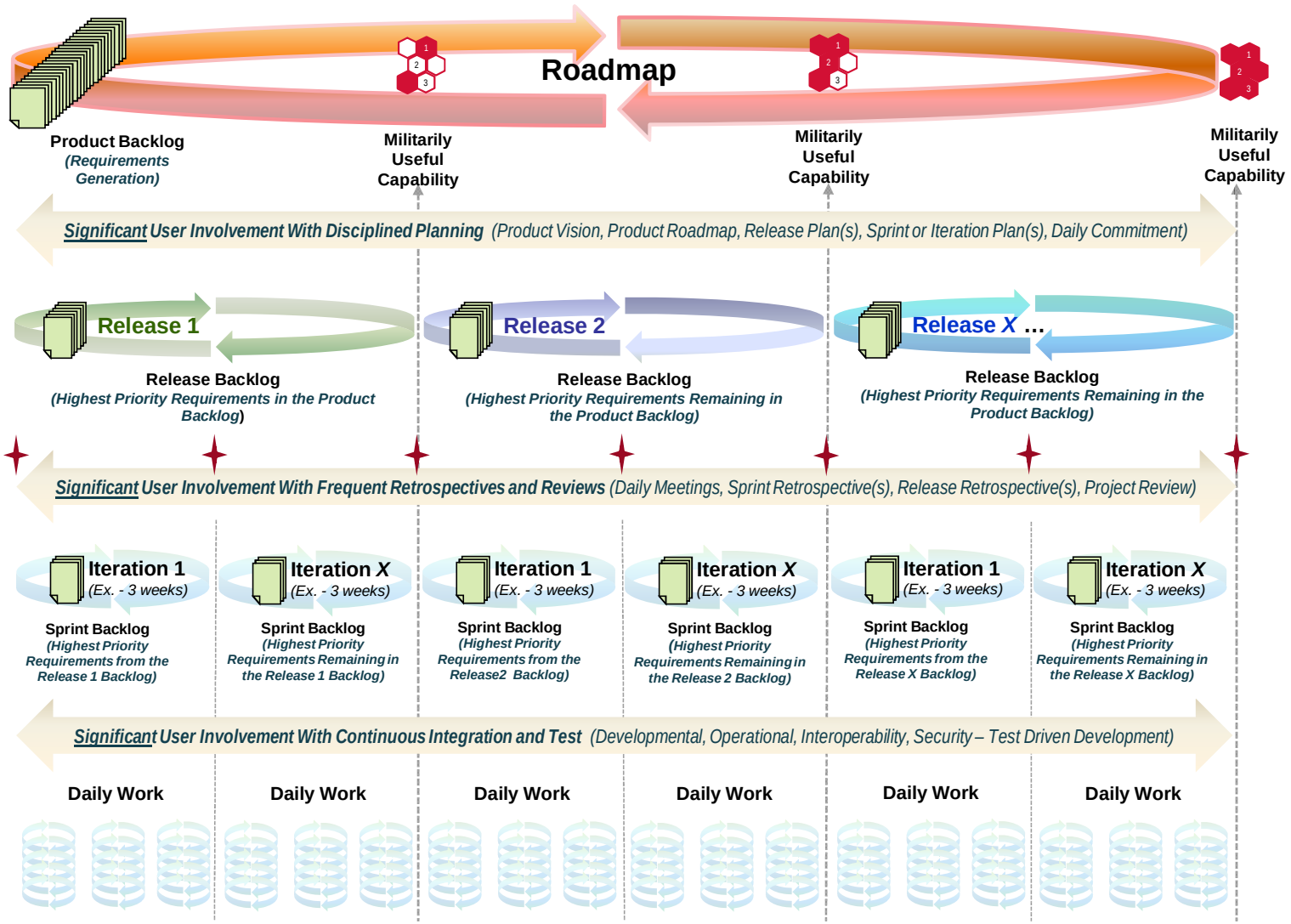
[22] Continuous integration and regression testing is typically employed at least daily, and sometimes more often depending on how the processes are instantiated.

[23] This is potentially shippable code. However, many times it is not deployed for operational reasons or perhaps because it provides some of the infrastructure for the rest of the software.

- Releases and iterations continue until
  - All desired capabilities have been delivered (the government can chose to accept a less-than-complete product if “enough” of the capabilities have been incorporated).
  - The money runs out.
  - The schedule deadline is reached.

Unlike waterfall, government user involvement is heavy during all phases of the work. In fact, the need for dedicated government representatives (product owners in Agile parlance) skilled in Agile methods is one of the biggest challenges that Agile brings to the Traditional World.

Figure 4 shows this Agile process.



1

Figure 4: Agile Life Cycle

An important point here is that Agile is a disciplined planning process, including understanding requirement dependencies, potential groupings and infrastructure needs. Agile planning also includes other technical practices such as configuration management, testing, and the like as part of this disciplined planning perspective.

2 Similarities—The Same Basic Building Blocks

2.1 Similarities—Traditional and Agile Share the Same Goal

What is sometimes lost in Traditional World/Agile World discussions is the fact that both groups have the same goal—to deliver a quality product in a predictable, efficient and responsive manner. Both worlds do the same "types" of things—define, gather, analyze, design, code, test, release, maintain, retire—it's how they do these things that are different.

However, we want to point out that while there are similarities, there are significant differences. The two methods cannot be thought of as the same.

2.2 Similarities—The Traditional World and the Agile World Use Many of the Same Principles

As we showed in a previous section, the Traditional World grew out of the perception that the best way to manage the "software crisis" was to:

- plan the work out completely before beginning
- lock down requirements early
- institute multiple reviews[24]
- move forward in a step-by-step, sequential manner
- move forward only when all parts of the previous steps were complete
- capture all details with extensive documentation

Taken individually, it's difficult to argue with these if they are appropriate (i.e., you really can state all your requirements up front) and they are done wisely. For example, gold-plating should be avoided; progress reviews are reasonable management tools; senior leaders do need to be kept informed of progress and issues; designs should be documented to support future work, etc.

Because these principles have value, they are used in the Agile World as well; here is an example of the parallels:

[24] It is ironic that what is frequently lost in the Agile-Traditional discussion is that Agile's emphasis on frequent demonstrations of working software constitute and facilitates the review process, only in a more realistic fashion.

Traditional Principles	Agile Instantiation
Plan the work—especially the budget, schedule, and deliverables—to the maximum extent possible before beginning any design or code.	• Near-term plans contain more detail, while plans further out on the time horizon contain fewer details. • The overall vision is broken down into a roadmap, which is further broken down into release plans, which are further broken down into sprint or iteration plans, which are further broken down into daily plans. • Requirements are prioritized. • Cost and schedule estimates are prepared for each capability at a high level. Relative estimation versus absolute estimation is employed. • Frequent planning sessions (at the beginning of each iteration) result in detailed, high-fidelity plans. • Risks are assessed and risk mitigation influences planning.
Lock down requirements to prevent gold-plating and scope creep.	• No requirements can be added to an iteration once it has started. • New requirements are evaluated by the stakeholders and prioritized thus preventing gold-plating and scope creep.
Institute multiple reviews to provide senior leadership oversight as well as to serve as gates for continued work.	• The customer is involved in all aspects of planning and testing. Customer (in the form of the product owner) is involved daily. • There are reviews at the end of each iteration that serve as gates to further work.
Move forward in a step-by-step, sequential manner and only when all parts of the previous steps were complete.	• The code base is integrated and tested daily. • The code base must pass all tests before and after integration. Regression testing is typically done each night.
Capture all details with extensive documentation.	• There is an overall plan. • There are requirements descriptions. • There are cost and schedule estimates. • There are risk assessments. • There is training material (as appropriate). • There is documentation (as appropriate). • There are lessons learned (based on retrospectives).

Table 1: Agile Instantiations of Traditional Principles

2.3 Similarities—The Traditional World and the Agile World Use the Same Basic Building Blocks

Both the Traditional World and the Agile World also work with the same basic programmatic building blocks:[25]

- scope
- cost
- schedule
- performance

In its simplest form, the Traditional World sets the scope up front (through requirements) and then allows cost, schedule, and performance to vary. Again in its simplest terms, the Agile World sets the cost, schedule, and performance up front and then allows the scope to vary.[26]

[25] For simplicity, performance includes all of the "ilities"—quality, interoperability, security, modifiability, etc.

In addition, both the Traditional World and the Agile World use the same technical or development building blocks:[27]

- *analyze* the requirement
- *design* a capability to satisfy the requirement
- *build* the capability
- *test* the capability to ensure the requirement is met
- *deploy* the capability

In the Traditional World, the requirements are fixed and the five building blocks move forward en masse each step in sequence with heavy documentation and formal approval required, as shown in Figure 5:

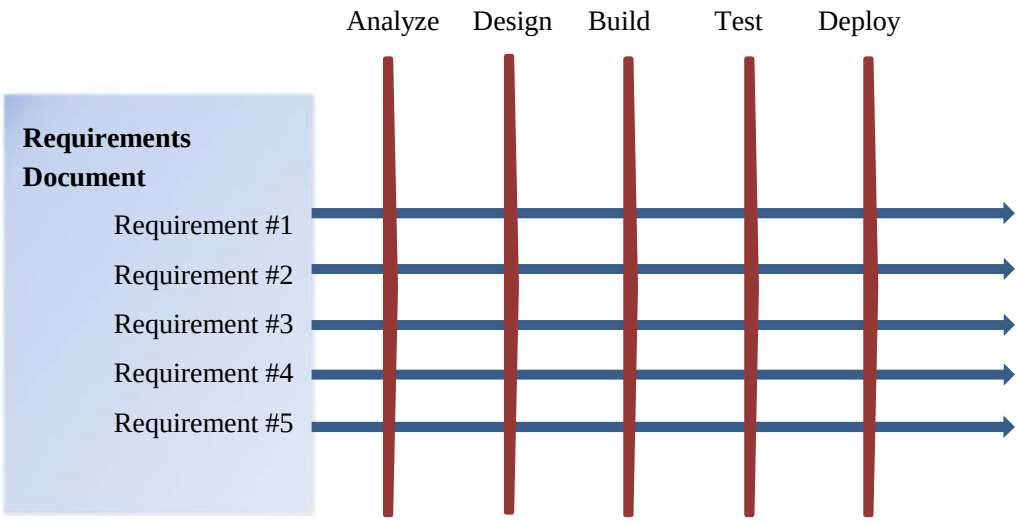


Figure 5: Requirements Moving En Masse Through the Process

Sometimes these requirements are "blocked out" or delivered in planned increments[28] as shown in Figure 5, but the effect is still the same because the requirements for all of the blocks are determined all the way to the left up front. Blocking and increments then are simply techniques to manage schedule and resources but the sequential paradigm remains (see Figure 6).

---

[26] As with all generalities, there is danger in this simplification. For example, many DoD project managers would argue that their schedule is fixed more than scope (requirements), though more accurately both schedule and requirements are fixed together.

[27] Material for this section—in particular the graphics—is adapted from http://www.agile-process.org/process.html

[28] Sometimes referred to as Pre-Planned Product Improvements, or P^{3}I

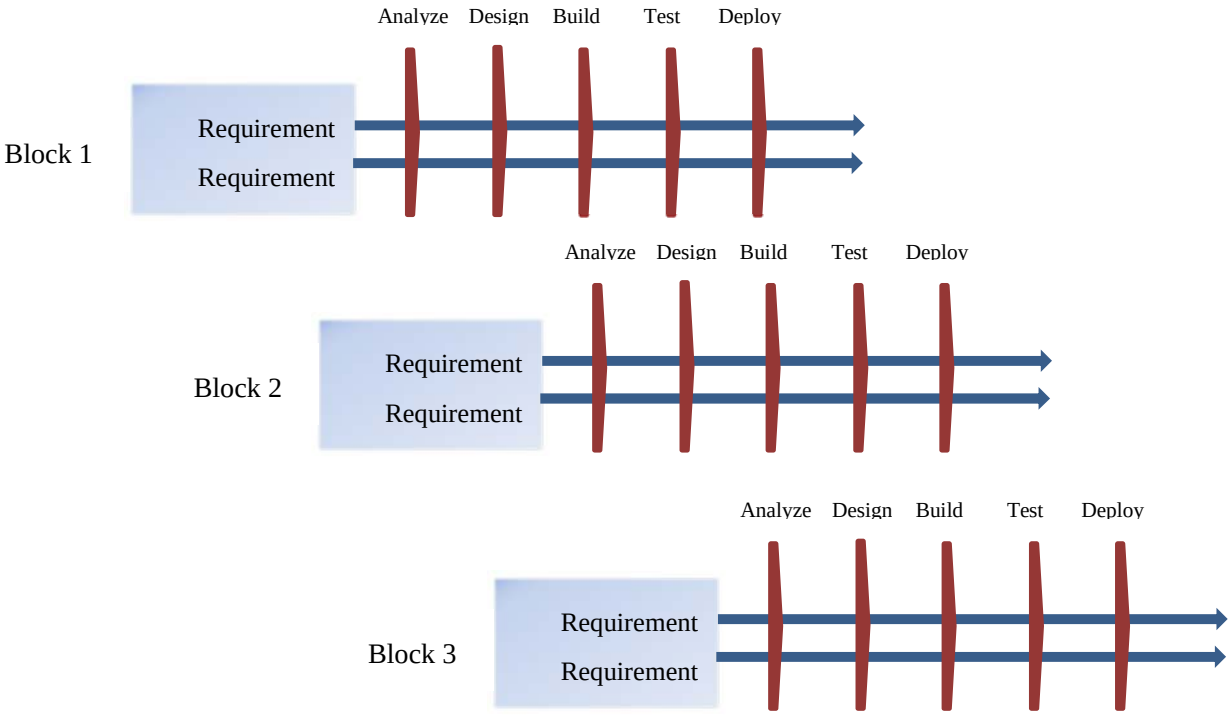


Figure 6: Blocking and Increment Techniques

The Agile World uses the same building blocks—it just looks at these things differently than the Traditional World as shown below (Figure 7).

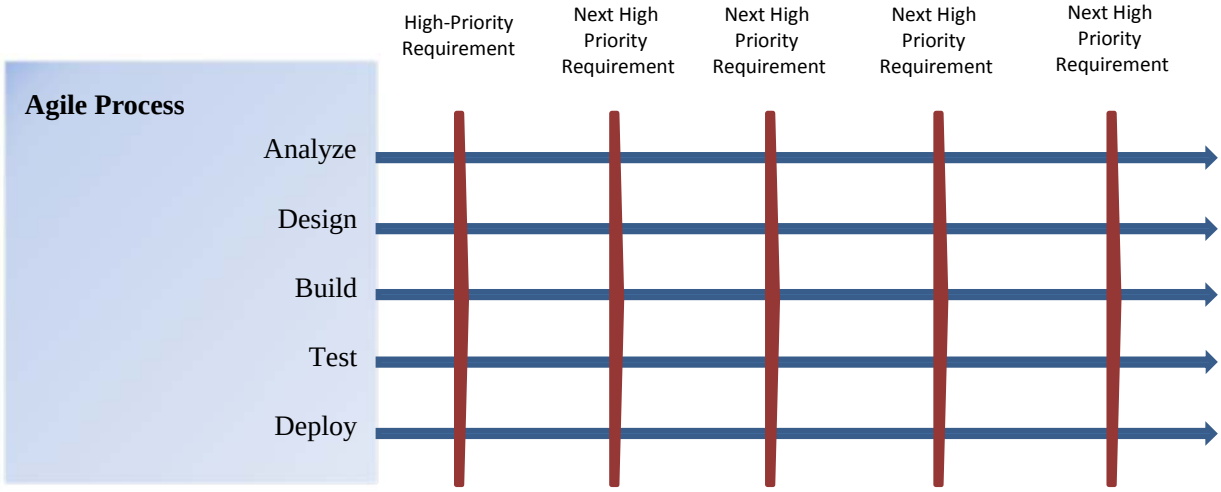


Figure 7: Agile Building Blocks

Pure Agilists may not fully agree with this representation, but it does convey two key points in the context of this technical note. The first is that Agile does all of the things with which a Traditional World person is familiar—they are captured on the left-hand side. That leads to key point number one—Agile methods are disciplined, managed processes.

Key point number two is that what's "built next" is always evolving—it's the highest priority as each iteration begins. One argument against Agile is that this flexibility is Agile's undoing in the world of immense DoD systems. The Traditional World already struggles with changing requirements when requirements are supposedly fixed—how could Agile possibly work in the DoD if customers were actually encouraged to add or reprioritize requirements?

And to do that *every 30 days*?

There are several flaws in this thinking. First, it's true that an undisciplined customer could attempt to continuously move their latest "flavor of the month" to the top of the backlog. However, for a requirement to rise to the top of the priority list it must be justified and it must have real return on investment determined.

Second, it is true that Agile—as with any development approach—can produce the wrong system if flaws exist in many areas:

- the initial description of the need (reacting to symptom, not causes, etc.)
- setting the initial scope
- breaking down the requirements
- initial risk assessments
- initial cost and schedule estimates
- and so forth

However, Agile is geared towards detecting these flaws—it is *designed* to fail early, correct course, learn, and improve. As a simple example, in an Agile program working software is tested and integrated daily and demonstrated to the user as often as every two to four weeks, which allows for frequent assessments as to whether the program is on track.

In the Traditional World, building the wrong system is more likely due to the fact that the requirements are set years in advance and are deliberately resistant to change. Given

- the time between the requirements studies and the creation of a capstone or requirements document
- the multiple years to get into the federal budget process
- the months-to-years to select a contractor
- the (possibly) multiple years before a product is delivered

programs are often lucky the requirements were only half a decade old.

In the Traditional World, requirements don't change. However, the likelihood that a product built to fixed, half-decade old requirements will still fully meets the users' needs and expectations in an ever-changing world of threats and technology is simply not realistic.

3 Differences—Significantly Different Perspectives

3.1 Differences—Forward-Looking Perspective vs. Backward-Facing Perspective

Perhaps the most important section in this technical note:

- In a dynamic environment like the DoD, the Traditional World struggles to deliver as it constantly looks back at long-fixed requirements and priorities.
- In a dynamic environment like the DoD, the Agile World adapts as it delivers by constantly looking forward at evolving requirements and priorities.

4 Differences—Terms and Concepts

4.1 Differences—The Traditional World and the Agile World Do Not Use the Same Words (Or If They Do, They Don't Always Have the Same Meanings)

It is a bit like British English and American English, where sometimes the same word can mean different things. For example, in England a caravan is a towed trailer that provides a place to sleep when on a vacation,[29] while in the U.S. a Caravan is a minivan made by Chrysler, or a stream of cars all going to the same place, as in "let's caravan to the lake this weekend."

Alternatively, the same item can be called by different names. Consider the large piece of sheet metal covering the engine bay on most cars. It's a basic component of every car in the world, but why do two people with presumably the same language and presumably the same goal (e.g., check the oil) call it by such different terms?

But then again, sometimes the engine is in the back or the middle, so "hood" or "bonnet" doesn't describe what you want in either case, which actually is a good segue back to the issue at hand.

The Traditional World and the Agile World both use the same principles and building blocks—just like all cars have engines—but they are put in different places for different design or performance considerations, and are sometimes accessed via different routes.

With that in mind, we've selected a small set of terms, activities, products, or roles from both the Agile World and the Traditional World to define and show how they do—or do not—relate. The number of terms could have been far higher—it seems that each one we picked led us to two or more terms that needed to be included. However, we strove to keep it reasonably compact, though we welcome any additions, deletions, corrections, or elaborations.

NOTE: The definitions were drawn from many sources. However, in almost all cases the definitions were modified or shortened to better fit the limitations of the table. In those cases where a suitable definition was not available, we created one based on our overall Agile research.

In all cases, the authors encourage readers to read the original source definitions if they have any questions about the term or concept. In addition, we welcome and encourage any feedback

4.1.1 Agile World and Traditional World Terms

We limited our selection to 25 terms or concepts from each world. Arguments can be made for or against including each term, and even stronger arguments can be made for including more, but we had to bound this paper. Readers will also see that some terms have escaped from their process context and have become general, like Kleenex—a brand name that has come to signify an object produced by many manufacturers. These terms are provided in that spirit, not necessarily in the context of the process.

[29] Even "vacation" can lead to misunderstandings as in England it would be called a "holiday" whereas in the U.S. a "holiday" generally refers to a specific holiday like Christmas, the Fourth of July, etc.

The terms and concepts are listed alphabetically, and are most easily read by going down each column rather than left-to-right on each row. There is not an intentional relationship between Agile terms and Traditional terms that line up across from each other in the tables; each term was selected based on its own significance.

Agile World
Agile
Backlog
Burn-Down Chart
Complexity Point
Continuous Integration
Done
Epic
eXtreme Programming (XP)
Feature
Five Levels of Agile Planning
INVEST
Pair Programming
Planning Poker
Product Owner
Refactoring
Release
Retrospective
Roadmap
Scrum
Sprint (or Iteration)
Story (or User Story)
Story Board
Technical Debt
Velocity
Vision

Traditional World
Ball Park Estimate
Bar Chart (or Gantt Chart)
Critical Path
Derived Requirements
Earned Value Management
Entry Criteria/Exit Criteria
Function Point
Increment
Inspection
Integrated Master Plan/Integrated Master Schedule
Integrated Product Team
Key Performance Parameters
Lightweight Process
Milestone A/Milestone B/Milestone C
Oversight
Peer Review
Performance Measurement Baseline
Preplanned Product Improvement (P^3I)
PERT (Program Evaluation and Review Technique)
Progressive Elaboration
Prototype
Requirements Scrub
System Specification
Traditional Waterfall Methods
Work Breakdown Structure

Agile Terms: Agile

<table>
<tr><th>Definition[30,31]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>As we use the term in this technical note, it is the group of various software development methodologies that emphasize most or all of these points:
requirements evolution
iterative development
continuous test and integration
frequent progress demonstrations
frequent delivery of working code
on-going, direct communication between the customer and developer
self-organizing, cross-functional teams
frequent retrospectives promoting continuous improvement</td><td colspan="2">As we've tried to show, many Agile concepts are not new to the Traditional World – it's a matter of emphasis and context.

However, perhaps the greatest difference between the Agile World and the Traditional World is Agile's explicit acknowledgement and support for evolving requirements.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?[32]</th></tr>
<tr><td rowspan="3"></td><td>Fake Agile (Fragile)</td><td>A project that declares itself Agile but doesn't embrace Agile; such a group typically dictates its own delivery schedule and stops writing documentation but doesn't adopt test-driven development or any other practice they dislike.</td></tr>
<tr><td>Hitting the Scrum Wall</td><td>An initial improvement in productivity and customer satisfaction after adopting Scrum management techniques that comes to an abrupt end because other Agile practices were not adopted.</td></tr>
<tr><td>ScrumBut</td><td>When a project claims to follow Scrum but misses or avoids important practices as in
"We do Scrum – but we don't have a product owner"
"We do Scrum – but the project manager allocates tasks."</td></tr>
</table>

30 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

31 Term and definition(s) adapted whole or in part from http://www.accurev.com/wiki/agile-glosssary

32 All three terms and definition(s) adapted whole or in part from http://www.develop.com/agiledemystified

Agile Terms: Backlog

<table>
<tr><th>Definition[33]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>The backlog is a prioritized list of stories (or user stories) and defects ordered from the highest priority to the lowest.

Backlogs were developed in the context of Scrum, but are now used widely in many Agile methods.

Backlogs include both functional and non-functional stories (or user stories) as well as technical team-generated stories. There are three types of backlogs.
The product backlog contains all of the requirements and is the highest level.
One level down is one or more release backlog(s), which contain the product backlog requirements as allocated into releases.
Two levels down is one or more sprint (or iteration) backlog(s), which contain the release backlog requirements as allocated into the sprints.
In addition, some Agile teams use the concept of a daily backlog, which are composed of the sprint or iteration backlog requirements as they are allocated into daily work assignments.</td><td colspan="2">This term is a frequent source of confusion as it has fundamentally different meanings and connotations.

In the Traditional World, having a backlog is not desirable because the backlog is what the program expected to do but didn't or couldn't perform. In the Traditional World, the backlog of unfinished or deferred work usually falls out of the work baseline.

However, it's true that in both worlds not all work planned is accomplished. In Agile, the unfinished work from a sprint (or iteration) is returned to the release backlog or the product backlog where it is prioritized and included in planning for future work.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>The concept of backlogs is a cornerstone of Agile and is used during each of the five levels of Agile planning:
Vision
Roadmap
Release
Sprint (or Iteration)
Daily Work</td><td>Backlog[34] (Traditional World definition)</td><td>Known work input that is beyond an organization's capacity or capability for any given period of time.</td></tr>
</table>

[33] Term and definition(s) adapted whole or in part from http://www.accurev.com/wiki/agile-glosssary

[34] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Agile Terms: Burn-Down Chart

<table>
<tr><th>Definition[35]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A visual tool displaying progress via a simple line chart representing remaining work (vertical axis) over time (horizontal axis).</td><td colspan="2">There isn't a direct peer in the Traditional World for the Agile burn down chart. However, both the Traditional World and the Agile World have charts with the same fundamental purpose—to show progress against time in an easily understood graphic.

In Agile, burn-down charts (tracking the work completed) and burn-up charts (tracking the work remaining) in essence capture the same type of information.
These same charts also characterize the information in another manner as the team's velocity refers to the slope of the line on the chart (this is not explicitly captured in most EVM processes).</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">As a visual display of progress, burndown charts are normally used at a release level as well as the sprint (or Iteration) levels.</td><td>Burn-Up Chart (or Graph) [36]</td><td>A visual tool displaying progress via a simple line chart representing work accomplished (vertical axis) over time (horizontal axis).

Burn-up charts are also normally used at a release level as well as the sprint (or iteration) levels.

Agile burn-up charts are conceptually equivalent to the Traditional World's earned value accumulated at a specific date [Cabri 2006].</td></tr>
<tr><td>Earned Value Management (EVM) [37] (Traditional World definition)</td><td>A method combining scope, schedule, and resource data into a measure of performance and progress by comparing what was budgeted for a task (time and resources) against what the task actually required (time and resources).

A key criterion in EVM is making the "percent complete" calculations, which is related to the Agile concept of "done." It is also related to EVM's schedule performance index (SPI) and budgeted cost of work scheduled (BCWS) which is also known as planned value.</td></tr>
</table>

[35] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[36] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[37] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

Agile Terms: Complexity Point

Definition	**Is there an equivalent in the Traditional World?**	
Complexity points are units of measure used to estimate development work in terms of complexity, but not effort—effort is measured by story points.	No direct peer *per se*; however the Traditional World's use of *function points* and *function point analysis* attempts to address similar issues (though the use and application of function points is not universally or consistently used).	
How is it used in Agile?	**Are there any related terms or concepts?**	
Complexity Points are most often used during planning at the release level as well as the sprint (or iteration) levels.	Story Points[38]	According to Cohn, "story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work…The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it and so on" [Cohn 2006].

[38] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Agile Terms: Continuous Integration

Definition[39]	**Is there an equivalent in the Traditional World?**	
Continuously integrating new development code into the existing codebase, which ensures that the code repository always reflects the latest working build of the software. Continuous integration helps identify and resolve issues more quickly than "end of build" integration.	No direct peer per se; however continuous integration as defined in Agile is practiced in some Traditional World projects. That being said, there is significant similarity between the Traditional World concepts of *automated verification system* and the *automated acceptance test*. In some traditional projects, unit and integration test cases are defined and built as part of the design phase in order to ensure the developers fully understand the requirements. However, in a strict interpretation the Traditional World's automated verification system isn't necessarily tied to acceptance, and does allow for the need for human intervention. Additionally, the Agile automated acceptance test is often incorporated into the daily work. In addition, some Agile practices call for the automated acceptance tests to be built prior to coding as a mechanism to ensure the developers fully understand the customer requirements.	
How is it used in Agile?	**Are there any related terms or concepts?**	
Continuous integration is most often used during daily work or at a sprint (or iteration) level.	Automated Acceptance Tests[40]	Tests written by the product owner which are run automatically against software and systems; they form part of the program specification.
	Test Driven Development (TDD) [41]	A technique where a test case is written before coding is started for a desired improvement or new function, code is written until it passes the test, the code is refactored to acceptable standards.
	Automated Verification System[42] (Traditional World definition)	A software tool that accepts as input a computer program and a representation of its specification and produces, possibly with human help, a proof or disproof of the correctness of the program
	Multi-Stage Continuous Integration[43]	Multi-stage continuous integration (CI); each team does CI on their branch such that if a problem occurs only that team is affected – not the entire project.

---

39 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

40 Terms and definition(s) adapted whole or in part from http://www.develop.com/agiledemystified

41 Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

42 Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

43 Term and definition(s) adapted whole or in part from http://www.accurev.com/wiki/agile-glosssary

Agile Terms: Done

<table>
<tr><th>Definition[44]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A story (or user story) is done when:
• All code is checked in
• All developer tests pass
• All acceptance tests pass
• Help text is written
• Product Owner accepted

A sprint (or Iteration) is done when:
• Product backup is complete
• Performance tested
• Defects fixed or postponed

A release is done when:
• Stress tested
• Performance tuned
• Security validation passes
• Disaster recovery plan tested
• Required documentation is complete</td><td colspan="2">There is no direct peer term per se in the Traditional World, though there is the peer concept of software being ready for acceptance testing or production.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="4">Done is used to express when work is complete at some level, which can vary.

Done is often defined uniquely to a team, and care must be taken to ensure all stakeholders share the same interpretation. In this manner it is very similar to confusion over the terms “complete” or “finished” in the Traditional World, where various stakeholders (developers, testers, users, etc.) frequently have different definitions of the terms.</td><td>Done Done</td><td>Done done means that all of the tasks needed to create the final, releasable product have been completed.</td></tr>
<tr><td>Shrink-wrapped (Traditional World definition)</td><td>Software that is ready to deliver; i.e., if it could be purchased at a local store as a consumer product it would be wrapped in cellophane or shrink wrapping.</td></tr>
<tr><td>Acceptance Criteria[45]</td><td>Those criteria by which a work item (user story) is judged successful or not; usually "all or nothing"—it is “done” or it is “not done.”</td></tr>
<tr><td>Acceptance Criteria[46] (Traditional World definition)</td><td>The criteria that a system or component must satisfy in order to be accepted by a user, customer, or other authorized entity.</td></tr>
</table>

[44] Terms and definition(s) adapted whole or in part from http://www.rallydev.com/help/definition-done

[45] Term and definition(s) adapted whole or in part from http://www.accurev.com/wiki/agile-glosssary

[46] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Agile Terms: Epic or Epic Stories

<table>
<tr><th>Definition</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A connected or bundled set of stories that result in a definable (in the case of software, desirable) capability or outcome. An epic is a large user story. It is possible to break up an epic into several user stories.[47]</td><td colspan="2">This is most similar to a system specification or top-level requirements (TLRs). Because Epics can represent an end-to-end capability, they are also similar to mission threads</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">Epics are most often used during planning at the vision and or roadmap level.</td><td>Story (or User Story) [48]</td><td>Often written on 3”x5” cards, a story (or user story) is a high-level requirement definition written in everyday or business language</td></tr>
<tr><td>Product Backlog[49]</td><td>The repository of requirements maintained by the product owner; typically high level requirements with high level estimates, and with the requirements in priority order.</td></tr>
<tr><td>Mission Threads (Traditional World definition)</td><td>A sequence of end-to-end activities and events beginning with an opportunity to detect a trigger of an event of interest and ending with an assessment of the effectiveness of any actions initiated in response to the event of interest.</td></tr>
</table>

[47] Term and definition(s) adapted whole or in part from http://www.targetprocess.com/LearnAgile/AgileGlossary/ThemeEpic.aspx

[48] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[49] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

Agile Terms: eXtreme Programming (XP)

<table>
<tr><th>Definition[50]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>XP is a discipline of software development based on values of communication, simplicity, feedback, courage, and respect.

XP consists of the following core practices; planning poker, on-site customer, small releases, metaphor, simple design, test-driven development, refactoring, pair programming, collective ownership, continuous integration, coding standards, and sustainable pace.</td><td colspan="2">There is not a peer process to XP in the Traditional World; in fact there are Agile proponents that feel XP is the antithesis of the Traditional World.

In extreme programming, every contributor to the project is an integral part of the team, and the team forms around a business representative called “the customer,” who sits with the team and works with them daily.[51]</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">XP is one of the major forms of Agile, and can be used during each of the five levels of Agile planning:
Vision
Roadmap
Release
Sprint (or Iteration)
Daily Work</td><td>Planning Poker[52]</td><td>A consensus-based technique used to estimate how long a certain amount of work will take to complete.</td></tr>
<tr><td>Refactoring[53]</td><td>Modifying/revising code to improve performance, efficiency, readability, or simplicity without affecting functionality; generally considered part of the normal development process and it improves longevity, adaptability, and maintainability over time.</td></tr>
<tr><td>Pair Programming[54]</td><td>Two developers (sometimes referred to as the “driver” for the person actually coding and the “observer”) working side-by-side to create a single feature; it provides real-time code review, allows one developer to think ahead while the other thinks about the work at hand, and supports cross-training.

The concept can also extend to pair designing and pair unit testing. It provides real time peer reviews.</td></tr>
</table>

[50] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com and from http://xprogramming.com/book/whatisxp/

[51] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com and from http://xprogramming.com/book/whatisxp/

[52] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[53] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[54] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Agile Terms: Feature

<table>
<tr><th>Definition[55]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A customer-understandable, customer-valued piece of functionality that serves as a building block for prioritization, planning, estimation, and reporting.</td><td colspan="2">While the DAU definition of a feature is much simpler (a distinguishing system characteristic), the underlying meaning is essentially the same in both the Traditional World and the Agile World.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="4">The concept of a feature is used during each of the five levels of Agile planning:
Vision
Roadmap
Release
Sprint (or Iteration)
Daily Work</td><td>Minimally-Marketable Feature (MMF) [56]</td><td>A smallest element of a marketable or operationally useful feature; it is marketable, or operationally useful because when it is released as part of a product, users would use (or buy) the feature.</td></tr>
<tr><td>Attribute[57] (Traditional World definition)</td><td>A property associated with a set of real or abstract things that is some characteristic of interest.</td></tr>
<tr><td>Feature-Based Planning[58]</td><td>An approach where features and scope take priority over date; plans are created by estimating the amount of time needed to build a set of features or a defined amount of scope.</td></tr>
<tr><td>Feature-Driven Development (FDD)[59]</td><td>FDD utilizes an incremental, model-driven approach based on five key activities:
Define the overall model
Build the feature list
Plan by feature
Design by feature
Develop by feature</td></tr>
</table>

---

55 Term and definition(s) adapted whole or in part from *Agile to waterfall Dictionary; Mike Griffiths;* http://www.pmhut.com/agile-to-waterfall-dictionary

56 Term and definition(s) adapted whole or in part from http://www.agilebok.org/index.php?title=Minimally_Marketable_Feature_%28MMF%29

57 Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

58 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

59 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Agile Terms: Five Levels of Agile Planning

<table>
<tr><th>Definition[60]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>The five levels of Agile planning are:
Vision - The highest level in agile planning, the vision is strategic in nature and is infrequently changed
Roadmap - The roadmap distills the vision into a high level plan that outlines work spanning one or more releases; requirements are grouped into prioritized themes, each with an execution estimate.
Release – A release is a planning segment of prioritized requirements, along with execution estimates
Sprint (or Iteration) – An iteration is a predefined, time-boxed and recurring period of time in which working software is created.
Daily Work – a brief, daily communication and planning forum where the development team and other stakeholders evaluate the health and progress of the iteration/sprint.</td><td colspan="2">The Traditional World also has different levels of planning (PMBOK's planning process, for example) but does not structure them in the same framework. Visions, roadmaps, and release plans are essentially the same as in the Traditional World; however their representations and scope may vary from an Agile approach.

The main distinction between the two approaches is that Agile planning is based on the assumption that change is inevitable and must be accommodated, while traditional methods are based on the assumption that deviations from a plan are problems that must be actively avoided.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>The five levels are used throughout the life cycle.</td><td>Planning Processes[61]
(Traditional World definition)</td><td>Those processes performed to define and mature the project scope, develop the project management plan, and identify and schedule the project activities that occur within the project.</td></tr>
</table>

[60] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[61] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

Agile Terms: INVEST

<table>
<tr><th>Definition[62]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>From eXtreme Programing Explored, INVEST is an acronym for a set of rules to create a story (or user story)
• Independent
• Negotiable
• Valuable
• Estimable
• Small
• Testable</td><td colspan="2">There is no direct peer for this acronym in the Traditional World; however in the Traditional World requirements are always seen as needing to be:
• Necessary
• Prioritized
• Unambiguous
• Verifiable
• Complete
• Consistent
• Traceable</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>As stated in the definition, INVEST is used during the planning stages.

The INVEST concept can be used during each of the five levels of Agile planning:
• Vision
• Roadmap
• Release
• Sprint (or Iteration)
• Daily Work</td><td>Story (or user story)[63]</td><td>Often written on 3”x5” cards, a story (or user story) is a high-level requirement definition written in everyday or business language.</td></tr>
</table>

[62] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[63] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

Agile Terms: Pair Programming

<table>
<tr><th>Definition[64]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>Two developers (sometimes referred to as the “driver” for the person actually coding and the “observer”) working side-by-side to create a single feature; it provides real-time code review, allows one developer to think ahead while the other thinks about the work at hand, and supports cross-training.

The concept can also extend to pair designing and pair unit testing. It provides real time peer reviews.</td><td colspan="2">There is not a peer in the Traditional World; however the concept of peer inspections captures an equivalent quality mechanism to that of paired programming.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>Used during product development (daily work). While reviewing, the observer also considers the strategic direction of the work, coming up with ideas for improvements and likely future problems to address.

This frees the driver to focus all of his/her attention on the "tactical" aspects of completing the current task, using the observer as a safety net and guide.[65]</td><td>eXtreme Programming [66]</td><td>XP is a discipline of software development based on values of communication, simplicity, feedback, courage, and respect.

XP consists of the following core practices; planning poker, on-site customer, small releases, metaphor, simple design, test-driven development, refactoring, pair programming, collective ownership, continuous integration, coding standards, and sustainable pace.</td></tr>
</table>

64 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

65 Term and definition(s) adapted whole or in part from http://en.wikipedia.org/wiki/Pair_programming

66 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com and from http://xprogramming.com/book/whatisxp/

Agile Terms: Planning Poker

<table>
<tr><th>Definition[67]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A consensus-based estimating technique using cards marked with one number from a modified Fibonacci sequence (0,1/2, 1, 2, 3, 5, 8, 13, 20, 40, 100, and optionally ? and ∞); the rules are:
The team selects a requirement as the baseline; this can have any value but is notionally assigned a value of “2.”
The team selects a new requirement, and team members discuss and clarify assumptions and risks.
Team members select a card whose value they feel reflects the complexity or risk of the new requirement as compared to the baseline (i.e., a “1” is half as difficult and a “20” is an order of magnitude more difficult).
The team members reveal their cards simultaneously by turning them over.
The people with the high and/or low estimates justify their selection.
Each person then selects a card again, and the process repeats until there is a consensus.
The process repeats until all the requirements have been scored.</td><td colspan="2">Similar to wide-band delphi (described below)</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>Planning poker is used during release and sprint (or iteration) planning.</td><td>Wide-Band Delphi[68] (Traditional World definition)</td><td>A group estimation technique when there are a many unknowns; steps include:
Describe what is being estimated.
Ask individuals to privately make their own estimates using their best judgment.
Present the estimates and discuss, usually begun by asking the high/low estimates to explain their thinking.
Repeat these steps (with anonymity dropped) until the estimates converge.</td></tr>
</table>

[67] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[68] Terms and definition(s) adapted whole or in part from http://leansoftwareengineering.com/wideband-delphi

Agile Terms: Product Owner

<table>
<tr><th>Definition[69,70]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>The “voice of the customer,” accountable for ensuring business value is delivered by creating customer-centric items (typically stories (or user stories), prioritizing them, and maintaining them in the product backlog.

In scrum, the product owner is the sole person responsible for managing the product backlog; they:
• Define the product backlog items
• Prioritize the product backlog to reflect goals and missions
• Keep the product backlog visible to all
• Define the sprint backlog items
• Ensure the developers fully understand the backlog items

However, the product owner does not specifically have to be one person; this role could be carried out by one person or a group could own it. The main point is that that there is “one voice” for the customer.</td><td colspan="2">Similar in concept to the Traditional World’s voice of the customer, but with the significant difference that in agile the product owner is much more involved in the software development daily operations as well as release and sprint planning, prioritization, etc. and thus has more influence on the developer’s decisions making process.

In the Traditional World, the actual implementation of the product owner role is shared among many different stakeholders such as the end user, systems engineer, product architect, solution analysis, program manager, etc.

In theory, sharing this role among so many stakeholders should ensure that at least one of the product owner(s) is always available to support the software development activities. However, that is frequently not the case because the role is not explicit.

Also, with the role split among many different stakeholders there are sometimes conflicting perspectives which can hamper or delay activities.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>The product owner is involved in all stages of an Agile project.

In fact the demands placed on the product owner (time commitment, knowledge of the domain, and knowledge of Agile development methods) are one of the key stumbling blocks in the Traditional World’s Agile adoption.</td><td>Voice of the Customer (Traditional World definition)</td><td>A survey technique to capture a detailed set of customer needs and desires using the customer’s language, translating these into technical requirements, and putting them in priority order.

HOWEVER—the Traditional voice of the customer is a survey technique, where the Agile product owner is an active role and responsibility.</td></tr>
</table>

[69] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[70] Terms and definition(s) adapted whole or in part from *The Scrum Guide – The Definitive Guide to Scrum: Rules of the Game; Schwaber and Sutherland, scruminc*

Agile Terms: Refactoring

<table>
<tr><th>Definition[71]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>Modifying/revising code in to improve performance, efficiency, readability, or simplicity without affecting functionality; generally considered part of the normal development process and it improves longevity, adaptability, and maintainability over time.</td><td colspan="2">There is not a direct peer for this in the Traditional World, although refactoring can be argued to be a general software development approach that has simply been most popularized by Agile.

There are also similarities with elements of the Traditional World's software redesign or reengineering. These processes result in the design and implementation of a new overall structure without changing its external behavior, and share the refactoring goal of correcting deficiencies in the software design and supporting future enhancements.

However, redesign and reengineering can also include adopting a new programming paradigm (such as a transition from unstructured to structured programming or to object-oriented programming), which is well beyond the normal use of Agile refactoring.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Refactoring was popularized as a practice in extreme programming, but was used prior to XP and is now used by most Agile methods as a normal part of the development process</td><td>eXtreme Programming [72]</td><td>XP is a discipline of software development based on values of communication, simplicity, feedback, courage, and respect.

XP consists of the following core practices: planning poker, on-site customer, small releases, metaphor, simple design, test-driven development, refactoring, pair programming, collective ownership, continuous integration, coding standards, and sustainable pace.</td></tr>
<tr><td>Simple Design[73]</td><td>To paraphrase the poet Wallace Stevens, simple design is "the art of what suffices."

Simple design means coding for today's specified requirements, and no more.</td></tr>
</table>

[71] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[72] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com and from http://xprogramming.com/book/whatisxp/

[73] Term and definition(s) adapted whole or in part http://www.versionone.com/Agile101/Simple_Design.asp

Agile Terms: Release

Definition[74]	**Is there an equivalent in the Traditional World?**
The third of the five levels of Agile planning: • Vision - The highest level in agile planning, the vision is strategic in nature and is infrequently changed. • Roadmap - The roadmap distills the vision into a high level plan that outlines work spanning one or more releases; requirements are grouped into prioritized themes, each with an execution estimate. • Release – A release is a planning segment of prioritized requirements, along with execution estimates. • Sprint (or iteration) – An iteration is a predefined, time-boxed and recurring period of time in which working software is created. • Daily Work – a brief, daily communication and planning forum where the development team and other stakeholders evaluate the health and progress of the iteration/sprint.	This is another term which can cause confusion. In Agile, a release is part of the planning process. However, at the end of each release the development contractor will deliver some element of a militarily useful capability to the government. (This does not mean that the capability is fielded at this point, but it does mean that it could be fielded.) What this means is that a release in Agile often does result in a deliverable—which in the Traditional World is often called a "release" (see the definition below).

How is it used in Agile?	**Are there any related terms or concepts?**	
Planning	Release[75] (*Traditional World definition*)	A delivered version of an application which may include all or part of an application.

[74] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[75] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Agile Terms: Retrospective

<table>
<tr><th>Definition[76]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A team meeting at the end of every iteration to review lessons learned and to discuss how the team can be more efficient in the future.</td><td colspan="2">The closest peer in the Traditional World is a hot wash (see below), but in Agile retrospectives are much more integrated into the project's rhythm.

The Traditional World also performs project post-mortems which share many of the same goals but are usually only performed once at the end of a project so the lessons learned can only improve future projects. Retrospectives, on the other hand, are designed to improve the current project as well as future projects.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>The retrospective is an integral part of Agile planning and process/product improvement; the most common examples are sprint (or iteration) retrospectives, and release retrospectives.</td><td>Hot Wash[77] (Traditional World definition)</td><td>Discussions and performance evaluations following an exercise, training session, or event to identify strengths, weaknesses and lessons learned; normally includes all the parties that participated in the exercise or event.</td></tr>
</table>

[76] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[77] Terms and definition(s) adapted whole or in part from http://www.develop.com/agiledemystified

Agile Terms: Roadmap

Definition[78]	**Is there an equivalent in the Traditional World?**
The second of the five levels of Agile planning: • Vision - The highest level in agile planning, the vision is strategic in nature and is infrequently changed. • Roadmap - The roadmap distills the vision into a high level plan that outlines work spanning one or more releases; requirements are grouped into prioritized themes, each with an execution estimate. • Release – A release is a planning segment of prioritized requirements, along with execution estimates. • Sprint (or iteration) – An iteration is a predefined, time-boxed and recurring period of time in which working software is created. • Daily Work – a brief, daily communication and planning forum where the development team and other stakeholders evaluate the health and progress of the iteration/sprint.	While elements of the roadmap could also comprise parts of the Traditional World's *project charter*, it is most like the Traditional World's *acquisition program baseline* (see below). It is also similar to the Traditional World's *integrated master plan* (IMP) or *master phasing schedule*.

How is it used in Agile?	**Are there any related terms or concepts?**	
Planning	Acquisition Program Baseline (APB) [79] (*Traditional World definition*)	Baseline reflecting threshold and objective values for the cost, schedule, and performance attributes that describe the program over its life cycle: • Life cycle cost estimate (LCCE) • Schedule dates including key activities such as milestones and the initial operational capability (IOC) • Performance attributes reflecting the operational performance required for the fielded system

[78] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[79] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Agile Terms: Scrum

<table>
<tr><th>Definition[80]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A process framework of team members and their associated roles (product owners, Scrum masters and team members such as developers, testers, etc.) who collaboratively define product and sprint backlogs that are executed in short, time-boxed sprints (or iterations).

At the end of each sprint, a working increment of the software is delivered or demonstrated to the product owner, and the entire process repeats itself.</td><td colspan="2">There isn't a direct peer in the Traditional World. There are many examples in the Traditional World that contain the iterative and incremental aspects of Scrum (particularly spiral development or the IBM Rational Unified Process® or RUP ®), but they lack key elements of a Scrum:
Daily close collaboration on a working level
Short iterations
Planning cycles are done at the start of each new body of work vs. all planning done up front before any body of work begins
Some elements of Scrum are also in the Traditional World such as design reviews and technical interchange meetings (TIMs), but these are usually on a "grander scale" than Scrum demonstrations.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">Management, planning</td><td>Scrum Master[81]</td><td>The Scrum master is not the development team leader per se – they buffer the team from distracting influences while they ensure the Scrum process is used as intended. The Scrum master also removes roadblocks, handles the paperwork, and generates the burn-down chart (metrics).</td></tr>
<tr><td>Scrum of Scrums[82]
(only used on larger teams)</td><td>A daily meeting that occurs after individual team's daily stand up, it allows multiple Scrum teams to stay synchronized and understand the flow and challenges of the other teams.

Each Scrum master addresses these questions:
What did my team complete?
What is my team working on next?
What barriers/issues are my team facing?</td></tr>
<tr><td>ScrumBut[83]</td><td>A project that claims to follow Scrum but doesn't:
"We do Scrum—but we don't have a product owner"
"We do Scrum—but the project manager allocates tasks."</td></tr>
</table>

[80] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[81] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[82] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[83] Terms and definition(s) adapted whole or in part from http://www.develop.com/agiledemystified

Agile Terms: Sprint (or Iteration)

<table>
<tr><th>Definition[84]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>A predefined, time-boxed, and recurring block of time in which working software is created— most commonly two, four, or six weeks long.

“Sprint” and “iteration” are effectively interchangeable, with sprint used by teams implementing Scrum.</td><td colspan="2">*Activity* or *work package* (see below) may be the closest peer in the Traditional World as the concept of a basic schedule building block is used in both Agile and the Traditional World.

However, the principal difference is that in Agile, the planning is taken “to the edge”—it is done by the people most affected by the planning, who are best suited to make realistic choices, and who have a vested interest in ensuring the work can be accomplished.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="4">Sprints (or iterations) are more than just the basic building blocks of Agile scheduling and product development; they represent a deliberate approach to planning and executing work in manageable “chunks” with known goals, known resources, and with the scope adjusted to fit the known schedule.</td><td>Sprint (or Iteration) Backlog[85]</td><td>The subset of stories (or user stories) and/or features from the product backlog planned to be completed in a specific sprint (or iteration). It reflects the priority and order of the release plan and product roadmap.</td></tr>
<tr><td>Sprint (or Iteration) Plan[86]</td><td>The detailed execution plan for a given (usually current) sprint (or Iteration); it defines the iteration goals and commitments by specifying the user stories, work tasks, priorities and team member work assignments required to complete the iteration. The sprint plan is normally produced by the entire development</td></tr>
<tr><td>Work Package[87] (*Traditional World definition*)</td><td>A work package is a deliverable or component at the lowest level of the work breakdown structure; it is a task, activity or grouping of work at the lowest level where work is planned, progress is measured, and earned value is computed.</td></tr>
<tr><td>Activity[88] (*Traditional World definition*)</td><td>A task or measurable amount of work to complete a job or part of a project.</td></tr>
</table>

84 Term and definition(s) adapted whole or in part from *Agile to Waterfall Dictionary; Mike Griffiths;* http://www.pmhut.com/agile-to-waterfall-dictionary

85 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

86 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

87 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

88 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Agile Terms: Story (or User Story)

<table>
<tr><th>Definition[89]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>Often written on 3”x 5” cards, a story (or user story) is a high-level requirement definition written in everyday or business language; it is a communication tool written by or for the customers to guide developers though it can also be written by developers to express non-functional requirements (security, performance, quality, etc.).

Stories (or user stories) are not vehicles to capture complex system requirements on their own. Rather, full system requirements consist of the body of stories (or user stories).

An epic is a large story (or user story) that will eventually be broken down into smaller stories (or user stories) that will be captured in the product backlog.</td><td colspan="2">Stories (or user stories) are most similar to requirements in the Traditional World.

Stories (or user stories) in Agile describe the same thing as requirements in the Traditional World—they capture what the system must do.

When combined together in the Agile product backlog, the stories (or user stories) form a comprehensive set of what the system “must do.”</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Stories (or User Stories) are used in all levels of Agile planning and execution:
Vision
Roadmap
Release
Sprint (or Iteration)
Daily Work</td><td>Requirement[90] (Traditional World definition)</td><td>A condition or capability needed by a user to solve a problem or achieve an objective.</td></tr>
<tr><td>Story Points[91]</td><td>According to Cohn, “story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work…The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it and so on” [Cohn 2006].</td></tr>
</table>

89 Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

90 Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

91 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Agile Terms: Story Board

Definition	Is there an equivalent in the Traditional World?
A wall chart (or digital equivalent) with markers (cards, sticky notes, etc.) for each task in a sprint or iteration; the board is divided into "to do", "in progress", "done", etc. and the movement of the markers across the board indicates progress. One goal of the story board is also to recognize the order and the dependencies of the stories in representing end-to-end functionality for the users. Similar to story maps[92], which are a visual technique to prioritize stories (or user stories) by creating a "map" of users, their activities, and the stories (or user stories) needed to implement the functionality needed	There are several similar concepts in the Traditional World. For example, some Traditional World *war rooms* have had the same goal as an Agile story board. The Traditional World's *project management dashboard* is also similar in concept as it is a simple-to-read graphical presentation of the current status and in some cases historical trends of performance; one prominent example is the federal government's IT dashboard (http://www.itdashboard.gov/). Also, the IMS (*integrated master schedule*) allows for teams to identify tasks that have not started (to do), in progress (in progress), and completed (done). However, the story board is usually much more detail than what is in an IMS.

How is it used in Agile?	Are there any related terms or concepts?	
A Story Board could be used in all levels of Agile planning and execution: • Vision • Roadmap • Release • Sprint (or Iteration) • Daily Work	Story Maps[93]	A visual technique to prioritize stories (or user stories) by creating a "map" of users, their activities, and the stories (or user stories) needed to implement the functionality needed.
	War Room[94] (*Traditional World definition*)	A room used for project conferences and planning, often displaying charts of cost, schedule status, and other key project data.
	Information Radiator[95] (Or Task Board)	A display posted in a public place showing readers relevant information; it should be: • easily visible to casual but interested readers • able to be understood at a glance • kept current

92 Term and definition(s) adapted whole or in part from http://www.agilelearninglabs.com/modules/story-mapping/

93 Term and definition(s) adapted whole or in part from http://www.agilelearninglabs.com/modules/story-mapping/

94 Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

95 Term and definition(s) adapted whole or in part from http://alistair.cockburn.us/Information+radiator

Agile Terms: Technical Debt

<table>
<tr><th>Definition[96]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>Technical debt includes those internal things (such as architectural elements, strategic development tasks such as common methods, etc.) that you choose not to do now, but which will impede future development if left undone. This includes deferred refactoring.

Technical debt doesn't include deferred functionality, except possibly in edge cases where delivered functionality is "good enough" for the customer, but doesn't satisfy some standard (e.g., a UI element that isn't fully compliant with some UI standard)</td><td colspan="2">As we write this in 2012, technical debt is principally used in Agile, and is not frequently used in the Traditional World.

However, the concept has been understood in the Traditional World for many years. In 1980 Manny Lehman wrote The Law of Increasing Complexity [Lehman 1980]:

"As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it."

The technical debt metaphor was coined by Ward Cunningham in a 1992 Object-Oriented Programming, Systems, Languages and Applications (OOPSLA) experience report:[97]

“Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt. Entire engineering organizations can be brought to a standstill under the debt load of an unconsolidated implementation, object-oriented or otherwise.”</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>The concept of technical debt is most applicable to Agile planning and execution at the release and sprint (or iteration) level.</td><td>Refactoring[98]</td><td>Modifying/revising code in to improve performance, efficiency, readability, or simplicity without affecting functionality; generally considered part of the normal development process and it improves longevity, adaptability, and maintainability over time.</td></tr>
</table>

[96] Terms and definition(s) adapted whole or in part from http://c2.com/cgi/wiki?TechnicalDebt

[97] http://c2.com/doc/oopsla92.html

[98] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Agile Terms: Velocity

<table>
<tr><th>Definition[99]</th><th colspan="2">Is there an equivalent in the Traditional World?</th></tr>
<tr><td>The rate at which work is completed, normally measured by the number of story points completed within an iteration; it is a predictive metric used for planning.</td><td colspan="2">This term is not used in the Traditional World, though the concept of “how much work will be accomplished in how much time” is fundamental in the Traditional World as it is in Agile. In both worlds, classes of work are defined, measures to the effort needed to complete that work are derived, and how much work was accomplished is measured.

There are similarities between velocity and aspects of the Traditional World’s earned value system, but as with all discussions of software productivity there are many issues regarding what to measure, how to measure, who should measure, etc. Velocity also uses story points which have an inherent level of uncertainty.

Velocity also has similarities with the Traditional World’s schedule performance index (SPI), which is a historical measure of how much work the team expected to complete and how much work was actually completed. However, velocity is not “guessed” ahead of time, but is refined as sprints are completed and becomes a predictor.</td></tr>
<tr><th>How is it used in Agile?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">Velocity is most applicable to Agile planning and execution at the release, sprint (or iteration), and daily work level.</td><td>Cadence</td><td>Cadence is an efficient and sustainable working rhythm, incorporating elements such as the daily work and stand-up meetings, weekly planning sessions and reviews, regular demonstrations and retrospectives, etc.</td></tr>
<tr><td>Sustainable Pace[100]</td><td>Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.</td></tr>
<tr><td>Story Points[101]</td><td>According to Cohn, “story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work…The number of story points associated with a story represents the overall size of the story. There is no set formula for defining the size of a story. Rather a story-point estimate is an amalgamation of the amount of effort involved in developing the feature, the complexity of developing it, the risk inherent in it and so on” [Cohn 2006].</td></tr>
</table>

99 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

100 For the principles behind the Agile Manifesto, see http://agilemanifesto.org/principles.html

101 Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Agile Terms: Vision

Definition[102]	**Is there an equivalent in the Traditional World?**	
The first of the five levels of Agile planning: • Vision - The highest level in agile planning, the vision is strategic in nature and is infrequently changed • Roadmap - The roadmap distills the vision into a high level plan that outlines work spanning one or more releases; requirements are grouped into prioritized themes, each with an execution estimate. • Release – A release is a planning segment of prioritized requirements, along with execution estimates • Sprint (or Iteration) – An iteration is a predefined, time-boxed and recurring period of time in which working software is created. • Daily Work – a brief, daily communication and planning forum where the development team and other stakeholders evaluate the health and progress of the iteration/sprint.	This term is also used in Traditional World, though sometimes called the goal. The vision would normally be captured in the Traditional World's project charter.	
How is it used in Agile?	**Are there any related terms or concepts?**	
Planning	Charter[103] (*Traditional World definition*)	Provides authority to conduct the program (within cost, schedule, and performance constraints); assigns personnel and resources; defines the PM's line of authority and reporting channels.

[102] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[103] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Ball Park Estimate

Definition[104]	**Is there an equivalent in the Agile World?**	
Rough estimate made with some knowledge and confidence that the estimated figure falls within a reasonable range of values. (i.e., this estimate is at least somewhere in the ball park …"). This is based upon expert judgment gained from experience. Also called rough order of magnitude.	Ball park estimates are normally done using *analogous estimating* (see below), which uses the experience from previous projects and extrapolates that onto the current project. In agile this technique is call relative estimation (see below). In Agile, initial estimates are done using relative estimation with tools such as *planning poker*, which could in some sense be thought of as a very thorough ball park estimate.	
How is it used in the Traditional World?	**Are there any related terms or concepts?**	
Initial budgeting and planning	Relative Estimation[105] *(Agile World definition)*	A technique to assess size and complexity by comparing the work under consideration to the size and complexity of other known requirements and work items.
	Cost Estimate[106]	A judgment or opinion regarding expected costs reached using an estimating process; it may constitute a single value or a range of values.
	Analogous Estimating[107]	An estimating technique that uses the values of parameters (such as scope, cost, budget, and duration) or measures of scale (such as size, weight, and complexity) from a previous, similar activity as the basis for estimating the same parameter or measure for a future activity.
	Parametric Estimating[108]	An estimating technique that uses a statistical relationship between historical data and other variables (e.g., lines of code) to calculate an estimate for activity parameters, such as scope, cost, budget, and duration.

[104] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[105] Term and definition(s) adapted whole or in part from http://www.agilebok.org/index.php?title=Relative_Prioritization_or_Ranking

[106] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[107] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[108] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

Traditional Terms: Bar Chart (or Gantt Chart)

<table>
<tr><th>Definition[109]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A graphic display of schedule-related information, normally with activities listed down the left side of the chart, dates across the top, and activity durations are shown as horizontal bars.</td><td colspan="2">Traditional World bar or Gantt charts do not have a direct peer in Agile, but the concept of graphically showing work assignments or other activities or efforts across a calendar is found in both worlds.

Perhaps the closest example of a Gantt chart in Agile is an epic board. The epic board is similar to a story or task board, but sits one level higher, i.e., at the project/program/portfolio level.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Bar charts are used throughout the life cycle, but primarily in pre-systems acquisition (material solution analysis and technology development) and systems acquisition (engineering and manufacturing development and production and deployment)</td><td>Burn -Up Chart (or Graph) [110] (Agile World definition)</td><td>A visual tool displaying progress via a simple line chart representing work accomplished (vertical axis) over time (horizontal axis)

Burn-up charts can be used at both a sprint (or iteration) and release level.</td></tr>
<tr><td>Burn-Down Chart (or Graph) [111] (Agile World definition)</td><td>A visual tool displaying progress via a simple line chart representing remaining work (vertical axis) over time (horizontal axis).

Burn-down charts can be used at both a sprint (or iteration) and release level.</td></tr>
</table>

[109] Terms and definition(s) adapted whole or in part from PMI Glossary Definitions http://www.timetrackingsoftware.com/help/dovtime10/pmi_glossary.htm

[110] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[111] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

Traditional Terms: Critical Path

Definition[112]	**Is there an equivalent in the Agile World?**	
The critical path is the "path" through a project with the shortest schedule and where the activities have the least flexibility or float. As such, any delay to an activity on the critical path will likely delay the project. The Critical Path Method is a technique that aids understanding of the dependency of events in a project and the time required to complete them. Activities that, when delayed, have an impact on the total project schedule are critical and said to be on the critical path.	There isn't a direct peer in Agile due to the different planning approaches. However, the Agile *roadmap* is the vehicle to coordinate dependencies across releases, and the Agile *release plan* is the vehicle to coordinate dependencies across sprints or iterations. Also, when working with a multi-team project, *epic boards* can be used to coordinate story dependencies and ensure iteration alignment of features.	
How is it used in the Traditional World?	**Are there any related terms or concepts?**	
Planning	Backward Pass[113]	Calculating late finish dates and late start dates for uncompleted activities by working backwards through the schedule network logic from the project's end date
	Forward Pass[114]	Calculating early start and early finish dates for uncompleted activities.
	Network Logic[115]	The collection of schedule activity dependencies that makes up a project schedule network diagram.
	Project Schedule Network Diagram[116]	A schematic display of the logical relationships among the project schedule activities, and always drawn from left to right to reflect project work chronology.

[112] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[113] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[114] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[115] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[116] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

Traditional Terms: Derived Requirements

Definition[117]	**Is there an equivalent in the Agile World?**	
A lower-level requirement that is determined to be necessary for a top-level requirement to be met.	This term means essentially the same in both the Traditional World and Agile. In Agile, this is often referred to as *splitting user stories*, and these are uncovered during many stages in the development, and are fed into the backlog for active consideration for the next release or iteration/sprint. In the Traditional World, however, they must (in theory) all be discovered and articulated at project start as part of the requirements refinement and architectural design processes.	
How is it used in the Traditional World?	**Are there any related terms or concepts?**	
During the user needs phase, and possibly into the technology opportunities and resources phase.	Requirement[118]	A condition or capability needed by a user to solve a problem or achieve an objective.
	Quality Attribute[119]	A requirement that specifies the degree of an attribute that affects the quality that the system or software must possess, such as performance, modifiability, usability.
	Story (or User Story) [120] *(Agile World definition)*	Often written on 3" x 5" cards, a story (or user story) is a high-level requirement definition written in everyday or business language.

[117] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[118] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[119] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[120] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

Traditional Terms: Earned Value Management (EVM)

<table>
<tr><th>Definition[121]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A method combining scope, schedule, and resource data into a measure of performance and progress by comparing what was budgeted for a task (time and resources) against what the task actually required (time and resources).</td><td colspan="2">Typically, it is difficult to use the Traditional waterfall-related concepts of earned value management (EVM) in Agile. However, Rawsthorne[122] proposes how a functional work breakdown structure (WBS) can provide a structure for business metrics (business value, earned business value), which when combined with burn-down charts can provide a good, composite understanding of the progress of a project.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="5">EVM can be used throughout the life cycle, but is primarily used during
• Pre-Systems Acquisition (Material Solution Analysis and Technology Development)
• Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
• Sustainment (Operations and Support)</td><td>Actual Cost of Work Performed (ACWP)[123]</td><td>The costs actually incurred and recorded in accomplishing the work performed within a given time period.</td></tr>
<tr><td>Budget at Completion (BAC) [124]</td><td>The sum of all the budgets for the work to be performed on a project; can also be done for a work breakdown structure component or for a schedule activity.</td></tr>
<tr><td>Cost Performance Index (CPI) [125]</td><td>The ratio of earned value to actual costs.</td></tr>
<tr><td>Estimate at Completion (EAC) [126]</td><td>The expected total cost when the defined scope of work has been completed; most EAC forecasts adjust the original cost estimate based on actual performance to date.</td></tr>
<tr><td>Schedule Performance Index (SPI) [127]</td><td>A measure of schedule efficiency on a project; it is the ratio of earned value (EV) to planned value (PV).</td></tr>
</table>

---

121 Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

122 Discussion adapted from http://www.agilejournal.com/articles/columns/column-articles/54-calculating-earned-business-value-for-an-agile-project

123 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

124 Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

125 Terms and definition(s) adapted whole or in part from PMI Glossary Definitions http://www.timetrackingsoftware.com/help/dovtime10/pmi_glossary.htm

126 Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

127 Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

Traditional Terms: Entry Criteria/Exit Criteria

<table>
<tr><th>Definition[128]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>Entry Criteria - The state of being that must be present before an effort can begin successfully.

Exit Criteria - The state of being that must be present before an effort can end successfully.</td><td colspan="2">While the terms are not in as common use in Agile as they are in the Traditional World, the concept that there are criteria that must be met before an effort can start is a management tenant in both worlds.

For example, one entry criteria for a sprint or iteration is a ready product backlog, which is a backlog that is broken down into small pieces, which are clear to the developers, immediately actionable, estimated in points by the team that will implement it, and testable.

Exit Criteria are also related to the Agile concept of “done.”</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Used for planning gates and reviews.</td><td>Done
(Agile World definition)</td><td>Defined differently at different stages of a project, done means within the context where the term is understood and accepted—that everything needed to advance to the next stage (be that to the next day’s work, the next sprint (or iteration), or release is complete.</td></tr>
<tr><td>Done Done
(Agile World definition)</td><td>Done done means that all of the tasks needed to create the final, releasable product have been completed.</td></tr>
</table>

[128] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Traditional Terms: Function Point

<table>
<tr><th>Definition[129]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A unit of measure for functional size that looks at the logical view:
• EI - external inputs
• EO - external outputs
• EQ - external inquiries
• EIF - external interface files
• ILF - internal logical files

However, function points do not count things like coding algorithms or database structure.</td><td colspan="2">This concept has utility in both the Traditional World and Agile (though the use and application of function points is not universally or consistently used).</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>When used, Function Points are primarily used during :
• Pre-Systems Acquisition (Technology Development)
• Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
• Sustainment (Operations and Support)</td><td>Complexity Points
(Agile World definition)</td><td>Complexity Points are units of measure used to estimate development work in terms of complexity, but not effort—effort is measured by story points.</td></tr>
</table>

[129] Terms and definition(s) adapted whole or in part from http://www.functionpoints.org

Traditional Terms: Increment

<table>
<tr><th>Definition[130]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A militarily useful capability that can be developed, produced, acquired, deployed and sustained; increments each have their own user-defined threshold and objective values.</td><td colspan="2">While the underlying concept is similar (a useful capability or an added value), a Traditional World increment is normally much larger than an Agile increment.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td>The concept of increments can be used throughout the life cycle but are most used during:
Pre-Systems Acquisition (Material Solution Analysis and Technology Development)
Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
Sustainment (Operations and Support)</td><td>Increment[131] (Agile World definition)</td><td>Agile software projects deliver the system in increments, which represent the value added to the system such as newly implemented features, removed defects, or an improved user experience.</td></tr>
</table>

[130] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[131] Terms and definition(s) adapted whole or in part from http://my.safaribooksonline.com/book/software-engineering-and-development/agile-development/9780735625679/return-on-investment/increment

Traditional Terms: Inspection

<table>
<tr><th>Definition[132]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>Visual examination of an item and associated documentation comparing it to predetermined standards to determine conformance; does not require the use of special laboratory equipment or procedures.</td><td colspan="2">This term means essentially the same (for a product) in both the Traditional World as in Agile. In the Agile method Scrum, however, it also relates to the Scrum process itself.

In Agile, a product being “ready for inspection” is more subjective because of the iterative nature of the development and the fact that products are continuously evolving. Therefore a product should be considered ready for inspection when it is "finished" for the time being (as in a sprint demonstration, for example). Note that finished does not necessarily mean “done” as more work on that product may be planned, but it does mean that it is in a stable state.[133]

Some Agile teams add an additional column or phase to their story board for team inspections or peer reviews, requiring that all code be inspected before it can be declared done.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">Inspections are most used during:
• Pre-Systems Acquisition (Material Solution Analysis and Technology Development)
• Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
• Sustainment (Operations and Support)</td><td>Inspection[134] (Agile World definition)</td><td>Assessment to determine if a process has deviated outside acceptable limits; four formal opportunities in Scrum:
• Sprint Planning meeting
• Daily Scrum
• Sprint review
• Sprint Retrospective</td></tr>
<tr><td>Peer Review[135]</td><td>A review of work products performed by peers during development of the work products to identify defects for removal.</td></tr>
<tr><td>Structured Walkthrough[136]</td><td>A systematic examination of the requirements, design, or implementation of a system, or any part of it, by qualified personnel.</td></tr>
</table>

[132] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[133] Terms and definition(s) adapted whole or in part from DSDM Public Version 4.2

[134] Terms and definition(s) adapted whole or in part from *The Scrum Guide – The Definitive Guide to Scrum: Rules of the Game; Schwaber and Sutherland, scruminc*

[135] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[136] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Traditional Terms: Integrated Master Plan/Integrated Master Schedule

<table>
<tr><th>Definition[137]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>Integrated Master Plan (IMP) - An event-driven plan capturing the major accomplishments necessary to complete a body of work that ties each accomplishment to a key program event.

Integrated Master Schedule (IMS) - An integrated schedule of the tasks needed to complete the work effort captured in the IMP; the IMS should include all IMP events and accomplishments.</td><td colspan="2">When taken in their total, Agile's five levels of planning (product vision, product roadmap, release plan(s), sprint (or iteration) plan(s), and daily commitment(s)) match or perhaps even exceed the information in a Traditional World IMP and IMS.

Also, an epic board can be used to visualize an integrated Agile plan.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">The IMP and IMS can be used throughout the life cycle but are most used during:
Pre-Systems Acquisition (Material Solution Analysis and Technology Development)
Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
Sustainment (Operations and Support)</td><td>Schedule Development[138]</td><td>The process of creating the project schedule by analyzing activity sequences, activity durations, resource requirements, and schedule constraints</td></tr>
<tr><td>Roadmap[139] (Agile World definition)</td><td>The roadmap distills the vision into a high level plan that outlines work spanning one or more releases; requirements are grouped into prioritized themes, each with an execution estimate.</td></tr>
</table>

[137] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[138] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[139] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Traditional Terms: Integrated Product Team (IPT)

<table>
<tr><th>Definition[140]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>Multi-disciplinary teams to identify, explore and resolve issues, and to provide recommendations to decision makers.
Working-level IPTs (WIPTs) focus on program issues and status, identify risks, and seek improvement opportunities.
Overarching IPTs (OIPTs) focus on strategic guidance, program assessment, and issue resolution.
Program-level IPTs (PIPTs) focus on program execution and may include both government and contractor representatives.</td><td colspan="2">The term means the same in Agile; however in Agile there is a significantly greater emphasis placed on the team with regard to planning; they have a much greater voice and are much more active.

Perhaps the best peer term in Agile is cross functional teams, which are groups of people who collectively represent the entire organization’s interests in a specific product or product family.[141]</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">IPTs are used in all aspects of Traditional waterfall.</td><td>Feature Teams[142] (Agile World definition)</td><td>Small, cross-functional teams focused on designing and building specific feature groupings.</td></tr>
<tr><td>Stakeholder[143]</td><td>An individual, group, or organization who may affect, be affected by, or perceive itself to be affected by a project’s activities, products, or services.</td></tr>
<tr><td>Red Team[144]</td><td>Red teams are groups of experts brought in by an enterprise to challenge plans, programs, assumptions, etc. as well as to play devil’s advocate and related roles.</td></tr>
</table>

[140] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[141] Adapted whole in in part from http://theagileproductmanager.blogspot.com/2008/07/whats-cross-functional-team-and-why.html

[142] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[143] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[144] Terms and definition(s) adapted whole or in part from *The Final Report of the Defense Science Board Task Force on the Role and Status of DoD Red Teaming Activities*

Traditional Terms: Key Performance Parameters (KPPs)

<table>
<tr><th>Definition[145]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A critical or essential system characteristic; normally has a threshold and an objective value.</td><td colspan="2">While there is no direct equivalent or peer for this term in the Agile World, this type of information is critical to both the Traditional World and the Agile World.

In Agile, this information is normally captured in Agile stories (or user stories), though the concept of technical user stories (those created by the development teams rather than those created by the product owner) is contentious within Agile.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">Planning and Testing</td><td>Requirements Analysis[146]</td><td>Definition and refinement of system, subsystem, and lower-level functional and performance requirements and interfaces to facilitate the architecture design process; establishes the detailed functional, interface, and temporal aspects of the system to unambiguously communicate system behavior in its intended environment, and the development of lower tier functional and performance requirements that need to be allocated to the system physical architecture.</td></tr>
<tr><td>Performance Specification[147]</td><td>A document that specifies the performance characteristics that a system or component must possess.</td></tr>
<tr><td>Quality Attribute[148]</td><td>A requirement that specifies the degree of an attribute that affects the quality that the system or software must possess, such as performance, modifiability, usability.</td></tr>
</table>

[145] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[146] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[147] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[148] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Traditional Terms: Lightweight Process

<table>
<tr><th>Definition[149]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A process with a single thread of control; a task.</td><td colspan="2">This is a heavily loaded term, as Agile development methods are frequently characterized as “lightweight” and traditional as “heavyweight.”

However, the underlying concept is valid in both the Traditional World and Agile.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Both the concept of lightweight and heavyweight processes are normally used during:
• Pre-Systems Acquisition (Technology Development)
• Systems Acquisition (Engineering and Manufacturing Development)</td><td>Process[150]</td><td>The combination of people, equipment, materials, methods, and environment that produces a given product or service.</td></tr>
<tr><td>Heavyweight Process[151]</td><td>A process with its own memory and multiple threads of control</td></tr>
</table>

[149] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[150] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[151] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Traditional Terms: Milestone A/Milestone B/Milestone C

<table>
<tr><th>Definition[152]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A scheduled event used to measure progress;
Milestone A precedes a program moving into Technology Development (TD)
Milestone B precedes a program moving into the Engineering and Manufacturing Development (EMD)
Milestone C precedes a program moving into Production and Deployment (P&D)</td><td colspan="2">This term means the same thing for a program in the Traditional World as well as a program using an Agile method. However, the data and documents normally required in the Traditional World normally exceed that which is produced in Agile (though Agile does produce the data and documents to address the primary intent of the milestones—is the program ready to proceed?)

However, when these concepts are used in an Agile World, care needs to be taken that the inherent conflicts are addressed so not to impact the program with unnecessary documentation while ensuring that essential documents are still included.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">Milestone decision points are fundamental to planning and management of programs</td><td>Technology Development Phase[153]</td><td>Designed to reduce technology risk and to determine the appropriate set of technologies to be integrated into the full system.</td></tr>
<tr><td>Engineering and Manufacturing Development Phase[154]</td><td>Consists of integrated system design (ISD) and system capability and manufacturing process demonstration (SC&MPD).</td></tr>
<tr><td>Production and Deployment Phase[155]</td><td>Designed to achieve an operational capability that satisfies the mission need, this phase consists of low-rate initial production (LRIP) and full-rate production and deployment (FRP&D) separated by a full-rate production decision review (FRPDR).</td></tr>
</table>

[152] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[153] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[154] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[155] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Oversight

Definition[156]	**Is there an equivalent in the Agile World?**	
Review activity by the Office of the Secretary of Defense (OSD), the Joint Staff (JS), DoD Components, and congressional committees of DoD programs to determine current status, ascertain if the law or other desires of Congress are being followed, or as a basis for possible future legislation.	Programs are subject to oversight whether they are in the Traditional World or in the Agile World. However, with the passage of such legislation as the NDAA of 2010 Section 804 and efforts such as the USN's IT Streamlining, the form of oversight will often vary depending on the whether the project is using traditional or Agile methods.	
How is it used in the Traditional World?	**Are there any related terms or concepts?**	
Oversight is fundamental to planning and management of programs.	Review[157]	A process or meeting during which a work product, or set of work products, is presented to project personnel, managers, users, customers, or other interested parties for comment or approval.
	Material Development Decision[158]	Formal entry point into the acquisition process that normally requires an initial capabilities document (ICD) and study guidance for the analysis of alternatives (AoA).
	Critical Design Review[159]	A multi-discipline technical review to ensure that a system can proceed into fabrication, demonstration, and test, and can meet stated performance requirements within cost, schedule, risk, and other system constraints.
	Full-Rate Production Decision Review[160]	A review conducted at the conclusion of low-rate initial production (LRIP) effort that authorizes entry into the full-rate production (FRP).

156 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

157 Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

158 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

159 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

160 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms*:* Peer Review

<table>
<tr><th>Definition[161]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A review of work products performed by peers during development of the work products to identify defects for removal.</td><td colspan="2">Paired programming is a specific type of peer review used in Agile (most consistently in eXtreme programming).

However, while the concept of peers participating in all phases of planning, design, execution, and review is fundamental to several Agile methods, peer reviews are generally confined to product review in the Traditional World.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Peer review is used during the development process</td><td>Pair Programming[162]
(Agile World definition)</td><td>Two developers (sometimes referred to as the “driver” for the person actually coding and the “observer”) working side-by-side to create a single feature; it provides real-time code review, allows one developer to think ahead while the other thinks about the work at hand, and supports cross-training.

The concept can also extend to pair designing and pair unit testing. It provides real time peer reviews.</td></tr>
<tr><td>Inspection[163]</td><td>Visual examination of an item and associated documentation comparing it to predetermined standards to determine conformance; does not require the use of special laboratory equipment or procedures.</td></tr>
</table>

[161] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[162] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[163] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Performance Measurement Baseline (PMB)

<table>
<tr><th>Definition[164]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>An integrated scope-schedule-cost plan for the project work against which project execution is compared to measure and manage performance.</td><td colspan="2">The closest concept for this is velocity, which like the performance measurement baseline is a relative measure of how much work is accomplished by the team. A release plan can also be viewed as a high-level PMB.

In Agile, the velocity calculated as the number of story points associated with stories (or user stories) that are finished by a team over a given period of time.
burn-down charts—which graphically show how much work remains or how much work has been “burned down” over time—and burn-up charts—which are the same except they show how much work has been accomplished over time— are also very similar.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">The performance measurement baseline is used throughout the life cycle but is most used during:
Pre-Systems Acquisition (Material Solution Analysis and Technology Development)
Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
Sustainment (Operations and Support)</td><td>Burn -Up Chart (or Graph) [165] (Agile World definition)</td><td>A visual tool displaying progress via a simple line chart representing work accomplished (vertical axis) over time (horizontal axis)

Burn-up charts are also normally used at a release level as well as the sprint (or Iteration) levels.

Agile burn-up charts are conceptually equivalent to the Traditional World’s Earned Value accumulated at a specific date [Cabri 2006].</td></tr>
<tr><td>Burn-Down Chart (or Graph) [166] (Agile World definition)</td><td>A visual tool displaying progress via a simple line chart representing remaining work (vertical axis) over time (horizontal axis).

Burn-down charts can be used at both a sprint (or iteration) and release level.</td></tr>
<tr><td>Performance Indicator[167]</td><td>An assessment indicator that supports the judgment of the process performance of a specific process.</td></tr>
</table>

[164] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[165] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[166] Term and definition(s) adapted whole or in part from http://www.telerik.com/agile-project-management-tools/agile-resources/vocabulary.aspx

[167] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

Traditional Terms: Preplanned Product Improvement (P^3I)

<table>
<tr><th>Definition[168]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>Planned future improvement of developmental systems for which design considerations are effected during development to enhance future application of projected technology.

Includes improvements planned for ongoing systems that go beyond the current performance envelope to achieve a needed operational capability.</td><td colspan="2">While there is some credence to the notion that Agile methods are similar to P^3I in that Agile methods stress delivering the highest priority requirements first, P^3I itself remains firmly rooted in the Traditional World as the improvements are all planned up front.

When applying Agile methods, the concept of P^3I can be used to reduce long term refactoring costs or technical debt. However, this must be balanced against the Agile tenet of simplicity—maximizing the amount of work not done.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Preplanned product improvement concepts can be used throughout the life cycle but are most used during:
Pre-Systems Acquisition (Technology Development)
Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
Sustainment (Operations and Support)</td><td>Big Design Up Front (BDUF)</td><td>An extensive up-front design effort which many Agilists see as the hallmark of Traditional waterfall.</td></tr>
<tr><td>Product Improvement (PI)[169]</td><td>Effort to incorporate a configuration change involving engineering and testing effort on end items and depot repairable components, or changes on other-than-developmental items to increase system or combat effectiveness or extend useful military life. Usually results from feedback from the users.</td></tr>
</table>

[168] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[169] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Program Evaluation and Review Technique (PERT)

<table>
<tr><th>Definition[170]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A technique for management of a program through to completion by constructing a network model of integrated activities and events and periodically evaluating the time/cost implications of progress.</td><td colspan="2">There is no peer per se in Agile. However, Agile does present the opportunity to associate stories (or user stories) that have dependencies on one another in the release plan. However, this is balanced against the goal that stories (or user stories) are able to stand alone and not be interdependent.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">The PERT technique and charts are used throughout the life cycle.</td><td>PERT Chart[171]</td><td>A graphic portrayal of milestones, activities, and their dependency upon other activities for completion and depiction of the critical path.</td></tr>
<tr><td>Story Maps[172] (Agile World definition)</td><td>A visual technique to prioritize Stories (or User Stories) by creating a “map” of users, their activities, and the Stories (or User Stories) needed to implement the functionality needed.</td></tr>
</table>

[170] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[171] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[172] Term and definition(s) adapted whole or in part from http://www.agilelearninglabs.com/modules/story-mapping/

Traditional Terms: Progressive Elaboration

<table>
<tr><th>Definition[173]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>Continuously improving and detailing a plan as more detailed and specific information and more accurate estimates become available as the project progresses, and thereby producing more accurate and complete plans that result from the successive iterations of the planning process.</td><td colspan="2">While not using the same terminology, this concept is fundamental to Agile methods and underlies the five levels of Agile planning.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">Progressive Elaboration concepts can be used throughout the life cycle but are most used during:
Pre-Systems Acquisition (Technology Development)
Systems Acquisition (Engineering and Manufacturing Development and Production and Deployment)
Sustainment (Operations and Support)</td><td>Rolling Wave Planning</td><td>Rolling wave planning involves planning near-term work in the greatest detail down to the lowest level of the WBS, while planning the mid-term and long-term at increasing higher levels of the WBS.

It is called “rolling” because the more-detailed planning for the next one-to-two work periods is done during the current work period such that planning “rolls forward” along with the project schedule.</td></tr>
<tr><td>Five Levels of Agile Planning[174] (Agile World definition)</td><td>The five levels of Agile planning are:
Vision - The highest level
Roadmap - The vision distilled into a high level plan.
Release – A planning segment of prioritized requirements and execution estimates.
Sprint (or Iteration) – A predefined, time-boxed period in which working software is created.
Daily Work – a daily communication and planning forum.</td></tr>
</table>

[173] Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

[174] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

Traditional Terms: Prototype

Definition[175]	**Is there an equivalent in the Agile World?**	
A model or preliminary implementation of a piece of software suitable for the evaluation of system design, performance or production potential, or for the better understanding of the software requirements.	This term means approximately the same in the Traditional World and Agile.	
How is it used in the Traditional World?	**Are there any related terms or concepts?**	
Prototypes can be used throughout the life cycle but are most used during: • ***Technology Opportunities and resources*** • ***Pre-Systems Acquisition*** (*Material Solution Analysis* and *Technology Development*) • ***Systems Acquisition*** (*Engineering and Manufacturing Development*)	Risk-Based Spike *(Agile World definition)*	A spike (a small iteration or experiment to research and answer a problem) driven by risk considerations.
	Spike *(Agile World definition)*	A spike is a small iteration or experiment to research and answer a problem.
	Brassboard[176]	An experimental device to determine feasibility and/or to develop technical or operational data; normally capable of being used in the field and may resemble the end item but it is not production ready.
	Breadboard[177]	An experimental device to determine feasibility and/or to develop technical or operational data; normally only used in a lab, it may not resemble the end item and is not production ready.

[175] Terms and definition(s) adapted whole or in part from ISA/IEC/IEEE 24765 *Systems and Software Engineering – Vocabulary*; December 15, 2010

[176] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[177] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Requirements Scrub

<table>
<tr><th>Definition[178]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A review of a draft requirements for adequacy and clarity; may also include reviewing comments regarding the requirements to validate and prioritize the requirements.</td><td colspan="2">A “requirements scrub” is at the heart of Agile planning (backlog pruning or backlog grooming); it is done when clarifying and prioritizing the product backlog, the release backlog, the sprint (or iteration) backlog, and (if used) the daily backlog.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">In a true Traditional waterfall program, the requirements scrub is done during user needs as part of the requirements process.</td><td>Relative Prioritization[179]</td><td>A 9-step process to prioritize requirements:
List the requirements
Estimate the relative benefit of each (1 to 9)
Estimate the relative penalty of not including each (1 to 9)
Sum 2 and 3 (applying weighting is optional)
Estimate the relative cost to build (1 to 9)
Estimate the relative cost to implement (1 to 9)
Estimate the relative degree of technical risk (1 to 9)
Calculate the priority number (formulas vary)
Sort the requirements in priority order</td></tr>
<tr><td>Backlog[180] (Agile World definition)</td><td>The backlog is a prioritized list of stories (or user stories) and defects ordered from the highest priority to the lowest.
Backlogs include both functional and non-functional stories (or user stories) as well as technical team-generated stories.</td></tr>
</table>

[178] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[179] Term and definition(s) adapted whole or in part from *Agile Glossary: Words and Terms Common to Agile Methods*; http://www.aspe-sdlc.com

[180] Term and definition(s) adapted whole or in part from http://www.accurev.com/wiki/agile-glosssary

Traditional Terms: System Specification

<table>
<tr><th>Definition[181]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A description of the system-level requirements, constraints, and interfaces (functional, performance, and design) along with the qualification conditions and procedures for their testing and acceptance.</td><td colspan="2">Most similar to the Agile epic, though perhaps a better comparison would be to a group of epics.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2">In a true Traditional World program, the system specification is initially reviewed at the preliminary design review and is approved at the critical design review.</td><td>System[182]</td><td>1) The organization of hardware, software, material, facilities, personnel, data, and services needed to perform a designated function with specified results, such as the gathering of specified data, its processing, and delivery to users.
2) A combination of two or more interrelated pieces of equipment (or sets) arranged in a functional package to perform an operational function or to satisfy a requirement.</td></tr>
<tr><td>Epics or Epic Stories
(Agile World definition)</td><td>A very large user story—too large to be accurately estimated or completed in a reasonably number of iterations.

Epics are common when creating the initial product backlog, and are broken down into smaller stories (or user stories) for planning and execution.</td></tr>
</table>

[181] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[182] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Traditional Waterfall Methods

<table>
<tr><th>Definition</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>A plan-driven software development methodology using distinct phases; phases are performed in a single-pass, sequential order, and the initiation of any subsequent phase requires the documented completion of the previous phase.

The notional phases are:
• Requirements elicitation
• Requirements analysis
• System design
• System construction
• System test and integration
• System operation
• System retirement

The main instantiations are the original waterfall paradigm and the V-shaped paradigm.</td><td colspan="2">There is an argument that an Agile sprint (or iteration) is a mini-waterfall; however there are key aspects of Agile methods not present in Traditional waterfall such as:
• Continuous integration
• Continuous test
• Sustainable pace
• Set scope and budget
• Daily meetings

In addition, a series of mini-waterfalls does not allow for the continuous reprioritization of requirements to be addressed in each mini-waterfall.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="2"></td><td>Incremental Approach[183]</td><td>Determines user needs and defines the overall architecture, but then delivers the system in a series of increments or builds where the first build incorporates a part of the total planned capabilities, the next build adds more capabilities, and so on, until the entire system is complete</td></tr>
<tr><td>Evolutionary Acquisition (EA) [184]</td><td>Preferred DoD strategy for rapid acquisition of mature technology which delivers capability in increments while recognizing upfront the need for future capability improvements; each increment is a militarily useful and supportable operational capability that can be developed, produced, deployed, and sustained.</td></tr>
</table>

[183] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

[184] Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

Traditional Terms: Work Breakdown Structure (WBS)

<table>
<tr><th>Definition[185]</th><th colspan="2">Is there an equivalent in the Agile World?</th></tr>
<tr><td>An organized method to break down a project into logical subdivisions or subprojects at lower and lower levels of details; very useful in organizing a project.</td><td colspan="2">When taken in their total, the five levels of agile planning (product vision, product roadmap, release plans, sprint (or iteration) plans, and daily commitments) match or even exceed the information captured in a Traditional World WBS.

A WBS can also be associated with the Scrum teams that make up the Scrum of Scrums on a larger program.</td></tr>
<tr><th>How is it used in the Traditional World?</th><th colspan="2">Are there any related terms or concepts?</th></tr>
<tr><td rowspan="3">A WBS is used to organize all work on a Traditional World program.

In addition, the structure of the WBS tends to match the structure of both the government and development contractor organizations (and vice versa), which may or may not improve the project's chances for success.</td><td>Organizational Breakdown Structure (OBS) [186]</td><td>A hierarchical depiction of an organization relating work packages to the organizational units performing the work.</td></tr>
<tr><td>Cost Breakdown Structure[187]</td><td>A system for subdividing a program into hardware elements and subelements, functions and subfunctions, and cost categories to provide for more effective management and control of the program.</td></tr>
<tr><td>Contract Work Breakdown Structure (CWBS) [188]</td><td>A complete WBS for a contract. It includes the DoD-approved program WBS extended to the agreed contract reporting level and any discretionary extensions to lower levels for reporting or other purposes. It includes all the elements for the products (hardware, software, data, or services) that are the responsibility of the contractor. This comprehensive WBS forms the framework for the contractor's management control system.</td></tr>
</table>

185 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

186 Terms and definition(s) adapted whole or in part from *PMBOK Guide® – Third Edition*

187 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

188 Terms and definition(s) adapted whole or in part from *DAU Glossary of Defense Acquisition Acronyms and Terms*, 14th Edition, July 2011

5 Summary

The Traditional World and the Agile World are simply two instantiations of the software world. In some ways they are related by cause and effect since the Agile Manifesto and related methods grew out of a group of developers' frustrations with the "heaviness" of DoD's implementation of waterfall, which had itself grown out of the need for rigor to address a "software crisis."

Agile methods emphasize structured, multi-level planning, continuous customer involvement, frequent (if not continuous) test, continuous integration, and frequent delivery of potentially deliverable software. As such, Agile methods can be argued to be a better way for many DoD projects to obtain the insight and control wanted when DoD imposed many of its current practices. There is no doubt that DoD adoption of Agile methods will require significant cultural and even perhaps legal (i.e., contracting) changes. However, the possibility that government programs could nimbly respond to changing environments and requirements on a pace measured in months instead of years is simply too good of a future to ignore.

We have tried to show, though, that Agile principles are not foreign to software development in the federal government; there are many examples of Agile or even eXtreme development efforts over the last decades. We have also tried to show the both the Traditional World and the Agile World use the same fundamental building blocks and have the same fundamental goals.

This is not to say that traditional methods are the same as Agile methods. They are *not*. However, they are two different ways to perform the same tasks—analyze, design, build, test, and deploy. In some ways, they can be thought of as parallel worlds, where the "what" to do is the same but the "how" to do it is different.

Thus, the difference is in perspective and application—and words. By considering the similarities, the goal is to ease fear and rejection of either method by the other community. By no means have we concluded they are equivalent.

We deliberately limited this paper to a total of 50 terms, though astute readers will point out we included dozens more in the "related terms and concepts" block. We were surprised as we began this paper that this type of Rosetta Stone didn't already exist, though we did find a number of smaller efforts.[189]

We hope this technical note stimulates discussions among practitioners in both communities and that regular revisions can be made to this report so that terms and definitions can be added, updated, or removed as needed.

Even more, we hope that this technical report helps facilitate DoD's adoption of Agile methods. We hope that we have shown that the two worlds are not as far apart as some believe.

[189] However, as we worked on the paper and the term count went well above 200 and was still climbing—as every term seemed to require two-to-three more in its definition—we did grasp part of the reason.

Appendix A Waterfall Software Development – DoD's Misplaced Emphasis?

Despite the widely-held view that the waterfall development paradigm is DoD's and the federal government's "tradition," there are many examples where federal software development could rightly be called Agile—even extreme—long before these terms assumed their current interpretations.

Larman and Basili provide several examples of this, starting in the 1960s [Larman 2003]. Project Mercury used half-day, time-boxed iterations and practiced test-first development for each micro-increment. In the 1970s, the Light Airborne Multipurpose System (LAMPS) incrementally delivered millions of lines of code in 45 time-boxed iterations that were each one month long. Each of the LAMPS iterations was delivered on time and under budget.

Also in the 1970s, the primary avionics software for the space shuttle was delivered in a series of 17 iterations over 31 months. They avoided waterfall (although they did call it the "ideal" software development cycle) because the requirements were not stable. Instead, they used "… an implementation approach (based on small incremental releases) … which met the objectives by applying the ideal cycle to small elements of the overall software package on an iterative basis."

While not mentioned in Larman and Basili's paper, in the 1970s the Department of Veteran Affairs made colocation a primary mechanism for the VistA system, when many VistA applications were built by doctors and/or clinicians working side by side with a programmer. In fact, in many cases the doctor or clinician were themselves the programmer using the MUMPS language (Massachusetts General Hospital Utility Multi-Programming System).

These examples and more indicate that many people in the federal government understood the benefits of an iterative, incremental approach. Larman and Basili also show that many people knew that what was to become known as a "waterfall" approach would not work for large, complex systems. For example, they provide a quote from Gerald Weinburg, who worked on Project Mercury: "… all of us, as far as I can remember, thought waterfalling of a huge project was rather stupid, or at least ignorant of the realities … I think what the waterfall description did for us was make us realize we were doing something else, something unnamed except for 'software development.'"

But while there were successes, there were failures—so much that by the late 1960s it was deemed a "software crisis." The NATO Science Committee held a conference in 1968 on "software engineering," apparently the first time this term was used and "… deliberately chosen as being provocative, in implying the need for software manufacture to be based on the types of theoretical foundations and practical disciplines, that are traditional in the established branches of engineering" [Naur 1969].

This conference focused on such issues as

- the problems of achieving sufficient reliability in the data systems which are becoming increasingly integrated into the central activities of modern society

- the difficulties of meeting schedules and specifications on large software projects
- the education of software (or data systems) engineers
- the highly controversial question of whether software should be priced separately from hardware

At almost the same time (1970), Dr. Winston Royce published *Managing the Development of Large Software Systems*, where he presented what he were his "personal views about managing large software developments" [Royce 1970]. Though Royce never used the term "waterfall" in his paper and the paper is in fact an argument for iterative development, many people consider his paper as the basis for the waterfall development methodology to the point where he has been called the "father of waterfall."

Royce began his paper with what I will call Royce Model #1, which he felt was the simplest form of a software development process:

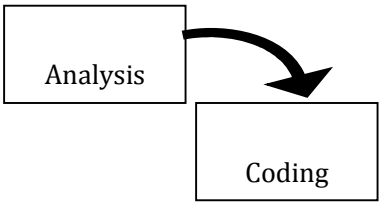


Figure 8: Royce Model #1

Royce captioned this *Implementation steps to deliver a small computer program for internal operations.*

Royce said Model #1 was potentially acceptable when "… the effort is sufficiently small and if the final product is to be operated by those who built it—as is typically done with computer programs for internal use." Royce then described Royce Model #2, which he called "more grandiose:"

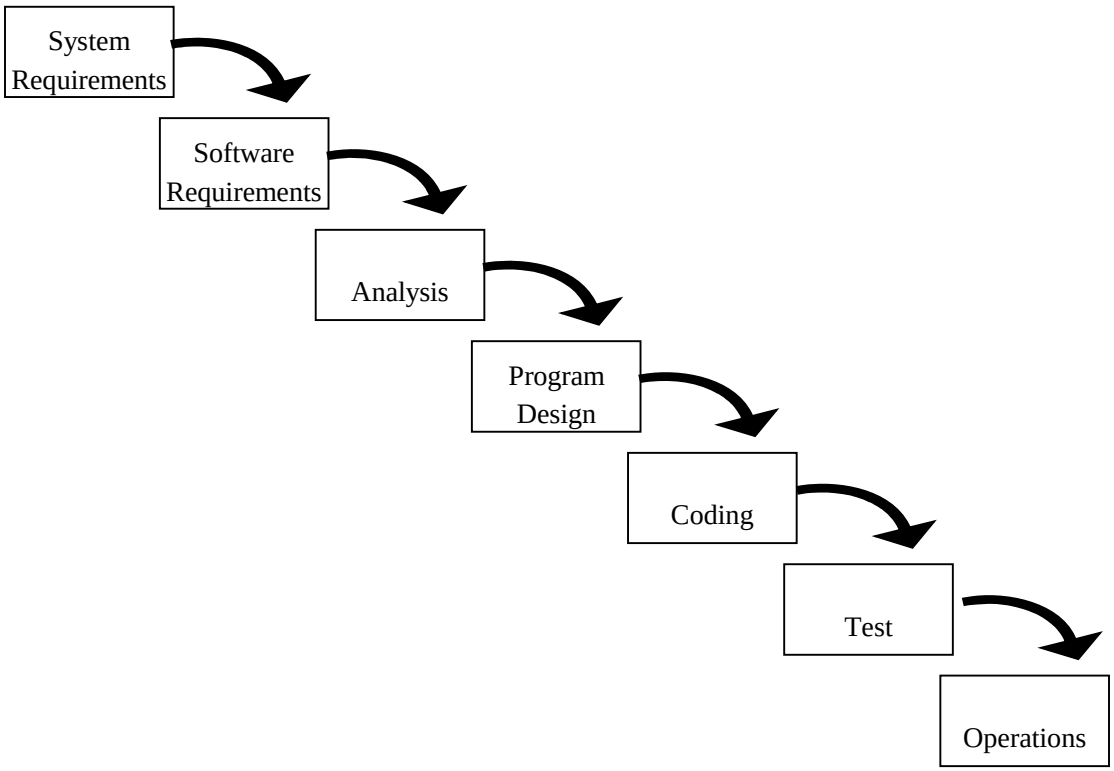


Figure 9: Royce Model #2

In a very good example of how the choice of words can have an impact far beyond what the author perhaps envisioned, Royce captioned Model #2 as *Implementation steps to develop a large computer program for delivery to a customer*.

Had a reader stopped at that caption, they would have thought Royce just described his recommended approach for delivering large computer programs, though a complete reading of his paper would have dispelled this. Royce's paper continued with the addition of the "… iterative relationship between successive development phases for this scheme," or Royce Model #3:

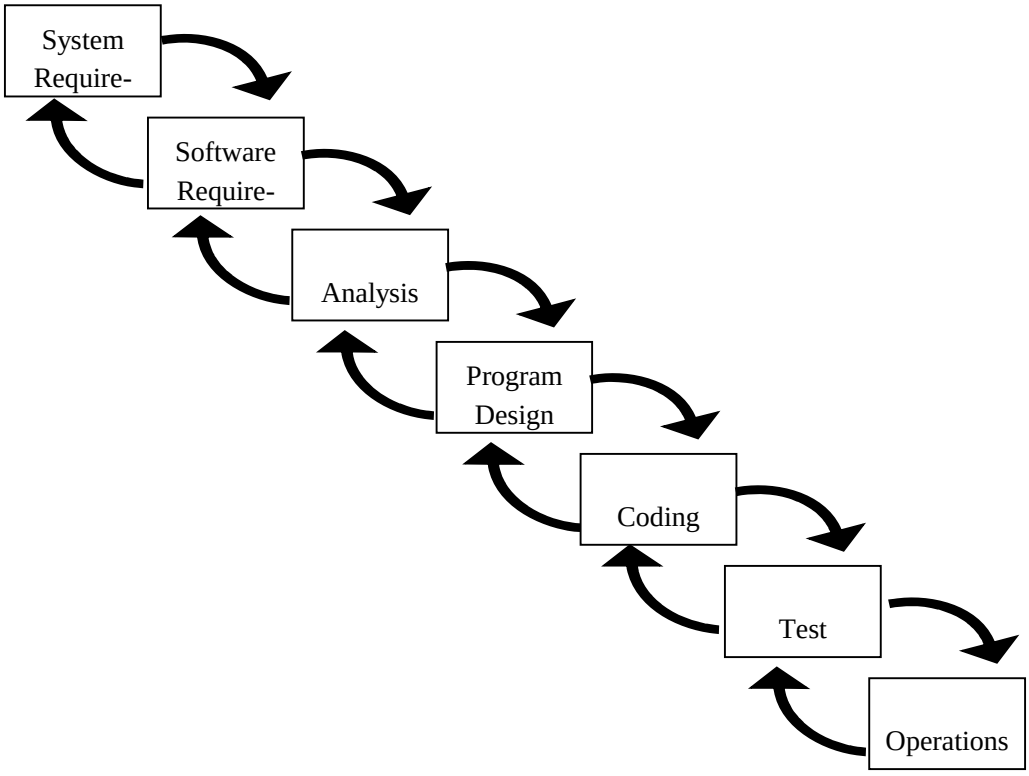


Figure 10: Royce Model #3

But even while Royce said that he believed in the concept of Model #3, he felt that even this model was "*risky and invites failure*" (italics added). As an example, he pointed out that with testing at the end of the development cycle, issues with timing, storage, input/output transfers, and the like, are not discovered until a major redesign is invariably required.

Royce observed: "The required design changes are likely to be so disruptive that the software requirements upon which the design is based and which provides the rationale for everything are violated. Either the requirements must be modified, or a substantial change in the design is required. In effect the development process has returned to the origin and *one can expect up to a 100-percent overrun in schedule and/or costs* (italics added)".

Royce devoted the remainder of his paper to the "… five additional features that must be added to this basic approach to eliminate most development risks." These included his recommendation that this model be run at least twice (iteratively), with the first time being a significant prototyping phase that was used to better understand the requirements, better understand the technologies involved, and ensure it was providing what the customers actually needed.

Royce also recommended that the customer be involved well before testing as "*for some reason what a software design is going to do is subject to wide interpretation even after previous agreement*" (italics added).

As stated earlier, Royce never used the term waterfall in his paper. Again quoting from Larman and Basili, Walker Royce, Dr. Royce's son, said this of his father and the paper:

"He was always a proponent of iterative, incremental, evolutionary development. His paper described the waterfall as the simplest description, but that it would not work for all but the most straightforward projects. The rest of his paper describes [iterative practices] within the context of the 60s/70s government contracting models (a serious set of constraints)" [Larman 2003].

But the federal government's push for engineering rigor and greater control settled on what was now known as "waterfall." Manifested by extensive documentation, a strong preference for a single-pass, sequential development method, and heavy oversight, the waterfall development method was perhaps best captured in DOD-STD 2167 (1985).

However, even as DOD-STD 2167 was released, the document itself and the waterfall method it espoused were already under attack. In 1986, a draft copy of Revision A to MIL-STD 2167 appeared which removed the emphasis on top-down design and called out rapid prototyping as an alternative to the waterfall. In 1987, the Defense Science Board recommended that 2167 be revised to " … to remove any remaining dependence upon the assumptions of the "waterfall" model and to institutionalize rapid prototyping and incremental development" [DSB 1987].

But the perception that waterfall was the federal government's preferred development approach had become firmly embedded. Federal software development and acquisition still retained a strong hardware-oriented, waterfall flavor, as was argued in a 2010 report issued by the National Research Council [NRC 2010]:

> For example, the terminology used to describe the engineering and manufacturing development phase emphasizes the hardware and manufacturing focus of the process ... Preliminary design reviews (PDRs) and critical design reviews (CDRs), hallmarks of the waterfall SDLC model, are prescribed for every program, with additional formal Milestone Decision Authority (MDA) decision points after each design review. At least four and potentially five formal MDA reviews and decision points occur in every evolutionary cycle.

This isn't to say that traditional waterfall does not or has not delivered quality products that are in the field and working today—it obviously has. However, more often than not the traditional world has delivered quality products despite waterfall, not because of waterfall.

Continuing to quote from the 2010 National Research Council report:

> As a result, although the oversight and governance process of DODI 5000 does not forbid the iterative incremental software development model with frequent end-user interaction, *it requires heroics on the part of program managers (PMs) and MDAs to apply iterative, incremental development (IID) successfully within the DODI 5000 framework* (italics added).
>
> Today, many of the DOD's large IT programs therefore continue to adopt program structures and software development models closely resembling the waterfall model rather than an IID model with frequent end-user interaction. Even those that plan multiple delivered increments typically attempt to compress a waterfall-like model within each increment.

So if the premise that DOD-STD 2167 and other waterfall-based standards and instructions fundamentally misunderstood Royce could be accepted, it follows that DoD's multi-decade emphasis on waterfall was misplaced. And if that is accepted, one can only wonder what DOD's current software environment would be like if 2167 had emphasized DoD's "Agile roots" instead.

Appendix B History of Agile[190]

For additional information, please see these other SEI Technical Notes:

- *Agile Methods: Selected DoD Management and Acquisition Concerns*; CMU/SEI-2011-TN-002
- *Considerations for Using Agile in DoD Acquisition*; CMU/SEI-2010-TN-002

In many ways there is nothing new in Agile. As we showed in Appendix A there were a number of software development efforts in the federal government that were agile well before agile became a well-known term. They were what we call 3I—inventive, iterative, and incremental.

What is new about Agile is how these "old" components are combined with some new components (ideas, practices, theories, etc.) to yield something more powerful and coherent. However, as Jim Highsmith asserts, "the Agile approaches scare corporate bureaucrats—at least those that are happy with pushing process for process' sake versus trying to do what is best for the 'customer' and deliver something timely and tangible 'as promised'—because they run out of places to hide."[191]

This coherence—and the current energy driving advocacy of Agile—was the result of a remarkable meeting among thought leaders and consultants[192] in software development who would normally have been competitors. In February 2001 17 people met to try to find common ground and ultimately produced the Agile Software Development Manifesto. This document detailed all of their commonalities—overlooking, for the moment, areas where they had differences of opinion.

Agile Manifesto and Principles

The self-named Agile Alliance shared allegiance to a set of compatible values promoting organizational models based on people, collaboration, and building organizational communities compatible with their vision and principles.[193]

From the 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

[190] This section draws extensively from the SEI technical note, *Considerations for Using Agile in DOD Acquisition*.

[191] http://agilemanifesto.org/history.html

[192] The signatories were representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, and others: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas.

[193] http://agilemanifesto.org/history.html

- 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.

As the Agile Alliance noted, the four dichotomies listed in the manifesto (such as "individuals and interactions over processes and tools") are not intended to suggest that what is on the left is important and what is on the right is unimportant; rather, what is on the right, while important, is simply less important than what is on the left.

For example, some believe that the Agile approach advocates providing no documentation other than the code itself. The Agile community would argue instead that documentation *is* important, but no more documentation should be created than is absolutely necessary to support the development itself and future sustainment activities. In fact, Agile emphasizes collaboration and the notion that when documentation replaces collaboration the results are problematic. Documentation should be the result of collaboration.

The Agile Alliance says the following 12 principles underlie the Agile Manifesto:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity—the art of maximizing the amount of work not done—is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.[194]

From these principles, it is understood that Agile is really a *philosophy* or *development approach*, and it comprises a number of more specific methods, for example, eXtreme programming (XP), Scrum, and Adaptive Software Development (ASD).

[194] http://agilemanifesto.org/principles.html

References/Bibliography

URLs are valid as of the publication date of this document.

[Beck 2001]
Beck, Kent, Beedle, Mike, van Bennekum, Arie, Cockburn, Alistair, Cunningham, Ward, Fowler, Martin, Grenning, James Highsmith, Jim, Hunt, Andrew, Jeffries, Ron, Kern, Jon, Marick, Brian, Martin, Robert C., Mellor, Steve, Schwaber, Ken, Sutherland, Jeff, & Thomas, Dave. *Manifesto for Agile Software Development.* 2001.
http://agilemanifesto.org/

[Boehm 1976]
Boehm, Barry W. "Software Engineering," *IEEE Transactions on Computers C-25*, 12 (December 1976): 1226-1241.

[Boehm 1988]
Boehm, Barry W. "A Spiral Model of Software Development and Enhancement." Computer (May 1988): 61-72.

[Cabri 2006]
Cabri, Anthony & Griffiths, Mike. *Earned Value and Agile Reporting*. Quadrus Development Inc., 2006.

[Cohn 2006]
Cohn, Mike. Agile Estimating and Planning. Pearson Education, 2006.

[Colburn 2008]
Colburn, Alex, Hsieh, Jonathan, Kehrt, Matthew, & Kimball, Aaron. There is no Software Engineering Crisis. 2008.
http://www.cs.washington.edu/education/courses/cse503/08wi/crisis-con.pdf

[DSB 1987]
Defense Science Board. Report of the Defense Science Board Task Force on Military Software, September 1987.
http://www.dtic.mil/cgi-bin/GetTRDoc?Location=U2&doc=GetTRDoc.pdf&AD=ADA188561

[Gugenberger 2011]
Gugenberger, Pascal. The Waterfall Accident.
http://pascal.gugenberger.net/thoughts/waterfall-accident.html

[Highsmith 2001]
Highsmith, Jim. *History: The Agile Manifesto*. 2001.
http://agilemanifesto.org/history.html

[Lapham 2010]
Lapham, Mary Ann; Williams, Ray; Hammons, Charles (Bud) ; Burton, Daniel; & Schenker, Alfred. *Considerations for Using Agile in DoD Acquisition (CMU/SEI-2010-TN-002).* Software Engineering Institute, Carnegie Mellon University, 2010.
http://www.sei.cmu.edu/library/abstracts/reports/10tn002.cfm

[Larman 2003]
Larman, Craig & Basili, Victor. "Iterative and Incremental Development: A Brief History." *Computer 36*, 6 (June 2003): 47-56.

[Lehman 1980]
Lehman, M.M. "Programs, Life Cycles, and Laws of Software Evolution," 1060-1076. *Proceedings of the IEEE 68*, 9: 1980.

[Naur 1969]
Naur, Peter & Randell, Brian, eds. *Software Engineering: Report on a Conference Sponsored by the NATO Science Committee*. 1969

[NRC 2010]
National Research Council. Achieving Effective Acquisition of Information Technology in the Department of Defense. Committee on Improving Processes and Policies for the Acquisition and Test of Information Technologies in the Department of Defense. 2010.
http://www.nap.edu/catalog.php?record_id=12823

[Royce 1970]
Royce, W.W. "Managing the Development of Large Software Systems,"1-9. Proceedings of IEEE WESCON, August 1970. IEEE, 1970 (originally published by TRW).

[Wills 2010]
Wills, Patrick. *The Defense Acquisition System*. Defense Acquisition University, 2010.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	October 2013	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Parallel Worlds: Agile and Waterfall Differences and Similarities	FA8721-05-C-0003

6. AUTHOR(S)

M. Steven Palmquist, PE, PMP, Mary Ann Lapham, PMP, CSM, Suzanne Miller CSM, Timothy Chick, Ipek Ozkaya

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2013-TN-021

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
HQ ESC/XPK 5 Eglin Street Hanscom AFB, MA 01731-2116	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

This technical note (TN) is part of the Software Engineering Institute's series on Agile in the Department of Defense (DoD). It primarily addresses what at first seems a small issue on the road to Agile adoption—the confusion of terms. However, this is a much larger issue, as ineffective communications among and between stakeholders is often cited as a significant stumbling block on any project. Confusion over simple terms is a needless hurdle.

Many terms and concepts used by Agile practitioners seem to confound those working in the DoD's Traditional World of waterfall-based environment, and vice versa. The goal of this paper is to assemble terms and concepts from both environments to show both the similarities (of which there are many) and differences (of which there are also many).

A comprehensive cross dictionary was beyond the scope of this work; the authors strove to select from those terms most commonly encountered when considering Agile adoption. Therefore, the authors selected terms based on suggestions from both inside and outside the SEI, but deliberately limited themselves to 25 terms from each environment.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Agile, waterfall	101

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL