MyUPMC Program Agility Handbook

Introduction

Why

One may rightly ask why we need a handbook at all when so much has been written about agile software delivery. This handbook has been constructed in an attempt to achieve the following goals.

- Firstly, to capture and document the wealth of experience and institutional knowledge among our members.
- Secondly, to provide a shared framework within which teams may innovate and experiment. Innovation always happens within a structure.
- Thirdly, to ensure program scalability and effectiveness through cross-team alignment. We must ensure that structures are in place to ensure program effectiveness no matter what our size is. Lastly, to use as an onboarding document for new members.

Sources

While other foundational works have been used in the construction and formulation of this handbook, none have been blanketly applied to our process but the principles and learnings from these diverse systems have been taken in combination with what our teams have found to work to produce a common method of working for the greater MyUPMC team.

The reader is encouraged to review linked resources to gain more familiarity with concepts and the reasoning behind the prescribed patterns described in this handbook. This work is descriptive, in that it captures how we currently work, and aspirational, in that it describes ideal patterns we hope to more fully embody with time.

The Prime Directive

Our highest value and commitment is to the iterative delivery of user-facing value on a sprintly basis with the aim of creating a transparent inspect-and-adapt feedback loop. This is the Prime Directive of the Agile Working Group (AWG). Everything else in this handbook is made in service to this value. Exceptions to the specific patterns prescribed in this handbook will surely exist but should be made in service to this value.

The individuals closest to the work are empowered and will be entrusted to make those hard decisions when they arise. If such exceptions prove, with time, to be more effective ways of working they should make their way back into this handbook as revisions in future editions to the benefit of the entire program.

Teams

Feature Teams

The core organization unit of the program is the feature team. Feature teams are the program's engine of customer delivery. Feature teams should be cross-functional in that they contain all necessary skills to deliver new cross-platform features soup to nuts. They should be long-lived, minimizing musical chairs or transient membership in order to facilitate team maturity and inter-team relationship building. Lastly, they should be semi-autonomous with empowered internal leadership with the requisite supported autonomy to make decisions closest to the feature development. See: Conway's law

Tribes

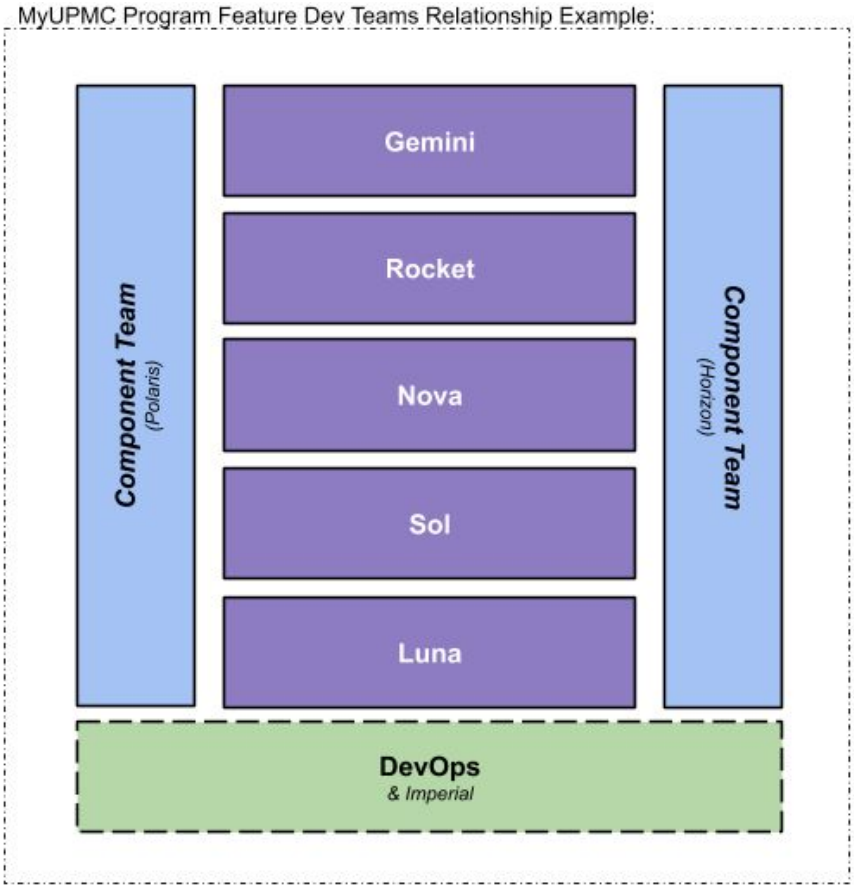
A tribe is any group of specialists shared across multiple feature teams. Also known as communities of practice in SAFe or Chapters in the Spotify model, this group exists to encourage mutual growth in a shared specialization and to ensure that the best technologies, design patterns, and methods are used in a compatible way across the organization. The concept of tribes can evolve as an organization matures. For example, more granular specialization groupings may be necessary.
Voluntary Tribe sessions are held on a sprintly basis to facilitate this. See: MyUPMC Tribes List

Component Teams

While organizing around feature teams has been shown to be the most effective way to deliver user value, research also shows that exclusive adherence to this pattern is suboptimal. It is, for this reason, that we have teams that are, intentionally, not feature teams called component teams. Component teams serve to deliver some subset component of the application stack or provide essential services required for the fluid operation of the rest of the program.

If feature teams serve the end-user then component teams could be said to serve the feature teams by ensuring unfruitful work is not started, that the necessary infrastructure is in place to effectively deliver new features, and to address time-sensitive updates and critical user support needs. Component teams should make up a minority of the structure of the program but they should still follow the regularly scheduled program events, flow-based metrics goals, and sprintly demonstrations of value delivered. See: Advanced Topic - Organizing by Features and Components

MyUPMC Program Structure



Roles and Relationships

Product Owner

Besides the specializations mentioned earlier (tribes), there exists a uniquely special role on each team called a Product Owner (PO). The PO role is typically filled by a product manager but may also be filled by an engineer depending on the needs of the team. What makes this role unique is its function. The PO's role is to ensure that the right things are being built. This means formulating and prioritizing new capabilities through the creation and curation of User Stories. This also entails shielding the team from extraneous noise by acting as a proxy to any additional external business stakeholders.

The team's responsibility, in turn, is to ensure that the things being built are built properly and sustainably. The team is entrusted to determine the technical specifics of implementation and responsible to create and manage Development Tasks as needed or to coordinate through additional intra-team sessions. The team is also responsible to address any technical debt needs to ensure sustained speed, scalability, and security through the same means. Product Owners and their respective teams work together to ensure that new capabilities, however small, are sustainably demonstrated at the conclusion of every sprint thus serving the direct and indirect needs of the end-user. See: What is Story Mapping?

Agilest/Scrum Master

In addition to the Product Owner, there exist other group roles in special relation to the team, that being the Agilest or Scrum Master. The collection of all the Program Agilests & Scrum Masters in a particular program forms that program's Agile Working Group (AWG).

- Program-level Agilests observe the processes, metrics, and interactions of the various teams within an organization or program. This role takes a holistic approach to encouraging best practices and defining areas of improvement.

- Scrum Masters work to promote cohesion within a single team or among a group of teams. This role seeks to help teams adhere to defined processes, assists in the removal of impediments, and gathers/shares feedback used to make larger-scale improvements.

While there are differences pertaining to the areas of focus for the roles within the AWG, the group exists to identify and implement system-level optimizations and serves the teams individually and collectively to advance the cause of 'The Prime Directive'. They ensure that program events are held and continue to be effective and that systems are in place to maintain visibility and compatibility across the whole program.

The Sprint

The sprint (or iteration) is the heartbeat of our software delivery life cycle. Like a CPU, it's the clock time of the operation. The current sprint length across the program is 2 weeks. A sprint is a machine for turning a product backlog into new product increments. All teams sprint. The sprint consists of the following:

Sprint Planning

A Sprint starts with a Sprint Planning meeting being held with the whole team. The meeting starts with the team reviewing the previous sprint in what is frequently referred to as a retrospective. During this segment, the team reviews the previous sprints metrics, discusses any ideas for self-improvement, and determines any changes to be made going forward.

Next, the team selects items from the backlog they believe they will be able to finish by the end of that sprint, judging from their velocity. This collection of work comprises that sprints sprint goal. For a Feature Team, the primary goal of any sprint is the holistic integration of new features to produce a new version of the product (an increment). This will include one or more User Stories and may include any number of supporting issues required to address bugs, tech debt, or clear the way for future work.

Sprint Review

The Sprint Review is the conclusion and culmination of a sprint. In it, the sprint goal is demonstrated in the form of a demo to any and all interested stakeholders. For Feature Teams this means demonstrating new user-facing application capabilities. For Component Teams, this means demonstrating concrete value delivered to the Feature Teams consistent with their team's stated specialization.

The purpose and emphasis in the Sprint Review should be on working software, not metrics such as story points or the number of completed issues, and on value delivered to the end-user over such things as tech debt, research, or documentation. Each team's PO is expected to talk through the delivered sprint goal, demo the new capabilities, and speak to their value.

Daily Scrum

Intersperses between the beginning and end of a sprint are daily scrums. These are 15 min time-boxed sessions where the progress of the sprint goal is addressed. As the sprint is the heartbeat of the program so the daily scrum is the heartbeat of the sprint.

The Daily Scrum is not a status report to the PO, Scrum Masters, or any other stakeholders. It's an event encouraging and providing inter-team coordination and communication meant to address blocked items, exceeded WIP limits, items in danger of not meeting the teams SLE, and any threats to the successful completion of that sprint's sprint goal.

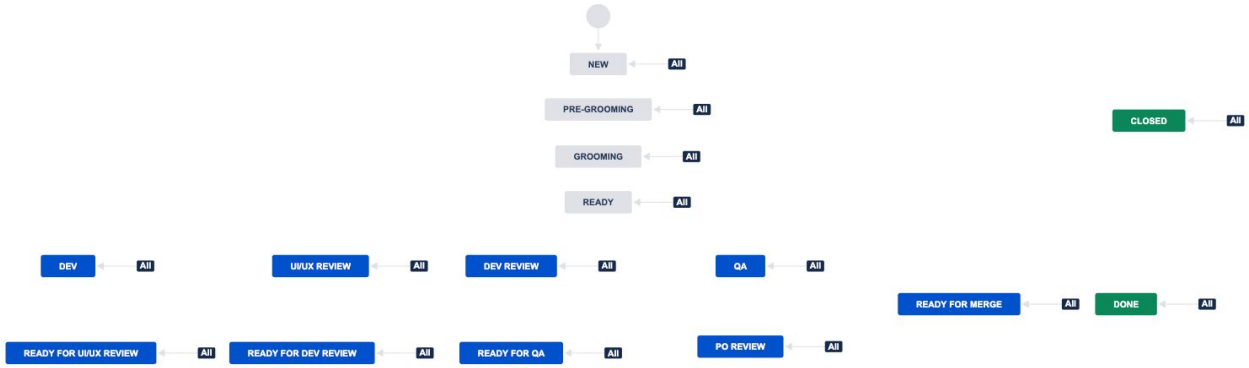
Workflows

Requirements and Usage

A team's workflow is the process it follows to complete work. They are visualized on a team's designated Jira board. The MyUPMC program employs similar development workflows across most teams; however, a team is free to request updates to their unique workflow, as long as it remains pull-based and active work states have WIP-limits.

Having a workflow that is pull-based and has WIP-limits means that additional work shouldn't enter the workflow until existing work has moved through it. It also means that previous work states shouldn't take on new work if future states are loaded or overcapacity (exceeded WIP limits). When WIP limits are being exceeded, team members with capacity should swarm on bottlenecked issues or engage in cross-training to minimize future bottlenecks and increase team resiliency.

MyUPMC Standard Workflow example:



Key States

A few critical key states of the default workflow deserve special attention.

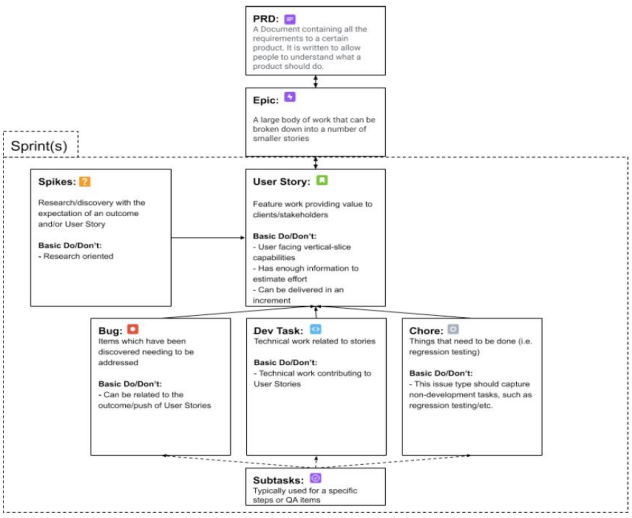
- *Ready*
  The Ready state denotes that a work item can be started. The process of making items ready usually takes place through the use of recurring weekly grooming sessions. See the program calendar for scheduled grooming sessions. Any item in a ready state means that it has the required acceptance criteria delineated, all dependencies identified and

linked, all necessary UI/UX artifacts attached, and that the team has some idea of how the work will be demonstrated in the Sprint Review in the case of a User Story issue type.

- *DEV (Progress Started)*
  Items in the Dev state mean that they have been started in a sprint. This is an important state because the cycle time clock has started for these work items and the team's average cycle time will be affected by their completion time. Metrics such as active item age can be used to better direct resources to items in danger of exceeding the team's SLE. See: Agility Metrics

- *Done*
  Done items have made their journey through all necessary states in a given workflow and been accepted by a team's PO in the case of user stories or by another person competent to judge their content. Done items are expected to have been design reviewed, code reviewed, unit tested, and tested against all existing test cases ensuring all acceptance criteria are met before conducting a demo for the PO for acceptance. Accepted work items, taken in combination with the code itself, largely function as our organization's product documentation so it's important that they get updated with any changes that may have occurred during the course of their journey through the workflow to completion.

Issue Types



[issue types chart link]

Release Planning

A release is a thematic collection of features and is the top-level bucket of all work released to production, thus all work should roll up to a release. Releases are synched across teams to provide maximum visibility and collaborative flexibility. To prevent confusion and miscommunication, teams should refrain from using semantic versions in release names unless that release is in the process of being released.

In most cases, releases encompass work requiring multiple sprints to complete. In these cases, Release Planning sessions (aka a Sprint Zero) should be conducted. In the following section, we'll define what the inputs, activities, and expected outcomes of Release Planning (a.k.a a Sprint Zero) should be.

Inputs

All major feature requests come to the teams in the form of an initial Product Requirements Document (PRD). PO's work with other external stakeholders to craft these documents and prepare them before presenting to their team. A unique Jira issue type of PRD acts as a single source of truth to capture all the compiled PRD documentation.

Once the PO has sufficiently prepared the PRD, created a PRD issue type, and linked all relevant information, it is ready to be reviewed with the team. Once a PRD issue type enters a sprint and is started by that team, the Sprint Zero has officially begun.

Activities

Release Planning (aka a Sprint Zero) is any activity conducted with the purpose of understanding, estimating, and planning a future release. These events may range from one hour to multiple days and initialized with the PO stepping through the details contained in the PRD.

The team gets familiarized with the scope and function of the work, asks questions, and provides feedback to the PO. The PO may create additional User Stories as a part of this dialogue and the engineering team is encouraged to create any number of other work items to capture required technical implementation details.

Outputs

At the conclusion of a Sprint Zero, the team should have a working understanding of the work to be completed in that release and a plan on how they will implement it. A codenamed release should be created for the body of work with at least one Epic created and linked to that release. At least two sprints of work items should be estimated and in a Ready state, and rough estimates (SWAG) placed on the epics, reflecting the expected level of effort for the remaining unestimated issues. A start date and projected completion date should be set on the release with considerations for the planned feature work, tech debt, and team velocity.

Program Calendar

In order to reduce ad hoc meetings, maximize visibility, promote program-wide structure, and minimize interruptions, recurring sprintly events are scheduled, shared, and maintained across all teams via the MyUPMC Development Calendar. The designated program events are intended to be regular, highly structured, and ruthlessly efficient providing a minimum of interruptions while supporting maximum informedness.

Required Events

The fundamental events of the sprint are required for every sprint. These include Sprint Planning, Daily Scrum, and the Sprint Review. They may incidentally shift in time or characteristic (voice vs text) as special needs or circumstances arise but their regular and predictable occurrence, attendance, and effectiveness are absolutely essential in framing the event loop (sprint) of the program.

Supportive Events

In addition to the essential required program events, there are a number of other regularly scheduled events that are intended to be strategic and supportive of overall program alignment and health. While these events are scheduled on cadence, they may be canceled or attended with greater flexibility based on current needs.

- *Grooming*
  Grooming sessions are scheduled weekly with the purpose of ensuring sufficient work (1-2 sprints worth) is in a ready state for the team and can be canceled or multiplied as needed to support this end. Grooming sessions should be attended by the whole team and involve discussing and ideating through backlog work items in order to understand and estimate them.

- *Tribe Sessions*
  Regular Tribe sessions are scheduled to provide opportunities for learning and sharing and to ensure compatible solutions, approaches, and design patterns are used across the organization. These sessions are intended to be voluntary and community-driven with a lunch-and-learn flavor.

- *Nexus*
  Regular Nexus sessions are designed to discuss and address cross-team issues related to cross-impact architectural changes and to be attended by a technical representative from each team. If the usual representative is unable to make the session, they are responsible to appoint a team representative to attend in their stead.

Agility Metrics

Purpose

A few specific metrics are highlighted and used to gauge the stability and predictability of teams and the organization's delivery process. These are intended to provide objective benchmarks and encourage experimentation and learning, not fear-mongering or blame gaming. Any failures or shortcomings are opportunities for learning and self-improvement and, as such, are perfectly consistent with the values of relentless self-improvement encapsulated in The Prime Directive.

KPI's

As can be seen in the preceding content of this document, there are three levels of delivery in our process. The first being the level of the individual work items or the continuous flow of valuable work. The second is the level of the sprint or new product increment. The third level is at the release. Producing stable and predictable processes at the lower levels will guarantee higher predictability at the higher ones. With this in mind, the following metrics are of material importance:

- *Flow Metrics*
  Service Level Expectation: An SLE, or just "average cycle time" for short, forecasts how long it should take a given item to flow from start to finish within your workflow. The SLE itself has two parts: a period of elapsed days and a probability associated with that period (e.g., "85% of work items will be finished in eight days or less").

Work Item Age: “Work Item Age is the amount of elapsed time between when a work item “started” and the current time. This metric complements a team’s designated SLE in highlighting items in danger of exceeding it relative to their age and location in the workflow. A team’s daily scrum is the ideal time to review this metric.

- *Sprint Metrics*
  Velocity: A team’s velocity is the average number of story points completed sprint over sprint and is intended to be used as a tool of planning. Taken in isolation, it is not to be used as a performance metric or as a comparison tool to other teams.

  Sprint Burndown: Show the work remaining until the sprint goal has been reached. Is a good visual indicator of how the team is converging on the sprint goal.

  Cumulative Flow Diagram (CFD): The CFD shows a number of valuable metrics, but a critical one easily and immediately seen is an undesirable increase in WIP and items experiencing thrash.

- *Release Metrics*
  Release Burndown: The release burndown shows how a team is converging on the completion of a release. It shows completed vs remain work and changes in scope with a projected completion date based on historical velocity.

Conclusion

Anything in this document is amendable and subject to change except for ‘The Prime Directive’. There’s no such thing as a permanent best practice. Practices should always be evolving and improving. Anyone is encouraged to submit changes to this document using the comments feature on Google Docs. Changes will be reviewed & merged, or rejected, based upon the approval of the AWG and program leadership.