An Overview of the Tempered Agility™ Project Framework

David A. Bly
Revision 1
June 19, 2011

ABSTRACT

This paper provides a brief overview of the Tempered Agility project methodology framework, a project-oriented application of Scrum blended with elements of traditional software engineering and project management disciplines. This balanced approach is more robust than either agile or waterfall extremes and is recommended for projects involving significant software development. Commonly tailored to meet the needs of the individual project, Tempered Agility has been proven successful across a wide range of projects from small to large and from simple to complex.

CONTENTS

INTRODUCTION

Wouldn't it be lovely if you could learn one way to run software projects and know that you could use that same approach all the time? You would get really good at it by using it over and over again. You would be able to estimate the costs and schedules of projects more accurately because you would have done it the same way several times. Everyone on the team would know what they need to do and everyone would know what the other folks on the team are doing. Everyone would know the outcomes of each task and how they would be used in downstream tasks. You would simply step through the process over and over with perfect knowledge of how things would turn out.

Unfortunately, that notion is an unrealistic fantasy. Many people have attempted to define a project **M**ethodology (capital M) as the one true way only to see it fail or fall into disuse, gathering dust on the bookshelf. The reason is that no two projects are ever alike. The team composition changes, the technology changes or the customer changes, with each change bringing different risks, experience levels, strengths and weaknesses. In addition, the objectives change, the timeframes change and the funds available change. So, the practical reality is that you need to be aware of the impacts these changes have on the way you engage in software projects and be able to tailor your approach to reflect the situation with which you are presented. It is within this reality that Tempered Agility offers a *methodology framework:*

- ***m****ethodology* is spelled with a lower case m. Tempered Agility is not a Methodology that prescribes a specific set and order of tasks with specific outcomes of each. It as a set of tools that you can pull out of your toolkit to apply when the situation warrants it. Think of it instead as a set of project fulfilment patterns that may be applied to situations where they fit, just as development patterns are used when they make sense.
- A *framework* is a general structure on which you apply specific activities that make sense for the context within which the project is accomplished. The activities described in Tempered Agility may change form from one project to the next to fit the situation determined to apply for the specific project.

At the beginning of a Tempered Agility project, the team needs to define the project life cycle model for the project (such as waterfall or agile), the degree of formality to apply to the project management approach, the specific artifacts to be produced in the product development process, the specific activities that will produce those artifacts and the sequence in which they will be undertaken. Tempered Agility refers to this project design process as Tailoring. Tailoring depends on several factors: the organization's culture, team preferences and experience levels, trust levels, the type of project and the risks associated with the project. It is covered in more depth in the section entitled "Tailoring Tempered Agility" after the basics of Tempered Agility are described.

Tempered Agility is intended to provide a middle-ground for software projects that covers a broader spectrum of applicability than extreme waterfall models or extreme agile models. It is a project-oriented application of the Scrum methodology balanced with traditional software engineering and project management disciplines. We believe this balanced approach is a stronger and more robust approach than either of the two extremes (agile or waterfall). Commonly tailored to meet the needs of the individual project, Tempered Agility has been proven to work through its repeated success across a wide range of projects from small to large and from simple to complex.

Why "*Tempered*" Agility?

The American Heritage Dictionary of the English Language[1] offers several definitions of *tempered*:

- To modify by the addition of a moderating element
- To bring to a desired consistency, texture, hardness, or other physical condition by or as if by blending, admixing, or kneading
- To harden or strengthen by application of heat or by heating and cooling
- To strengthen through experience or hardship; toughen
- To adjust finely; attune

These notions fit well with the concept of Tempered Agility. Think of it as a moderated and strengthened balance of agile and traditional concepts:

- **Blended** – Tempered Agility blends agile concepts with traditional analysis, software engineering and project management concepts.
- **Balanced** – It balances the two ends of the spectrum to a "happy medium".
- **Tailored** – It is easily tailored to meet the needs of the individual project.
- **Disciplined** – It preserves software engineering and project management disciplines while benefitting from the team-oriented values of agile.
- **Strengthened** – The combination is stronger and more robust than either of the two extremes.
- **Scaled** – The combination scales more easily to accommodate the needs of large scale and complex projects.
- **Tested** – The combination has been proven to work in adverse conditions.

Tempered Agility provides several distinct advantages compared to traditional approaches:

- Better fit-for-purpose by involving customers more
- Faster results through frequent, small releases
- More efficient results by avoiding costly rework
- Better accommodation of mid-course changes
- Avoidance of big-bang integration problems that so commonly plague software projects
- Better support of management's needs as well as the needs of the project team

Principles of Tempered Agility

Several important operating principles are inherent in the processes, artifacts and mechanisms that comprise Tempered Agility:

- Plan to reduce uncertainty with evolutionary discovery.
- Discover, measure, learn and adapt: measure, learn and adapt frequently through continuous questioning and introspection.
- Incremental improvement is better than postponed perfection.
- Embrace change: plan to accommodate change based on the new learning that emerges in the process of discovery.

[1] *The American Heritage® Dictionary of the English Language, Fourth Edition*. Houghton Mifflin Company, 2004.

- Facilitate client-driven adaptation of plans and deliverables as new things are learned and as priorities change.
- Develop teamwork based on shared customer/producer responsibility
- Deliver results in frequent, short, time-boxed sprints; provide incremental results early and often.
- Maximize business value early.
- Practice deliverables-based planning vs. based planning.
- Detect problems and failures early.
- Repeat successes.
- Employ cross-functional teams.
- Facilitate team success.
- Apply only as much structure as appropriate for the situation.
- Practice active daily management.
- Lead – do not just manage.
- Practice just-in-time planning.
- Practice just-in-time specification.
- Practice transparency.

KEY POINTS

The primary characteristics of Tempered Agility include:

- Iterative and incremental solution development with frequent re-planning
- Frequent integration, testing and deployment of solution increments
- High customer involvement
- User-centric design employing use cases, user interface prototypes and flow diagrams
- Scrum for daily team efforts
- Self-managing teams

Iterative and Incremental Solution Development

Iterative development means repeating the solution cycle of definition, design, development and deployment several times. Each cycle creates some of the total set of desired features in each iteration or "sprint".

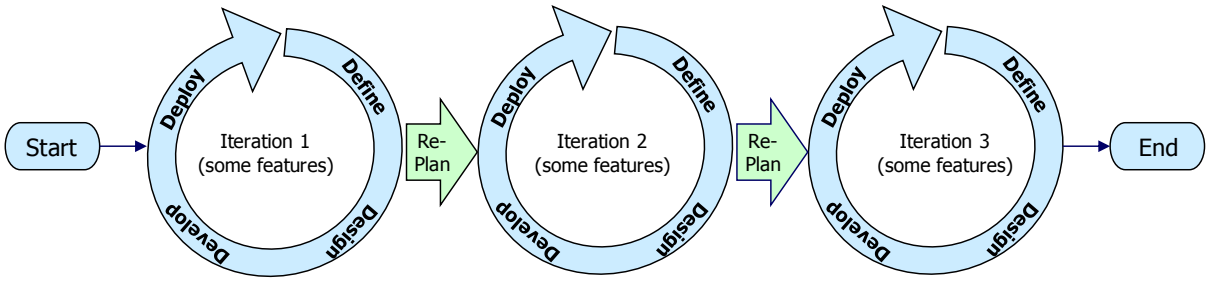


Figure 1 – Iterative and Incremental Development

In between each sprint, the client and the project team have an opportunity to adapt plans as needed to accommodate changes, new learning and to resolve procedural defects.

Frequent Integration, Testing and Deployment of Solution Increments

Too often in traditional software development approaches, problems with the integrated system lay undiscovered until the end of the development cycle and into the full-system testing stage. In such cases, sheer panic ensues as errors skyrocket during the integration period at the end. As a result, schedules and budget both run over initial estimates.

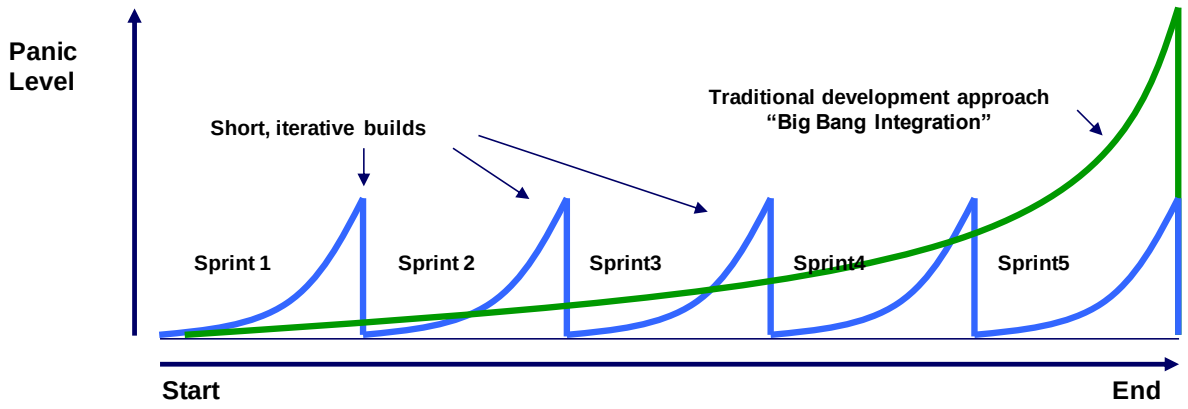


Figure 2 – Frequent Integration, Testing and Deployment

Tempered Agility tackles this problem in small, time-boxed increments, each of which are integrated, deployed to a testing environment, tested by quality assurance representatives and remediated to resolve major defects, one increment at a time. Effectively, one can think of it as several "small panics" along the way compared to one panic one at the end. Time and again, this approach has proven to improve overall cost and schedule predictability for the project.

High Customer Involvement

Involving the business users on a daily basis with the project team provides several advantages:

- Quicker decisions about requirements, design and acceptability
- Quicker clarification on points of confusion
- Better communication and understanding both ways
- Better fit with requirements
- Quicker resolution of defects
- Quicker delivery of solutions

This is accomplished by having an authoritative user/customer representative, referred to as the "Product Owner," attend brief, daily, stand-up meetings with the project team and small, problem-solving discussions as needed to help the team make decisions on a daily basis. The Product Owner also works to manage the client stakeholder community in conjunction with the project manager. This dyadic management team is sometimes supplemented with a representative from the client's IT community to provide an effective management trio.

User-Centric Definition and Design

Our approach to definition and design of the solution involves three interrelated facets: use cases, user interface prototypes and flow diagrams.

- Use cases document the conversations that are required for users to achieve their goals of system usage. They provide a unifying framework of understanding for the various perspectives that contribute to the project's success.

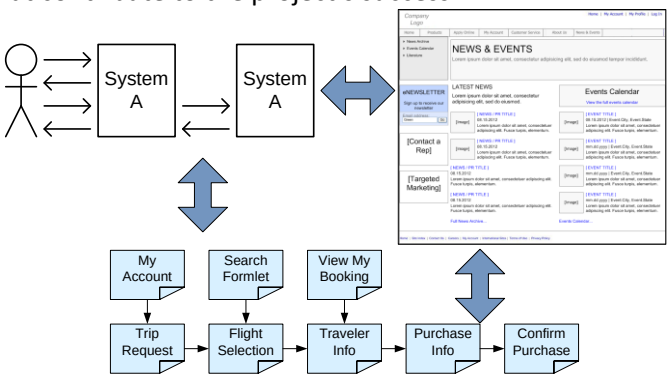


Figure 3 – User-Centric Definition and Design

- User interface prototypes may include wireframes, page mock-ups, or low-fidelity "white-board" simulations of the pages or screens through which the user will interact with the solution.
- Flow diagrams help the team design better user experiences by focusing on the sequence of activity through the various pages or screens associated with one or more use cases.

Taken together, Tempered Agility's three-pronged approach to user-centric design ensures that the solution provided to the client is one that is readily usable by the using community, improving its acceptance and adoption in the organization.

Scrum

Scrum [1] is the most widely used flavour of agile methodologies. It provides the methodology framework for accomplishing iterative and incremental development as well as frequent integration, deployment and testing as the solution is built incrementally. With Scrum, a project is constructed as a series of sprints, each of which will last from 2 – 4 weeks.

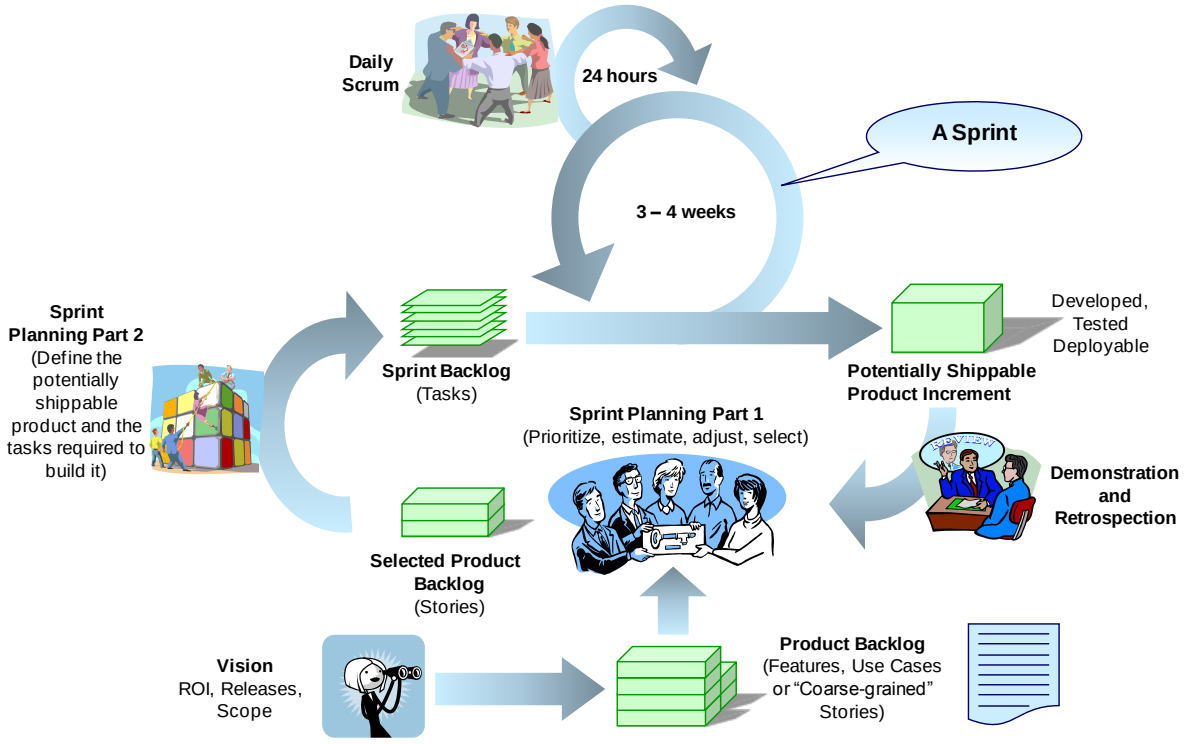


Figure 4 –Scrum

Each sprint starts with a two-part planning session to provide detailed task planning of the work to be accomplished in the upcoming sprint. The goal of each sprint is to produce a "product increment" that has been tested and remediated within the sprint. Finally, each sprint is concluded with a demonstration of the sprint results and retrospection which helps the team figure out ways of improving their performance in the next sprint.

High-Performing, Self-Managing Teams

High-performing, self-managing project teams exhibit the following characteristics:

- They are aligned on a common set of objectives that they all understand and share a strong sense of ownership in achieving those objectives together.
- They have reached consensus on the approach they will follow to achieve those objectives.
- They focus on the work at hand to achieve their objectives.
- They know each others' strengths and weaknesses and support each other to fill in the gaps.
- They challenge each other to do better and enforce their own rules.

- They regularly engage in joint problem-solving.
- They communicate clearly, honestly and without hidden agendas.

Project teams are neither high-performing nor self-managing the moment the project starts. They need to go through a process to get there. It is a painful process sometimes and one that is easily derailed. Helping them get there requires patience, understanding, coaching and facilitating by project managers that respect the process and know that it pays off if successful.

Project managers of Tempered Agility projects perform the role of scrum master, but also as project leader. They need to recognize the value of servant leadership. It is their job to lead the team, to guide the team, to coach the team, to facilitate group problem-solving and to support and encourage the team to accomplish its goals in a self-managed manner.

Tempered Agility Project Flow

The following diagram provides a graphic view of the project flow which incorporates all of the Tempered Agility features discussed above. The basic Scrum framework is preceded by a definition sprint and followed by a deployment sprint. These two book ends allow us to apply the principles of Scrum more effectively in a project environment.

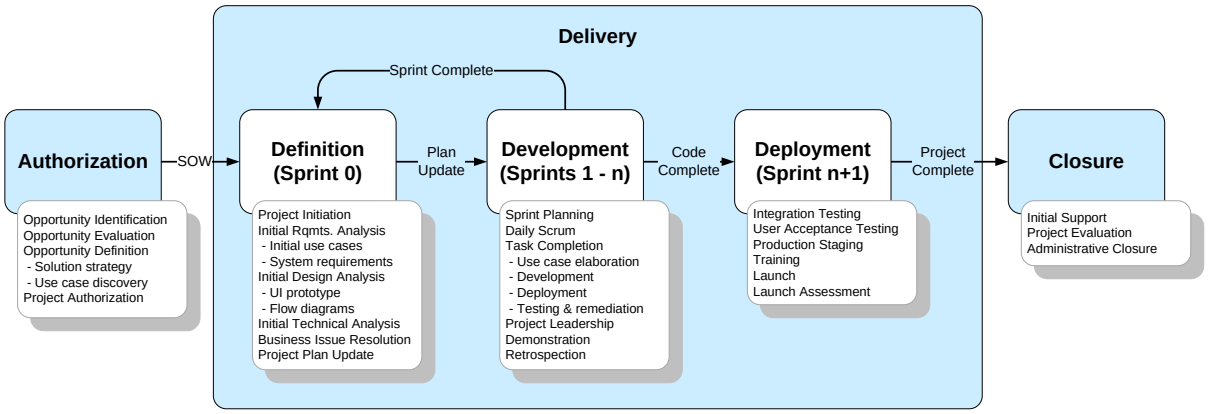


Figure 5 – Tempered Agility Project Flow

The definition sprint focuses on understanding the initial user and technical requirements and establishing key design concepts that will be applied throughout the project. The solution definition is then elaborated and developed in a series of development sprints. The number and length of development sprints varies from one project to the next based on the complexity and size of the resulting product solution. At the end of each development sprint, the project plan developed in the definition sprint is revisited to incorporate changes, reprioritization or new learning from the sprint just completed. The deployment sprint concludes the project with a disciplined approach to testing, training and deployment of the solution to its production computing environment.

Each of the five stages shown above is described in the following sections.

Balancing Waterfall and Agile Approaches

The Definition Sprint provides an opportunity to pursue a balance between the strictly Waterfall approach and the strictly Agile approach as illustrated in Figure 6 below.

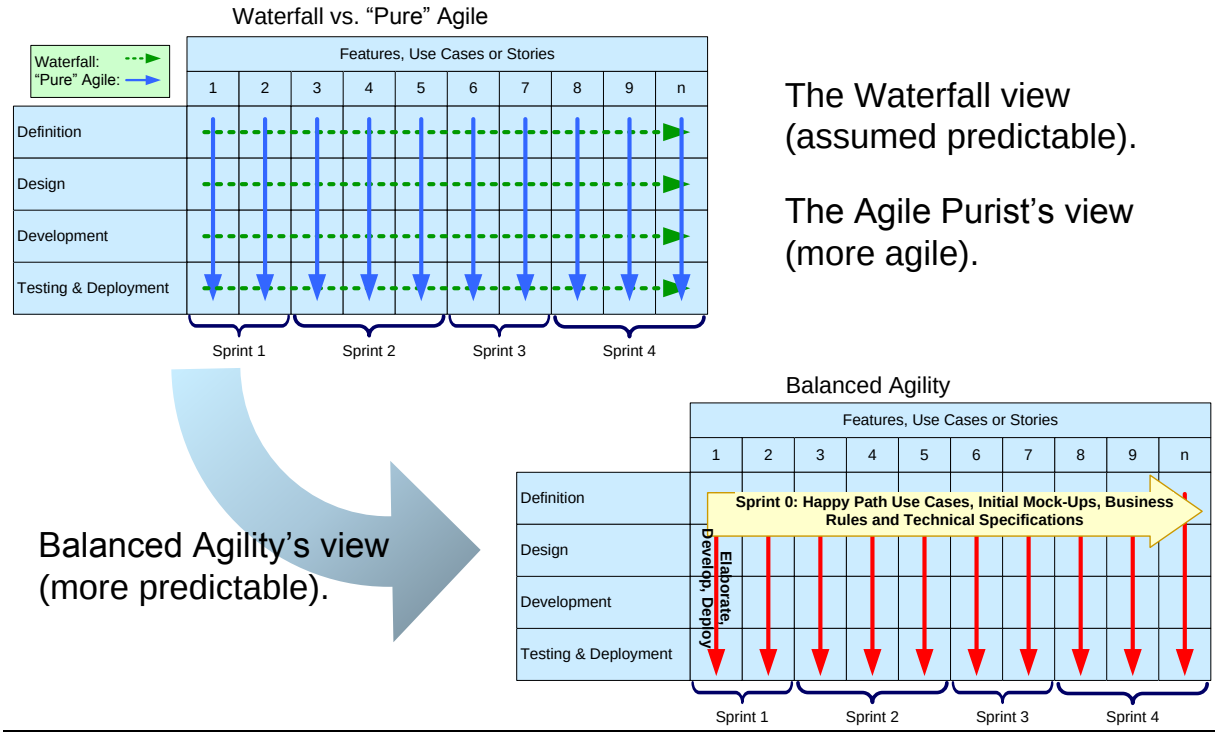


Figure 6 – Balancing Waterfall and Agile Approaches

In a strictly waterfall approach, the definition stage is completed for all features (or use cases or stories) and approved before any design work is begun. All design work is completed and approved before any development is begun. All development in completed before any testing or deployment is begun. This is perceived by management to be more predictable, though it is not generally perceived as such by those actually doing the work. Experience has shown that this approach takes too long to deliver usable capability to users, results in bloated requirements as users throw in every requirement they can think of for fear that the next opportunity will be far in the future and results in very costly correction if mistakes or absences are not discovered until late in the project.

In the strictly agile approach, the features (or use cases or stories) are identified without very much analysis work taking place before commencing with the first iteration. The definition, design, development, testing and deployment are accomplished in the first iteration for the selected features and then another set of features is selected for the next iteration. This approach works well for the people doing the work, minimizes unneeded features, gets usable capability to the user sooner than waterfall approaches and readily accommodates change. However, the strictly agile approach does not address the legitimate needs of customers and management to know how much a project will cost and how long it will take early in the life cycle. Nor does is support design/cost trade-offs early in the project.

In a product-oriented application of agile, where the team is continually enhancing an application that is already in production, there will always be another iteration as long as the system remains useful. Therefore, features which are not completed in the planned iteration can overflow into the next iteration without much angst. However, in a project-oriented application of agile, the project only has a defined amount of budget and time in which to complete the goals and objectives of the

project. And, these budgetary and schedule-oriented constraints need to be defined up front in order to fit within the broader portfolio investment profile and resource plans of the company. This means that over flow from one sprint to the next is significantly more problematic and must be avoided.

Tempered Agility seeks to address these concerns in a project-oriented application of agile which improves the cost and schedule predictability of the project while simultaneously allowing the team to enjoy the benefits of the agile approach. Further, it strikes a happy medium between the two waterfall and agile extremes by doing some of the definition, analysis and design work up front in the Definition Sprint and then completing the definition and design work in the iterations just prior to developing the capability. This provides an opportunity to improve the predictability of how long the project will take as well as how much it will cost. And, it allows the team to make intelligent design/cost trade-offs early in the project.

ROLES AND RESPONSIBILITIES IN TEMPERED AGILITY

This section describes the roles and responsibilities involved in Tempered Agility projects.

What is a Role?

A role may best be thought of as a part that is played in a project. Each role has responsibilities to be fulfilled by a person associated with the project. A person may fulfil more than one role on the project depending on the resources available and the skill sets of each of the team members.

Roles are often confused with positions. A position is the job that an individual holds in the company. One person fills one position. However, one person in their position may fulfil more than one role on more than one project. Conversely, several different people in different positions may all fulfil one role on a project.

The Roles in Tempered Agility

The following roles are involved in Tempered Agility projects. The first four roles (Product Owner, Project Manager, Business Analyst and Technical Architect) form the "Core Team". They are often assigned first so they can scope out the project and determine the right skill mix and number of team members that will be required to complete the project.

Product Owner

The Product Owner represents the business owner for the project and is also well-defined in the Scrum methodology [1]. His/her responsibilities revolve around coordinating efforts in the business community to support the project. This includes stakeholder identification, communication and coordination, business issue research and resolution, business rule definition, development of functional test scripts, etc.

Project Manager

The Project Manager coordinates efforts of business and technical team members to accomplish the project goals and objectives. In Tempered Agility, the project manager is also the Scrum Master as described in the Scrum methodology [1].

Business Analyst

The Business Analyst is responsible for defining the business requirements with the detail and precision needed for the developers. Ideally, the Business Analyst leads the use case analysis sessions. They may also actively participate in defining business rules and resolving business issues for the project.

Technical Architect

The technical architect defines the architecture for the solution and works throughout the project with all team members to preserve the conceptual integrity of the solution as the project progresses. They actively participate in use case analysis discussions to understand the business requirements and to ensure that the envisioned solution addresses those requirements. They play a key role in helping the team analyze design/cost trade-offs. In other words, the architect may be able to offer alternative design approaches that could be accomplished more inexpensively. In

addition, the technical architect has the primary responsibility to capture non-functional requirements that may emerge in use case discussions.

Developer

The developer takes part in use case analysis and associated design activities to represent the development perspective in those discussions and to elicit requirements from the business community. They also help to capture non-functional requirements as they arise. If developers are not yet staffed when the use case analysis takes place, the technical architect represents the developer's perspective.

Usability Analyst

The Usability Analyst, variously also referred to as User Experience Analyst, Information Architect, User Interface Designer or Interaction Designer provides expertise in designing a pleasing, intuitive and positive user experience in the design of the system solution. They will generally prepare wireframes and/or UI Prototypes for the team to review while discussing a use case.

Quality Assurance Analyst

The Quality Assurance (QA) Analyst participates in the use case analysis process to understand requirements as they emerge, to ensure that requirements are valid and testable, to formulate testing strategies and plans, to capture information for test scripts and to assist with developing the functional test scripts.

Roles and Responsibility Matrix

The following diagram illustrates the responsibilities of each of these roles in a somewhat typical Tempered Agility project.

"ORCA" Responsibility Matrix		Product Owner	Project Manager	Business Analyst	Technical Architect	Information Architect	Graphic Designer	Developer	System Engineer	Quality Assurance
WBS	Task Legend:	O = Owner		R = Review		C = Contribute		A = Approve		
1	Definition - Sprint 0									
1.1	Project Initiation									
1.1.1	Project Brief	A	O	C	C	C	R	R	R	C
1.1.2	Staffing Confirmation		O							
1.1.3	Workspace Preparation		O							
1.2	Solution Definition									
1.2.1	Initial Requirements Analysis									
1.2.1.1	Define Initial Use Cases	A	R	O	C	C		R		C
1.2.1.2	Elaborate Sprint 1 Use Cases	A	R	O	C	C		C		C
1.2.1.3	Define Non-Functional Requirements	A	R		O			C	C	R
1.2.2	Initial Design Analysis									
1.2.2.1	Define Initial Site Map	A	R	C	R	O	R	R		R
1.2.2.2	Define Initial Wireframes	A	R	C	C	O	C	R		R
1.2.2.3	Design for Sprint 1 Use Cases	A	R		R	O	C	R		R
1.2.3	Write Technical Specifications Document	A	R		O			C	C	R
1.2.4	Conduct Initial Issue Analysis	O	C	C	C	R				R
1.3	Project Plan Update	A	O	C	C	C	R	R	R	C
2	Development - Sprint 1									
2.1	Build Out Sprint 1 Use Cases	A		C	O			C		
2.2	Write Test Scripts for Sprint 1 Use Cases	A	R	C	C			R		O
2.3	Test/Fix Sprint 1 Use Cases	A		C	C	C	C	C		O
2.4	Elaborate Sprint 2 Use Cases	A	R	O	C	C		C		C
2.5	Design for Sprint 2 Use Cases	A	R		R	O	C	R		R
3	Development - Sprint 2									
3.1	Build Out Sprint 2 Use Cases	A		C	O			C		
3.2	Write Test Scripts for Sprint 2 Use Cases	A	R	C	C			R		O
3.3	Test/Fix Sprint 2 Use Cases	A		C	C	C	C	C		O
3.4	Elaborate Sprint 3 Use Cases	A	R	O	C	C		C		C
3.5	Design for Sprint 3 Use Cases	A	R		R	O	C	R		R
4	Development - Sprint 3									
4.1	Build Out Sprint 3 Use Cases	A		C	O			C		
4.2	Write Test Scripts for Sprint 3 Use Cases	A	R	C	C			R		O
4.3	Test/Fix Sprint 3 Use Cases	A		C	C	C	C	C		O
5	Deployment - Sprint 4									
5.1	Full System Integration Testing	A	C	C	C			C	C	O
5.2	User Acceptance Testing (UAT)	A	C	C	C			C	C	O
5.3	Solution Staging	A	C		O				C	
5.4	Training	A	R	O	C	C				
5.5	Migration to Production	A	R		O			C	C	C
5.6	Initial Evaluation	A	C		O					

Figure 7 – Responsibility Matrix

AUTHORIZATION

The Authorization stage includes the activities that precede the official start of the project. The specific activities will vary significantly from one organization to another. As such, this section describes fairly generic activities that typify this stage. Furthermore, the activities in this stage depend on your perspective as the client, dealing with internal projects, or as the vendor.

The View of the Client

From the client's perspective or from the perspective of a person working on internal projects, the authorization stage is generally characterized by some variation of a portfolio planning and procurement cycle.

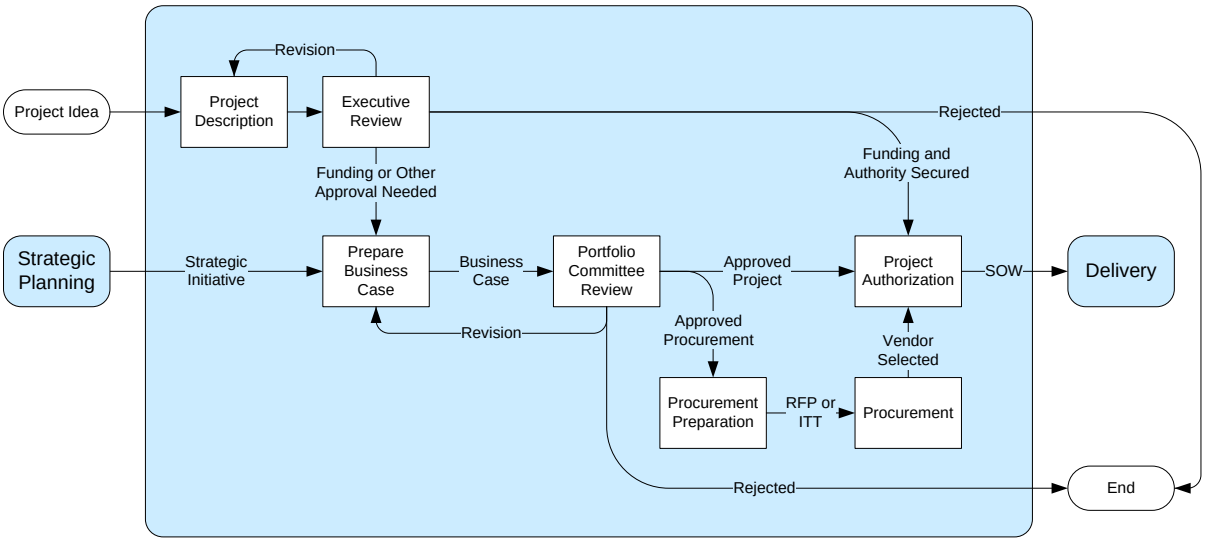


Figure 8 – Example of Authorization – Client or Internal Project View

- Opportunity Identification – Strategic planning activities, internal suggestions, service requests or suggestions from vendors may lead to the recognition of a potential project or initiative.
- Opportunity Definition – The opportunity is described to some degree of formality by an internal sponsor, perhaps as a business case, so that it can be evaluated by the appropriate decision-making body. In some cases, a vendor may assist in the development of the business case.
- Opportunity Evaluation – Someone with purchasing authority or a portfolio management committee of some sort evaluates the opportunity and makes a decision to whether or not to proceed with the opportunity and provide the necessary budget. Where procurements are required, the procurement processes lead to vendor selection.
- Project Authorization – Projects or initiatives receiving approval and budget are often authorized through some form of statement of work, project charter or similar document. In some cases, this may trigger a purchase order if products or services are to be purchased to complete the project.

The View of the Vendor

The vendor's perspective of the Authorization stage is better represented by some form of sales cycle.

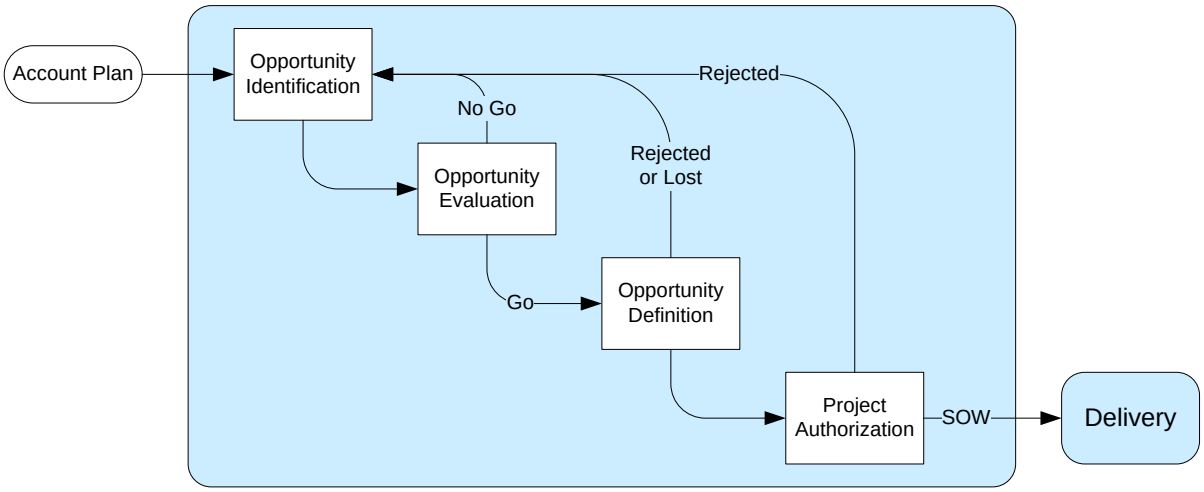


Figure 9 – Example of Authorization – Vendor View

- Opportunity Identification – Whether through lead generation activities or through prospecting efforts, the first step in securing project authorization is the identification of an opportunity to provide a service to a prospective or existing client.
- Opportunity Evaluation – Interviews and other research provide an evaluation (otherwise known as qualification) of the opportunity's validity and desirability from both the vendor's and the client's perspective. The vendor may support the client's evaluation in the form of a business case. Some opportunities may not pass this evaluation.
- Opportunity Definition – If the opportunity qualifies, both parties begin discussions to solidify their mutual expectations associated with the opportunity. Activities may include the definition of a "solution strategy" to clarify delivery approaches, workshops to identify candidate use cases, preparation of cost and schedule estimates and presentation of a proposal to the client describing the opportunity and its delivery. In some cases, the opportunity definition may occur in the context of the client's procurement process via Request for Proposal (RFP) or Invitation to Tender (ITT).
- Project Authorization – Assuming that both parties agree to the expectations and benefits of the opportunity, some sort of formal authorization takes place, often in the form of a service agreement or statement of work.

The viewpoints of the client, internal project and vendor all come together at project authorization where some form of document such as a statement of work or project charter is generated which authorizes the opportunity as a project and serves as the principal input to initiating the project in the Definition stage.

DEFINITION

The purpose of the Definition stage, sometimes referred to as "Sprint 0", is to develop a solid project plan and backlog of stories to be completed in the project. Getting to this point requires starting up the project and defining the desired solution in enough detail to support the creation of the plan and backlog. Solution definition requires some initial analysis of requirements, design, technology and business issues.

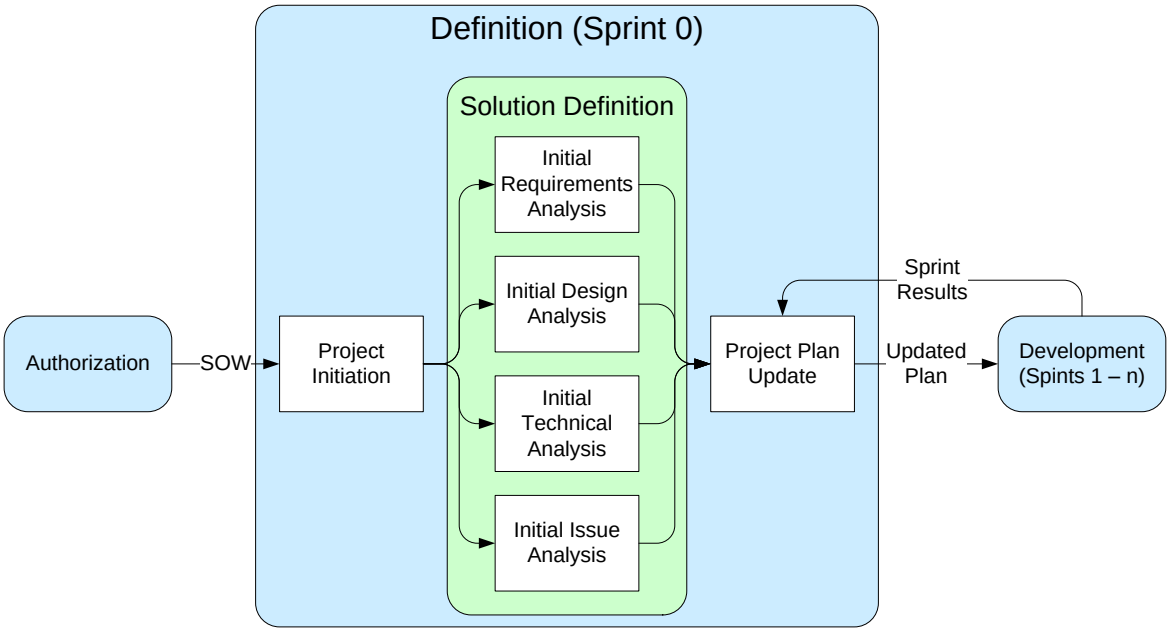


Figure 10 – The Definition Stage (aka Sprint 0)

Entry into the Definition stage assumes some form of project authorization, often consisting of a statement of work or authorization memo. When this authorization is secured, a project manager is assigned, if that has not already occurred. The PM then orchestrates the completion of the activities in the Definition stage. The tasks described below are recommended.

Project Initiation

The activities associated with Project Initiation are intended to get the project started on a good footing and get the team up and functioning with the needed workspace, software, etc. to be productive.

Project Brief

The Project Brief is a document that describes the project as well as its key goals, objectives, scope and plans in such a way that all necessary stakeholders can agree to about the project. Stakeholders include the client focal point, the associated client business and using community, the project team and the functional managers of the project team members. The Project Brief varies significantly from one type of project to another, e.g., technology vs. creative, but common sections will generally include:

- Background
- Objectives/deliverables/scope
- Project roles/ responsibilities/organizations
- Top Risks

- Constraints
- Communication plan
- Change management plan
- Approach/milestones/timeline
- Assumptions
- Approvers
- Glossary

Often, the first draft is put together by the core team, then reviewed and revised through stakeholder reviews and finally approved by the project sponsor(s).

It is important to do a project brief even though a statement of work may have already been written. This is because the process of putting the project brief together fosters stakeholder alignment and identifies previously undiscovered scope expectations among the stakeholder community early in the project. It also helps the project leaders better understand the broader stakeholder community in general and the politics that may lurk therein.

This is where tailoring occurs. In the process of putting together the project brief, and as the stakeholder community is better understood, the project leadership may want to tailor its project approach to better fit the situation and context within which the project will take place.

Staffing Confirmation

This activity confirms the availability of the necessary skilled analysts, designers, programmers, testers, etc. and assures commitment to the required levels of effort by the relevant functional managers or discipline leads. Sometimes, people will need to be hired or subcontracted as part of this activity as well.

Workspace Preparation and Accounting Initialization

In this activity, the team workspace is arranged, software and hardware needed by the project is arranged and the company systems that will keep track of time and expenses are initiated for this project.

Solution Definition

The activities comprising Solution Definition provide a description of the solution to be provided in the project that is clear and complete enough to build a solid project plan and backlog for the remainder of the project. In order to do support that objective, initial analysis must be conducted of functional requirements, external design and technical considerations. In addition, most of the business issues associated with the solution need to be researched and resolved to remove as much ambiguity as possible early in the project.

Tempered Agility recommends that the requirements and design analysis efforts take place in a series of workshops that include the core team of product owner, project manager, technical architect and business analyst plus a usability analyst to focus on design.

Initial Requirement Analysis

Depending on the nature of the project and the intended solution, initial requirements analysis will consist of a case-dependent mix of initial use cases, business rules and interviews of subject matter experts.

Tempered Agility strongly recommends the employment of use cases to define the users' functional requirements. If use cases are employed, this stage should see the completion of all candidate use cases down through the main success scenarios and a list of all expected alternate paths and exceptions. Elaboration of the alternate paths and exceptions is not accomplished at this point. Rather, that happens in the sprint just prior to the build-out of that use case so that it can include the developer that will be working on it and to ensure that the specifications are current.

Use Case Name: Make reservation
Narrative: The user of airline.com makes a reservation on line.
Context of Use: The user is proficient, sitting at their home computer with their browser open.
Primary Actor: airline.com user
Scope: airline.com
Stakeholders and Interests:

Stakeholder	Interest
airline.com user	Wants to make a reservation
Reservations	Wants to reduce number of calls

Precondition: The user has accessed the Internet
Success Post Condition: The user has successfully reserved a seat on a flight.
Minimal Guarantees: airline.com returns an error message
Trigger: The user has accessed the airline.com website home page to make a reservation.

Main Success Scenario (MSS):

1. The airline.com user enters trip request and elects to shop for a flight.
2. airline.com displays matching flights to user.
3. The airline.com user selects the desired flight and elects to continue.
4. airline.com requests passenger information.
5. The airline.com user enters the passenger information and elects to make a reservation without purchasing.
6. airline.com makes reservation in SABRE and displays a confirmation.
7. The airline.com user leaves airline.com

Extensions:

2a. There are no available matching flights

1. airline.com advises the user to shop again as no flights are available.
2. The user re-enters trip request (back to MSS Step 1)

Figure 11 – Fully-Dressed Use Case Example

The related paper, Use Case Analysis, should be consulted for more on use cases.

Initial Design Analysis

Also dependent on the nature of the project and the preferences of the customer and the project team, initial design analysis consists of a case-dependent mix of site maps, storyboards, UI prototypes, wireframes and flow diagrams.

Initial Technical Analysis

Also dependent on the nature of the project, initial technical analysis includes consideration of the following technical elements:

- Applications
- Databases
- Integration requirements and mechanisms (design)
- Data migration requirements and designs
- Infrastructure
- Non-functional requirements
- Programming languages and tools
- Programming practices, e.g. coding conventions, source code control, builds

Tempered Agility recommends that the initial technical analysis be documented in a Technical Specification document that is developed concurrently with the solution definition workshops by the technical architect. Doing so surfaces non-functional requirements early and ensures that the architecture is defined and agreed prior to beginning development. The Technical Specification is also one of the primary sources for identifying technical (non-functional) stories for the project backlog.

Business Issue Resolution

As the initial requirements and design analysis activity takes place, it is natural that questions and issues will arise with respect to business policies, procedures and processes. These issues should be logged in an Issues Log, researched and resolved by the business community, often by the product owner, and communication back to the project team to better inform the requirements and design analysis.

Project Plan Update

Once the solution definition activities have been completed, the team can develop a project plan with much more certainty than the initial plan in the Project Brief. This planning will need to include the following steps:

- Define the project backlog – User stories can be derived from the use cases and other stories can be derived from the Technical Specifications. Stories will also need to be included to elaborate the initial use cases into their full precision levels.
- Story definition – Each of the intended stories will need some degree of description as to the expected outcomes.
- Story estimation – The team will need to estimate the hours required to complete each story by role.
- Story prioritization – In conjunction with the product owner, the team will need to derive a prioritization of all of the stories based on value contributed.
- Sprint assignment – Each story is assigned to a sprint in the project plan. This enables the team to definitively determine how many development sprints will be needed to complete the project. Along the way, the team will need to define the length of the sprint and the number of people that will be working on the stories.
- Project plan update – The information determined above is incorporated into an updated project plan.
- Project plan review and approval – The resulting plan will need to be reviewed and approved by the relevant project stakeholders.

Please note also that these activities comprising the project plan update will need to be repeated at the conclusion of each sprint to accommodate scope changes, priority changes and lessons learned from the sprint just completed, including any stories that may not have been completed in the previous sprint. The plan update then becomes the input to Sprint Planning, the first activity of the next development sprint.

About Stories

A full discussion of stories is beyond this overview, but a few words may help. Stories are of two fundamental types: user stories and technical stories. User stories represent small, deployable and testable pieces of functionality. Technical stories (other than user stories, sometimes referred to as

non-functional stories) represent small, independent units of work generally associated with infrastructure or other technical aspects of implementing the solution such as data migration, acceptance testing, etc. They are generally derived from the Technical Specification.

User stories represent small, "vertical slices" of functionality that generally require integration of code across the presentation, logic and data layers of the system architecture as shown in the following figure.

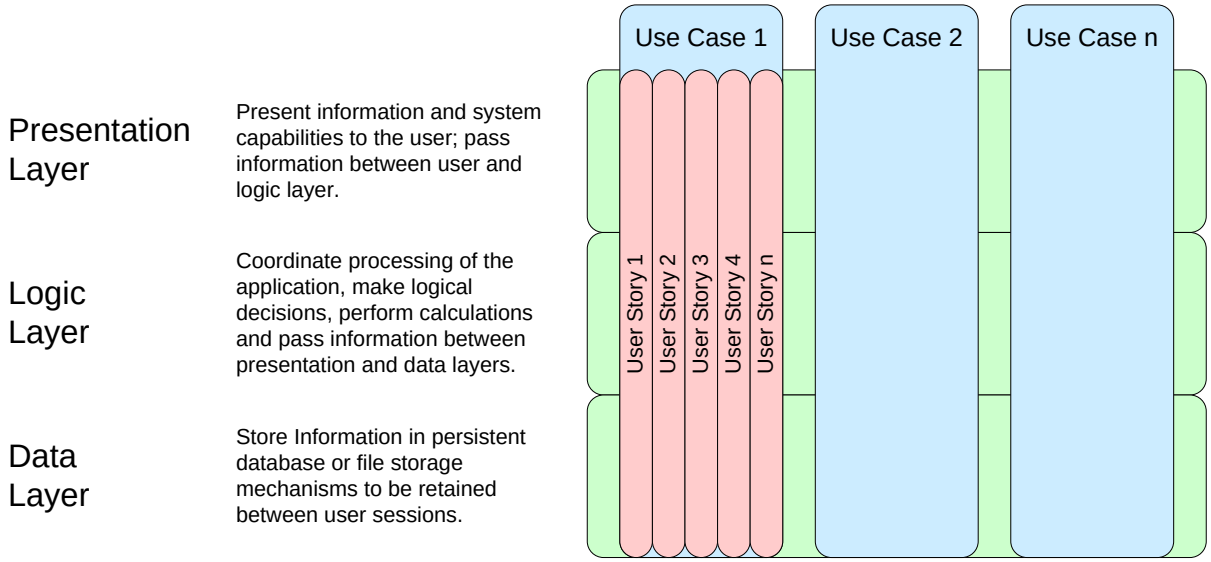


Figure 12 – User Stories

This is significant because full development of user stories leads to an early testing and debugging of the riskiest parts of the architecture. The concept of user stories is important for Tempered Agility because use cases may be larger than can be developed within one sprint as illustrated in the following diagram.

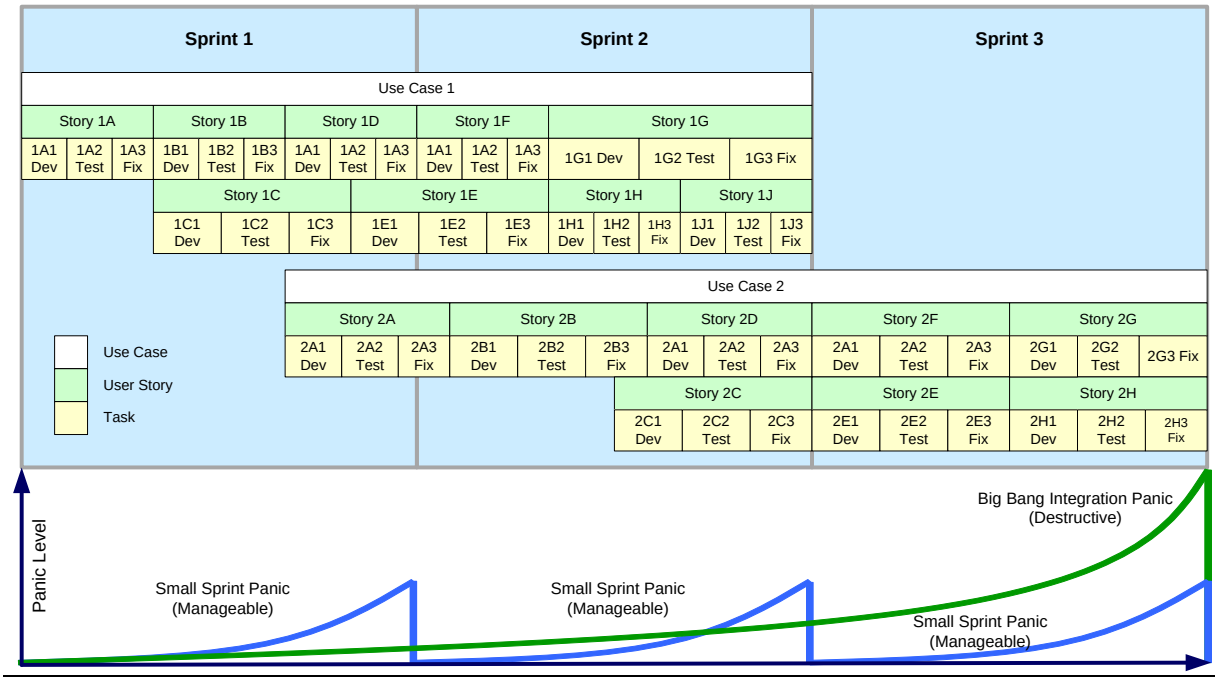


Figure 13 – The Importance of User Stories

- Avoiding the big-bang integration panic is an important goal for which the team should strive. Accomplishing that goal requires discipline to test and remediate functionality in the same sprint in which is has been developed. It also requires decomposing the sprint functionality into small, deployable and testable units of work.
- Breaking the work down into user stories accomplishes that objective. Many use cases are too big to fit in a sprint and tasks can neither be deployed nor tested. However, user stories are small enough to fit in a sprint and can be deployed and tested.
- User stories are the units of work that enable potentially shippable product units, make smaller releases practical, and help you avoid big bang integrations.
- Coding, deployment and testing are all going on concurrently with story-driven development, so people need to be available for it.
- Without deployment and testing within the sprint, you are stuck with big bang integrations at the end.

By dividing the use case into multiple user stories, the complete "done-list" for the subset of stories selected for the sprint can be completed within the sprint. For example:

- Develop code
- Develop unit test scripts
- Unit test and fix
- Code review and fix
- Deploy to test environment
- Develop functional test script
- Functional test and fix

The do-list captures the specification and development life cycle for each story. It is the completion of the done-list within the sprint for each selected user story that ensures that the resultant product increment from the sprint will be "potentially shippable" and minimises the big-bang integration phenomenon that so often characterises software development.

Project Planning

The Gantt chart portrayed below illustrates a representative Tempered Agility project plan.

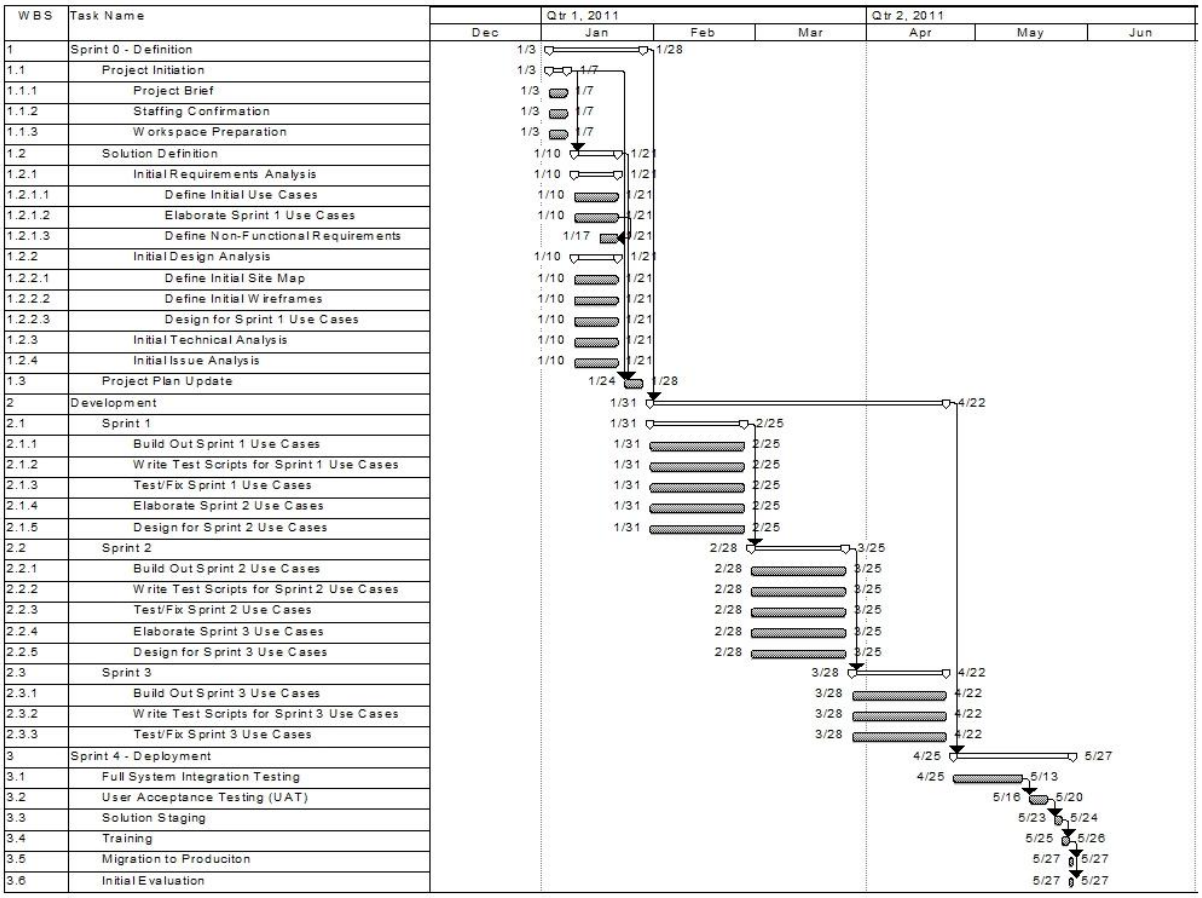


Figure 13 – Tempered Agility Project Plan

During the Definition Sprint (Sprint 0), the initial use cases for the entire project are defined through Precision Level 2 (Sunny Day Scenario or Main Success Scenario). In addition, those use cases that are planned for build out in Sprint 1 are fully elaborated and the design work for those use cases is completed in Sprint 0.

During Sprint 1, the team builds out the use cases (or the relevant user stories) targeted for Sprint 1 and elaborates the use cases planned for Sprint 2. This approach, referred to as "pipelining", is followed for Sprint 2 and 3 as well. Design work is completed concurrently with the elaboration of the use cases as tasks comprising the associated user stories. In larger projects the definition work may be accomplished by a business scrum team and the build out by a technical scrum team.

Sprint 0	Sprint 1	Sprint 2	Sprint 3	Sprint 4	Sprint 5
UC1 Elaboration	UC1 Build-out	UC1 Test/Fix			
UC2 Elaboration	UC2 Build-out	UC2 Test/Fix			
UC3 MSS	UC3 Elaboration	UC3 Build-out	UC3 Test/Fix		
UC4 MSS	UC4 Elaboration	UC4 Build-out	UC4 Test/Fix		
UC5 MSS		UC5 Elaboration	UC5 Build-out	UC5 Test/Fix	
UC6 MSS			UC6 Elaboration	UC6 Build-out	UC6 Test/Fix
UC7 MSS			UC7 Elaboration	UC7 Build-out	UC7 Test/Fix
			Test/Fix Overflow	Test/Fix Overflow	Test/Fix Overflow

Figure 13 – Pipelining

Carrying the concept of pipelining to the other activities in a project, the project activities take on the look shown in the following diagram.

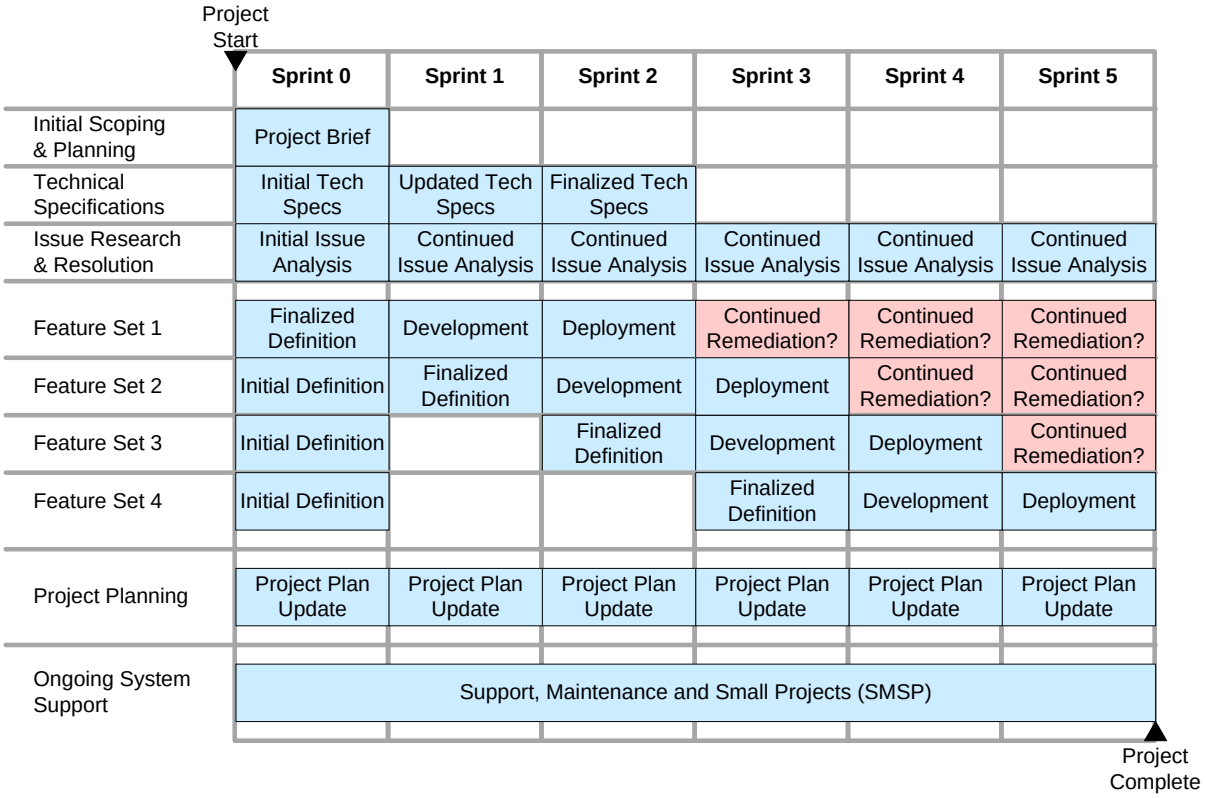


Figure 13 – Pipelining a Tempered Agility Project

Some things to be aware of in this approach include the following. It may look good on paper but it is very difficult to achieve in practice.

- Beware remediation growth – it drains development availability. Work the done list!
- The support load grows step-wise over time, further draining development availability.
- The cycle could repeat over the system's whole life by simply adding more feature sets.

Definition of terms:

- Initial, Updated and Finalised Tech Specs – stages in the creation of the Technical Specification document.
- Initial Definition – A list of candidate uses, a narrative description of each use case, the definition of the happy path for each use case and list of the expected alternate paths and exceptions for each use case. Also includes page mock-ups or wireframes for most pages and flow diagrams to illustrate the flow of events through the pages or screens of the system.
- Finalised Definition – fully dressed use cases, finalised page mock-ups or wireframes for every page and finalised flow diagrams for each use case.
- Development – completion of each story as per the done list: code complete, code reviewed and remediated, unit tested and remediated, deployed to QA and QA tested and remediated. Also includes completion of non-user stories assigned to the sprint.

- Deployment – Includes integration testing, UAT, user training, launch preparation, launch, launch assessment
- Project plan update – project backlog, story definition, story estimation, story prioritization, sprint assignment, plan revision and update
- Support, Maintenance and Small Projects – BAU support requirements

Prioritizing Use Cases and Stories

The Project Plan Update task in the Definition Sprint includes the prioritization of use cases, decomposition of the use cases into user stories, completion of the project backlog with prioritized stories and iteration planning that identifies the stories to be completed in each planned sprint. Prioritization of both use cases and user stories should be based on value as illustrated below.

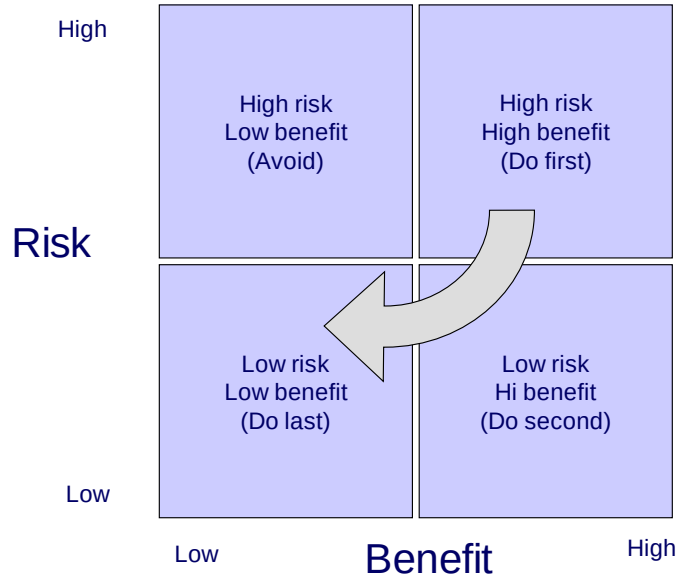


Figure 23: Value-Driven Prioritization

Value is a function of both risk reduction and benefits realization. This implies that those use cases which most greatly reduce project risk and which provide high benefits realization should be defined and built out before use cases of lesser value. In practical terms, risk reduction in a web application or custom development project often amounts to proving out the reference architecture of the application and/or the system integration requirements. This means that the first use cases to be built out would be those that entail the riskiest parts of the architecture.

Building out those initial use cases will probably therefore entail building out a proportionately large amount of the lower logic and data layers of the application. As such, those first use cases will take longer and require more resources to complete than the use cases that are built out later in the project. Project plans should take that into account.

Finally, in large and complex projects, it may make sense to define and build out the initial set of use cases in a separate Proof of Concept (POC) project. This affords the additional benefit of applying the learning gained from the POC in the planning for the project which builds out the remainder of the use cases.

DEVELOPMENT

The development stage is comprised of one-to-many development sprints. Each sprint develops a portion of the total solution envisioned by the project. This is referred to as a potentially shippable product increment, meaning that code that is developed in the sprint should also have been functionally tested and bugs fixed all within the same sprint. The number of the sprints and the length of each sprint are decided during the project plan update activity. Each sprint should be 5 – 20 workdays long. Tempered Agility recommends 20 days to enable full testing and remediation of stories developed in the sprint.

The activities that make up a development sprint are illustrated and described below.

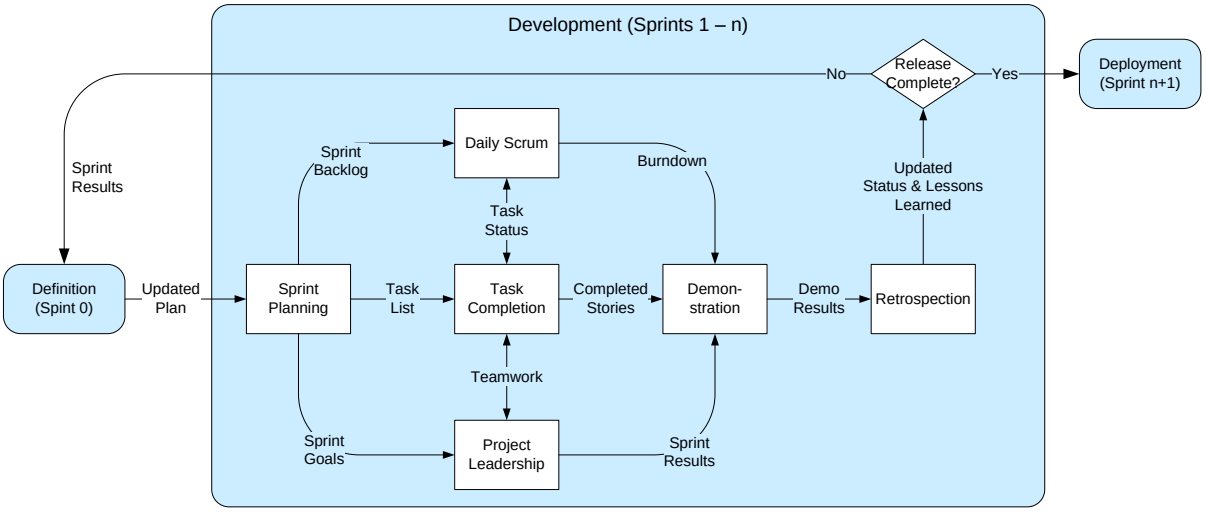


Figure 14 – The Development Sprints

Sprint Planning

Sprint Planning is an intensive, team planning session that takes place on the first day of the sprint. The objective of sprint planning is to define the sprint goals and all of the tasks that will need to be completed in order to complete all of the stories assigned to this sprint that will achieve the goals. Each story should be assigned a story owner who meets with the product owner and other team members to list the tasks needed to complete the story. Several people may be involved in this as task owners. At a minimum, each task should have a task owner, an estimate of the hours needed to complete the task and clearly defined exit criteria so all team members can know what each task produces and when it is completed.

Defining what "done" means for each story is sometimes a challenge. User stories, which are small, deployable and testable units of functionality, need to not only be specified and developed. They also need to be deployed to a test environment, independently tested and remediated (bugs fixed) all within the sprint. It may help for the team to establish a standard "done list" for the user stories that reflects the life cycle of that story. Each step in the done list becomes a task for that story. An example of a done list is:

- Finalize specifications
- Develop unit test scripts
- Code complete
- Deploy to test environment
- Develop functional test scripts
- Complete functional testing

- Unit test and fix
- Conduct code review and fix defects
- Resolve all defects

Many teams make the mistake of not enforcing this discipline with the result that work tends to pile up in the later sprints and throwing the costs and schedule targets into overrun.

Sprint Planning is complete when all tasks have been defined in the sprint backlog and the scrum manager (or other tool) is set up for running the daily scrums.

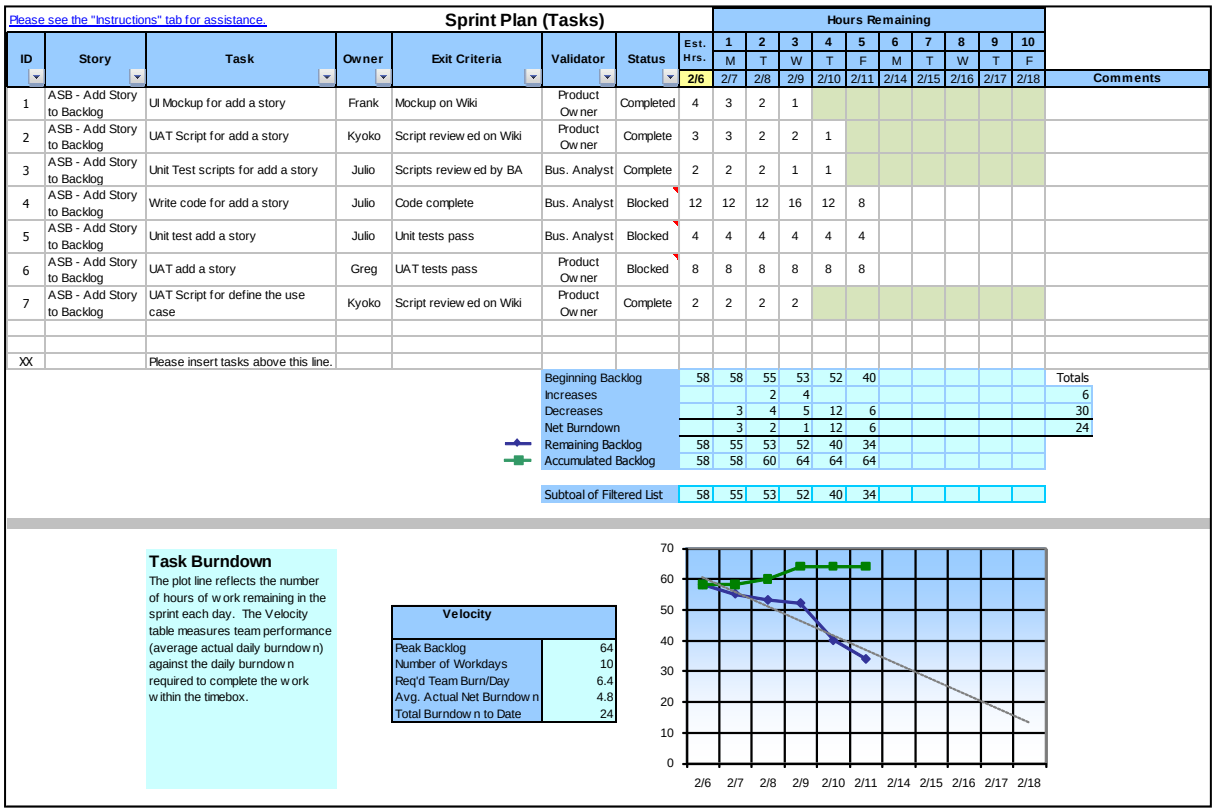
Daily Scrum

The daily scrum is a short meeting between 15 and 30 minutes long that occurs each day. It is run by the scrum master (generally the project manager in Tempered Agility project) and attended by everyone on the team. Each member of the team provides a quick status update and quickly coordinates with team mates. Generally, each person answers s few semi-standard questions:

- What have you worked on since last scrum?
- Any status updates on any of your assigned tasks?
- How many hours do you have left on each of your assigned tasks?
- What are you going to be working on today?
- Are there impediments that keep you from making progress?

The first three are classic Scrum questions. The last two are not.

In Tempered Agility projects, the team generally uses the Scrum Manager spreadsheet to track sprint status in the scrum.



Please see the "Instructions" tab for assistance.

Sprint Plan (Tasks)

ID	Story	Task	Owner	Exit Criteria	Validator	Status	Est. Hrs.	Hours Remaining 1 M 2/6	2 T 2/7	3 W 2/8	4 T 2/9	5 F 2/10	6 M 2/11	7 T 2/14	8 W 2/15	9 T 2/16	10 F 2/17	2/18	Comments
1	ASB - Add Story to Backlog	UI Mockup for add a story	Frank	Mockup on Wiki	Product Owner	Completed	4	3	2	1									
2	ASB - Add Story to Backlog	UAT Script for add a story	Kyoko	Script reviewed on Wiki	Product Owner	Complete	3	3	2	2	1								
3	ASB - Add Story to Backlog	Unit Test scripts for add a story	Julio	Scripts reviewed by BA	Bus. Analyst	Complete	2	2	2	1	1								
4	ASB - Add Story to Backlog	Write code for add a story	Julio	Code complete	Bus. Analyst	Blocked	12	12	12	16	12	8							
5	ASB - Add Story to Backlog	Unit test add a story	Julio	Unit tests pass	Bus. Analyst	Blocked	4	4	4	4	4	4							
6	ASB - Add Story to Backlog	UAT add a story	Greg	UAT tests pass	Product Owner	Blocked	8	8	8	8	8	8							
7	ASB - Add Story to Backlog	UAT Script for define the use case	Kyoko	Script reviewed on Wiki	Product Owner	Complete	2	2	2	2									
XX		Please insert tasks above this line.																	
						Beginning Backlog	58	58	55	53	52	40							Totals
						Increases			2	4									6
						Decreases		3	4	5	12	6							30
						Net Burndown		3	2	1	12	6							24
						Remaining Backlog	58	55	53	52	40	34							
						Accumulated Backlog	58	58	60	64	64	64							
						Subtotal of Filtered List	58	55	53	52	40	34							

Task Burndown

The plot line reflects the number of hours of work remaining in the sprint each day. The Velocity table measures team performance (average actual daily burndown) against the daily burndown required to complete the work within the timebox.

Velocity	
Peak Backlog	64
Number of Workdays	10
Req'd Team Burn/Day	6.4
Avg. Actual Net Burndown	4.8
Total Burndown to Date	24

Figure15 – The Scrum Manager Workbook

The scrum master keeps the meeting short by asking members to meet after scrum when conversations take too long. Everyone on the team that has tasks assigned must attend the scrum either in person or by phone unless previously agreed by the scrum master.

Task Completion

In the Sprint Planning session, each story targeted for the current sprint was broken down into the tasks needed to complete the story. Each day, the team members work together to make progress on their assigned tasks. Tasks generally include such activities as:

- Use case elaboration
- Development of graphic elements, wireframes, code, test scripts, documentation, etc.
- Deployment of code to test environments
- Testing
- Fixing bugs

Tasks status is updated in the daily scrum.

Project Leadership

Project leadership is generally provided by the project manager on Tempered Agility projects. The term, project leadership, includes project management but also implies that the PM must be a leader for the team as well. On Tempered Agility projects, leadership is provided through facilitation, coaching, mentoring, guiding, nurturing, shepherding and serving, but not commanding.

Project Leadership includes several types of activities as discussed later in this paper in the section entitled “The Activities and Artifacts of Tempered Agility Projects”.

Demonstration

At the conclusion of the sprint, the team reviews its progress with the product owner and other selected members of the stakeholder community. Sometimes, this includes reviews of documents, prototypes or other artifacts and sometimes it includes demonstration of the working software that was developed, tested and remediated in the sprint. Formality should be kept to a minimum in these demonstrations to reduce the amount of preparation time required.

Retrospection

After the team has completed its demonstration of the sprint results, a retrospection session is held. The goal of the retrospection is for the team to review its progress in the sprint just completed and to discuss ways in which it can improve its performance in future sprints. The retrospection is conducted by the project manager and includes all team members that took part in the sprint.

If this is the last sprint and the project is code complete, the project leaves the development stage and moves into the Deployment sprint.

DEPLOYMENT

The Deployment stage moves the code complete system through the activities required to getting the system launched into its intended production environment.

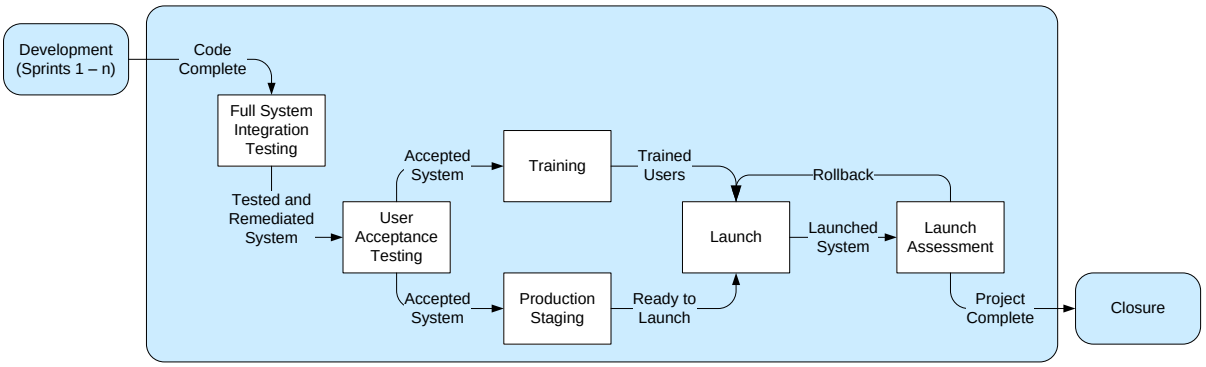


Figure 16 – Deployment (Sprint n+1)

This stage is comprised of several activities designed to assure the client that the system and the using organization are ready for production usage. Sometimes, additional activities may be required for data migration or extended training for multiple using communities and support functions.

Getting complex software to a state of readiness for production is a painful process during which the outcome may seem uncertain. The main thing that carries it through to success is a healthy sense of teamwork and collaboration

Integration Testing

Integration testing, which takes place in the QA or Alpha environment, is intended to get all of the major defects in the system cleared up or otherwise resolved. This activity may also be known as Full System Testing or Functional Testing. Generally, test scripts based on use cases should be used in this activity to ensure that the system functions as intended and it fit for its intended purpose. Other types of testing may also take place at this point depending on the needs of the project. Examples might include performance testing or regression testing, which ensures that the functionality in the previous release of the system still functions properly.

During the Integration Testing activity, the combined producer/vendor and client/customer team works in close cooperation, with daily bug-triage sessions, frequent cross-team communications and often with deployments made daily (mostly incremental deployments) as defects are corrected in order to confirm their resolution.

Integration testing may take two – three weeks of time, depending on the complexity of the system and the amount of testing deemed appropriate by the client and project team. Generally speaking, this stage of testing is not considered complete until all major defects have been resolved.

User Acceptance Testing

User Acceptance Testing (UAT) takes place in the UAT or Beta environment. It is intended to provide assurances to the using community that the system is ready for use. While some new defects may arise in UAT, there should not be many as they should have been identified and resolved in

Integration Testing. Once again, test scripts based on user-approved use cases are often used as the basis of acceptance.

Production Staging

In Production Staging, the system remains implemented in the UAT or Beta environment, but is connected to production data stores and external links to ensure proper functioning in the intended production environment. The acceptance test scripts may once again be processed to ensure proper functioning.

In addition, the production staging activities should include detained deployment planning and rollback procedures for the upcoming deployment to production.

Training

In parallel with production staging, the intended users and supporters of the system are trained in its usage or new features in the case of a system enhancement project. Separate training may also be provided to the people that will support the system in varying capacities, e.g., user support, IT infrastructure and data administration.

Launch

After testing has been successfully completed and the training has been conducted, the system is deployed to the intended production environment.

Launch Assessment

Immediately following deployment to production, the team conducts a set of tests to make sure the system is functioning properly in production. If problems cannot be rectified in a timely manner, the system may need to be rolled back to its previous production state so that users can continue to be productive while the problems get resolved.

A successful assessment of the production deployment completes the project, save for the activities required to provide on orderly closure of the project in the Closure stage.

CLOSURE

The closure stage provides an orderly shut-down and administrative closure of the project as illustrated in Figure x.

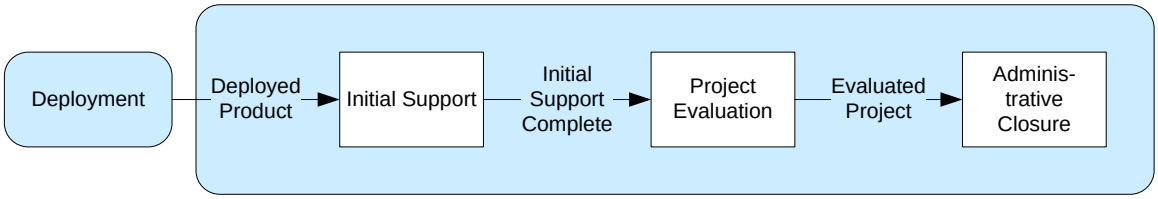


Figure 17 – Project Closure

Project Closure is comprised of three main sets of activities: Initial Support, Project Evaluation and Administrative Closure.

Initial Support

Once the system is launched into production, there is often an initial support period. During this initial support period, the project team provides support to the using community, answering questions and providing spot instruction as needed to assist the using community in its initial adoption of the new system. Often, support requests need to be prioritized in triage sessions and the resolution of these initial support issues are often tracked through to resolution just as in the testing activities.

Project Evaluation

Project evaluation may include interviews of the users or client personnel to measure their satisfaction level. Key metrics are also gathered at this point to help in future estimation efforts. Finally, the project team may conduct a final retrospection or lessons learned session to provide input to future project teams.

Administrative Closure

The very final set of activities provide for orderly closure of the project. This may include contract closure activities as specified in service agreements and warranties provided in those agreements. Project artifacts, code and documentation will be archived in safe storage, servers will be returned to be used on other projects, the team will be disbanded and tracking systems turned off.

THE ACTIVITIES AND ARTIFACTS OF TEMPERED AGILITY PROJECTS

This section describes the activities and artifacts associated with Tempered Agility projects. First, the development activities and artifacts are described and then the project management activities and artifacts are described.

Development Activities and Artifacts

The development activities and artifacts are those related to the definition, design, development and deployment of the solution that the project is intended to deliver to the client.

Development Activities

Earlier sections in this paper described the development activities that take place in the definition sprint, the development sprints and in the deployment sprint. The following diagram provides a one-page summary of all of these activities. It connects the primary stages into one comprehensive project flow and provides a quick reference of the steps involved in each of the activities.

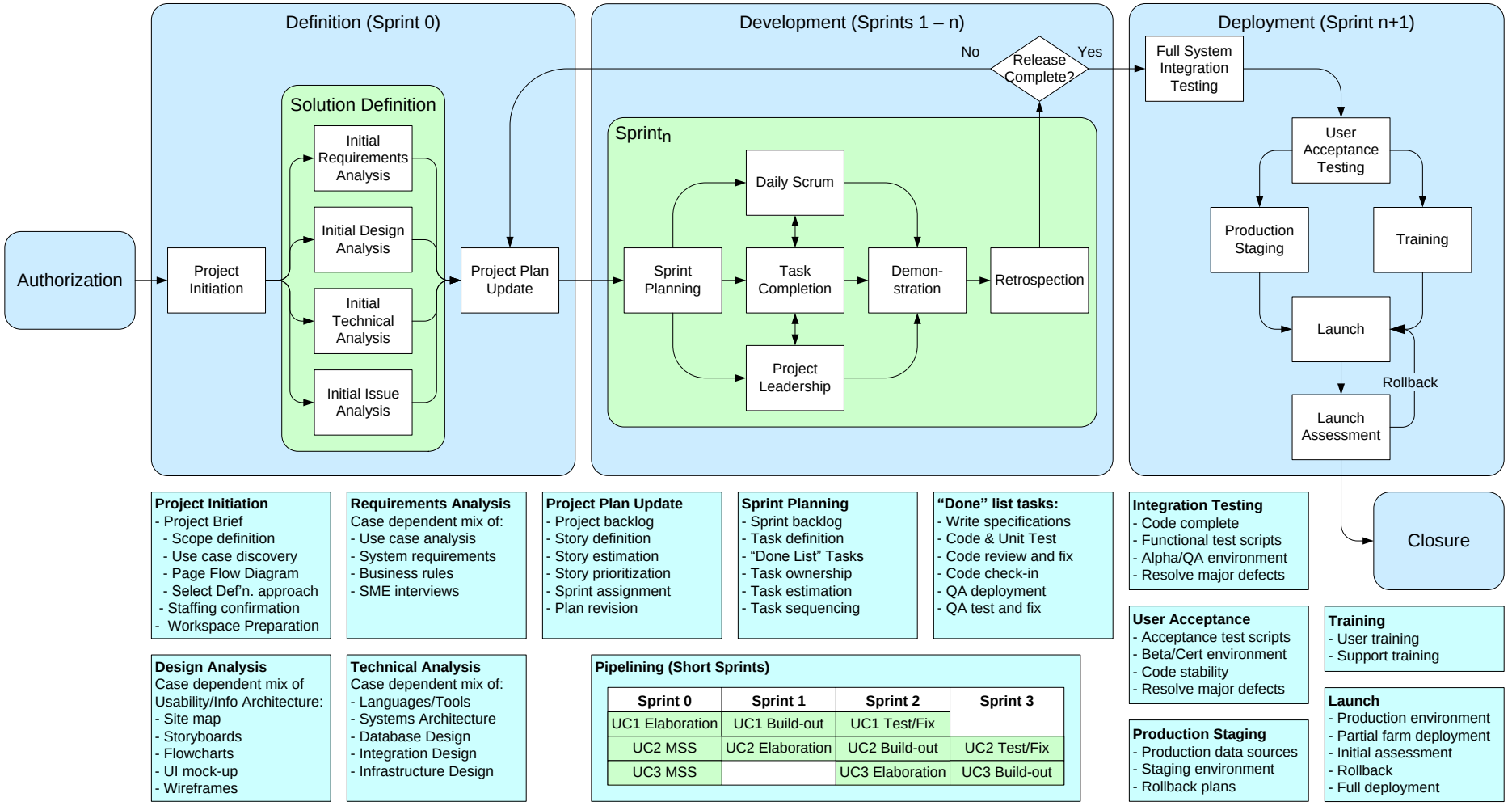


Figure 18 – Development Activities

Development Artifacts

The diagram shown below identifies the major development artifacts produced in a Tempered Agility project by the development activities, how they inform each other, build to a solution and which Sprint of the project produces them.

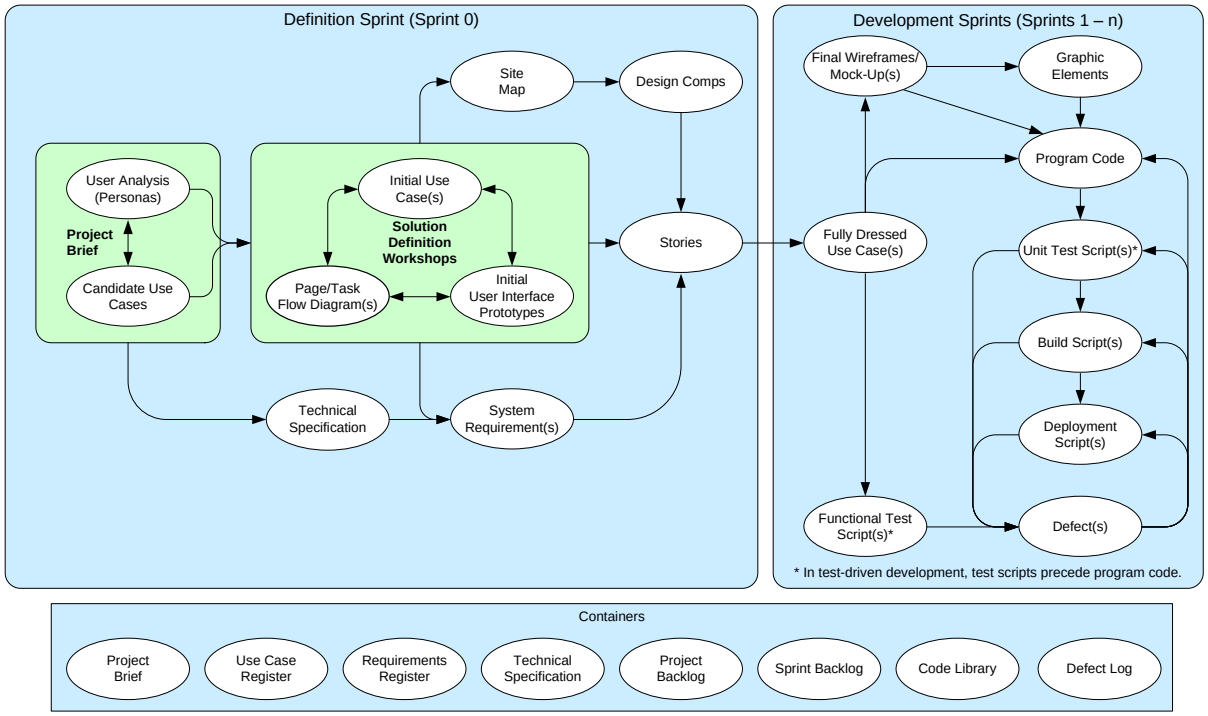


Figure 19 – Tempered Agility Development Artifacts

Each artefact is briefly described in the following paragraphs.

- User Analysis – User analysis includes a description of each category of user that will interact with the system. Sometimes, user analysis takes the form of a set of personas for each of the types of users. Doing so helps team members identify actors and to visualize each type of user in a meaningful way during design and development activities.
- Candidate Use Cases – Candidates use cases include the actors, the goals of each of those actors and a narrative paragraph describing the process through which the actor achieves the goal by interacting with the system under discussion. They are called candidate use cases because further analysis is probably going to change the list by merging splitting, adding and deleting use cases.
- Initial Use Cases – Initial use cases expand upon the candidate use case information by including definition of stakeholders, pre-conditions, post-conditions and the main success scenario. Initial use cases should also include a list of expected alternate paths and exceptions without fully dressing them out.
- Page/Task Flow Diagrams – Flow diagrams define the sequence of pages or screens through which the user navigates to achieve the goal of a use case.
- Initial User Interface Prototypes – Defining the solution will also generally require some design work into the usability of the system by preparing one or more flavours of user interface prototypes such as wireframes or page mock-ups.
- Site Map – A site map provides a hierarchical organization of the pages that comprise a website.

- Design Comps – Design comps provide several alternate design approaches that can be compared in the process of designing the look and feel of the site.
- Technical Specifications – The Technical Specifications document documents the technical architecture approaches and designs involved in the project such as applications, databases, systems integration mechanisms, data migration requirements, infrastructure requirements and programming tools and practices.
- System Requirements – System requirements may reflect user requirements that are not tied to a single use case or non-functional requirements that relate to the technical architecture.
- Stories – Stories are small, independent units of work that are derived from use cases or technical specifications.
- Fully Dressed Use Cases – Building on the Initial Use Cases, the Fully Dressed Use Cases include full definition of all alternate paths, exceptions and other supplementary information.
- Final Wireframes/Mock-Ups – Final page designs may be expressed as wireframes or page mock-ups.
- Graphic Elements – Graphic elements are the .JPG and .GIF files that are implemented in the web pages to produce the desired look and feel of the website.
- Program Code – Program Code refers to the collection of programs that comprise the system. Taken loosely, it may also include the configurations and customizations associated with vendor packages.
- Unit Test Scripts – Unit test scripts are written by the developer to test the smallest units of code. Sometimes, they are organized into a hierarchical test harness that can effectively regression test an entire system if implemented properly.
- Build Scripts – Build scripts are small programs that automate the aggregation of all system components into deployment packages.
- Deployment Scripts – Deployment scripts are small programs that automate moving the deployment packages to the desired computing environment.
- Functional Test Scripts – Functional test scripts are generally written one for each use case to test the functional performance of the resulting solution.
- Defects – Defects are bugs or other problems encountered during testing.
- Development Containers
    - Project Brief – The project brief is a document which defines the goals and objectives of the project and lists the users and candidate use cases.
    - Use Case Register – The use case register lists all use cases and key information about them.
    - Requirements Register – The requirements register lists all system requirements.
    - Technical Specifications – The technical specifications document contains technical architecture and design details.
    - Project Backlog – The project backlog lists all of the stories to be accomplished in the project and assigns them to sprints.
    - Sprint Backlog – The sprint backlog lists all of the tasks to be completed in order to complete all of the stories assigned to the sprint.
    - Code Library – The code library manages all of the code in the solution. This is generally some form of source code control repository such as Team Foundation Server (TFS) or CVS.
    - Defect Log – The defect log lists all know defects and their status. It is also common to use some form of defect tracking tool such as Bugzilla, Mantis or TFS.

Relating Development Activities and Artifacts

The following tables lists all of the development activities and the artifacts that are produced by their accomplishment.

Activity	Artifacts Produced
Authorization	Proposal Consulting or Master Services Agreement Statement of Work Candidate Use Cases (hopefully)
Definition Sprint	
Project Initiation	Project Brief User Analysis Event Analysis Candidate Use Cases
Initial Requirements Analysis	Initial Use Cases Business rules System Requirements (User Requirements) Requirements Register
Initial Design Analysis	Initial User Interface Prototypes Wireframes (possibly) Page Mock-ups (possibly) Page/Flow Diagrams Site Map Design Comps
Initial Technical Analysis	Technical Specifications document System Requirements (non-functional) Requirements Register
Initial Issue Analysis	Business Issues and their resolution Business rules Issue Log
Project Plan Update	Stories defined, estimated and prioritized Project Backlog Sprint assignments
Development Sprints	
Sprint Planning	Sprint Backlog Task defined, estimated and prioritized
Task Completion	Fully Dressed Use Cases Final Wireframes or Mock-Ups Graphic Elements Program Code Code Library Unit Test Scripts Build Scripts Deployment Scripts Functional Test Scripts Defects (and hopefully resolved) Defect Log Training Materials
Demonstration	
Retrospection	Lessons Learned
Deployment Sprint	

Activity	Artifacts Produced
Integration Testing	Program Code Code Library Defects and their resolution Defect Log
User Acceptance Testing	Program Code Code Library Defects and their resolution Defect Log
Production Staging	Deployment Scripts Rollback Plans and Scripts
Training	Trained users and administrators
Launch	Live System
Launch Assessment	Assessment results
Closure	Lessons Learned Client Satisfaction Survey results Defects (and their resolution) Closed contracts

Figure 20 – Relating Development Activities and Artifacts

Project Management Activities and Artifacts

Project Management is about managing the delivery of the solution to the client in a Tempered Agility project. It includes two major sets of activities: Project Planning and Project Leadership.

Project Management Activities

Project Management activities are shown in the following diagram. Project Management takes place throughout the project in parallel with the development activities being accomplished by the team.

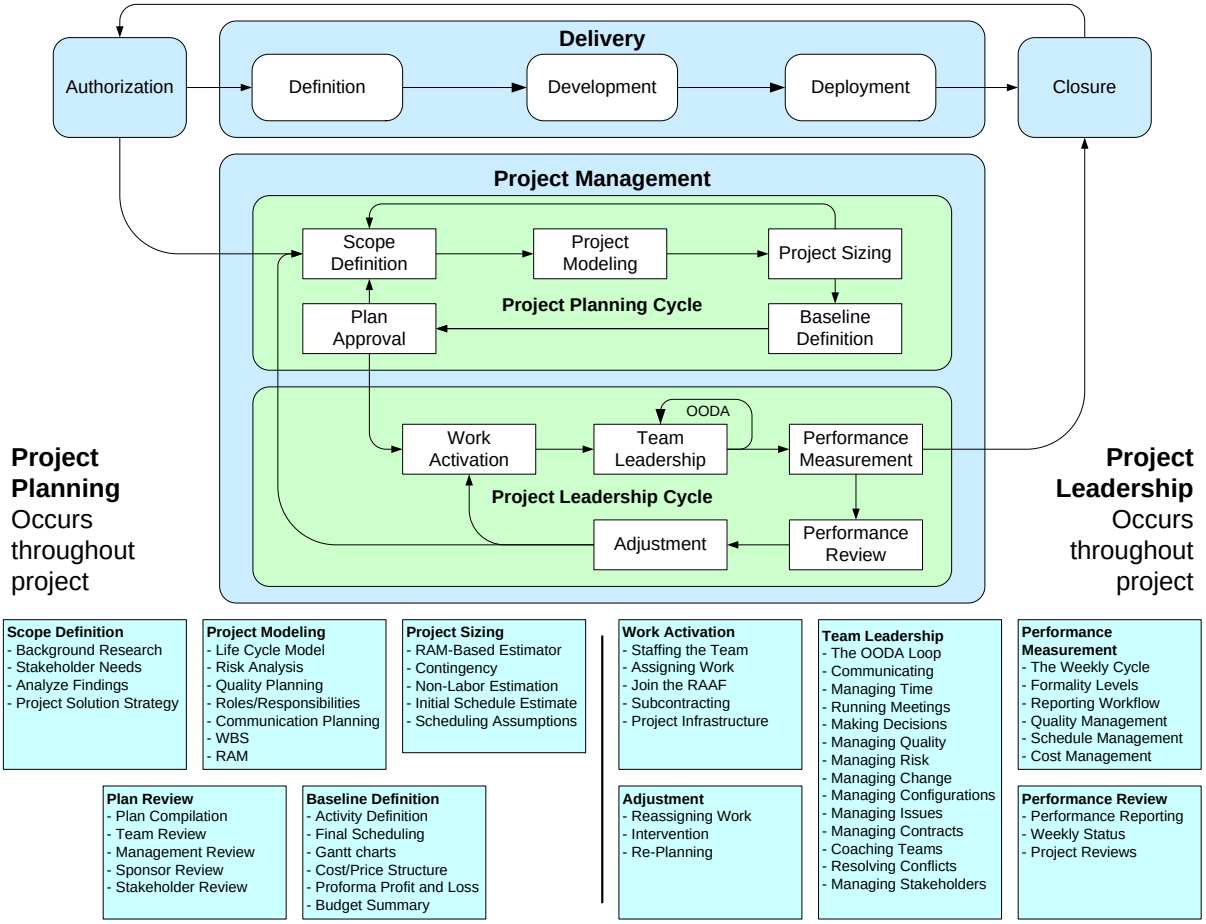


Figure 21 – Project Management Activities

Project Planning activities occur throughout the project. The activities shown above reflect an iterative approach to planning to reflect the fact that projects are planned not just once at the beginning but at several times in the life of a project. The most effort is generally spent up front in the Authorization stage and in the Project Initiation activity. However, the project plan is updated at least at the completion of the definition sprint and at the end of each development sprint to reflect new understanding that has recently been gained or to reflect changes in priorities. It is further considered good practice to update project plans at least once a week to reflect the most recent work efforts and status. The project planning cycle is made up of the following activities.

- Scope Definition – Scope definition addresses the need to define with clarity what will be delivered at the end of the project and the work that will be undertaken to develop those results.

- Project Modelling – Project modelling provides a model for the way the project will be managed and accomplished. It includes such considerations as selecting the project lifecycle model, defining the Work Breakdown Structure (WBS), defining the roles, responsibilities and project governance model, conducting a risk analysis and defining the communication plans.

- Project Sizing – Project sizing means developing an estimate of the project cost and schedule based on the scope definition and the project model.

- Baseline Definition – Baseline definition refers to the definitization of project plans after scope, cost and schedule have been brought into balance.

- Plan Approval – Finally, plans are reviewed and approved with the project team, project sponsor(s), management and client.

Project Leadership takes places every day as the Project Manager provides leadership for the team on Tempered Agility projects. The term, project leadership, includes project management but also implies that the PM must be a leader for the team as well. On Tempered Agility projects, leadership is provided through facilitation, coaching, mentoring, guiding, nurturing, shepherding and serving, but not commanding. Project Leadership includes several types of activities as discussed below.

- Work Activation – This set of activities includes ensuring that the tasks are assigned appropriately and working with the team to re-assign them if needed. It includes helping the team recognize its responsibilities, authorities and accountabilities associated with the project as well as each task, artifact and deliverable.

- Team Leadership – Effective team leadership is an art form that takes years to master. It can easily make or break a project team's effectiveness depending on how it is handled. Note that the team in this case includes the product owner and key stakeholders. The team leader helps the team accomplish it goals and objectives by supporting it in several ways:

    - Communicating effectively and fostering open communication
    - Helping the team and the members of the team manage their time more effectively
    - Running effective meetings
    - Helping the team make better group decisions
    - Helping the team manage quality
    - Helping the team understand project risks and how to mitigate them
    - Helping the team manage change
    - Ensuring proper configuration controls are followed
    - Managing issues and escalating them as needed
    - Helping the team understand and meet its contractual obligations
    - Resolving conflicts in a constructive manner
    - Managing stakeholder communications, decisions and support

- Performance Measurement – Performance Measurement means tracking progress against the plans for the project. This includes task completion, burn down, schedule progress, timekeeping, budget tracking and quality management.

- Performance Review – Performance Review includes reporting on project progress with stakeholders in weekly status reports as well as discussing and resolving issues, problems, questions and concerns with the stakeholder community.

- Adjustment – When performance measurements or stakeholder reviews call for it, work adjustment includes such activities as re-assigning work among the team, adding or removing team members, intervening in dysfunctional team behaviors and re-planning the remaining work on the project.

Project Management Artifacts

The diagram below portrays the project management artifacts.

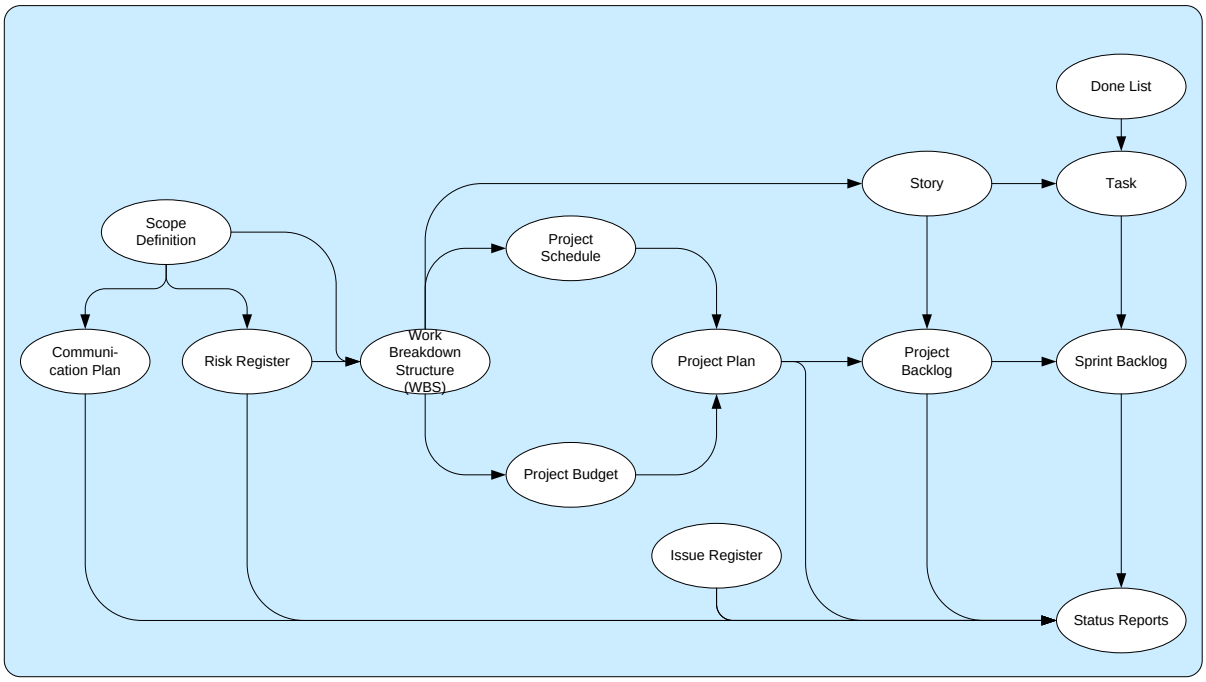


Figure 22 – Project Management Artifacts

- Scope Definition – The definition of what will be delivered by the project.
- Communication Plan – A plan of communication methods and vehicles to be produced during the project to keep all interested stakeholders informed about project progress.
- Risk Register – A list of identified risks and the actions taken to mitigate their impacts.
- Work Breakdown Structure (WBS) – A hierarchical definition of all of the deliverables to be produced in the project and the activities that will be accomplished to develop them. The lowest level in the tree structure on any given branch in the WBS is referred to as a Work Package.
- Project Schedule – The timelines for completing the activities in the WBS.
- Project Budget – The costs for completing the activities in the WBS.
- Project Plan – A document that conveys the project scope, cost and schedule.
- Issue Register – A listing of the issues that have arisen in the project and the actions taken to address them.
- Story – A small, independent unit of work that is to be accomplished to complete part of a Work Package.
- Project Backlog – A list of all the stories for the project.
- Task – A step in the process of completing a story.
- Done List – A prescribed set of steps to be followed in the completion of user stories.
- Sprint Backlog – A list of all the tasks to be accomplished in a sprint to complete all of the stories assigned to the sprint.
- Status Report – A document that provides the current status of the project to the stakeholders of the project.

Relating Project Management Activities and Artifacts

The following table provides an overview of all of the project management activities in the Tempered Agility project methodology framework and the artifacts produced by those activities.

Activity	Artifacts Produced
Project Planning	
Scope Definition	Scope Definition
Project Modelling	Work Breakdown Structure (WBS) Communication Plan Risk Register
Project Sizing	Project Schedule (initial) Project Budget (initial)
Baseline Definition	Project Schedule (final) Project Budget (final) Project Plan (draft)
Plan Approval	Project Plan (approved or rejected)
Project Leadership	
Work Initiation	Stories Project Backlog Tasks Done List
Team Leadership	Issue Register
Performance Measurement	Cost and Schedule progress updates
Performance Reporting	Status Report
Work Adjustment	Changes to all of the above

Figure 23 – Relating Project Management Activities and Artifacts

TAILORING TEMPERED AGILITY

No two projects are alike. Each project presents a different set of challenges and is completed by a unique team in a unique organizational and cultural context. As such, practitioners should expect to tailor the use of Tempered Agility to reflect the specific situation presented for each project.

Tailoring occurs in a couple of different ways. First, it might be the way certain activities are accomplished. For example, there are many different ways of gathering and analyzing requirements, designing user interfaces, integrating systems and testing. The activities that comprise the Tempered Agility Framework may stay the same, but the way the activities are completed may differ depending on the needs of the project, the abilities and preferences of the project team and the preferences and experiences of the customer. These aspects of tailoring tend to affect the product development process. That is the process the team goes through to produce the product.

Tailoring may also occur in the project management processes. This may mean the degree of formality of the project management approach. Differing project management approaches may be required to address cultural differences in the customer organization, the procedures and regulations of the customer and the needs of the stakeholder community.

Factors to Consider in Tailoring

Several different types of factors may come into play when tailoring Tempered Agility.

Process Context

It may help to think of the context of the various processes at play in a project as illustrated in Figure x below. The project approach must be adapted for at least three different kinds of processes: product development processes, project management processes and organizational management processes. For a project to succeed, those three processes must be consistent with each other and support each other. If any one layer is inconsistent with the others, friction arises that undermines the effectiveness of the project team and the project gets into trouble. Two simple examples would include using agile product development processes in a project managed by a command-and-control project management style or trying to be agile in an organization that requires strict adherence to rigid, waterfall-oriented stage-gate review processes imposed by the Project Management Office (PMO).

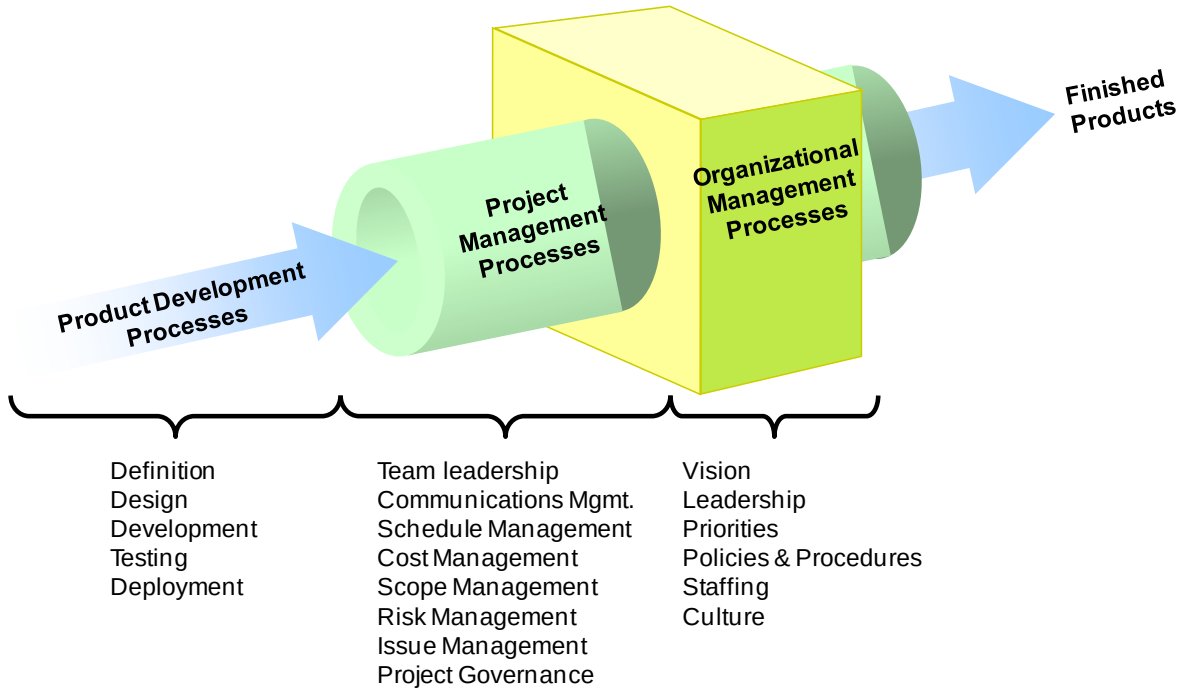


Figure 24 – Project Life Cycle Context

In general, organizational processes are probably beyond the control of the project team or project manager. They represent the context with which the project management and product development processes must be consistent.

The project management approach is then adapted to fit the organizational context. Finally, the team and the project manager determine the product development process that fits within the context of both organizational processes and the project management approach. To some extent, the project manager can act as a buffer between the organizational context and the team development processes. The project manager can shield the team from dysfunction in the organizational layer so that they can be more productive in getting the hard work accomplished. At the same time, the project manager may be able to add value by helping sort out the organizational needs encountered.

Project and Team Characteristics

The diagram shown below in Figure x lists several factors that should be considered in deciding how to tailor the project between agile and traditional approaches.

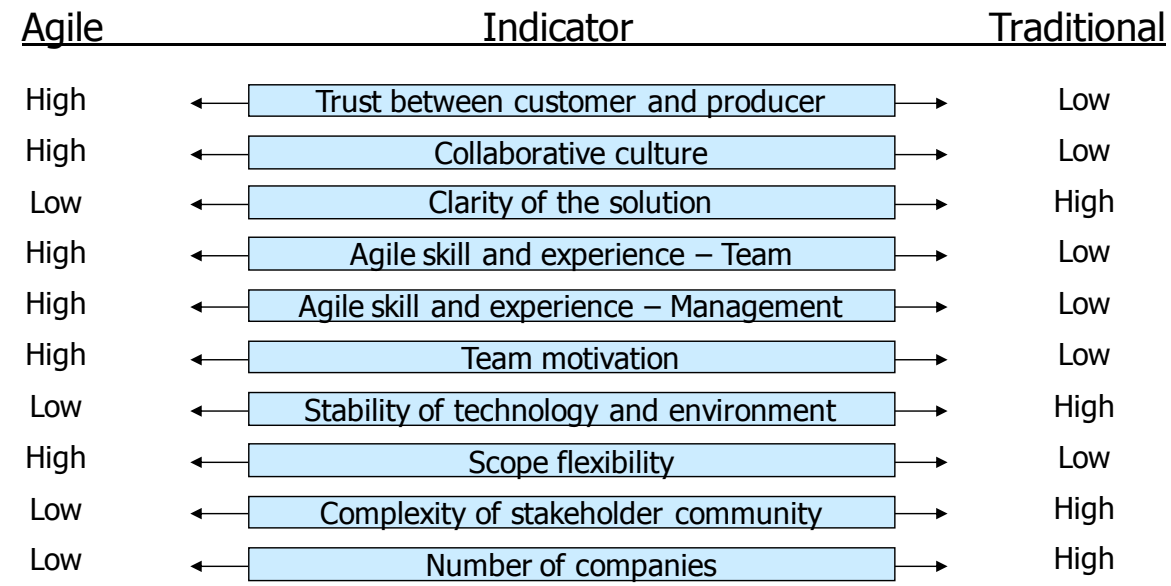


Figure 25 – Factors in Tailoring Tempered Agility

As indicated a high degree of trust may be conducive to agile approaches while low degrees of trust may indicate a more traditional approach, e.g., a waterfall life cycle. Conversely, lack of clarity of the intended solution may argue for a more agile approach.

Project Types

Not all types of projects are good fits for Tempered Agility as portrayed in Figure x below.

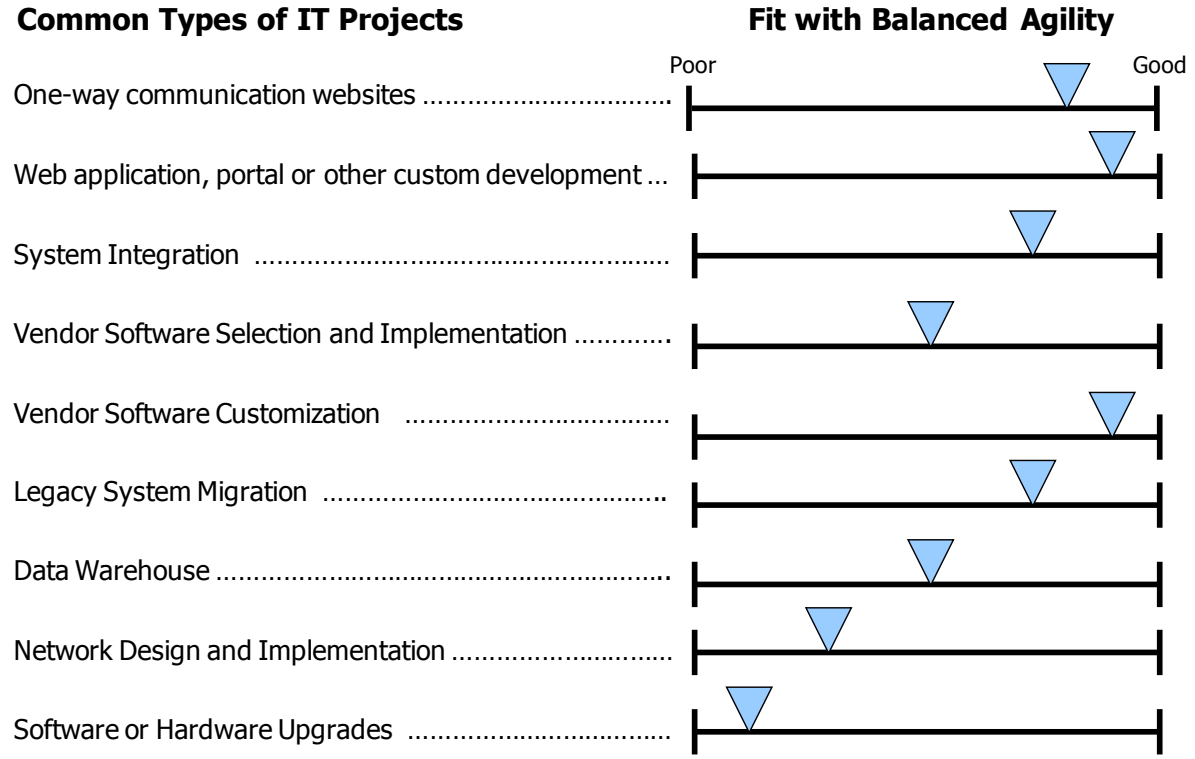


Figure 26 – Project Type Fit with Tempered Agility

Formality Spectrum

To simplify the tailoring process, the table shown in Figure 21 below describes three categories of project formality with the project characteristics that should drive the selection.

Formality Category	Small, Informal Project	Medium Project	Large, Formal Project
Situational Indicators - Size - Risk - Proficiency - Complexity - Trust	2-4 person team Short duration (1-3 iterations) Low risk Discretionary division funds Project lead, no PM Few reporting requirements Customer comfort high Management comfort high Proficient team Established teamwork environment	5-15 person team 3-5 iterations Medium risk Product and/or Project Manager Schedule reporting required Status report required Customer comfort good Management comfort good Team needs or prefers some structure	Business case 16+ person team(s) Possibly multiple teams Multiple releases PM, Sr. PM or PgM assigned Cost & schedule reporting High schedule expectations Steering Committee reporting Customer new or skeptical Management new or skeptical Team wants or needs structure
Agile Practices & Tools (Variations Occur)	Sprint backlog - Index cards or wiki. Flip charts with swim lanes and Post-It notes or Agile Task Manager at team's discretion. Story point sizing.	Project and sprint backlogs Use Cases, stories, tasks Agile Task Manager workbook Estimated hours sizing Sprint burn-down	Use Cases, stories, tasks Architecture Analysis Project and sprint backlogs Agile Project Manager workbook Estimated hours sizing Project and sprint burn-down Mercury Quality Center
Project Management Practices	Informal status reporting, e.g., weekly group meetings Informal issue escalation	Status reporting Schedule management Qualitative risk management Issue management	Status reporting Schedule management Cost management Quantitative risk management Issue management Change management

Figure 27 – Tempered Agility Project Formality Categories

RESOURCES

Some resources to help in adoption of Tempered Agility are listed below.

Tools

- Scrum Manager – This spreadsheet can be used to run the daily scrums and to manage the stories and tasks that comprise the project backlog.

ScrumManagerV2_10
0623.xlsx

- Project Brief Template – This Word document provides a template for the Project Brief created in Project Initiation.

TA_ProjectBrief_Tem
plate.docx

- Technical Specifications Template – This Word document provides a template for the Technical Specifications created initially in the Definition stage.

TA_TechnicalSpecific
ation_Template.docx

Training

- Tempered Agility Course Description – This document describes a 32-hour training class entitled, “Managing Projects with Tempered Agility”, which covers the Tempered Agility methodology in significantly more depth with hands-on exercises and case studies.

TemperedAgility_Cou
rseDescription.docx

References

1. Agile Software Development Using Scrum, Ken Schwaber and Mike Beedle, © Prentice Hall 2001

Related Readings

A. User Stories Applied: For Agile Software Development, Mike Cohn, © Addison-Wesley 2004