MPHASIS TUTORIALS POINT READING MATERIALS AGILE

Contents

AGILE – PRIMER

What is Agile?

It is a SDLC where the software is built incrementally, with each incrementations happening in short iterations of 1-4 weeks. This means that as the needs of the client changes or becomes clearer or more specific, the development process can keep up with the changes, and the requirement plan can be adjusted accordingly.

Instead of a single-pass development of 6 to 18 months where all the requirements and risks are predicted upfront, Agile adopts a process of frequent feedback where a workable product is delivered after 1 to 4-week iteration.

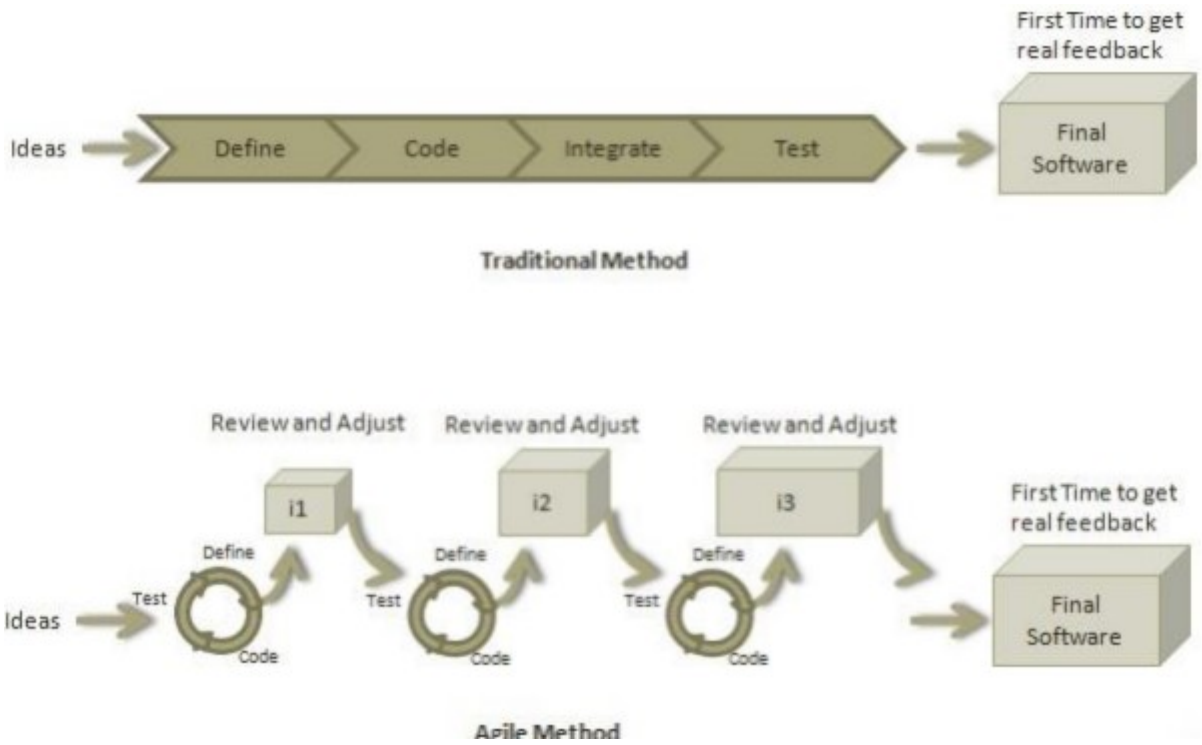


FIGURE 1 AGILE METHOD

Roles in Agile

There are 2 main leadership roles in an Agile team

Scrum Master

Team leader, helps the team to follow scrum practices. Works as an interface between the technical and business aspects of the team. Ensures close cooperation between members, and ensures that agile 'Inspect and Adapt' practices are leveraged properly:

1. Daily stand-ups
2. Planned meetings
3. Demo
4. Review
5. Retrospective Meetings

Product Owner

They drive the product from the business perspective. Their job to ensure the product satisfies all the business requirements. Their responsibilities are:

1. To define the requirements and prioritize their values.
2. To determine the release date and contents.
3. To take an active role in iteration planning and release planning meetings.
4. To ensure that team is working on the most valued requirement.
5. To represent the voice of the customer.
6. To accept the user stories that meet the definition of done and defined acceptance
7. criteria.

Cross-functional Team

An agile team generally consists of developers, testers, one technical lead, one product owner, and one scrum master. Product Owner and Scrum master are considered to be a part of Team Interface, whereas other members are part of Technical Interface.

How an Agile Team Plans its Work?

An Agile team works in iterations to deliver user stories where each iteration is of 10 to15 days. Iterations are also known as sprints. Each user story is planned based on its backlog prioritization and size. The team uses its capacity – how many hours are available with team to work on tasks – to decide how much scope they have to plan. In each release, there are 3 stages.

Iteration planning

Planning what is going to be in each user story.

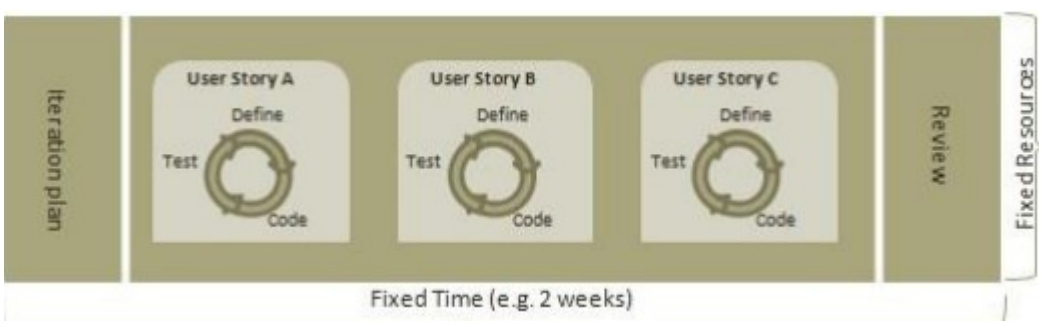
User Stories

Cycles of Define, Code, Test, Define, Code, Test, etc.

Review

Review of the work done, usually culminating in an After-Action Review plan document.

In each release, a number of User Stories are selected, and points given to them in a rough estimate. In iteration planning, User Stories are broken down into tasks and subtasks.



A Point is the amount of time a team can commit to the software user story. A point is usually 8 hours. Each story is estimated in points. The total amount of points adds up to the amount of time a team can work in the fixed time, generally 2 weeks.

Capacity is the amount of time an individual can commit. Total capacity of the team is equal to the total amount of points x (hour/point), and is equivalent to the amount of time a team can work in the fixed time, generally 2 weeks.

User Story

A user story is a requirement which defines what is required by the user as functionality. A user story can be in two forms:

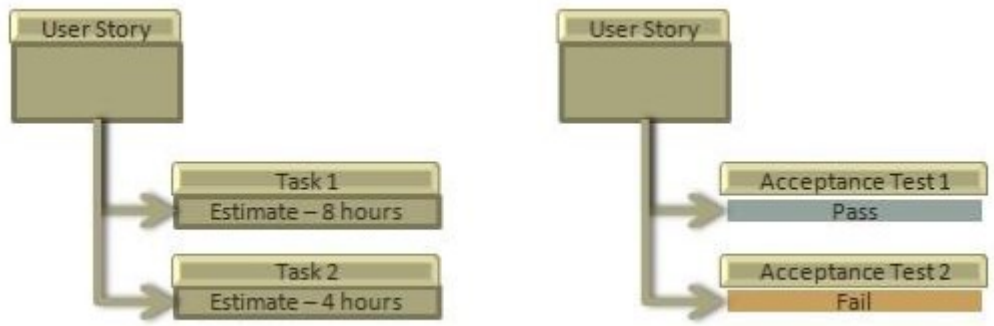
- As a <User Role> I want <Functionality> so that <Business Value>
- In order to <Business value> as a <User Role> I want <Functionality>

User story is a requirement that a particular role wants to realize the need for a particular business value. During release planning, a rough estimate is given to a user story using relative scale as points. During iteration planning, the story is broken down into tasks.

Relationship of User Stories and Tasks

User story talks about what is to be done. It defines what a user needs. User story is divided into tasks during planning. Task talks about how it is to be done. It defines how a functionality is to be implemented. Stories are implemented by tasks. Each story is a collection of tasks. Tasks are estimated in hours, typically from 2 to 12 hours.

Stories are validated using acceptance tests. Tasks are validated using unit, integration and regression testing.



When a Story is Done

The team decides what done means. The criteria may be:

- All tasks (development, testing) are completed.
- All acceptance tests are running and are passed.
- No defect is open.
- Product owner has accepted the story.
- Deliverable to the end-user.

What are Acceptance Criteria?

Criteria defines the functionality, behavior, and performance required by a feature so that it can be accepted by the product owner. It defines what is to be one so that the developer knows when a user story is complete. It is the minimum number of requirements that must be satisfied by a story for the product owner to be ok with it.

How the Requirements are Defined?

Requirements are defined as a User Story, with acceptance criteria, and tasks to implement the story.

AGILE – MANIFESTO

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

- Individuals and interactions over Processes and tools
- Working software over Comprehensive documentation
- Customer collaboration over Contract negotiation
- Responding to change over Following a plan

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

Basically, hippie shit.

Principles of agile

Customer Satisfaction - Highest priority is given to satisfy the requirements of customers through early and continuous delivery of valuable software. Customer's business needs are the priority.

Welcome Change - Changes are inevitable during software development. Ever-changing requirements should be welcome, even late in the development phase. Agile processes should work to increase customers' competitive advantage. Change in requirements should be encouraged.

Deliver a Working Software - Deliver a working software frequently, ranging from a few weeks to a few months, considering shorter time-scale. Getting the software working properly should be the priority, with frequent releases to the customer.

Collaboration - Business people and developers must work together during the entire life of a project. Devs, customers, and product owners should work together closely to ensure the software is exactly what is needed.

Motivation - Projects should be built around motivated individuals. Provide an environment to support individual team members and trust them so as to make them feel responsible to get the job done.

Face-to-face Conversation - Face-to-face conversation is the most efficient and effective method of conveying information to and within a development team.

Measure the Progress as per the Working Software - Working software is the key and it should be the primary measure of progress. Nothing else matters more than if the software works or not—whether it satisfies all the customer's requirements.

Maintain Constant Pace - Agile processes aim towards sustainable development. The business, the developers, and the users should be able to maintain a constant pace with the project. The keyword is constant—proper and practical planning should not assume crunches or miracles. Breakneck speeds are not sustainable.

Monitoring - Pay regular attention to technical excellence and good design to enhance agility. Nothing can compensate for technical faults, and good design adds value everywhere.

Simplicity - Keep things simple and use simple terms to measure the work that is not completed. Complex stuff doesn't really help.

Self-organized Teams - An agile team should be self-organized and should not depend heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams. No one is coming to help.

Review the Work Regularly - Review the work done at regular intervals so that the team can reflect on how to become more effective and adjust its behavior accordingly. Regular reviews ensure that the team is on the right track.

AGILE – CHARACTERISTICS

There are 3 main characteristics

Iterative/incremental and Ready to Evolve

Face-to-face Communication

Feedback Loop

Iterative/incremental and Ready to Evolve

Most of the agile development methods break a problem into smaller tasks. There is no direct long-term planning for any requirement. Normally, iterations are planned which are of vary short period of time, for example, 1 to 4 weeks. At the end of iterations, the software is handed over, and this is called a release. Planning only occurs until a release, further plans are not concrete. After demo, review comments are taken and are planned to be incorporated in the working software as required.

Face-to-face Communication

> "Each agile team should have a customer representative such as a product owner in scrum methodology. This representative is authorized to act on behalf of the stakeholders and he can answer the queries of the developers in between iterations. An information radiator (physical display) is normally located prominently in an office, where passers-by can see the progress of the agile team. This information radiator shows an up-to-date summary of the status of a project."

Face to face communication is emphasized, but the real idea is effective communication. Hashing out most of the details over email, and talking out the rest in a meeting is also fine, if it is more effective than face to face communication.

And instead of an information radiator, a dashboard or a checklist can also be used if it is more efficient.

Feedback Loop

> Daily stand-up is a common culture of any agile development; it is also known as daily scrum. It is a kind of a brief session where each team member reports to each other regarding the status of what they have done, what to do next, and any issues they are facing.

A feedback loop must be established, incorporating the cross-functional team. Other loops ma also be established to communicate with stakeholders.

AGILE – DAILY STAND-UP

Daily stand-up, as the name suggests, is a daily status meeting among all the members of an agile team. It not only provides a forum for regular updates but also brings the problems of team members into focus so that it can be quickly addressed. Daily stand-up is a must-do practice, no matter how an agile team is established regardless of its office location.

Structure of Daily Stand-up

Daily status meeting attended by all members and it is held roughly for 15 minutes.

Every member has to answer three important questions:

1. What I did yesterday?
2. What I'll do today?
3. Any impediment I am facing…/ I am blocked due to…

Daily stand-up is for status update, not for any discussion. For discussion, team members should schedule another meeting at a different time. Participants usually stand instead of sitting so that the meeting gets over quickly. However, this seems incredibly ableist.

Why Stand-up is Important?

The benefits of having a daily stand-up in agile are as follows:

1. The team can evaluate the progress on a daily basis and see if they can deliver as per the iteration plan.
2. Each team member informs all about his/ her commitments for the day.
3. It provides visibility to the team on any delay or obstacles.

Who Attends a Stand-up?

The scrum master, the product owner, and the delivery team should attend the stand-up on a daily basis. Stakeholders and Customers are encouraged to attend the meeting and they can act as an observer, but they are not supposed to participate in stand-ups. It is the scrum master's responsibility to take note of each team member's queries and the problems they are facing.

In case the teams are geographically dispersed, one person from each remote team should attend the standup.

AGILE – DEFINITION OF DONE

The definition of done varies according to user story, iteration, and release. However, this ultimately depends on the product, the team, and the customer.

User Story

A user story is done when

1. All the related code has been checked-in.
2. All the unit test cases have been passed.
3. All the integration and acceptance test cases have been passed.
4. Help text is written.
5. Product Owner has accepted the story.

Sprint

A sprint is done when

1. Product backup is complete.
2. Performance has been tested.
3. User stories have been accepted or moved to the next iteration.
4. Defects have been fixed or postponed to the next iteration.
5. All that was planned for the iteration has been done/moved to the next iteration/discarded, and a review has been conducted.

Release

A release is done when

1. System is stress tested.
2. Performance is tuned.
3. Security validations are carried out.
4. Disaster recovery plan is tested.

Releases should be frequent, and right after one or more iterations.

AGILE – RELEASE PLANNING

The purpose of release planning is to create a plan to deliver an increment to the product. It is done after every 2 to 3 months.

Who is Involved?

- Scrum Master - The scrum master acts as a facilitator for the agile delivery team.
- Product Owner - The product owner represents the general view of the product backlog.
- Agile Team - Agile delivery team provides insights on the technical feasibilities or any dependencies.
- Stakeholders - Stakeholders like customers, program managers, subject matter experts act as advisers as decisions are made around the release planning.

Prerequisites of Planning

The prerequisites of release planning are as follows:

1. A ranked product backlog, managed by the Product Owner. Generally, five to ten features are taken which the product owner feels that can be included in a release. This is used to select what features to work on for the release.
2. Team's input about capabilities, known velocity or about any technical challenge. This is vital as the team knows best about what their capability is.
3. High-level vision, to focus on the overall picture.
4. Market and Business objective, to have a set goal to work towards.
5. Acknowledgement whether new product backlog items are needed, so that the product is better aligned with business needs.

Planning Data

The list of data required to do release planning is as follows:

1. Previous sprints or release planning results, past data basically.
2. Feedback from various stakeholders on product, market conditions, and deadlines to determine the scope of the plan.
3. Action plans of previous releases / sprints, for templating and to make it easier to determine challenges and correct past mistakes.
4. Features or defects to be considered
5. Velocity from previous releases/ estimates.
6. Organizational and personal calendars
7. Inputs from other teams and subject matter experts to manage any dependencies

Output

The output of a release planning can be the following:

1. Release plan, that will be formed for the whole meeting
2. Commitment, or agreement to a set of plans
3. Issues, concerns, dependencies, and assumptions which are to be monitored, so that they can be mitigated or solved for
4. Suggestions to improve future release planning meetings, because even they need After Action Reviews.

Agenda

The agenda of a release planning can be:

1. Opening ceremony - Welcome message, review purpose and agenda, organizing tools and introduction to business sponsors.
2. Product Vision, Roadmap - Show the large picture of the product.
3. Review previous releases - Discussion on any item which can impact the plan.
4. Release name / theme - Inspect the current status of roadmap themes and do the required adjustments, if any.
5. Velocity - Present the velocity for the current release and of previous releases. Time taken and work done.
6. Release schedule - Review key milestones and decision on time boxes for release and sprints within release. Fix the amount of time to be given to the release and sprints. This is based on data from deadlines, etc.
7. Issues and concerns - Check any concerns or issue and record them.
8. Review and Update the Definition of Done - Review the definition of done and make appropriate changes based on technology, skill, or changes in team members since the last sprint / release.
9. Stories and items to be considered - Present the user stories and features from the product backlog to be considered for scheduling in the current release.
10. Determine sizing values - If the velocity is unknown, then plan the sizing values to be used in the release planning.
11. Coarse the size of stories - The delivery team determines the appropriate size of the stories under consideration and splits the stories into multiple sprints if a story is too large. The product owner and the subject matter experts clarify the doubts, elaborate the acceptance criteria, and make proper story splits. The scrum master facilitates the collaboration. If a story is too large, split it into parts.
12. Map stories to sprints - The delivery team and the product owner map the stories to the sprints based on the size and velocity. The scrum master facilitates the collaboration.
13. New concerns or issues - Check any new issues based on previous experience and record the same.
14. Dependencies and assumptions - Check any dependencies/assumptions planned during the release planning. This ensures that they are well communicated and monitored for.
15. Commit - The scrum master calls for committing to the plan. Delivery team and Product owner signal it as the best plan and then commit to move to the next level of planning, that is, sprint planning.
16. Communication and logistics planning - Review/Update the communication and logistics planning for the release.
17. Parking lot - Process parking lot means all items should be either resolved or set as action items. Things done and things to be worked on.
18. Distribute Action items and action plans - Distribute the action items among their owners, process the action plan.
19. Retrospect - Solicit feedback from participants to make the meeting successful.
20. Close - Celebrate the success.

AGILE – ITERATION PLANNING

The purpose of sprint planning is for the team to complete the set of top-ranked product backlog items. This commitment is time boxed based on the length of sprint and team velocity.

Who is Involved?

- Scrum Master - The scrum master acts as a facilitator for the agile delivery team.
- Product Owner - The product owner deals with the detailed view of the product backlog and their acceptance criteria.
- Agile Team - Agile delivery defines their tasks and sets the effort estimates required to fulfil the commitment.

Prerequisites of Planning

- Items in product backlog are sized and have a relative story point assigned.
- Ranking has been given to portfolio items by the product owner.
- Acceptance criteria has been clearly stated for each portfolio item.
- A clear idea of the capacity of the team
- Concrete data about time available for the sprint

Planning Process

Following are the steps involved in sprint planning:

1. Determine how many stories can fit in a sprint.
2. Break these stories into tasks and assign each task to their owners.
3. Each task is given estimates in hours. These estimates help team members to check how many task hours each member have for the sprint.
4. Team members are assigned tasks considering their velocity or capacity so that they are not overburdened.

Velocity Calculation

An agile team calculates velocity based on past sprints. Velocity is an average number of units required to finish user stories in a sprint. Planned velocity tells the team how many user stories can be completed in the current sprint. If the team quickly finishes the tasks assigned, then more user stories can be pulled in. Otherwise, stories can be moved out too to the next sprint.

Task Capacity

The capacity of a team is derived from the following three facts:

- Number of ideal working hours in a day
- Available days of person in the sprint
- Percentage of time a member is exclusively available for the team.

Suppose a team has 5 members, committed to work full time (8 hours a day) on a project and no one is on leave during a sprint, then the task capacity for a two-week sprint will be: 5 x 8 x 10 = 400 hours

Planning Steps

1. Product Owner describes the highest ranked item of product backlog.
2. Team describes the tasks required to complete the item.
3. Team members own the tasks.
4. Team members estimate the time to finish each task.
5. These steps are repeated for all the items in the sprint.
6. If any individual is overloaded with tasks, then his/her task is distributed among other team members.

AGILE – PRODUCT BACKLOG

A product backlog is a list of items to be done. Items are ranked with feature descriptions. In an ideal scenario, items should be broken down into user stories.

Why Product Backlog is Important?

- It is prepared so that estimates can be given to each and every feature.
- It helps in planning the roadmap for the product.
- It helps in re-ranking the features so that more value can be added to the product.
- It helps in determining what to prioritize first. Team ranks the item and then builds value.

Characteristics of Product Backlog

- Each product should have one product backlog which can have a set of large to very large features.
- Multiple teams can work on a single product backlog.
- Ranking of features is done based on business value, technical value, risk management or strategic fitness.
- Highest ranking items are decomposed into smaller stories during release planning so that they can be completed in future iterations.

AGILE – USEFUL TERMS

Acceptance Criteria

It is the conditions set by the product owner or the customer in order to accept a feature to be valid and adhering to their requirements. Functions like a rubric for features of the software.

Backlog Grooming

It is an ongoing process in which the product manager or the customer manages the product backlog by getting feedback from agile teams. This process involves prioritizing the portfolio items, breaking them in smaller items, planning them for future iterations, creating new stories, updating acceptance criteria or elaborating acceptance criteria in details.

Capacity

It is the amount of work a team can take to complete in one iteration.

Feature

An improvement done to a product or capability of value to stakeholder which can be developed in a release.

Sprint

A theme-based work item that can be completed within a time box and accepted within the release of a product. Sprint work is defined during sprint planning and it finishes with demo and review meeting. It is also termed as Sprint.

Increment

An increment is the changing state of a product as it undergoes gradual development. It is normally represented by milestones or number of fixed iterations. Basically, the changelog for a release.

Product Owner

The product owner is a member of the Agile delivery team, responsible to collect and rank business requirements in the product backlog. A product owner communicates what is to be done in a release/sprint. He/she sets the commitments and is responsible to protect team from any change in requirements during an iteration.

Product Backlog

Set of functional and non-functional product requirements.

Product Backlog Items

May be user stories, defects, features which are to be developed by the agile team.

Points

A common unit used to set the relative size of user stories, features, or any other portfolio items.

Release

A time box where work is done to support delivery of testable increment to a software. In scrum, a release consists of multiple sprints.

Requirement

A specification of a software product to satisfy a stated contract or functionality. User stories and portfolio items are types of requirements.

Story Points

A unit used by the agile team to estimate relative sizes of user stories and features.

Sprint

Same as Iteration.

Timebox

A fixed duration of time in which a deliverable is to be developed. Normally, along with fixing start and end date of a timebox, the number of resources is also fixed.

Task

It is a unit of work that contributes towards the completion of a user story within a sprint. User stories are decomposed into multiple tasks and each task can be divided between team members marking them as owner of the tasks. Team members can take responsibility of each task, update estimates, log work done or to-do as desired.

User Story

A listed acceptance criterion to fulfil certain requirements of a user. It is normally written from the perspective of an end-user.

Velocity

A measure to weight the accepted work in an iteration or timebox. Normally it is the sum of story points accepted in an iteration. Speed of work in fixed time

Further reading:

The Scrum Guide,

The Lean Startup by Eric Ries - Google Search

A Sense of Urgency - Google Search

Leading the Transformation: Applying Agile and DevOps Principles at Scale - Google Search

Extreme Programming Explained: Embrace Change - Google Search

The Agile Enterprise: Building and Running Agile Organizations by Mario E. Moreira - Google Search

Project Management the Agile Way: Making it Work in the Enterprise by John Goodpasture - Google Search

Agile Project Management with Scrum, by Ken Schwaber - Google Search

Agile Project Management: Creating Innovative Products - Google Search

The Software Project Manager's Bridge to Agility by Michele Sliger and Stacia Broderick - Google Search

Coaching Agile Teams: A Companion for Scrum Masters, Agile Coaches, and Project Managers in Transition by Lyssa Adkins - Google Search

Scrum Mastery: From Good to Great Servant-Leadership by Geoff Watts - Google Search

Scrum: The Art Of Doing Twice The Work In Half The Time - Google Search

Agile Software Development with SCRUM - Google Search

Agile Estimating & Planning Your Sprint with Scrum - Google Search

Agile Estimating and Planning - Google Search

Agile Retrospectives: Making Good Teams Great - Google Search

Software in 30 Days - Google Search

How to FAIL with SCRUM?: Learning from Experience - Google Search

The Mythical Man-Month: Essays on Software Engineering - Google Search

The Agile Developer's Handbook - Google Search

Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation by Jez Humble | Goodreads

Continuous Integration: Improving Software Quality and Reducing Risk - Google Search

Google - Site Reliability Engineering

User Stories Applied: For Agile Software Development - Google Search

Agile UX Storytelling: Crafting Stories for Better Software Development - Google Search

Working Effectively with Legacy Code - Google Search

More Agile Testing: Learning Journeys for the Whole Team - Google Search

Fit for Developing Software: Framework for Integrated Tests - Google Search

Test Driven Development: By Example - Google Search