

Section	Descriptions
<a href="#">LEADS Overview</a>	
<a href="#">Engineering Operations</a>	
Agile Operations	

##Agile Operations TOC: | Section | Description | |--|--| | [Sprint Workflow](#) | Documented process of our sprint workflow. This covers areas such as how the sprints are organized, what each of the states within the workflow is for, and what is expected of each of the team members throughout the Sprint | | [SDLC Workflow](#) |Software Development Lifecycle workflow. This is the documented process of how we move our work through the different stages of the development process | | [Design Workflow](#) | This workflow is used for design-centric operations and is flexible enough to allow a variety of request types to use this workflow| |[Analysis Workflow](#) |This workflow is used for technical analysis operations and is flexible enough to allow a variety of request types to use this workflow | | [Work Items Types](#) | Definitions and processes around our sprint work items| |[Terms & Definitions](#) | Documented descriptions of general Scrum Operations terms |

#Operational Principals:

**Operational Excellence**

Ability to support development, run workloads effectively, gain insight into our operations, and continuously improve supporting processes and procedures to deliver business value.

**Predictability**

Transparency and consistency in our scrum operations model predict what we can accomplish and how we support our business.

**Accountability**

Providing clear expectations within each team role to ensure all team members are on the same page about what is expected and how we can support each other effectively.

**Balance**

We are establishing processes that balance what we can accomplish versus what may be unrealistic, helping us better understand what we can deliver while continuing to support our business teams.

## Risk, Issue, and Decision Tracking

<> This isn't the greatest wiki page in the world, this is just a tribute.

##ADO Work Items

RISKS

Required fields

- Description (Text field)
- Risk Type (Drop down: Budget, Scope, Resource, Quality, Other)
- Risk Priority (1 = High, 4 = Low)
- RiskImpact (Drop down: H, M, L)
- Risk Probability (H, M, L)
- Escalate (Y/N)
- Created Date
- Due Date

Optional Fields

- Risk Trigger (Text field)
- Contingency Plan
- Mitigation Triggers

Work item flow and States

- Resolved - This item has been resolved
- Owned - An owner has been identified and will track and complete necessary actions to avoid and issue
- Accepted - We know this is a risk but chose to move forward anyway
- Mitigated - We have a plan if this risk materializes
- Open - The risk has been identified but action is pending

graph LR; N[Open] --> R[Resolved]; N[Open] --> O[Owned]; N[Open] --> A[Accepted]; N[Open] --> M[Mitigated];

**Backlog level** = Epic/Feature

**Visibility and Permissions** - Visible and able to be created by all team members

ISSUES

### Required fields

- Description (Text field)
- Issue Type (Drop down: Budget, Scope, Resource, Quality, Other)
- Issue Priority (1 = High, 4 = Low)
- Issue Impact (Drop down: H, M, L)
- Escalate (Y/N)
- Created Date
- Due Date

### Optional Fields

- Root Cause (only when closed)

### Work item flow and States

- Open - Issue is new and has not yet been reviewed
- Closed - Issues has been resolved either by corrective action or time
- Active - Issue is being worked

graph LR; N[Open] --> V[Active]; V[Active] --> C[Closed];

Backlog level = Epic/Feature Visibility and Permissions

## DECISIONS

### Required fields

- Description
- Decision Type (Drop down: Budget, Scope, Resource, Quality, Other)
- Escalate (Y/N)

### Optional Fields

- Decision Maker
- Decision Date
- Created Date
- Due Date

### Work item flow and States

- Open
- Pending Review
- In Review
- Approved
- Declined

graph LR; D[Open] --> P[Pending Review]; P[Pending Review] --> I[In Review]; I[In Review] --> Q[Approved]; I[In Review] --> x[Declined];

Backlog level = Epic/Feature

## Visibility and Permissions

- Linkage to work items and roll up
- Review meetings
  - Participants
    - TPM, EM, Others as needed
  - Frequency
    - Every 2 weeks
- Question: Include Actions and Decisions? (Decisions but not Actions at this time).
- Question: Dashboard/metrics
- Open Item: A way to pull risks into a status report.
- Open Item: Determine a way to identify what teams owns (label, naming convention, assignee...)

## Setup

### RISKS

1. In project settings go to "Project configuration" and Select "Areas"
2. Underneath the parent enter "Risks and Issues" This will create the team that will house Risks and Issues and keep from loading up the teams backlogs
3. Next go to the "go to the process customization page"
4. In the "Work item types" click on the "+ New work item type"
5. Name the item "Risk", provide a description and chose an icon and color and hit "Create"
6. On the "Layout" tab verify that the field "Description" (multiple line) exists
  1. FYI, this Maps to Risk Description multi-text field in Kendis
7. Click the "New field" tab at the top, Click on "Create a field" and type in Risk Type into the Name field, Type is a Picklist String: Budget, Scope, Resource, Quality, Other)
8. Click the "New field" tab at the top, Click on "Create a field" and type in Risk Priority into the Name field, Type is a Picklist String (1 = High, 4 = Low)
9. Click the "New field" tab at the top, Click on "Create a field" and type in Risk Probability into the Name field, Type is a Picklist String (High, Medium, Low)
  1. Probability drop down field: Kendis and ADO both have a Probability field but they are different field types. Kendis' Probability field has Low, Medium, High
10. Click the "New field" tab at the top, Click on "Create a field" and type in Risk Impact into the Name field, Type is a Picklist String (High, Medium, Low)
  1. Impact drop down field: Kendis and ADO both have an Impact field but they are different field types. Kendis' Impact field has Low, Medium, High field values. ADO has an Impact multi-text field.
11. Click the "New field" tab at the top, Click on "Use an existing field" and type in Escalate into the Name field
12. Click the "New field" tab at the top, Click on "Use an existing field" and type in Created Date into the Name field

13. Click the “New field” tab at the top, Click on “Use an existing field” and type into the Name field
  1. Optional Fields
    1. Risk Trigger (Text field)
    2. Mitigation Plan (Text field) (Already in ADO)
14. Mitigation Actions multi-text field in Kendis 1. Contingency Plan 1. Mitigation Triggers
15. Click on the “States” tab at the top, and create the following structure
  1. Proposed
    1. Open
  2. In Progress
    1. Completed
    2. Resolved - This item has been resolved
    3. Accepted - We know this is a risk but chose to move forward anyway
    4. Mitigated - We have a plan if this risk materializes
    5. Owned - An owner has been identified and will track and complete necessary actions to avoid and issue
16. Click on the process name at the top of the screen and chose the “Backlog levels” tab. “Risks” should show at the top level Backlog level = Epic/Feature
17. Visibility and Permissions - Visible and able to be created by all team members
  1. Going back to project settings click on the “Team configuration” and for the “Risk and Issues” team verify that Risk, Issues and Decisions are all visible and nothing else, then verify that all other teams do not have those items visible.

## ISSUES

1. Next go to the “go to the process customization page”
2. In the “Work item types” click on the “+ New work item type”
3. Name the item “Issue”, provide a description and chose an icon and color and hit “Create”
4. On the “Layout” tab verify that the field “Description” (multiple line) exists
  1. FYI, this Maps to Risk Description multi-text field in Kendis
5. Click the “New field” tab at the top, Click on “Create a field” and type in Issue Type into the Name field, Type is a Picklist String: Budget, Scope, Resource, Quality, Other)
6. Click the “New field” tab at the top, Click on “Create a field” and type in Issue Priority into the Name field, Type is a Picklist String (1 = High, 4 = Low)
7. Click the “New field” tab at the top, Click on “Create a field” and type in Issue Impact into the Name field, Type is a Picklist String (High, Medium, Low)
  1. Impact drop down field: Kendis and ADO both have an Impact field but they are different field types. Kendis’ Impact field has Low, Medium, High field values. ADO has an Impact multi-text field.
8. Click the “New field” tab at the top, Click on “Use an existing field” and type in Escalate into the Name field
9. Click the “New field” tab at the top, Click on “Use an existing field” and type in Created Date into the Name field
10. Click the “New field” tab at the top, Click on “Use an existing field” and type in Due Date into the Name field
  1. Optional Fields
    1. Root Cause (only when closed) this will require a rule being created based on state.
11. Click on the “States” tab at the top, and create the following structure
  1. Proposed
    1. Open
  2. In Progress
    1. Active
  3. Completed
    1. Closed
12. Click on the process name at the top of the screen and chose the “Backlog levels” tab. “Issues” should show at the top level Backlog level = Epic/Feature
13. Visibility and Permissions - Visible and able to be created by all team members
  1. Going back to project settings click on the “Team configuration” and for the “Risk and Issues” team verify that Risk, Issues and Decisions are all visible and nothing else, then verify that all other teams do not have those items visible.

## DECISIONS

1. Next go to the “go to the process customization page”
2. In the “Work item types” click on the “+ New work item type”
3. Name the item “Decisions”, provide a description and chose an icon and color and hit “Create”
4. On the “Layout” tab verify that the field “Description” (multiple line) exists
  1. FYI, this Maps to Risk Description multi-text field in Kendis
5. Click the “New field” tab at the top, Click on “Create a field” and type in Decision Type into the Name field, Type is a Picklist String: Budget, Scope, Resource, Quality, Other)
6. Click the “New field” tab at the top, Click on “Use an existing field” and type in Escalate into the Name field
7. Optional fields
8. Decision Maker
  1. Decision Date
  2. Created Date
  3. Due Date
9. Click on the “States” tab at the top, and create the following structure
  1. a. Proposed
    1. Open
  2. In Progress
    1. Pending Review
    2. In Review
  3. Completed
    1. Approved
    2. Declined
10. Click on the process name at the top of the screen and chose the “Backlog levels” tab. “Issues” should show at the top level Backlog level = Epic/Feature
11. Visibility and Permissions - Visible and able to be created by all team members
  1. Going back to project settings click on the “Team configuration” and for the “Risk and Issues” team verify that Risk, Issues and Decisions are all visible and nothing else, then verify that all other teams do not have those items visible.

## DECISIONS

### Required fields

- Description
- Decision Type (Drop down: Budget, Scope, Resource, Quality, Other)
- Escalate (Y/N)

### Optional Fields

- Decision Maker
- Decision Date
- Created Date
- Due Date

**Work item flow and States**

- Draft
- Pending Review
- In Review
- Approved
- Declined

graph LR; D[Draft] --> P[Pending Review]; P --> I[In Review]; I --> Q[Approved]; I --> x[Declined];

**Backlog level** = Epic/Feature

**Visibility and Permissions**

# ISSUES

**Required fields**

- Description (Text field)
- Issue Type (Drop down: Budget, Scope, Resource, Quality, Other)
- Issue Priority (1 = High, 4 = Low)
- Issue Impact (Drop down: H, M, L)
- Escalate (Y/N)
- Created Date
- Due Date

**Optional Fields**

- Root Cause (only when closed)

**Work item flow and States**

- Open - Issue is new and has not yet been reviewed
- Closed - Issues has been resolved either by corrective action or time
- Active - Issue is being worked

graph LR; N[Open] --> V[Active]; V --> C[Closed];

**Backlog level** = Epic/Feature **Visibility and Permissions**

# RISKS

**Required fields**

- Description (Text field)
- Risk Type (Drop down: Budget, Scope, Resource, Quality, Other)
- Risk Priority (1 = High, 4 = Low)
- RiskImpact (Drop down: H, M, L)
- Risk Probability (H, M, L)
- Escalate (Y/N)
- Created Date
- Due Date

**Optional Fields**

- Risk Trigger (Text field)
- Contingency Plan
- Mitigation Triggers

**Work item flow and States**

- Resolved - This item has been resolved
- Owned - An owner has been identified and will track and complete necessary actions to avoid and issue
- Accepted - We know this is a risk but chose to move forward anyway
- Mitigated - We have a plan if this risk materializes
- Open - The risk has been identified but action is pending

graph LR; N[Open] --> R[Resolved]; N --> O[Owned]; N --> A[Accepted]; N --> M[Mitigated];

**Backlog level** = Epic/Feature

**Visibility and Permissions** - Visible and able to be created by all team members

The work intake workflow identifies standards and processes for evaluating requests in order to use our resources to their fullest capabilities.

Section	Description
<a href="#">Objectives</a>	
<a href="#">Intake Request Workflow</a>	
<a href="#">Types of Intake Requests</a>	

- ##Objectives
- We want to help individual developers be more effective in the face of multiple streams of requests **by giving them a process with which to manage those requests.**
  - We want to improve planning across the entire organization **by collecting, centralizing, and articulating all the requests that would otherwise be hidden.**
  - We want to make prioritization an explicit conversation **by making it obvious what will get bumped if something else gets prioritized.**
  - We want to identify long-term trends and act upon them **by collecting, organizing, and analyzing the data we need during our day-to-day work.**

##Intake Request Workflow

graph LR; A[Submit Intake Request] --> B[Business Review] B --> C[Tech Review] C --> D[Feature Ranking] D --> E[Release Planning]

Task	Status	Description	Notes
Submit Intake Request	New	Intake request is submitted by a business sponsor	<ul style="list-style-type: none"><li>• EDS is using MSFT Forms for submitting requests and ADO to track the request</li><li>• <a href="https://forms.office.com/Pages/ResponsePage.aspx?id=7hWi8RBpE0Kuxqw2_cpgSGxfYAUPcmxBvgp-SkYCpk5UMk9HS1hXQldWOFJETEM2MFo0T0dUQTUxNi4u">https://forms.office.com/Pages/ResponsePage.aspx?id=7hWi8RBpE0Kuxqw2_cpgSGxfYAUPcmxBvgp-SkYCpk5UMk9HS1hXQldWOFJETEM2MFo0T0dUQTUxNi4u</a></li><li>• SLA (TBD)</li><li>• Communication plan (TBD)</li></ul>
Business Review	Active	Intake request is reviewed with the business sponsor.	<ul style="list-style-type: none"><li>• Items reviewed vs. vision &amp; strategy</li><li>• Benefits (business and ROI) &amp; Acceptance Criteria (business requirements)</li><li>• User Workflows</li><li>• UX Designs</li></ul>
Technical Review	Active	The intake request is reviewed by technology	<ul style="list-style-type: none"><li>• Solution outlined (this could include spikes to define a solution)</li><li>• Dependencies</li><li>• Risks</li><li>• Arch. Enablers</li><li>• Level of Effort</li></ul>
Feature Ranking	In Progress	In Progress	<ul style="list-style-type: none"><li>• Features ranked by business value/cost of delay</li></ul>
Release Planning	Active	Feature is ready for release planning	<ul style="list-style-type: none"><li>• Feature &amp; Story Writing</li><li>• Security Reviews</li><li>• Test Cases</li></ul>

Design States & Descriptions

State	Status	Description
New	New	The intake request is recorded and pending review by intake coordinator
Ready	Ready	The intake request is ready for review with the business sponsor (et. all)
Active	In Progress	The intake request is under business and technical review
Blocked	In Progress	The intake request is blocked or impeded and requires special attention.
Removed	Removed	The work item was de-prioritized.
Archived	Closed	The intake request is accepted by the business sponsor

##Types of Intake Requests

- **Bug:** Errors caused by defects in programming.
- **Change:** Work needed due to missing or incorrect requirements, or a change in environment.
- **Enhancements:** A small addition to existing functionality.
- **Feature:** new functionality to support business initiatives and road maps.
- **Operational Support:** A task that requires an engineer’s involvement (e.g. running a script, setting up an environment, or grabbing data).
- **Tech Debt:** Anything that slows down the delivery team from producing high quality software.

##To Do: Processes to define

- Roles and responsibilities for all participants in the intake process.
- Clarify what information needs to be submitted with the intake request (artifacts, cash benefit analysis)
- Communication plan for new work items. Avoid spamming the world and disrupt workflow unnecessarily.
- How are intake requests approved for intake?
- How are intake requests accepted?
- Change Control once work items are approved Intake Request is added to the backlog
- How is the backlog prioritized?
  - Ideally this is ongoing, and the PO has defined sprint goals and prioritized the ‘User Story Backlog’ prior to Sprint Planning.

Section	Description
<a href="#">Schedule</a>	Details the daily sprint iteration schedule.
<a href="#">Ceremonies</a>	Details all Sprint ceremonies.
<a href="#">Processes</a>	Information, guidelines, and responsibilities on sprint processes.

##Sprint Cadence 2 Weeks (10 Business Days)

Sprint Focus Breakdown

DAY	ACTIVITIES	FOCUS
-----	------------	-------

DAY 1	<a href="#">Sprint Commitment</a> , <a href="#">Team Standup</a>	Alignment
DAY 2	<a href="#">Team Standup</a>	Focus Time
DAY 3	<a href="#">Team Standup</a>	Focus Time
DAY 4	<a href="#">Team Standup</a>	Focus Time
DAY 5	<a href="#">Team Standup</a>	Focus Time
DAY 6	<a href="#">Team Standup</a>	Focus Time
DAY 7	<a href="#">Team Standup</a>	Focus Time
DAY 8	<a href="#">Team Standup</a> , <a href="#">Backlog Refinement</a>	Focus Time
DAY 9	<a href="#">Team Standup</a> , <a href="#">Sprint Planning</a>	Wrap-up & Planning
DAY 10	<a href="#">Team Standup</a> , <a href="#">Technical Planning</a> , <a href="#">Sprint Review</a>	Wrap-up & Planning

## Focus Areas

FOCUS	DESCRIPTION
Alignment	Understanding the key goals of the sprint. Coordinating your efforts and key priorities.
Focus Time	Core focus time is on committed sprint work. Activities outside of that are limited.
Wrap-up & Planning	Concluding work items for closure and planning activities work for next sprint.

##Ceremonies & Definition

Ceremony	Description
	<p><a href="#">Backlog Refinement</a>   Review, organize and clarify work items based on priority from stakeholders.   <a href="#">Planning</a>   The team reviews stories, estimate complexity, and/or request further clarification   <a href="#">Technical Planning</a>   Assign work to engineers to review the story, create subtasks, provide time estimations, and write brief notes on the implementation of the work   <a href="#">Commitment</a>   Work items are ready to be committed into a sprint. PMs, Developers, and Stakeholders sign off on work items.   <a href="#">Stand-up</a>   Daily meetings are conducted with the team in which each team member will share their updates   <a href="#">Retrospectives</a>   Scrum team and other participants meet and review the sprint work items and identify areas we did well and areas we can improve.   <a href="#">Sprint Review</a>   This optional event is where the team has an opportunity to demo their work and review what was accomplished this last sprint.  </p>

Section	Description
<a href="#">Guidelines</a>	Describes the general standards to follow during the standup ceremony
<a href="#">Post-Topic Discussion</a>	Topics that may need to be addressed that are not directly connected to the discussion agenda or immediate collaborative efforts at hand.
<a href="#">Story Review</a>	Review, organize, and clarify work items based on priority from stakeholders after the <a href="#">Post-Topic</a> portion of standup.
<a href="#">Responsibilities</a>	A breakout of the roles and responsibilities related to the standup ceremony

## Guidelines

### General standup

- Meeting should take 30 minutes or less (preferable less, Ideally ~15 minutes)
- Please allow each person to provide their updates
  - What did you accomplish yesterday?
  - What do you hope to accomplish today?
  - Do you have any [risks](#), [blockers](#), and [Post-Topic](#) items to discuss?
  - If there are other areas where they need technical help they should be taken offline to keep standup brief.

## Post-Topic Discussion

Post-Topic (or sometimes referred to as "Parking Lot" items) items are common and sometimes leak into the general updates portion of the standup ceremony. It is a collective responsibility to be conscious of what constitutes a discussion topic that should wait until all general updates have concluded before the extended discussion on key items that need to be further addressed in the standup.

### Post-Topic Discussion Etiquette

- Healthy Post-Topic discussions usually constitute things that are relevant to the entire group that is on the call.
- Ask yourself if further discussion is really required within the standup. Can it be something that we can engage in outside of the standup meeting? Perhaps in chat or within a smaller circle of the specific person(s) after the standup.
  - Be direct about what specifically you are looking for resolution on.

### Post-Topic Item Examples

1. An urgent bug was discovered and discussion is needed to establish the priority to address the bug and who would be able to look into it.
2. A team member is blocked on their work which requires some discussion of who will be assisting to help unblock them
3. Cross-team dependency callout - an element of the current work has a dependency that we need to make everyone aware of. Discussion is needed about who can coordinate and communicate with the other team.

## Responsibilities

### Technical Project Manager (TPM)

The [TPM](#) is responsible for hosting the standup meetings. If the TPM is unable to attend the call, they must delegate the responsibilities to another team member and allow them enough time to prepare.

#### Before Standup

- Remind the team to update their assigned work before standup
- Review Sprint Board prior to standup and review what has changed from yesterday
- Review the [ADO Activity](#) to see what each team has submitted the prior day. Leave comments or make notes of work items that require your input.
- Evaluate and note if any of the team members are over their [allocated capacity](#).
- Check the statuses of the releases and determine if releases need to be discussed as a [Post-Topic](#) item

#### During Standup

- Call out each team member to announce their updates
- Take note of any [Post-Topic](#) items team members may have
- Take note of any [blockers](#) or [risks](#) the team members may have

#### Post-Topic Discussion

- Enforce [Post-Topic Etiquette](#)

#### Concluding Standup

- Remind the team to use the [Retrospective Board](#) if someone worth noting arises
  - Remind the team to update their work items for the following day
- 

### Staff Engineer (Lead)

#### Before the Standup

- Note the status of current and future releases. Work with the TPM to determine if further discussion is needed as a [Post-Topic](#) item
  - Review the [ADO Activity](#) to see what each team has submitted the prior day. Leave comments or make notes of work items that require your input.
- 

### Technical Business Analyst

#### Before the Standup

- If there are stories that require a review, please declare that in the standup chat and link to the story asking for the team to review and be prepared to discuss.

#### Story Review

- Share your screen and provide an overview of the story and see if the team has any questions or is ready to provide [Story Points](#)
- 

### Team Responsibilities

#### Before the Standup

- Validate that all assigned user stories have been updated to reflect up-to-date information and time estimations
- Note what your accomplished yesterday and the key items that may be relevant for the team to know
- Note what your plan is today and what you hope to accomplish
- Note any risks or blockers that you are experiencing and make the team aware of them in order to plan accordingly
- Note any [Post-Topic items](#) you wish to discuss after the team has delivered their daily updates

#### During Standup

- Deliver your update based on the notes you compiled before the Standup meeting
  - Please keep your updates as concise as possible. The average update of each team member should be around 1 minute
  - Please call out any risks, blockers, or items that require further discussion either within the standup Post-Topic portion or in another discussion forum (Chat, Call, Scheduled Meeting, etc)

#### Post-Topic Discussion

- Please be ready to discuss your items
  - If the discussion item has the potential to take longer than the time is allotted, please be sure to be direct on what you require and who you would need to converse with outside of the standup forum

Section	Description
<a href="#">Guidelines</a>	Describes the general standards to follow during this ceremony
<a href="#">Responsibilities</a>	Team responsibilities related to the Commitment activities

---

#### ##Guidelines

#### Commitment Review & Validation

- Establish the team's available [Sprint Capacity](#) - IN REVIEW
- Work is in [Order of Priority](#)
- Work items generally meeting the [Work Item](#) required information
- Work items have [Subtasks and Estimations](#)

## Commitment Standards

- All Work items must be in [Ready](#) state. Some exceptions can be made as long as there is a plan to get them in a ready state within the same sprint day.
- Sprint capacity needs to be established
- Commitment work shall not go over the available capacities of the team or individual

## Sprint Capacity

## Responsibilities

### Technical Project Manager (TPM)

- Organizing and hosting the Commitment ceremony.
- Establish the team's available [Sprint Capacity](#).
- Make sure that all the required participants are present in the meeting.
- Communicating to the team what items are priority over others.
- Notating project action items, risks, and discussion follow-ups related and unrelated to the meeting.
- Enforcing [commitment standards](#).

### Engineering Manager (EM)

- Enforcing [Sprint Capacity](#) and making sure that the team does not get [oversubscribed](#).
- Validating that work items are clear in terms of priority levels.
- Make sure the developers understand what the sprint priorities are.
- Notating technical action items, risks, and discussion follow-ups related and unrelated to the meeting.

## Staff Engineer (Lead)

### General Responsibilities

- Speaking up if a work item is not clear or have questions or concerns.
- 

Section	Description
<a href="#">Ceremony Goals</a>	
<a href="#">Guidelines</a>	
<a href="#">Responsibilities</a>	

### Ceremony Goals

- Review stories that are marked for the upcoming sprint to make sure they are clear and generally meet the [work items criteria](#)
- If the User Story is marked for the Engineers to work on they MUST sign off that the user story is clear to each of them and generally understand how to complete the work
  - If the story is not clear to them and requires more information the story is redirected to the author for clarification
  - If the story requires a high level of research in order to better understand the complexity or more upfront planning in the form of a [Research Spike](#)

### Guidelines

#### Review and Validation

- Confirm that the work items have all the required information

#### Attendees

- Technical Project Manager (Required)
- Product Owner (Required if applicable)
- UI/UX Designer (Required If applicable)
- Engineering Manager (Required)
- Staff Engineer (Required)
- Software & Quality Engineer(s) (Required)
- Business Stakeholders (Optional)

## Responsibilities

⚠ This ceremony is option and discretionary from team to team.

Section	Description
<a href="#">Guidelines</a>	Outlined guidelines that provide clarity on how to conduct the ceremony.
<a href="#">Planning Considerations</a>	Things to consider when planning that may result in a more successful execution of the sprint
<a href="#">Creating Tasks</a>	Creating sub-tasks that break up the work item and validate the planning element of how the work will be executed.
<a href="#">Submitting Estimations</a>	Providing time estimations when tasking our the work which will measure against available sprint capacity.
<a href="#">Best Practices</a>	Tips and tricks around technical planning that contribute to a more successful outcome.
<a href="#">Responsibilities</a>	Outlined descriptions of responsibility for each of the involved roles.

## Planning Guidelines

As part of our sprint planning, we'll be grouping related items together to target for release and will be paring on estimations.

The goal is to achieve more-frequent releases that move into the higher environments sooner to enable QA and UAT within the same sprint. **Related items** can be stories that deal with the same code, database objects, or are categorically similar. Basically a group of items that make sense to work on together and deploy together.



Planning Considerations

User Stories and Bugs

- Am I including time to test my implementation and write QA notes?
- Am I including time to ask questions to teammates or the functional team?
- For database updates, am I including time to update the data layer and application code?
- Am I including time to document my work in the user story and update any related wiki items?

Research Spike

- Am I including a task and allocating time for another team member to assist with this research?
- Will this spike require a team-wide review of the research output?
- Will I need to meet with anyone outside of the team?
- How much research documentation will be required? Am I tasking that and adding the appropriate amount of time needed?

Analysis Work

- Will I need to meet with anyone outside of the team?
- Am I including a task and allocating time for another team member to assist with the analysis?
- Will I need to organize and coordinate additional requirements gathering or analysis working sessions?
- Will this work require any design or technical resources?

Design Work

- Are there existing assets that I can leverage or will I need to design them?
- Will I need to meet with anyone outside of the team?
- Am I including a task and allocating time for another team member to assist with any design research?
- Will this work require any business analysis or technical resources?

##Creating Tasks

##Submitting Task Time Estimations

##Responsibilities

Technical Project Manager (TPM)

- Owns the calendar invitation and facilitates any additional meeting logistics required.

Engineering Manager (EM)

- Provides support for the team to ensure that have what they need to review, plan, and estimate their assigned work.
- Validates the work is in the correct order of priority.
- Validates the work assigned for planning and estimation can be properly reviewed within the allocated time
- Reviews and validates the Research Spikes are properly tasked and estimated.

Staff Engineer (Lead)

- Call out any [Pair Programming Opportunities](#) for consideration
- Reviews each of the User Stories, Bugs, and Research Spikes to ensure tasks are created with time estimations.

General Responsibilities

- Review the assigned stores to ensure the request is clear.
- Plan out the work by providing tasks and task descriptions.
- Provide the estimated [Effort](#) for each of the tasks.

Section	Description
<a href="#">Guidelines</a>	General guidelines for conducting the Retrospective ceremony
<a href="#">Retro Categories</a>	
<a href="#">Best Practices</a>	Outlines the best practices in conducting a meaningful retro
<a href="#">Retro Setup</a>	Covers the general base setup of how to organize a traditional Retrospective
<a href="#">Responsibilities</a>	Outlines the team responsibilities when contributing and participating in retro

##Guidelines Sprint Retrospectives are important to have because they allow the team to reflect on how the previous sprint went based on three core criteria. What did we do well? What didn't go well? Where can we improve? Retrospective culture is based on the [Prime Directive](#).

The sprint retrospective is meant to be a safe area for the team to communicate and reflect on how the previous sprint unfolded. Below are some general guidelines to remember in order to engage in a healthy discussion around how we can continue to improve our sprint cycles and general team operations.

General Understanding

- All Inclusive - All team members, including other participants of the sprint, should be offered the ability to participate in the retrospective discussion.
- It is expected that each team member will add their own retrospective feedback during the course of each of the sprints.
- Positive environment - Healthy retrospective discussions need to remain positive while allowing the team to address challenges and pain points. Please review [Best Practices](#) on tips for keeping the discussions healthy and positive.
- Retrospective contribution and discussion should centralize around the [Standard-Retro-Categories](#)

### Attendees

- Technical Project Manager (Required)
- Product Owner (Required if applicable)
- UI/UX Designer (Required If applicable)
- Engineering Manager (Required)
- Staff Engineer (Required)
- Software & Quality Engineer(s) (Required)
- Business Stakeholders (Optional)

### Goals of this ceremony

- Review the sprint and discuss how it went for the group
- Review any process changes and validate that they worked as expected
- Maintain a Team-centric, positive, and supportive attitude

## Retro Categories

### Standard Retro Categories

Category	Description
What did we do well?	Events or activities within the sprint that we believe went well and/or provided a positive effect on work activities. <a href="#">See examples</a>
What didn't go well?	Events or activities within the sprint that we believe did not go according to plan or caused some disruption in making progress. <a href="#">See examples</a>
Where can we improve?	Areas where we believe we have an opportunity to improve and/or enhance something that is believed to generate a better result. <a href="#">See examples</a>

### Additional Retro Categories

Category	Description
Action Items	
Process Change Review	Items that resulted in a process change from the previous sprint that needs to be reviewed and discussed to see if the change is working as desired. <a href="#">See examples</a>

### Examples

#### What did we do well?

- We completed all the work items that were committed in the sprint resulting in a completed time estimation [burndown](#).

#### What didn't go well?

- We had an event that caused us to shift our focus resulting in some planning work being deferred. [What can we do better?](#)
- Our deployment process is missing a step to gather approvals before we release it into production. We should consider adding this step to our release process.

#### Actions items

- Submit a process change in our release management workflow to gather approvals before a production release.

#### Process Change Review

- We changed the time of our standup from 9AM to 8AM.
- We added semantic versioning to our release process.

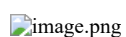
## Best Practices

- Positive reinforcement - Recognize areas within the team that the group or individual did well and show your appreciation for their contribution.
- Avoid casting blame - Refrain from singling groups or individuals out and "pointing the finger". If finding a way to express your feeling about how to communicate effectively in addressing an issue, please consult with the [TPM](#) and [EM](#)
- Be proactive - As the sprint unfolds please keep in mind things that we may want to identify in the team retrospective. Don't wait until the end because you may not remember a key item to discuss.
- Offer solutions - It's best to address a topic where we could improve in which you can offer a positive suggestion in the spirit of constant improvement and how might we be able to achieve that.

## Retro Setup

### Retrospective Board

Retrospective boards traditionally consist of a grid-like board that allows the team members to add cards to each of the discussion [categories](#)



## Responsibilities

### TPM

- ☐ Hosting the retrospective reacquiring meeting.
- ☐ Setting up the retrospective board using [Core Discussion Categories](#) as a base.
- ☐ Making sure the board is available at the beginning of each sprint
- ☐ Monitoring the retrospective board throughout the sprint.
- ☐ Reminding team members to review and contribute to the retrospective board.

- ☐ Enforce retrospective guidelines.

## Individual Responsibilities

- ☐ Contribute to the discussion with a positive, constructive, and respectful disposition.

---

Section	Description
<a href="#">Ceremony Goals</a>	
<a href="#">Guidelines</a>	
<a href="#">Best Practices</a>	
<a href="#">Responsibilities</a>	

### ##Ceremony Goals

- Review work items that have been prioritized in an upcoming sprint to ensure all the information required is present and accounted for.
- Ensure that the team(s) are on the same page.
- Ask questions if something is not clear and call out any information that should be represented in the work item
- Provide [story points](#) to work items that have all the necessary information and are clear to the team.

### ##Guidelines

#### ###General Setup

- Meeting event is hosted by the [Technical Project Manager](#)
- The Technical Business Analyst facilitates the meeting discussion and guides the attendees through the backlog

#### ###Attendees

- Technical Project Manager (Required)
- Product Owner (Required if applicable)
- UI/UX Designer (Required If applicable)
- Engineering Manager (Required)
- Staff Engineer (Required)
- Software & Quality Engineer(s) (Required)
- Business Stakeholders (Optional)

#### ###Things to look for

- When grooming the story please make sure that the story generally meets the [work items criteria](#).
- If the story is not clear and requires more information the story is redirected to the author for clarification.
- If the story requires a more detailed level of engineering research (beyond what was provided by the [Technical Business Analyst](#)) in order to better understand the complexity or more upfront planning in the form of a [Research Spike](#)

### ##Best Practices

#### ###Recording Action Items

- What is the action?
- Who will be the individuals responsible for the action?
- When is it expected to be completed?

### ##Responsibilities

#### ####Technical Project Manager

- Makes sure the event is scheduled at an appropriate time where all necessary team members are present.
- Takes notes of any callouts or dependencies that may require action and coordination with other teams
- Takes note of any action items that may result from the discussion and assigns them to the appropriate person.
- Posts the meeting notes and action items after the event is concluded
- Follows up on action items and estimations on completion dates.

#### ####Product Owner (If Applicable)

- Ensures that the initiatives and priorities are clear from a business standpoint
- Provides additional business or product context or each of the features or work items being reviewed.
- Asks questions on areas of the backlog that may not make sense and requires clarification.

#### ####Technical Business Analyst

- Facilitates the meeting and discussion topics
- Outlines the work items or features in the meeting making sure everyone understands what the objectives are.
- Records any additional notes that they need to follow up with specific to their role.
- Setups up the ability for the team to easily provide [story points](#) for each of the work items in the discussion

#### ####Engineering Manager

- Coordination with the TPM, BA, and PO\* to review the prioritized backlog items to ensure that the work is generally ready to be discussed and properly estimated using [story points](#)
- Ensures that the Software & Quality Engineers are present and engaged in the discussion.

#### ####Software & Quality Engineers

- Ask questions to get clarification if something is not clear.
- Call out any additional information that may be required for the work item to be properly estimated

Process	Definition
<a href="#">Measuring Capacity</a>	The process of establishing and monitoring a team or individual's availability to help determine how much work they can take in a given amount of time allocated in a Sprint
<a href="#">Story Points &amp; Scoring</a>	The process of establishing a general <a href="#">Level of Effort</a> using the Fibonacci scoring system.
<a href="#">Tracking Task Effort</a>	
<a href="#">Managing Spillover</a>	
<a href="#">Pull Request Review</a>	

## Working Agreement

**A working agreement is a set of guidelines or ground rules established by a team to ensure effective communication, collaboration, and productivity in their work. Working agreements can be formal or informal, written or unwritten, but they should be agreed upon by all members of the team**

## Rubicon Team Working Agreement

### Communications

- We will hold daily stand-up meetings to discuss progress, challenges, and upcoming tasks..
- We will use Teams to communicate outside of meetings, and we will respond to messages in a timely manner
- We will use email for formal communications, such as meeting agendas and project updates.
- We will use email for formal communications, such as meeting agendas and project updates..
- If a team member is unreachable, we will follow up with them via another communication channel or escalate to the Scrum Master/TPM

### Subject Matter Experts

- LPO (Lien Payoff)- Daniel Rhodes, Amber Smith
- FPO (Flooring Payoff)- Brandon Ditty
- FPR (Flooring Payoff Request)- Omar Lopez, Ashley Langhi
- VDT (Vehicle Document Tracker)- Anne Triplett, Marti Garrett
- LOF (Lifetime Oil and Filter)- Brittany Tramonte, Susie Acosta Burton

### Work Process

- We will use the Agile framework to manage our work.
- We will plan work in two-week sprints, and we will hold sprint planning and retrospective sessions at the beginning and end of each sprint.
- We will use Azure Dev Ops to manage our codebase, and we will use pull requests and code reviews to ensure code quality.
- We will use Azure Dev Ops ADO for continuous integration and delivery.

### Standards

- We will adhere to coding conventions and best practices, including writing clean, maintainable code and documenting our code.
- We will conduct code reviews to ensure code quality, consistency, and maintainability.
- We will adhere to testing standards and practices, including writing automated tests, unit tests and performing manual testing as needed.

### Tools

- Azure Dev Ops version control and code management
- Azure Dev Ops for continuous integration and delivery
- Microsoft Teams for team communication
- Azure Dev Ops for project management
- SharePoint for documentation

### Accountability

- We will hold each other accountable for meeting our commitments and deadlines.
- If a team member is struggling to meet a deadline or complete a task, they will communicate with the team as soon as possible.
- If a team member fails to meet a commitment or deadline, they will take responsibility and work with the team to address the issue.
- Ensure accountability for adherence to team agreements.

### Continuous Improvement

- We will hold sprint retrospectives to identify areas for improvement and make changes to our processes and practices.
- We will encourage feedback from stakeholders and use it to inform our development process.
- We will seek out opportunities to learn new skills and improve our technical expertise.

## Rubicon Definition of Ready

### User Stories

- The user story is clear, concise, and written from the perspective of the end user.
- Acceptance criteria are well-defined and have been reviewed by the product owner.
- Description, goals/expectations, current state, considerations, assumptions, included.
- Dependencies and risks have been identified and communicated to the team.
- Ensure all appropriate tags are added to user stories.

### Design

- The design has been reviewed and approved by the necessary stakeholders.
- The design is clear and well-defined.
- All necessary design assets have been created and are available to the team.

### Requirements

- Requirements have been identified and documented.
- Any legal or compliance requirements have been identified and documented.
- Engineering Manager/Technical Business Analyst has approved the requirements.
- Stakeholders review/validate requirements for sign-off.

### Estimation

- The team has reviewed the user story and acceptance criteria and estimated the effort required.
- The estimation has been reviewed and approved by the team.

### Dependencies

- All dependencies on other teams or systems have been identified and communicated to the necessary parties.
- The necessary approvals and/or agreements for any external dependencies have been obtained.

## Rubicon Definition of Done

### Code

- Code is written in accordance with coding conventions and best practices.
- The code has been reviewed by at least one other team member.
- All code comments have been added and are up to date.
- Code has been committed to the version control system.

### Testing

- Automated tests have been written and are passing.
- Manual testing has been performed to verify functionality.
- All defects have been identified and investigated with an action plan set in place.

### Documentation

- The user guide is up to date and accurate.
- Technical documentation has been created and is up to date.
- All documentation has been reviewed by at least one other team member.

### Integration

- Code has been integrated with the main/develop code branch.
- All code conflicts have been resolved.
- All dependencies have been identified and are up to date.

### Deployment

- Code has been successfully deployed to the appropriate environment.
- All necessary configuration changes have been made.
- Deployment has been verified by at least one other team member.
- Test automation has been validated to build stability (future state)

### Review

- The Engineering Lead/ TBA have reviewed and accepted the work.
- The acceptance criteria have been met.
- Completed UAT process and demonstrated the work to the stakeholders.

⚠ This proposal is in DRAFT state.

Regular updates to active work are critical to ensure progress or risks are called out for the team. Below is an example of a good daily update to an active story

8/10/2022

- Met with Roger and aligned on the frontend user story and how we will test the feature end to end. More information on that is forthcoming
- Wired up the POST API endpoint for /users.
- Unit tests were added to the /users endpoint.
- Story progress is looking good. No Risks.

NOTE: All comments in work items roll into the Teams channel, where everyone can review daily updates. This channel is usually called "DEVOPS Work Feed"

⚠ This proposal has been adopted as of 10/15/2022 and more information will be added.

The team will use story points and not track hours in sprint 56(2022).

- Engineers will be using the standard refinement process to set story points.
- Engineers will assign themselves to a story and task it out starting at the beginning of the sprint
- As the engineer finishes the story, they will assign themselves to the next story continuing the same process listed above.

⚠ This document is not currently practiced with the comment work items as of 10/1/2022. the process will remain discretionary from team to team. Please refer to Story Points and Velocity if this process does not apply.

Section	Description
<a href="#">Establishing Capacity</a>	Describes how to use Azure DevOps to establish the team member available <a href="#">sprint capacity</a> .
<a href="#">Reviewing Team Capacity</a>	Describes how to review capacity with in an active sprint.
<a href="#">Reviewing Individual Capacity</a>	Reviewing each team member's capacity versus the remaining work they have assigned.
<a href="#">Measuring Capacity Per Day</a>	Measuring capacity per day helps with accuracy to make sure the team has enough time for items that are not accounted for within their sprint

## Establishing Capacity

1. Days off for each of the team members will account for the time they will not be available. This reduces their capacity based on their capacity per day (example: 2 days off with 6 hours per day will be 12 hours subtracted from their overall capacity)
2. Team days off will apply the days off for all team members and reduce their capacity based on each of their established capacity per day.
3. Activity allows the TPM to know what type of work is provisioned for the sprint for closer monitoring and planning
4. Capacity per day is the measurement of how many hours a person has available to work within the sprint per day.

## Reviewing Team Capacity

1. The burndown chart shows how capacity is affecting the remaining time estimated for each of the work items within the sprint.
2. Work capacity is the cumulative measurement to indicate if the team is working within the allocated capacity limits against the remaining time estimated all the work items within the sprint
3. Work by Activity provides a more detailed view of remaining capacity by work type.

## Reviewing Individual Capacity

1. Team Member 1 has no work assigned that has tracked time. They have 12 hours of capacity remaining in the sprint and appear to have the bandwidth to take on work
2. Team Member 2 has 41 remaining hours on their assigned work with only 28 hours of availability for the rest of the sprint. Team Member 2 is oversubscribed and in danger of not finishing his assigned work. This would be a clear [risk indicator](#) in which the TPM to reach out to Team Member 2 to find what challenges they are running into and how they can help.

## ##Measuring Capacity Per Day

Section	Description
<a href="#">Story Point Precision</a>	
<a href="#">Scoring Considerations</a>	
<a href="#">Fibonacci Scale Chart</a>	

## Story Point Precision

graph TD; A[Sufficiently Precise ] --> B[1]; A --> C[2]; A --> D[3]; A --> E[5]; A --> F[8];

G[Sufficiently Broad ] --> H[13]; G --> I[20]; G --> J[40]; G --> K[?];

## Scoring Considerations

Please consider these areas when scoring the request based on [Risk and Effort](#)

### Health & Clarity

- Does the work item meet some or all of the [work item criteria health indicators](#)?
- Are the requirements clear enough to know the level of effort to achieve what is being asked?

### Testability

- Is the work items easily testable to ensure the work produced meets the acceptance criteria or request?

### Coding Impact

- If this is a user story, what is the estimated complexity of code that may need to be added, changed, or removed?
- Is this net new code or will the majority of the changes require modification of existing code? Both?
- Do you anticipate any refactorization within the project to achieve the requested?

### Technical Dependencies

- Are there any dependencies with this work within other predecessor work?
- Are there any new libraries, technologies, or environmental changes that have to be added, change, or removed?

### Operational Dependencies

- Are there any cross-team dependencies we would have to coordinate with to accomplish the request?

Notes for revision:

### ##Risk

- consideration 1
- consideration 2

### ##Effort

- consideration 1
- Scores above 13 points should be considered for breaking down into smaller stories

△ This document is not currently practiced with the comment work items as of 10/1/2022. the process will remain discretionary from team to team. Please refer to Story Points and Velocity if this process does not apply.

##Tasks and Effort Layout This is an example of a task that has some general information around what is to be done.

#	Title	Description
1	Original Estimation	The amount of time total you think is needed to complete the specific task
2	Remaining	The amount of time you think is remaining in order to complete the task
3	Completed	The amount of time you have spent on this task.

△ It is okay to increase the remaining time if more time is required.

△ It is okay to record more time spent than originally estimated.

### ##Examples

###Example 1 This is an example of what it looks like to provide an initial effort within a task

###Example 2 This is an example where I needed to spend more time on this task and still have one 1 hour estimated to complete the work.

###Example 3 This is an example where my original estimation was off and needed to adjust my new estimation by increasing the amount of time that I feel is remaining with this task. I also added some time that I spent working on this task and factored this into the estimated remaining time.

###Example 4 This is an example where the task was clearly under-estimated and would require an inquiry about what the complication was for this effort and how we could better task this effort in the future for better planning and forecast ability.

Section	Description
<a href="#">Guidelines</a>	
<a href="#">Review Process</a>	
<a href="#">Best Practices</a>	
<a href="#">Responsibilities</a>	

### ##Guidelines

#### Time Tracking

- Incorporate subtask of dev review (1 story point?)

#### Asking for Changes (PR Reviewer)

- Suggest vs Request
  - Suggest = Question, small refactor with maybe some context, non-blocking, casual
    - ex: I suggest if you did A, it would be better for B. What do you think?
  - Request = Harder rule, going against standards, "Changes Requested" that prevents approval
    - ex: This function should be refactored, not optimal. This code is breaking. Too long of comments, etc.

#### Finding Additional Work In PR (PR Reviewer + PR Creator)

- Refactors / Scope
  - Just make notes / add work item to ADO
  - Prefix comment with 'TODO - ...'
  - If possible create work item, tag as tech debt

#### Nitpicking Code / Being Too Specific With Changes (PR Reviewer)

- Just make sure code is readable, functions well, not redundant.
  - Ex: Object.assign() vs {...someVar} = Nitpicking / Preferential (avoid this, although it could be a suggestion instead of a request)

### ##Review Process

#### PR Sizing and Review Process

Size	# of Files	Lines of code	Process
SMALL	1 - 3	20-100	

Size	# of Files	Lines of code	Process
MEDIUM	3 - 5	100-150	PR creator will determine if the complexity is great enough that it warrants more than one required reviewer
LARGE	5 - 8	200+	Requires a call with the reviewing to walk through the code. Changes of this size should be called out and the reviewer should have a task allocated for reviewing the changes

Process When Reviewing (PR Reviewer)

- Any FE changes must be tested locally before being merged
  - PR Reviewer must pull FE Changes and test locally after the PR Creator has done the same (try to be thorough)

After PR Has Been Approved

- Merging of branches responsibility should still belong to tech lead/senior developer.
  - Andy will still be the only one merging into environment branches (develop/main), subject to change as the team grows but we don't want to introduce too much right now.

##Best Practices

- Efficient to have a quick call with all developers to review changes for large features
  - Ex: New Lien Flow + work
  - I will develop a set of criteria for us to review once the proposal is in place (Keep the criteria general)
- When PR's are large or complex, it helps to have a quick 1-on-1 or High-Level breakdown of what is happening in the PR. EX: 10+ files changed, going over changes in a call will speed the PR process up

Section	Description
---------	-------------

[What is Spillover?](#)

[How does DevOps handle it?](#)

[ADO Gotchas](#)

[Best Practices](#)

△ Azure DevOps way of managing work that does not get completed is very different from the JIRE 'Closure' concept. Please note this difference and review the best practices to ensure work can be planned to ensure it could be completed within the sprint.

##What is Spillover? Spillover is the result of work that was partially completed but has remaining tasks that require the work to "spill over" into the next sprint. Ideally, spillover should be avoided as much as possible. Work items should be designed to be completed within the sprint.

##How does DevOps handle it? If a work item is introduced into a sprint and one or more of its tasks are closed, the work item will be anchored to the sprint it was introduced in. If the sprint closes and the work item still has remaining tasks, the tasks are moved to the next sprint.

The diagram below demonstrates how work items can span across sprints if spillover tasks remain uncompleted.



SPRINT 1

SPRINT 2

SPRINT 3

### USER STORY 1

TASK 1

TASK 3

TASK 3

TASK 2

TASK 4

TASK 4

TASK 4

ACTIVE

ACTIVE

CLOSED

### USER STORY 2

TASK 1

TASK 2

CLOSED

### USER STORY 3

TASK 1

TASK 3

TASK 3

TASK 2

ACTIVE

CLOSED

#### ##ADO Gotchas

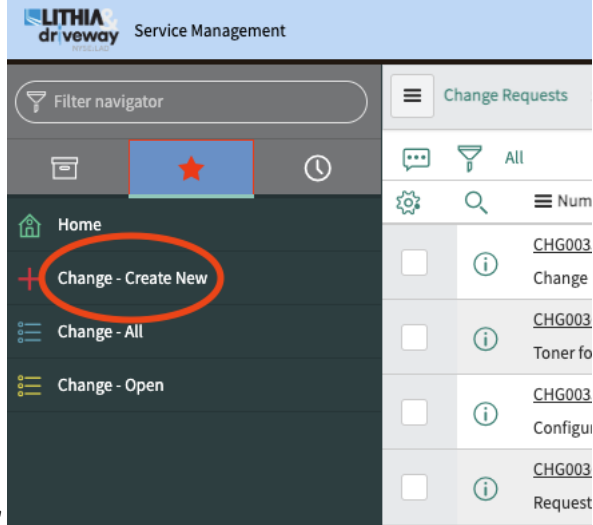
- If you have a work item that is anchored in a previous sprint, that work item is treated as THE SAME work item in future sprints. Avoid changing the state in a previous sprint thinking the work item is different.

#### ##Best Practices

- Goal is to complete a work item in the sprint assigned
- When a work item is not able to be completed, make this visible as early in the sprint as appropriate
- 

For any production release, it is Lithia practice to submit a CAB ticket and attend the CAB meeting (usually on Fridays). The purpose is to ensure that other teams are aware of the changes being done in Production.

#### Process



Navigate here: [Service Now - New Requests Menu](#) Click '+ Change - Create New'

# Standard or Emergency Release

- **Standard Release:** A normally scheduled release
- **Emergency Release:** A release that we need to get out immediately due to Business Needs

Field	Value	Type	Standard	Category	Enhancement	Risk	Depends on release, but AllPay is usually 'Low'	Impact
Multiple Stores	Assignment Group	Application Development		Short Description	production release v.X.X.X	Description	Give product name/version, as well as where release notes are located	

**Change Request**  
CHG0037522

Follow Update Save

---

New ✓ Assess ✓ Authorize ✓ Scheduled ✓ Implement ✓ Review ✓ Closed

Number	CHG0037522	State	Closed
Type	Standard	Risk	Low
Service		* Impact	Multiple Stores ▼
Service offering		Planned start date	12/05/2022 12:53:05 PM
Category	Enhancement	Planned end date	12/05/2022 12:53:11 PM
Configuration item		Assignment Group	Application Development ⓘ
Requested by	Richard McClure ⓘ	* Assigned to	Richard McClure 🔍 ⓘ
Approved by	Andy Barnes ⓘ		
* Short description ⓘ	All Pay Production release App v1.32.0		
Description ⓘ	All Pay Production release App v1.32.0 notes available here and include URL for release on Wiki : <a href="https://dev.azure.com/LithiaMotors/Advanced%20Incentive%20Program/_wiki/wikis/Advanced-Incentive-Program.wiki/5201/Release-v1.32.0">https://dev.azure.com/LithiaMotors/Advanced%20Incentive%20Program/_wiki/wikis/Advanced-Incentive-Program.wiki/5201/Release-v1.32.0</a> NOTE: The actual items in this release may change between now and release, and will be updated in the wiki above.		
Created by	RichardMcClure@lithia.com		

Planning\* CAB Conflicts Notes Closure Information

release notes are located. |

New ✓Assess ✓Authorize ✓Scheduled ✓Implement ✓Review ✓Closed

NumberCHG0037638

TypeEmergency

Service

Service offering

CategoryEnhancement

Configuration item

Requested byRichard McClure ⓘ

Approved byAndy Barnes ⓘ

\* Short description ⓘ

📎

DescriptionAll Pay Production release App v1.34.0 notes available here and include URL for release on Wiki :  
[https://dev.azure.com/LithiaMotors/Advanced%20Incentive%20Program/\\_wiki/wikis/Advanced-Incentive-Program.wiki/5301/Release-v1.34.0](https://dev.azure.com/LithiaMotors/Advanced%20Incentive%20Program/_wiki/wikis/Advanced-Incentive-Program.wiki/5301/Release-v1.34.0)  
NOTE: The actual items in this release may change between now and release, and will be updated in the wiki above.

Created byRichardMcClure@lithia.com

StateClosed

RiskLow

\* ImpactMultiple Stores ▾

ⓘ \* PriorityExpedited for business need ▾

Planned start date12/15/2022 01:08:44 PM

Planned end date12/15/2022 01:08:48 PM

Assignment GroupApplication Development ⓘ

\* Assigned toRichard McClure 🔍 ⓘ

##After Submit |Status | Description | |Request Approval | Once a CAB request is submitted, be sure to then click Request Approval. This will forward the request to the manager. The manager will then approve the request.

Once approved, the ticket will be 'Scheduled', then when its implemented, Click the button to 'Implement'.

At this point, once the release to production is complete, close the ticket. For emergency releases, you'll need to go to CAB to talk about the release and what it entailed.

⚠ This proposal is in DRAFT state.

SDLC Flow Diagram

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[DEV REVIEW] D --> C D --> E[CLOSED]

graph LR; A[Release Version Cut];

graph LR; A[QA READY] --> B[UAT READY] --> C[PROD READY] --> D[CLOSED];

NOTE: BLOCKED is not represented in this flow

NOTE: This workflow applies to most User Stories and Bugs

Development States & Descriptions

State	Status	Description
<a href="#">New</a>	Proposed	The work item has been created but is not fully ready to be committed.
<a href="#">Ready</a>	Proposed	The work item has been reviewed and is accepted as ready for commitment.
<a href="#">Active</a>	In Progress	The work item is assigned and actively being worked on.
<a href="#">Review</a>	In Progress	The work item requires review from another developer.
<a href="#">Blocked</a>	In Progress	The work item is blocked or impeded and requires special attention.
<a href="#">Closed</a>	Closed	The work has been deployed to the dev environment.

Release States & Descriptions

State	Status	Description
<a href="#">QA Ready</a>	In Progress	A release has been cut and this work item is ready for QA.
<a href="#">UAT Ready</a>	In Progress	The work has passed QA and is ready for UAT.
<a href="#">Prod Ready</a>	In Progress	The work item needs to be deployed into production.
<a href="#">Blocked</a>	In Progress	The work item is blocked or impeded and requires special attention.
<a href="#">Closed</a>	Complete	Complete - No further work is required. Sign-off needed to close.
<a href="#">Removed</a>	Removed	--
<a href="#">Archived</a>	Removed	--

State	Status	Description
Blocked	Blocked	The work item can't proceed because of an impediment that is blocking the way forward. This status requires special attention and should be brought up in stand-up or as early as possible.

Section	Description
<a href="#">Guidelines</a>	
<a href="#">Responsibilities</a>	

## Guidelines

The Dev Blocked state is part of the "In Progress" group in our ADO process management, and represents a condition for the work item where further progress is blocked.

### Blocked State Criteria

- Generally, If there are circumstances that do not allow the specific work item to progress.
- Examples:
  - No data available for the development of the feature.
  - Development tools are not responding, preventing me from working.

## Responsibilities

- The story owner is responsible for updating the ticket and keeping the status visible.
- A comment shall be added whenever a work item goes into a Blocked state.
- The comment can include:
  - Cause of Block
  - Next steps to resolve the blockage
  - Ticket numbers if there is a helpdesk ticket entered
  - If 3rd party help is needed (DWH?) then add who is helping to resolve the condition
  - Updates as needed to maintain visibility (at least every couple of days or as appropriate)
- Upon removing or clearing the blockage, please update the work item that describes the resolution so that it can be addressed in the team retrospective.

Section	Description
<a href="#">Guidelines</a>	
<a href="#">Responsibilities</a>	

## Guidelines

The Dev Review state is a part of the "In Progress" group in our ADO process management. The developer's branch has been pushed to the repository and a pull request has been created for a developer to review it. The reviewer can either accept the changes or respond to the developer for further changes.

### General Reviewer Checklist

- An ADO work item has been associated with the PR
- Link parent user story or bug (not tasks).
- Run tests with 100% passed (Eventually when we have tests)
- un the project successfully (don't assume)
- The engineer has added QA notes
  - If the story includes DB changes, attach SQL script(s) for QA to run for testing.
  - If front-end, should be written in a way that could be passed onto the functional team for UAT.
  - Time for this activity should be factored into hour estimates moving forward.
  - Consider adding the output of QA Notes into parent story instead of child task (but also include child task to track hours for it).
- Review PR code and check for the following
  - Large blocks of commented code
  - Large functions that can easily be refactored
  - Naming conventions make sense
  - Unused variables or functions
  - etc..

For further information about working with Pull Requests beyond the standard review process, please connect with your specific team for additional information.

### ##Responsibilities

Section	Description
<a href="#">Guidelines</a>	
<a href="#">Best Practices</a>	
<a href="#">Responsibilities</a>	

△ This workflow state is shared with multiple work types: Analysis, Bug, User Story, Research, Release...

### ##Guidelines

The active state is a part of the "In Progress" group in our ADO process management. The work is assigned and is actively being worked on by a developer.

### Active State Criteria

- The work is assigned and is actively being worked on by a developer.
- If the developer is no longer actively working on the item it should be moved back to Ready
- If the developer is blocked the work item should be moved to Blocked.
- If the work item requires another developer to review it should be moved to developer review and assigned to another developer to review the work item.
- This includes Pull Requests, Developer QA, Technical Reviews

- If the work item does not require developer review, it can be moved merged into the 'Development' branch and deployed to Staging. The work item can be moved to QA Review.

## Best Practices

Regular updates to active work are critical to ensure progress or risks are called out for the team. Below is an example of a good daily update to an active story

8/10/2022

- Met with Roger and aligned on the frontend user story and how we will test the feature end to end. More information on that is forthcoming
- Wired up the POST API endpoint for /users.
- Unit tests were added to the /users endpoint.
- Story progress is looking good. No Risks.

NOTE: All comments in work items roll into the Teams channel, where everyone can review daily updates. This channel is usually called "DEVOPS Work Feed"

### ##Responsibilities

1. Individuals with assigned work in an active state should provide regular updates on their progress.

Section	Description
---------	-------------

[Guidelines](#)

[Responsibilities](#)

△ This workflow state is shared with multiple work types: Analysis, Bug, User Story, Research, Release...

### ##Guidelines

Ready state is a part of the "Proposed" group in our ADO process management. The work item has been reviewed and is accepted as ready to be committed to a Sprint.

### Ready State Criteria

- The [work item](#) has all the required information
- The work has a [score](#) indicating the risk level of effort.
- The work has [tasks](#) or agreed that it does not require any.

### ##Responsibilities

- Each of the team members is responsible to move work items to the Ready state during [technical planning](#) if all the conditions are met.

Section	Description
---------	-------------

[Guidelines](#)

[Responsibilities](#)

△ This workflow state is shared with multiple work types: Analysis, Bug, User Story, Research, Release...

### Guidelines

### New State Criteria

- The work item has been opened but requires more information before it can be placed in a [Ready](#) state and placed into a sprint.
- New work items (besides [Epics](#) and [Features](#)) should have a parent Feature to ensure better tracking and management of the work

## Responsibilities

### Technical Project Manager

- Has the ability to create new items as placeholders
- Tracks all work items in [New](#) state and keeps them on the [backlog refinement](#) queue
- Validates that the work item has the information it needs before it gets placed into a sprint.
- Has the ability to review and change the state into [Ready](#) as long as all the required information is accounted for.

### Team Responsibilities

#### ###QA Ready

- The work item has been merged into the `test` branch and deployed to the test environment.
- The work item is ready for internal QA by the TPM or BA.
- QA work should be tracked in designated User Story, not as task on development User Story.
  - This makes it easier to track QA that extends past the sprint (expected) and provides more accurate reporting around "carried over" items.
- If the work passes acceptance criteria the item will be designated with the tag "QA-Complete" the BAs/TPM will give the go ahead to merge the test branch into UAT
  - *NOTE! - BAs and Developers need to coordinate right before we cut a release from development so that the code is "Frozen" as we merge it into the test branch. Please make sure we know what items are in that release.* Please refer to the [Environment Release Versions](#)

SDLC Flow Diagram

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[DEV REVIEW] D --> C D --> E[RELEASE READY]

graph LR; A[Release Version Cut];

graph LR; A[QA READY] --> B[UAT READY] --> C[PROD READY] --> D[CLOSED];

QA workflows needs to be integrated into the SDLC, so the workflow above is the same as shown on the SDLC workflow page. Integrating QA earlier into the process allows QA to work directly with Business Analyst to help define features and stories and inject quality concerns earlier into the process. Flowing into Developer Review, QA is then able to continue to advocate for/illustrate quality concerns with Developers, and help shape the design accordingly.

Below is the SDLC workflow, but now including the QA role in the associated status.

State	Status	Description
<a href="#">New</a>	Proposed	QA works with BA and Dev to understand the requirements and add use cases applicable to our quality standards. QA creates test cases per story at this point.
<a href="#">Ready</a>	Proposed	The work item has been QA reviewed and accepted the story as ready.
<a href="#">Active</a>	In Progress	The work item is assigned and actively being worked on.
<a href="#">Developer Review</a>	In Progress	Required peer review between Dev and QA to ensure Dev testing(unit testing) has been defined and performed.
<a href="#">Release Ready</a>	In Progress	The work has been deployed to develop and is ready for release to QA. (QA has signed off at the PR). QA creates test plan based on the stories in the Release Ready Status.
<a href="#">QA Ready</a>	In Progress	A release has been cut and deployed to the Test environment.
<a href="#">QA In Progress</a>	In Progress	QA performs manual/automation regression testing, and creates or updates test automation. Bugs found will return the story back to Dev for resolution. (Want: We want to start this automation sooner, which may not be available in the current build process)
<a href="#">QA Complete</a>	Complete	QA has signed off on the release, and informs Stakeholders the release is ready for UAT
<a href="#">UAT Ready</a>	In Progress	The work has passed QA and is ready for UAT. QA may collaborate with Stakeholders as needed.
<a href="#">Prod Ready</a>	In Progress	The work item needs to be deployed into production. QA may collaborate with Stakeholders as needed.
<a href="#">Blocked</a>	In Progress	The work item is blocked or impeded and requires special attention.
<a href="#">Closed</a>	Complete	Complete - No further work is required. Sign-off needed to close.
<a href="#">Removed</a>	Removed	--
<a href="#">Archived</a>	Removed	--
Blocked	Blocked	The work item can't proceed because of an impediment that is blocking the way forward. This status requires special attention and should be brought up in stand-up or as early as possible.

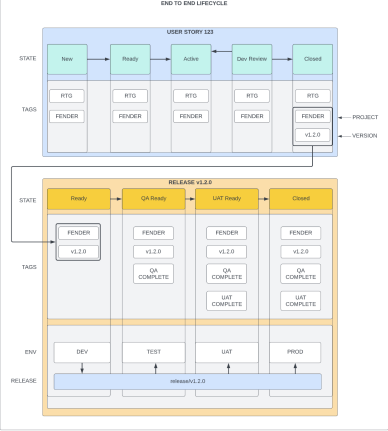
#UAT Ready

- When the code frozen work has passed QA and is ready for UAT
- NOTE! The BA/TPMs will determine when the test branch should be merged into UAT and deployed to the UAT environment.
  - Developers should not be merging into UAT without permission from the BA/TPM/SDM
  - This deployment has a checklist which and should be followed when releasing updates into UAT
- Once deployed, BA/TPM will Assign the item to a representative of the functional team and if necessary organize time for the functional team to have a session with the BA or Developer to walk through how to test and sign-off on the work.
- Once the functional approves of all changes the functional team member assigned will note their approval in the comments of the item in ADO and change the state to “Prod Ready” and assigned to the technical lead. The BA/TPM will work with the functional team to schedule a time to deploy the changes to production

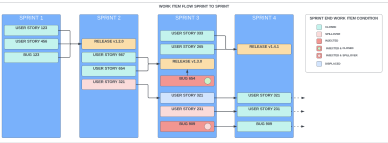
#Prod Ready

- When the work item is ready to be deployed into production
- The release developer will follow the [release checklist](#) prior to deploying to production
  - Please use semantic versioning for release tagging (ie v1.0.0 > 1.1.0) - NOTE! There could be more then one change which is why there needs to be release notes of what will be deployed so that the functional team can validate the changes in production. Release developer will verify that release notes are available in [Environment Release Versions](#) - Release notes will be shared with the team of what was released post deployment.

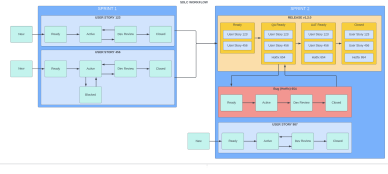
[End to End Lifecycle Diagram](#)



[Work Items Flow Sprint to Sprit Diagram](#)



[SDLC Workflow Diagram](#)



△ Analysis Work and Research Spikes follow this workflow

##Analysis Workflow Diagram

graph LR; A[New] --> B[Ready] B --> C[Active] C --> D[Review] D --> C D --> E[Closed]

Analysis States & Descriptions

State	Status	Description
<a href="#">New</a>	Proposed	The work item has been created but is not fully ready to be committed.
<a href="#">Ready</a>	Proposed	The work item has been reviewed and is accepted as ready for commitment.
<a href="#">Active</a>	In Progress	The work item is assigned and actively being worked on.
<a href="#">Review</a>	In Progress	The work item requires review the requestor.
<a href="#">Blocked</a>	In Progress	The work item is blocked or impeded and requires special attention.
<a href="#">Closed</a>	Complete	Complete - No further work is required. Sign-off needed to close.
<a href="#">Removed</a>	Removed	--
<a href="#">Archived</a>	Removed	--

###New

- The work item has been created but is not fully ready to be committed.

###Ready

- The work meets the required information outlined in the [Ready Criteria](#)

###Active

###Review

- The

###Blocked

###Closed

###Removed

###Archived

△ Design Work Items follow this workflow

##Design Workflow Diagram

graph LR; A[New] --> B[Ready] B --> C[Active] C --> D[Review] D --> C D --> E[Closed]

Design States & Descriptions

State	Status	Description
<a href="#">New</a>	Proposed	The work item has been created but is not fully ready to be committed.
<a href="#">Ready</a>	Proposed	The work item has been reviewed and is accepted as ready for commitment.
<a href="#">Active</a>	In Progress	The work item is assigned and actively being worked on.
<a href="#">Review</a>	In Progress	The work item requires review the requestor.
<a href="#">Blocked</a>	In Progress	The work item is blocked or impeded and requires special attention.
<a href="#">Closed</a>	Complete	Complete - No further work is required. Sign-off needed to close.
<a href="#">Removed</a>	Removed	--
<a href="#">Archived</a>	Removed	--

###New

- The work item has been created but is not fully ready to be committed.

###Ready

- The work meets the required information outlined in the Design WIC

###Active

###Review

- The

###Blocked

###Closed

###Removed

###Archived

Summary definition of work items

Type	Description
<a href="#">Epic</a>	A large chunk of work that can be divided into smaller, more focused features. Epics can hold multiple project <a href="#">workstreams</a> .
<a href="#">Feature</a>	A focused body of work usually constitutes one single <a href="#">workstream</a> . A feature is made up of several other work types that contribute to the completion of the feature.
<a href="#">User Story</a>	A user story is an informal, general explanation of a software feature written from the perspective of the end user. Its purpose is to articulate how a software feature will provide value to the customer.
<a href="#">Bug</a>	A bug consists of an issue that is commonly unexpected and impacts the expected behavior or criteria of what was originally requested.
<a href="#">Research</a>	Time-boxed research, and exploration to learn about the issue or the possible solutions. As a result of the spike, the team can break down the features into stories and estimate them.
<a href="#">Analysis</a>	Defined work that contributes to planning through working with the product owner to define the vision for a product and the purpose of the project.
<a href="#">Design</a>	Design work that contributes to providing a high level of resolution related to user experience and visual aesthetic requirements within a given feature.
<a href="#">Release</a>	Tracks the promotion of completed work as it moves through our <a href="#">SDLC Workflow</a> .
<a href="#">Task</a>	More tactical definition around the breakdown of a parent work item. Tasks are chunked out to consumable, measurable work output.
<a href="#">Proposal</a>	---

Item Hierarchy

graph TD; A[Epic] --> B[Feature ] B --> E[Analysis] B --> H[Design] B --> I[Spike] E --> J[Task] H --> K[Task] I --> L[Task]

A --> LK[Feature ]  
LK --> M[User Story]  
LK --> N[Bug]  
LK --> O[Release]

M --> P[Task] N --> Q[Task] O --> R[Task]

Section	Description
<a href="#">Workflow States</a>	
<a href="#">Information Criteria</a>	
<a href="#">Epic Example</a>	
<a href="#">Best Practices</a>	



Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[CLOSED]

##Overview [Atlassian](#) describes Epics as a helpful way to organize your work and to create a hierarchy. The idea is to break work down into shippable pieces so that large projects can actually get done and you can continue to ship value to your customers on a regular basis. Epics help teams break their work down while continuing to work towards a bigger goal.

##Information Criteria Consider the following when writing an Epic before any additional actions will be needed that contribute to the planning and execution of the outlined Epic.

Criteria Type	Requirement	Description
<a href="#">Description</a>	Required	Describe a high-level overview of what the Epic is for and what it is set out to accomplish.
<a href="#">Stakeholders</a>	Recommended	Identify who the main stakeholders specific to the Epic to ensure all team members know who to contact and communicate with as the Epic is being designed, planned, executed, and delivered.
<a href="#">Definition of Ready</a>	Required	Set of criteria that work items must meet before they are considered ready for development.
<a href="#">Definition of Done</a>	Required	Checklist of the necessary activities and quality criteria that must be met before a work item can be considered done.
<a href="#">Goals &amp; Expectations</a>	Required	Break down what the Epic's goals are as well as describe the expectations as a result of hitting these goals.
<a href="#">Use Cases</a>	Recommended	<a href="#">Use case</a> examples can help the development team identify and understand how the work is expected to be used.
<a href="#">Assumptions</a>	Optional	There are times when the story writer assumes something that is not validated as fact. Assumptions can be made about the work but would require others to review, validate, or correct the assumption(s) <ul style="list-style-type: none"><li>• Does this story require documentation or additional closure requirements to call this "done"?</li><li>• What areas of the system may be impacted?</li><li>• Does this story require any follow-up work or <a href="#">Technical Debt</a> that we need to identify that this work will ultimately replace?</li></ul>
<a href="#">Considerations</a>	Optional	

Epic Example

The following is a fictional example of a sales department making a request to build an application solution that will help make the review and approval process of sales commissions more efficient.

Description

The Sales team has asked for an application tool that will better manage the approval process of their sales records so that the sales team can receive official approval in order to get a commission for their individual sales

Stakeholders

Stakeholder	Role	Responsibility
Jane Smith	Executive Sponsor	Project Reviews, and approvals
Jon Doe	Business Owner	Requirements Reviews, and approvals
Grace Adams	Business <a href="#">SME</a>	Requirements Gathering Resource

Definition of Done

- The application has completed development and passed quality control
- The application has passed user acceptance testing
- Training has been administered with the Sales team on using the application
- Application is live in production and working as expected based on the requirements established

Goals and Expectations

Goal	Expectation
Allows the sales team to review and approve requests	The expected result of this will allow the Sales team to easily go to the application to review/approve requests more efficiently

Use Cases

Scenario	Description
Sales Manager will review and approve a team member's monthly sales	In this case, A Sales Manager would go to the application and choose a team member with the goal of reviewing their sales from the last month. The sales manager would review all the sales records and approve each of the sales.
Sales representative will submit their sales transactions for review/approval	

Assumptions

Assumption	Action
It is assumed that the sales records will be loaded each morning by the Lead Sales Representative	Follow-up is required to verify this assumption and document the expectation
It is assumed that the sales team will know how to get to the application to review their assigned sales records.	Training of the new tool will be required to ensure the sales team knows where to go and how to use the application
It is assumed that the Business Owner will only allow the sales staff access to the application and will require security roles to grant access to the same team.	Follow-up is required to verify the security requirement and move this to an expected requirement

Considerations

- Be aware that this application will replace their current solution which is an excel spreadsheet.
- Please consider that there will be a growing number of team members that will use this application over time.

##Best Practices

Section	Description
---------	-------------

- [Workflow States](#)
- [Information Criteria](#)
- [Feature Example](#)
- [Best Practices](#)

△ Features adhere to the Epic Workflow States

Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[CLOSED]

##Overview

##Information Criteria Consider the following when writing a Feature before any additional actions will be needed that contribute to the planning and execution of the outlined Feature.

Criteria Type	Requirement	Description
<a href="#">Description</a>	Required	Describe a high-level overview of what the Feature is for and what it is set out to accomplish.
<a href="#">Definition of Ready</a>	Required	(DOR) The Definition of Ready includes criteria related to the clarity and completeness of the user story, the acceptance criteria, and any dependencies or risks associated with the story. It also includes criteria related to the design and requirements of the story, as well as the estimation of effort required to complete it.
<a href="#">Definition of Done</a>	Required	(DoD) The Definition of Done includes criteria related to coding standards, testing, documentation, integration, and deployment. By establishing a clear Definition of Done, the development team and stakeholders have a shared understanding of the expected outcome for each user story or task, ensuring that the team is delivering high-quality work that reflects the scope of the <a href="#">business requirements document</a>
<a href="#">Goals &amp; Expectations</a>	Required	Break down what the specific goals well as describe the expectations as a result of hitting these goals.
<a href="#">Use Cases</a>	Recommended	<a href="#">Use case</a> examples can help the development team identify and understand how the work is expected to be used.
<a href="#">Assumptions</a>	Optional	There are times when the feature assumes something that is not validated as fact. Assumptions can be made about the work but would require others to review, validate, or correct the assumption(s) <ul style="list-style-type: none"><li>• Does this feature require documentation or additional closure requirements to call this "done"?</li><li>• What areas of the system may be impacted?</li><li>• Does this story require any follow-up work or <a href="#">Technical Debt</a> that we need to identify that this work will ultimately replace?</li></ul>
<a href="#">Considerations</a>	Optional	
<a href="#">References &amp; Artifacts</a>	Optional	Links that help the developer better understand what the feature is for and what they can compare it to for context. Visual aids, diagrams, and mockups are also encouraged to help communicate the feature and provide better guidance.

##Feature Example

Description

The sales team needs to be able to see all of their submitted sales records within the sales commission approval application.

Definition of Done

- Users can log into the Sales Commission App and view all the records associated with their user id.
- Users can select a sales record and view the details
- Users can edit a sales record details and resubmit
- Users can withdraw a sales record from their submissions

Please review the project [Business Requirements Document](#) for further information of the scope of this feature

Goals and Expectations

Goal	Expectation
Ability to View/Edit/Remove Sales Records	The expectation is that the user will be able to easily see all the records associated with them and manage all their submission details easier than they are currently today.

Use Cases

Scenario	Description
Sales Rep logs into the app and goes to the sales records summary table	

Assumptions

Assumption	Action
------------	--------

## Assumption

It is assumed that the sales records will be loaded each morning by the Lead Sales Representative

It is assumed that the sales team will know how to get to the application to review their assigned sales records.

It is assumed that the Business Owner will only allow the sales staff access to the application and will require security roles to grant access to the same team.

## Action

Follow-up is required to verify this assumption and document the expectation

Training of the new tool will be required to ensure the sales team knows where to go and how to use the application

Follow-up is required to verify the security requirement and move this to an expected requirement

## Considerations

- Please consider that this feature will have several stories to break up the work in smaller portions

## Section Description

[Workflow States](#)

[Information Criteria](#)

[User Story Example](#)

[Best Practices](#)

△ User Stories adhere to the SDLC Workflow

## Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C--> D[REVIEW] D --> C D --> E[CLOSED];

##Information Criteria | Criteria Type | Requirement | Description | |-----|-----|  
-----|-----|  
-----| | [User Personas](#) | Required | For Whom? If there are multiple end users, consider making multiple stories. Provide context and give an explanation of why. What is the goal of this story and the expectation of the result of it beyond the general persona need? | | [Goals & Expectations](#) | Recommended | What are the general goals that this story should achieve once the work is complete? | | [Acceptance Criteria](#) | Required | The story's acceptance criteria need to be clear as to what is acceptable so that we can define the work as "done" when it meets all of the conditions written. | | [Parent Feature](#) | Required | User stories must have a parent link that tie back to the feature. | | [Current State](#) | Recommended | What is the current "as-is" state of the work item before the desired work take effect. Explained what the future state will be after the desired work is completed. | | [Considerations](#) | Recommended |

- Does this story require documentation or additional closure requirements to call this "done"?
- What areas of the system may be impacted?
- Does this story require any follow-up work or "technical Debt" that we need to identify to ensure things we are not able to get to be tracked?
- Do we need diagrams or data flow to support this story? - Called out in associated [Analysis Work](#)

| | [Test Cases](#) | Recommended |

- Please provide if needed test cases in which the story shall be testing
  - Write out ways in which one would try and break it and cause it to not work properly
  - If no test cases are needed, please indicate that so the QA Rep can review and validate this.
- | | [Assumptions](#) | Optional | There are times when the story writer assumes something that is not validated as fact. Assumptions can be made about the work but would require others to review, validate, or correct the assumption(s) |

## User Story Example

The following is a fictional example of a sales department making a request for the ability to filter sales records by each of the Sales Representatives.

## Description

As a Sales and Support Specialist, I want the ability to manage the sales records and ensure the records are displaying correctly on my screen when I filter them by Sales Representative. I would expect that when I choose a person to filter by, it would show me all sales records that person has submitted

## Goals and Expectations

### Goal

### Expectation

Filter sales records by individual Create the ability for the Sales Support Specialist to filter by Sales Representative

## Goal

Clear and easy to understand

## Expectation

The UI is simple enough to understand how to easily filter by Sales Representative

##Best Practices Consider the following during the [Planning](#) ceremony before the user story is assigned to a team member for subtasks and further evaluation within the [Technical Planning](#) ceremony.

- If there are conditions where there are multiple steps in the process of completing the work in a story, call that out in the description to provide even greater clarity around dependencies or other condictions that may need to be factored in when planning out the work of the story. Write a story for each step in a larger process.
- Listen to feedback
  - Talk to your users and capture the problem or need in their words. No need to guess at stories when you can source them from your customers.
- Outline subtasks or tasks
  - Decide which specific steps need to be completed and who is responsible for each of them.
- Time
  - Since stories should be completable in one sprint, stories that might take weeks or months to complete should be broken up into smaller stories or should be considered their own feature or epic.

##Reviewing the output of the work conducted within the User story

- Dev Review Stage
  - User stories normally have a Dev Review process in which a fellow developer will review the pull request of the code that was written or something along those lines to validate that the work delivered meets the engineering standard we have set. Please review the [Developer Review](#) stage gate.

Section	Description
<a href="#">Workflow States</a>	Defined states of how a bug flows through the SDLC
<a href="#">Information Criteria</a>	Outlined criteria and descriptions of each
<a href="#">Bug Example</a>	Example of detailed bug might
<a href="#">Best Practices</a>	Stated best practices when reporting and submitting a bug
△ Bugs adhere to the SDLC Workflow	

## Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[REVIEW] D --> C D --> E[CLOSED];

##Information Criteria | Criteria Type | Requirement | Description | |-----|-----|  
-----|-----|  
-----| | [Description](#) | Required | Describe what the issue is and the impact it is having. | | [Repro Steps](#) | Recommended | Please list the steps required to reproduce the issue. Outlining this information will help the developer quickly recreate the issue in an effort to diagnose and prescribe a fix. | | [System Info](#) | Recommended | Describes the environment or conditions in which the bug is occurring. Adding this information will help the developer better provide clues as to the possible circumstances that may be causing the bug. | | [Additional Clues](#) | Optional | Provide any additional information that may be helpful to the developer in troubleshooting or diagnosing the issue. Additional clues benefit the developers, which can focus on the solution. |

## Bug Example

The following is a fictional example of a sales department connecting to application XYZ and getting an error in which data is not loading on the screen.

### Description

The XYZ application generates an error when the page loads "Unable to load data" in the browser. The error appears to be impacting the entire user base, and they cannot complete their business tasks.

### Repro Steps

1. Go to the XYZ application URL
2. The browser will load the application UI elements
3. After the UI elements load, the application shows an error "Unable to load data".
4. Refresh the browser does not fix the issue.

### System Information


1. This only happens in the DEV, TEST, and UAT. the PROD environment is unaffected.
2. Tested with both a Mac and PC with the same issue.
3. Tested with multiple browsers with the same issues.

### Additional Clues

1. I opened the developer tools and found that a 500 error was returned from the /data endpoint
2. I tested with multiple browsers and other users and got the same issues.
3. The issue may come from the API, not the front end.

Best Practices

- 1. Screenshots and visual aids
- 2. A little investigation and troubleshooting go a long way
- 3. Describe the impact and context of the urgency of fixing the issue

Section	Description
<a href="#">Workflow States</a>	Defined states of how a bug flows through the SDLC
<a href="#">Information Criteria</a>	Outlined criteria and descriptions of each
<a href="#">Best Practices</a>	
	Research Items adhere to the Analysis Workflow

##Research Spikes

The need for a spike is usually identified by the team during our sprint planning sessions and engineering team meetings but it might also arise from other discussions. Once agreed, it is added to our agile task tool along with a description and a list of starter questions to guide areas of research. We categorise a spike as a chore because this story type does not involve a story point estimate. We'll also tag it with the label 'spike' along with project specific labels to keep track of it and distinguish it from other chores.

Please refer to [this article](#) on best practices and usage of Spike research

Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[REVIEW] D --> C D --> E[CLOSED];

##Information Criteria   Criteria Type   Requirement   Description	----- ----- -----
-----	<a href="#">Description</a>   Required   Write a clear description of the work that will be conducted and why it is being conducted.
	<a href="#">Goals and Expectations</a>   Recommended
	<a href="#">Acceptance Criteria</a>   Required   What are we trying to learn, discover, or achieve with the Spike? The research and findings should be well documented and attached to the story before the SPIKE is moved into Dev Review

Best Practices

Outline subtasks or tasks

Decide which specific steps must be completed and who is responsible for each.

Time

Time is a touchy subject. Many development teams avoid time discussions, relying instead on their estimation frameworks. Since stories should be completable in one sprint, stories that might take weeks or months to complete should be broken up into smaller stories or should be considered their own epic.


Testing Impact

It's a good practice to consider if the research will result in future work involving testing and quality assurance.

- Will there be any testing elements involved in the research?
- Test cases and scenarios that may be involved

##Reviewing

- Research Spikes normally have a Review process in which a team member will review the research documented (usually attached to the spike or linked elsewhere)

Section	Description
<a href="#">Workflow States</a>	
<a href="#">Writing an Analysis Item</a>	
<a href="#">Review and Scoring</a>	
	Analysis Requests adhere to the Analysis Workflow

Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[REVIEW] D --> C D --> E[CLOSED]

##Writing an Analysis Item Consider the following when writing these work items before they are moved into the planning ceremony and assigned for sub tasking and estimations

- Write a clear description of the work that will be conducted and why it is being conducted.
- Is there an associated Epic, Feature, User Story, or Bug for this analysis work?
  - Does the associated feature or request have acceptable information for proper requirements gathering?
  - Does the associated feature, epic or request have clearly written expectations of what the result of the requested work will produce?
  - Does this work require engineering resources for requirements gathering, technical analysis, extended consultation and/or review?
  - If yes, please add subtasks with proper time allocated and find a resource to assign it to.
- Clear definition around what the resulting output will be
  - Provide information around the expected delivery results of the BA work item.
    - Example: Technical analysis will result in a data dictionary and diagram workflow representing how the data flows for X to Y to Z

##Grooming and evaluating user stories before sprint commitment Consider the following when reviewing and planning before commitment of this work into a future sprint.

- Considerations TBD

Section	Description
<a href="#">Workflow States</a>	
<a href="#">Writing a Design Request</a>	
<a href="#">Review and Scoring</a>	
△	Design Requests adhere to the Design Workflow

Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> D[REVIEW] D --> C D --> E[CLOSED];

##Writing a Design Request Consider the following when writing user stories before they are moved into the planning ceremony and assigned to developers for sub tasking and estimations

- User personas (Required)
  - For Whom? If there are multiple end users, consider making multiple stories. Provide context and give an explanation of why. What are the goals of this story and the expectation of the result of it beyond the general persona need?
- Goals to achieve (Required)
  - What are the general goals that this story should achieve once the work is complete and working in production
- Considerations (Recommended )
  - Does this story require documentation or additional closure requirements to call this "done"?
  - What areas of the system may be impacted?
  - Do we need diagrams or data flow to support this story? - Called out in [Analysis Work times](#)
- Acceptance Criteria (Required)
  - The story's acceptance criteria need to be clear as to what is acceptable so that we can define the work as "done" when it meets all of the conditions written.
  - Does the story account for additional user story writing or user experience requirements in an existing story or feature?

##Reviewing and Scoring

Section	Description
<a href="#">Workflow States</a>	
<a href="#">Writing an Release Item</a>	
<a href="#">Review and Scoring</a>	
△	Analysis Requests adhere to the Analysis Workflow

Workflow States

graph LR; A[NEW] --> B[READY] B --> C[ACTIVE] C --> E[CLOSED]

Term	Definition
Blocker	Something that is preventing a team or individual from proceeding with their assigned task
Burndown	A burn down chart is a graphical representation of work left to do versus time.
Capacity	The amount of availability a team or individual has in a given amount of time
Cutting a Release	To create a <a href="#">code frozen</a> package of software ready for distribution and QA testing
Code Frozen	The source code of the software is locked and changes are restricted unless under rare conditions to change it.
Demos	Demonstration of work conducted within the sprint or related to our project work
Dependency	
Effort	Commonly used to track the estimated time for a work item task. See <a href="#">Tracking Task Effort</a>
Order of Priority	A term used to distinguish what work items the team should be focusing on in priority order
Oversubscribed	A term used if a team or individual has more work assigned than they can reasonably complete within a given amount of time.
Pair Program	
Risk Indicators	
Sprint	
Subject Matter Expert	
Technical Debt	
Weekly Status Report	
Work Item Criteria	
Velocity	

Questions
<a href="#">What are we using story points for?</a>
<a href="#">Why are we using time estimations to track sprint work?</a>
<a href="#">What's the process when work is not fully completed in a sprint?</a>

##What are we using story points for?

Answer:

We use Story points to measure and track the team's [velocity](#) and well as help identify work items that may be too big to commit without breaking it up.

##Why are we using time estimations to track sprint work?

Answer:

##What's the process when work is not fully completed in a sprint ####Answer

## KPIs (Key Performance Indicators) Sometimes also referred to as Metrics

This is a proposal to increase the usefulness of the work item health metric. The original metric was based on reviewing a work item and then ranking it from 1-5. (1 low → 5 very high)

The focus for our working group was to propose a user story format, and how to rank each work item. Below, you will find the proposal details:

### Problem

Work items with incomplete information can cause ambiguity, which may lead to scope creep and loss of quality, or not solve the customer's request.

#Definition The object of this metric is to ensure that the work item has the required information appropriate to the work item type. Not all items will require the same level of information and should be ranked accordingly.

#Timing and Ranking Ranking should be done at the time of refinement, with the team present, to ensure that each work item has the necessary information for story pointing. This provides actionable feedback, allowing the team to make adjustments in a timely manner.

#Work Items ##User Story User stories must clearly present what needs to be built and why. The following are the User Story field definitions:

Criteria	Description
Title	What is the goal of the story What is the business problem or opportunity?
Description	<ul style="list-style-type: none"><li>• User Persona (the who)</li><li>• Requirement (the what)</li><li>• Business Value (The why)</li></ul> Any pieces of information that can help in the execution of the story. Not all of these are required, but if applicable, can be used.
Considerations	<ul style="list-style-type: none"><li>• Links to documentation</li><li>• Workflows</li><li>• Assumptions - either need to prove or disprove</li><li>• UX Designs(figma)</li><li>• Dependencies - outside of ADO story dependencies</li><li>• Pointers or tips to implement and/or test</li></ul> What does it mean for the story to be accepted? <ul style="list-style-type: none"><li>• Must be testable</li><li>• User and Technical functional requirements</li></ul>
Acceptance Criteria	<ul style="list-style-type: none"><li>• Data requirements</li><li>• Business rules, workflows, calcs</li><li>• Roles and Permissions</li><li>• Implementation requirements (as applicable)</li><li>• Documentation</li></ul> (Happy path / unhappy path) <ul style="list-style-type: none"><li>• User Scenarios (How is the system expected to function for the user?)</li></ul>
Test cases -> Developer/QA Notes	<ul style="list-style-type: none"><li>• Technical Scenarios (How can an engineer test this functionality)</li><li>• Test Data such as sample inputs and expected outputs</li><li>• Integration Points: Developers may provide information on the integration points between different components or modules of the application.</li><li>• Known Limitations: Developers may provide a list of known limitations of the application, such as performance issues or missing features.</li></ul>

##Analysis Story Analysis stories are for researching and defining business or technical workflows, or functionality. There are instances where this work may not be completely definable, and the story's objective will be needed.

Criteria	Description
Title	What is the goal of the story What is the business problem or opportunity?
Description	<ul style="list-style-type: none"><li>• User Persona (the who)</li><li>• Requirement (the what)</li><li>• Business Value (the why)</li></ul> Analysis stories must state the "why" and "what" regarding the expected result.
Resulting Requirement	<b>Note:</b> Some analysis stories may be partially open-ended. The point of such stories will be to provide the requirement.

##Design Story Design stories are for UX design of work flows or functionality.

Criteria	Description
Title	What is the goal of the story What is the business problem or opportunity?
Description	<ul style="list-style-type: none"><li>• User Persona (the who)</li><li>• Requirement (the what)</li><li>• Business Value (The why)</li></ul>
Work Needed	What is the deliverable to be provided?
Expected Result	What is the deliverable to be provided? <ul style="list-style-type: none"><li>• ASK: Is 'Work Needed' still needed or can Expected Result be used?</li></ul>

#Release

Criteria	Description
Title	What is the target release
Description	Release stories will have target versions being released, but otherwise be very succinct.

#Spike

Criteria	Description
Title	What is the goal of the story
Description	Spike stories are technical research stories, and must have a defined goal to the story. Spikes must have a defined goal and expectation set. The description should provide the "What" and the "Why" for the purpose of the story.
Acceptance Criteria	What does it mean for the spike to be complete? <ul style="list-style-type: none"><li>Research is documented - Success/Fail of the spike</li><li>Work items defined and ready to be refined as appropriate</li></ul>
Expected Results	For some Spikes, if the goal is to conduct an experiment, then enter the success criteria in this field.

#Bug

Criteria	Description
Title	What is the defect found
Reproduction Steps	<ul style="list-style-type: none"><li>Reproducible?</li><li>Steps to reproduce?</li><li>App Version</li><li>Expected Results vs Actual</li><li>Logging if available</li><li>Developer tool information from browser if web app</li></ul> All pertinent information related to the system under test.
System Information	<ul style="list-style-type: none"><li>Browser</li><li>OS if applicable</li></ul>

#Introduction As a team we need to know how the health of our stories affects our sprint execution. This is accomplished by adding a new field to work items 'Work Item Health' and monitoring items for trends over sprints.

#What is it This is field added to all work items. The TPM will audit stories at the start of a sprint, and use the following field options:

Option	Definition
1 - Very Unhealthy	The work item is not defined well, and missing most of the required information
2 - Unhealthy	The work item has some definition that has followed process, but is still unclear and not meeting process expectations
3 - Healthy	The work item meets the process requirements
4 - Very Healthy	The work item exceeds the process requirements.

#Why it matters Having work items that are unhealthy causes avoidable delay or confusion. The goal of this metric is to help ensure stories are sprint ready with all the requirements to be successfully executed.

#Considerations Key goal is to ensure work items have clear definitions. Some thing to consider is the complexity of the work item. The more complex, the more detail needed to ensure comprehension and quality. This will require some judgement, as the expectation is not all work items need to be 'Very Healthy' if it is not required.

#Measuring Each work item at the beginning of every sprint will be measured at the time of commitment and evaluated using the criteria listed here. Over time, the engineering manager and TPM will evaluate the trends to gauge the teams work item quality, and mentor as needed.

#Example

Example	Details
Unhealthy	Little to no detail in Description or Acceptance Criteria
Healthy	Detailed requirements in the description, along with accurate Acceptance Criteria
Very Healthy	Detailed requirements with supporting images, screenshots, or attachments. Detailed acceptance criteria



Actual Example:

124658MONTEGO - SAR Database Health Analysis

No one selected

0 Comments

MONTEGO

State

New

Area

LEADS - Corporate AppsRainmaker

Reason

Moved to state New

Iteration

LEADS - Corporate AppsRainmaker 2023Sprint 2

Description

Planning

We are moving in on the Schedule Audit Review project "MONTEGO." Before getting into the project scoping effort, we need to get a high-level analysis of the SAR database and its current health and complexity.

PROJECT DOCUMENTATION

MONTEGO

Goals and Expectations

Links

1. Establish a high-level understanding of the SAR application

2. Establish a high-level understanding of the SAR database

3. Deliver a health assessment of the current state of the SAR database project

4. Deliver recommendations of the future state of the SAR database

+ Add Child

Acceptance Criteria

Actions

1. Request access to the SAR database (DEV and PROD) - Note that these are on-prem

2. Request access to the SAR application (DEV and PROD) - Note that this app resides within the Performance Dashboard

3. Review the previously completed analysis summary of SAR (Link attached)

4. Work with a member of the Super Nova team on a recorded session to give a tour of the current SAR database

Results

1. Develop a high-level assessment of the current health of the database, table architecture, and stored procedures.

2. Write out and present some recommendations related to the future of this database and how we might be able to improve it so that it is more sustainable long term

3. Please consider how we might be able to phase the recommendation in a way that will allow us to deliver value for the business along the way.

Discussion

RM

Add a comment. Use # to link a work item, @ to mention a person, or ! to link a pull request.

#Introduction As an engineering organization, it is valuable to measure the Sprint health by looking at work items committed and executed upon in a sprint. This measure, combined with others will allow the team to measure the health of the team and project.

#What is it? This metric tracks the work items committed at the beginning of a sprint, and then again at the end of a sprint.

#Why it matters Planning projects, features or sprints requires predictability of how the team executes on work items. This metric will measure the committed work at the beginning of each sprint, and then a subsequent measurement at the end of a sprint. This data helps describes how changes affected the delivery of sprint work.

#Example Work Commitment at beginning of sprint vs end of sprint

2023 Sprint Velocity Statistics		
Sprint	# Stories	# Story Points
4	29	138
Injected	3	13
Normal	26	125

- Example: Commitment for sprint 4: 26 stories / 125 Story points committed
- During the sprint there were 3 injected stories at the cost of 13pts.
- Post Analysis: These 3 stories were Spike stories as the result of Discovery found in sprint. This disrupted other work and caused continuation.

#Considerations Tracking this work allows the team and leadership to monitor for injections during a sprint, and track trends in the amount of injection occurring each sprint.

#Conclusion Since the objective is to keep injected work at zero, any occurrences should trigger the team / leadership to question if this is a priority, or if it can be moved to the following sprint.

## Cycle and Lead Time

- Intro
  - Prior to making large changes to process, procedures, org structures, or toolset it's important to establish a baseline to be able to test your hypothesis against. Without this you may be making decisions of success based on gut feel or just doing change for the sake of change. One Key Performance Indicator (KPI)/Metric that has been useful is to understand current Lead time and specific process cycle times.
- What is it:

- Lead Time is how long something takes from start to finish (Item entered into the system until it is complete. In other words, how long from requirement to value realization or the speed of your value chain.
- Cycle time is the amount of time it takes once work is started to finish the work and a subset of Lead Time. Cycle Time helps identify any inefficiencies once work is started to once it is completed.
- Why it matters:
  - These can help test hypothesis made to altering processes and procedures. What is the baseline? What is the change introduced? What is the expected outcome? What is the actual impact to Lead or Cycle Time
  - Cycle Time helps identify any inefficiencies once work is started to once it is completed. If Cycle Time is longer than 1 sprint teams will not be able to meet their sprint commitments.
  - You can utilize Cycle time to measure smaller processes within the larger process. This can be used in a waterfall portfolio as well. (how long does it take items to move from Dev to UAT or BRD signoff once requirements are complete)
- Example:
  - Lead time is the time from once you get into the queue to get your breakfast beverage until the time you have it in your hand and have paid for it. Cycle time is the amount of time once the worker grabbed the cup to start fulfilling your order until the time it's placed on the counter.
- Considerations
  - If changes are made that help cycle time but hurt lead time this change is considered bad. Pieces of the process are being optimized at the cost of the Whole process. This may occur if teams are incentivized only to make their portion of the process better without consideration for the larger SDLC.
- Related topics
  - Throughput – This is how many items were completed (processed start to finish) during a specific period of time.

graph LR; A[Item Enters ADO Backlog] --> B[Item enters sprint] B[Item enters sprint] --> C[Work Starts] --> D[Work In Progress] --> E[Work Complete] --> F[Work Delivered] ;A[Item Enters ADO Backlog] --> L[**Lead Time**] --> F[Work Delivered] C[Work Starts] --> G[**Cycle Time**] --> E[Work Complete]