
WEBINAR

# AGILE REQUIREMENTS DOCUMENTATION: TIPS AND TRICKS FOR MODERN TEAMS

NIRAV ASSAR

# INTRODUCTION


- Nirav Assar
- Grails developer since 2009
- DFW area - worked in financial, retail, defense
- Agile consultant
- Worked in over 20 green field Agile projects – Grails and Java
- Experienced Waterfall, Spiral Model, mini-waterfall, Agile and Iterative (Larman style), Xtreme, Scrum, Safe, FDD, BDD, Ad-hoc, and NDD (Nothing Driven Development)

# WHAT IS THIS PRESENTATION ABOUT?

OBJECT COMPUTING

## It is NOT about:

- Explaining Agile or scrum values
- Defining terms or roles
- Contrasting to waterfall
- Issuing rules
- Estimating or planning

## It IS about:

- Recommendations for Agile documentation
- Ideas to organize documentation for a project
- Real-life examples
- Being efficient

# WHAT'S THE NEED?


- Teams and organizations revert to old habits
- Emails and attachments to emails become the norm
- Teams evolve and adapt for the better, but without a plan up front, it becomes a mess
- Inertia

# AGENDA


1. Artifacts of Agile documentation
2. Agile principles for documentation
3. Documentation framework for an Agile project
4. Recommended formats for documentation

# ARTIFACTS FOR AGILE DOCUMENTATION

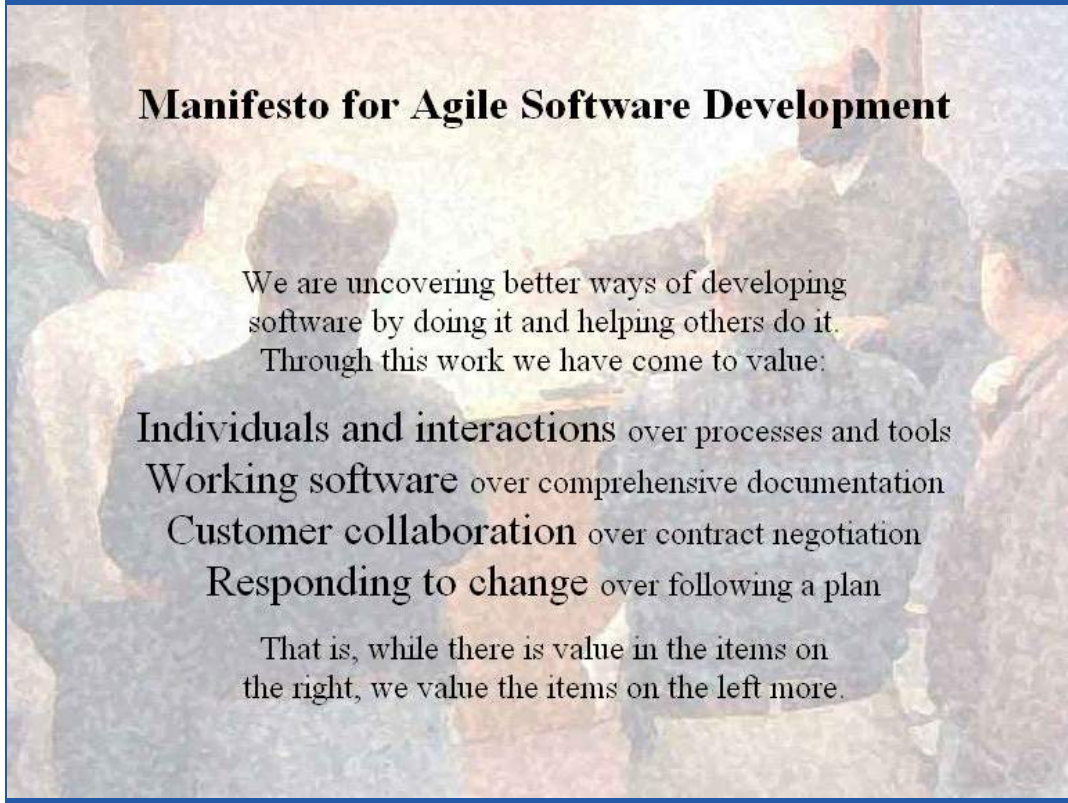

- User stories, Epics
- Daily Standups, Meeting Notes
- Product Roadmap
- Team Working Agreement
- Planning Session
- UI Storyboarding
- Design Documents

# ARTIFACTS FOR AGILE DOCUMENTATION


- Acceptance Tests
- Demos
- Technical Tutorials
- Retrospectives

# AGILE PRINCIPLES

- Working software over comprehensive documentation
- Customer collaboration over contract negotiation



# AGILE PRINCIPLES

OBJECT COMPUTING

2. Welcome changing requirements

3. Deliver working software frequently

4. Business people and developers must collaborate throughout the project

10. Simplicity, the art maximizing the amount of work not done

12. At regular intervals, the team reflects

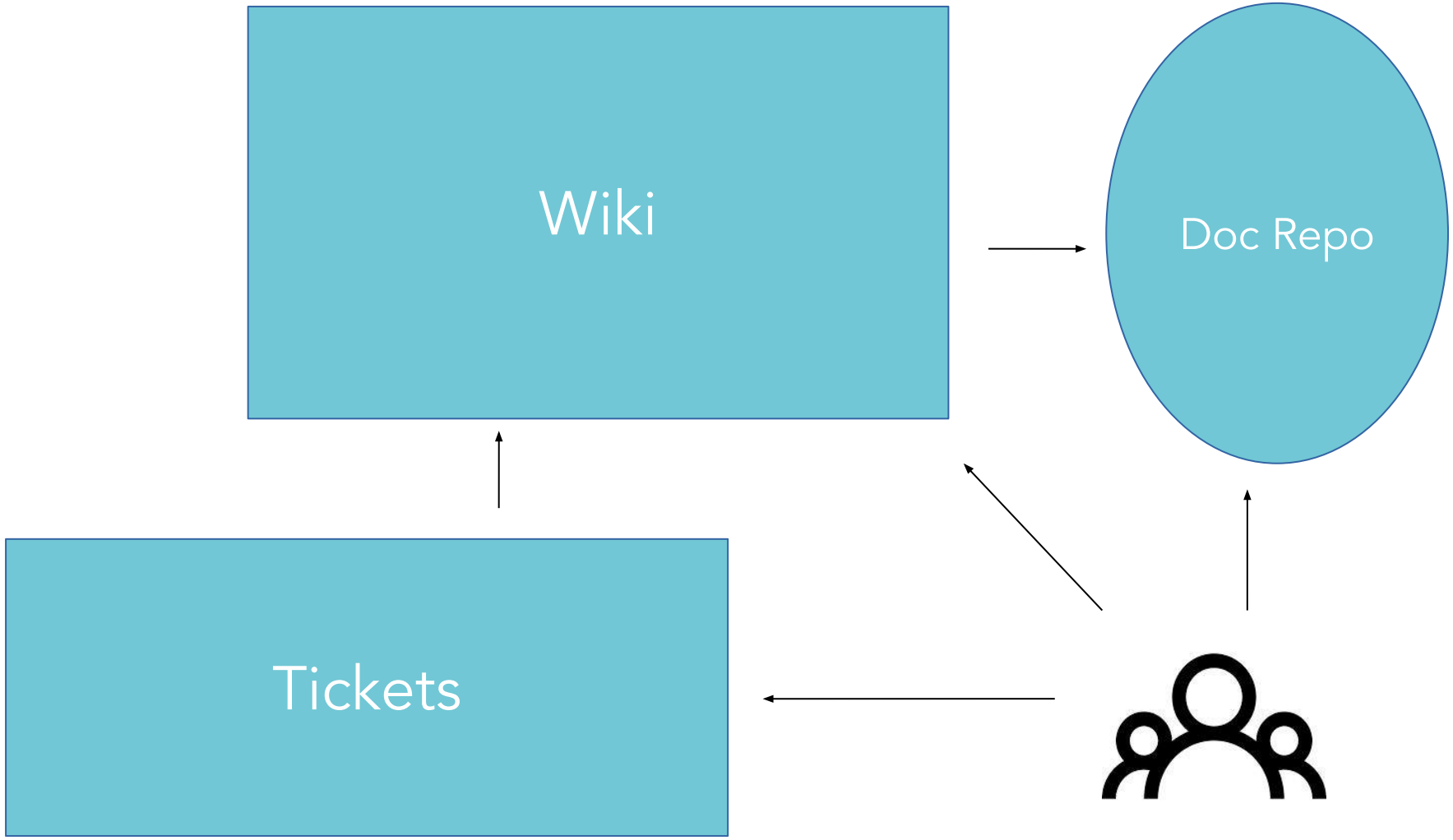
From https://agilemanifesto.org/principles.html

# CORRESPONDING PRINCIPLES


- Emergent design
- Emerging requirements
- Low ceremony, lighter documentation
- Having a plan is planning to succeed
- Plans can be rendered useless, but planning is indispensable

# DOCUMENTATION FRAMEWORK FOR AN AGILE PROJECT

OBJECT COMPUTING



# TOOL EXAMPLES


## Ticketing

- JIRA, Agilean, Sprintground, Github

## Wiki

- Conflience, Slimwiki, Github wiki

## DocumentRepo

- Shared Drive, Google Drive, Sharepoint, Dropbox

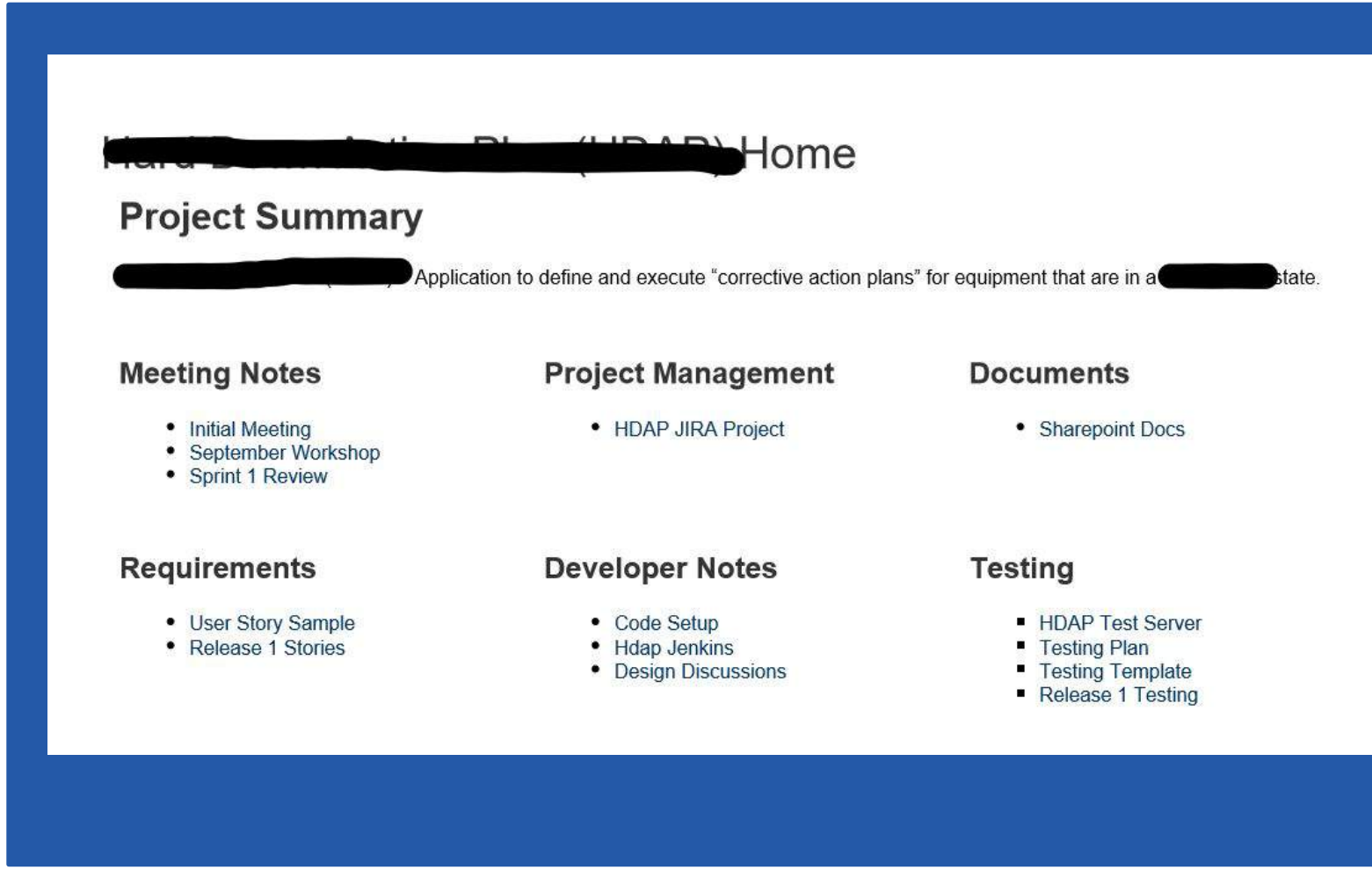
# DOCUMENT FRAMEWORK BENEFITS


- Integration between artifacts
- Fosters collaboration
- Searchable
- Traceable
- Evolving
- Binds tickets to stories and documents

# RECOMMENDATIONS FOR AGILE DOCUMENTATION

## Wiki

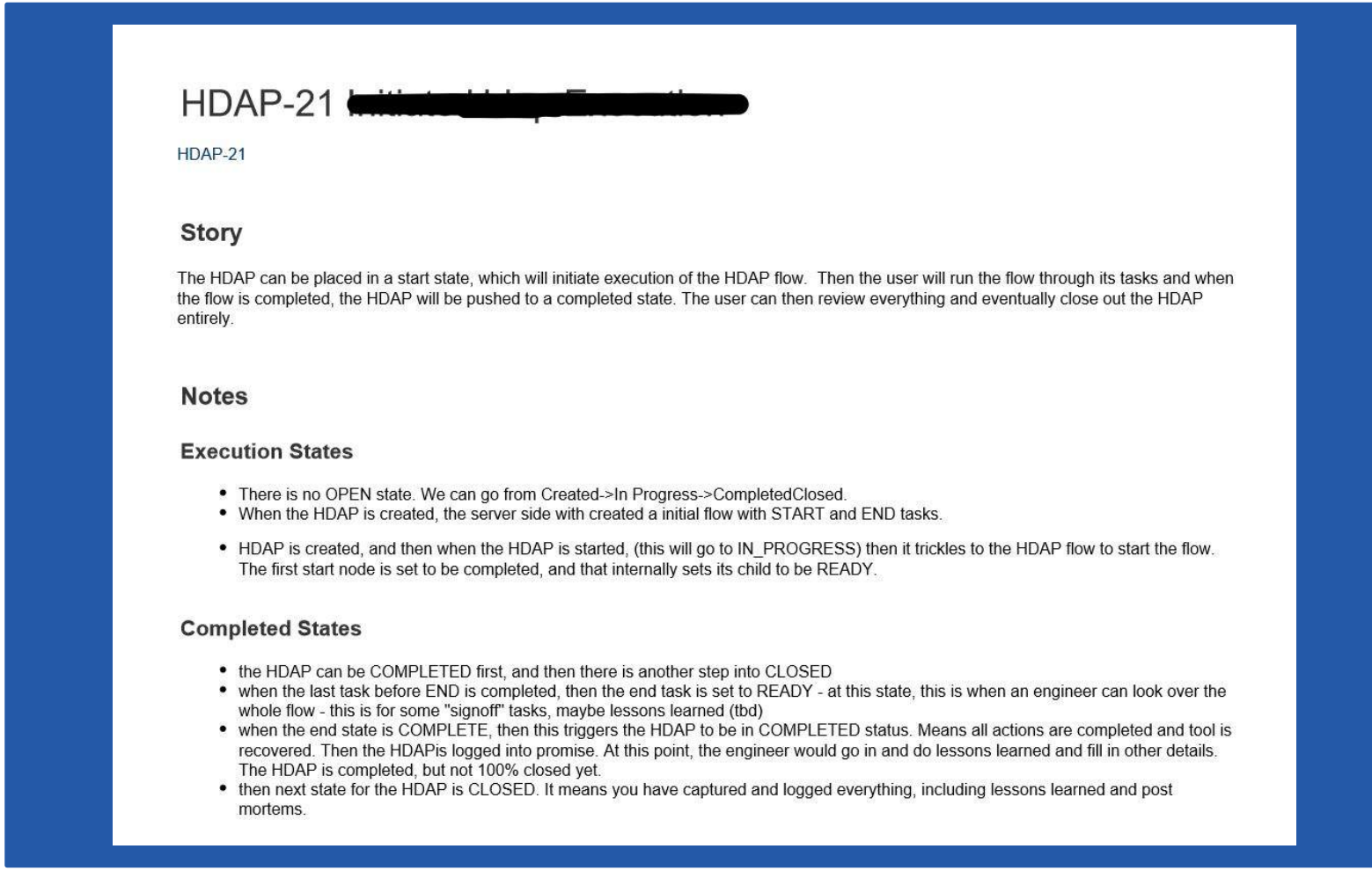
- Open to all team members
- Central hub
- Section into themes



# RECOMMENDATIONS FOR AGILE DOCUMENTATION

## User Stories

- Title goes in the ticket
- Story elaboration into the wiki



HDAP-21 [redacted]

HDAP-21

**Story**

The HDAP can be placed in a start state, which will initiate execution of the HDAP flow. Then the user will run the flow through its tasks and when the flow is completed, the HDAP will be pushed to a completed state. The user can then review everything and eventually close out the HDAP entirely.

**Notes**

**Execution States**

- There is no OPEN state. We can go from Created->In Progress->CompletedClosed.
- When the HDAP is created, the server side with created a initial flow with START and END tasks.
- HDAP is created, and then when the HDAP is started, (this will go to IN_PROGRESS) then it trickles to the HDAP flow to start the flow. The first start node is set to be completed, and that internally sets its child to be READY.

**Completed States**

- the HDAP can be COMPLETED first, and then there is another step into CLOSED
- when the last task before END is completed, then the end task is set to READY - at this state, this is when an engineer can look over the whole flow - this is for some "signoff" tasks, maybe lessons learned (tbd)
- when the end state is COMPLETE, then this triggers the HDAP to be in COMPLETED status. Means all actions are completed and tool is recovered. Then the HDAPis logged into promise. At this point, the engineer would go in and do lessons learned and fill in other details. The HDAP is completed, but not 100% closed yet.
- then next state for the HDAP is CLOSED. It means you have captured and logged everything, including lessons learned and post mortems.

# RECOMMENDATIONS FOR AGILE DOCUMENTATION


## Daily Standups

- Accomplished yesterday
- Goals for day
- Roadblocks
- Per person or by ticket (Kanban style)

# RECOMMENDATIONS FOR AGILE DOCUMENTATION

OBJECT COMPUTING

## Team Working Agreement

- Values, commitment, practices
- "Don't be afraid to ask for help"
- "Ask questions in JIRA"
- "Post information on the wiki and avid long technical emails"
- "Be on time for meetings"
- "Mark your calendar when you are out"
- "Definition of Done"
- "Commit often"
- "Broken build is the team's responsibility"
- Sign the wiki

# RECOMMENDATIONS FOR AGILE DOCUMENTATION

OBJECT COMPUTING

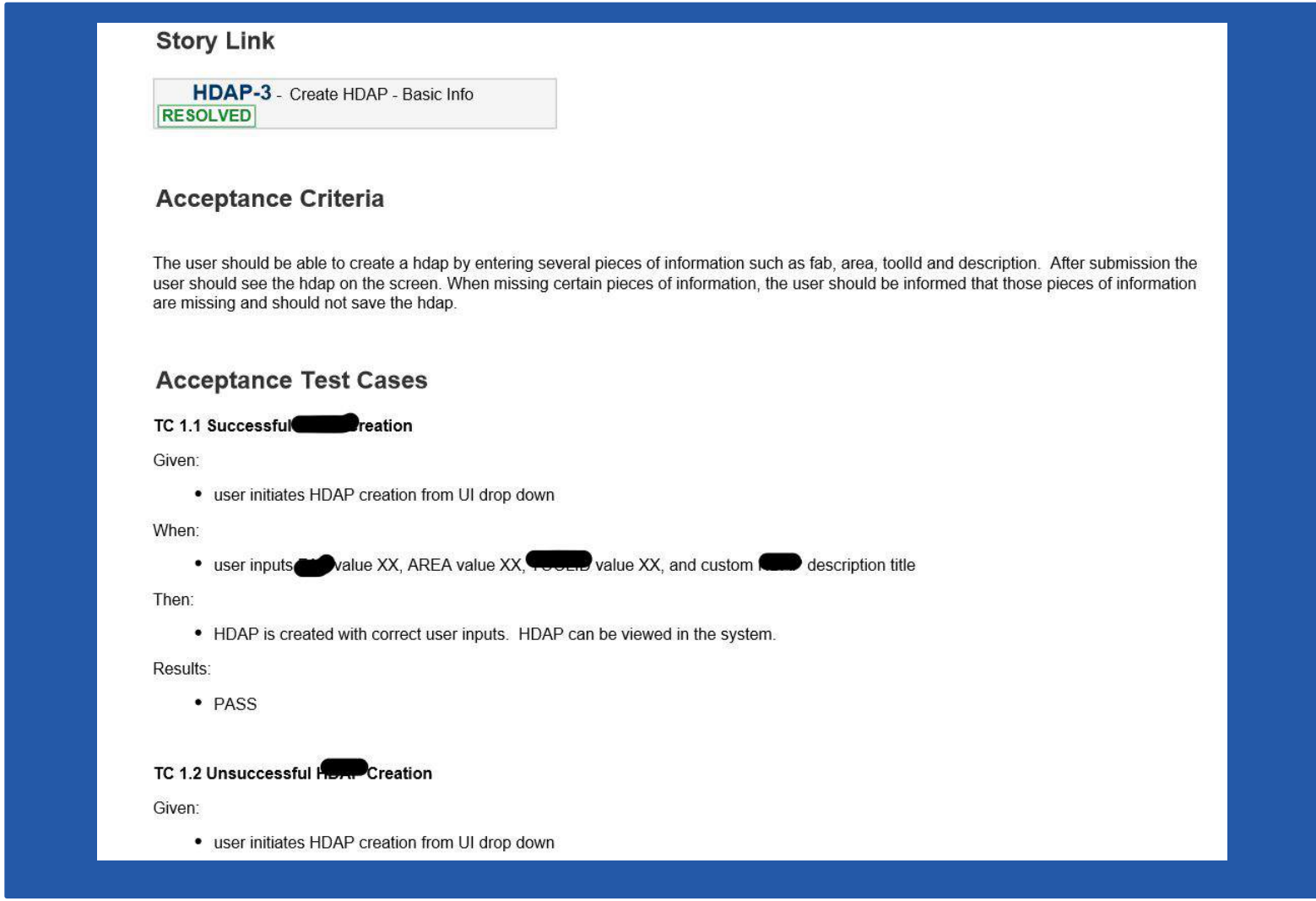
## Retrospectives

- The good
- Could have done better
- Improvements
- Personal complaints
- Action items

# RECOMMENDATIONS FOR AGILE DOCUMENTATION

## Acceptance Tests

- Acceptance criteria first
- Acceptance test
- Include border cases



**Story Link**

**HDAP-3** - Create HDAP - Basic Info
RESOLVED

**Acceptance Criteria**

The user should be able to create a hdap by entering several pieces of information such as fab, area, toolId and description. After submission the user should see the hdap on the screen. When missing certain pieces of information, the user should be informed that those pieces of information are missing and should not save the hdap.

**Acceptance Test Cases**

**TC 1.1 Successful [redacted]reation**

Given:

- user initiates HDAP creation from UI drop down

When:

- user inputs [redacted]value XX, AREA value XX, [redacted] value XX, and custom [redacted] description title

Then:

- HDAP is created with correct user inputs. HDAP can be viewed in the system.

Results:

- PASS

**TC 1.2 Unsuccessful [redacted]Creation**

Given:

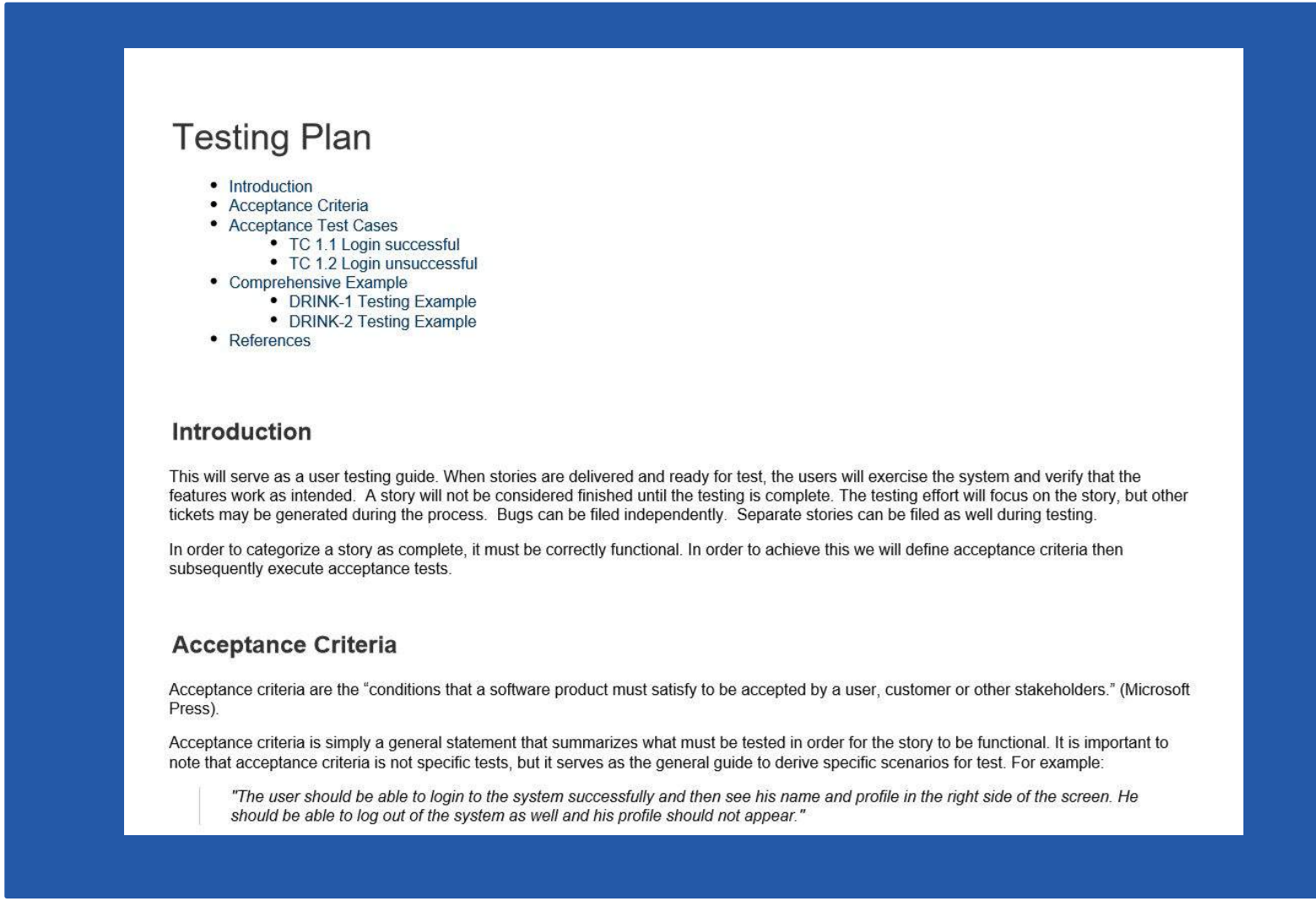
- user initiates HDAP creation from UI drop down

# RECOMMENDATIONS FOR AGILE DOCUMENTATION

## General Format for Other Artifacts

- Summary, Details, Diagrams
- Purpose, Problem, Details,

  Action



Testing Plan

- Introduction
- Acceptance Criteria
- Acceptance Test Cases
  - TC 1.1 Login successful
  - TC 1.2 Login unsuccessful
- Comprehensive Example
  - DRINK-1 Testing Example
  - DRINK-2 Testing Example
- References

**Introduction**

This will serve as a user testing guide. When stories are delivered and ready for test, the users will exercise the system and verify that the features work as intended. A story will not be considered finished until the testing is complete. The testing effort will focus on the story, but other tickets may be generated during the process. Bugs can be filed independently. Separate stories can be filed as well during testing.

In order to categorize a story as complete, it must be correctly functional. In order to achieve this we will define acceptance criteria then subsequently execute acceptance tests.

**Acceptance Criteria**

Acceptance criteria are the "conditions that a software product must satisfy to be accepted by a user, customer or other stakeholders." (Microsoft Press).

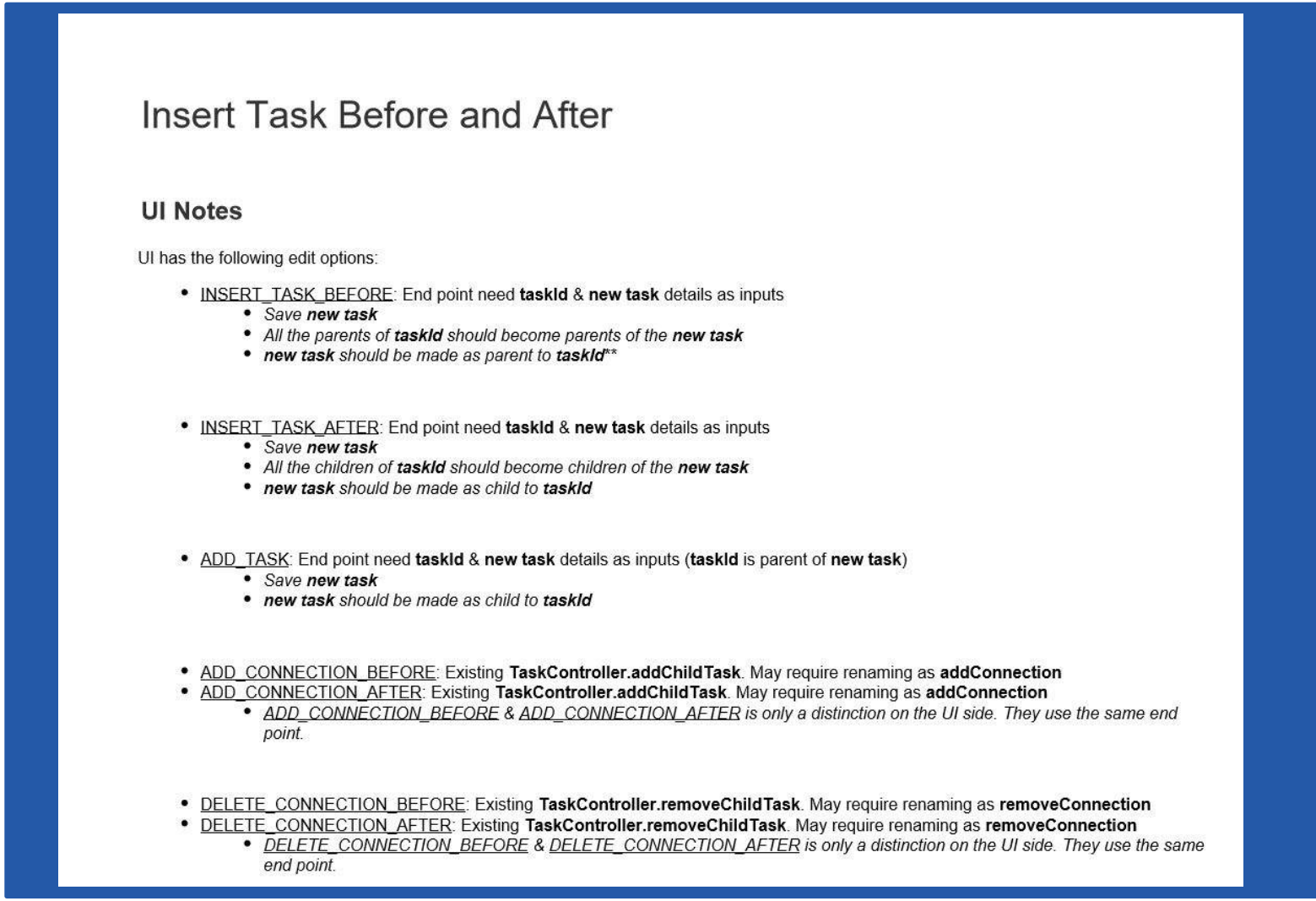
Acceptance criteria is simply a general statement that summarizes what must be tested in order for the story to be functional. It is important to note that acceptance criteria is not specific tests, but it serves as the general guide to derive specific scenarios for test. For example:

> *"The user should be able to login to the system successfully and then see his name and profile in the right side of the screen. He should be able to log out of the system as well and his profile should not appear."*

# RECOMMENDATIONS FOR AGILE DOCUMENTATION

## UI Storyboarding example

- Freeform



### Insert Task Before and After

**UI Notes**

UI has the following edit options:

- INSERT_TASK_BEFORE: End point need **taskId** & **new task** details as inputs
  - *Save **new task***
  - *All the parents of **taskId** should become parents of the **new task***
  - ***new task** should be made as parent to **taskId*****
- INSERT_TASK_AFTER: End point need **taskId** & **new task** details as inputs
  - *Save **new task***
  - *All the children of **taskId** should become children of the **new task***
  - ***new task** should be made as child to **taskId***
- ADD_TASK: End point need **taskId** & **new task** details as inputs (**taskId** is parent of **new task**)
  - *Save **new task***
  - ***new task** should be made as child to **taskId***
- ADD_CONNECTION_BEFORE: Existing **TaskController.addChildTask**. May require renaming as **addConnection**
- ADD_CONNECTION_AFTER: Existing **TaskController.addChildTask**. May require renaming as **addConnection**
  - *ADD_CONNECTION_BEFORE & ADD_CONNECTION_AFTER is only a distinction on the UI side. They use the same end point.*
- DELETE_CONNECTION_BEFORE: Existing **TaskController.removeChildTask**. May require renaming as **removeConnection**
- DELETE_CONNECTION_AFTER: Existing **TaskController.removeChildTask**. May require renaming as **removeConnection**
  - *DELETE_CONNECTION_BEFORE & DELETE_CONNECTION_AFTER is only a distinction on the UI side. They use the same end point.*

# AVOID LONG TECHNICAL EMAILS


- Big pet peeve – why is this bad?
- Email list changes, can't keep track of replies, not sequential
- Attachments not in context
- Not sustainable or easily retrievable
- Not in a central location
- Ephemeral lifetime for potential long term useful information

# FINAL RECOMMENDATIONS

OBJECT COMPUTING

- Keep as much as you can on the wiki
- Avoid blog-like entries
- If something is valuable in an email, export and attach it to a wiki page with context
- Use document repo for formal documents; export from wiki into document repo
- Encourage all team members to edit and contribute
- Refactor wiki when necessary

# LEARN MORE ABOUT OCI EVENTS AND TRAINING


Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.

OBJECT COMPUTING

## CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com