Issues are dead!
Long live issues, goals, and actions!

Fernando Colleone, Principal Agile Practitioner, Red Hat
Adam Samalik, Software Engineer, Red Hat
Agile Manifesto
What's the problem?
Mindset
“Mindsets are just beliefs. They're powerful beliefs, but they're just something in your mind, and you can change your mind.”

Carol Dweck
Being vs. Doing

Satisfy the customer
Welcome change
Deliver frequently
Work together
Trust and Support
Face-to-Face conversation
Working software
Sustainable development
Continuous attention
Maintain Simplicity
Self-organizing teams
Reflect and Adjust

Sprints
Retrospectives
Daily Standups
Sprint Reviews
WIP limits
Workflow
Extreme Programming - XP
Real Team Example
Mindset
Goals (epics)

Actions (cards)

Issues (feedback)
Project Tracking

The work and an overall progress is being coordinated and tracked on the Modularity WG Kanban Board. User feedback such as bugs and feature requests is being collected in the Modularity issue tracker.

What is being tracked

1. **Goals (epics)** — A specific definition of a state we want to achieve, ideally a SMART goal. Sets the direction of the project.

2. **Actions (cards)** — Tasks that get us closer to a specific goal. Defines actionable work items contributors can complete.

3. **Feedback (issues)** — User feedback such as bugs, ideas, questions. Influences the project from outside.

How we operate
How we operate

We work with an Agile mindset. We have designed our system to transparently communicate our goals and intentions without the need to understand every single piece of work that is being done. It also allows us to display actionable tasks that are clearly defined and prioritized so anyone can help us move forward.

Because Agile is not making things up as we go, but rather iteratively working towards specific goals, the first thing that needs to exist are goal definitions.

Setting and tracking the goals

Goal definitions are ideally SMART — specific, measurable, actionable, relevant, and time-bound. At the absolute minimum, they always must have a clearly defined purpose — so we know why do we want to achieve them, and a definition of done — so we know when exactly are they achieved. This way we can communicate the direction of the project.

Goals (epics) are owned by a single person who acts as a feature driver. The owner doesn’t necessarily do all the work, but makes sure that actionable tasks that lead towards a completion exist at all times, and is responsible for the goal being completed. This way the project can be scaled across a large community of contributors.

Epics are defined in the Epics view of our board. This view is used for strategic planning and for monitoring the overall progress of the project. It also allows to drill down into individual epics for more specific details.
Setting and tracking the goals

Goal definitions are ideally SMART — specific, measurable, actionable, relevant, and time-bound. At the absolute minimum, they always must have a clearly defined purpose — so we know why do we want to achieve them, and a definition of done — so we know when exactly are they achieved. This way we can communicate the direction of the project.

Goals (epics) are owned by a single person who acts as a feature driver. The owner doesn't necessarily do all the work, but makes sure that actionable tasks that lead towards a completion exist at all times, and is responsible for the goal being completed. This way the project can be scaled across a large community of contributors.

Epics are defined in the Epics view of our board. This view is used for strategic planning and for monitoring the overall progress of the project. It also allows to drill down into individual epics for more specific details about each goal.

The specific epic detail shows the purpose of the goal, the definition of done, and other information such as a delivery deadline. It also lists all the actions (cards) that lead to a completion of this epic. Epic owners use this view to define and manage individual cards.

Making progress
Making progress

To work on the project, contributors visit the Kanban view of our board showing all existing cards sorted into different columns, each representing a specific state. Contributors can choose any card in the "Next" column they want to work on.

To work on a specific card, contributors can simply assign themselves to any card in the "Next" column and move it to the "In Progress" column. This way they indicate to other contributors that this card is already being worked on.

While working on a card, updates can be posted to the comment section of the card. Contributors can also mention other people in the comments to ask questions or to notify them about specific information. It is also encouraged to coordinate with other using the usual communication channels such as IRC.

When a card is complete, contributors are expected to post an update to the comments, and move the card to the "Done" column in the Kanban view of our board. This step updates the progress of the epic the card belongs to which is useful for tracking progress.

If new actions appear as a result of a card completion, contributors should coordinate with the Epic owner about creating new cards for the new actions to ensure there is always next step in the queue.

Reporting issues
also encouraged to coordinate with other using the usual communication channels such as IRC.

When a card is complete, contributors are expected to post an update to the comments, and move the card to the "Done" column in the Kanban view of our board. This step updates the progress of the epic the card belongs to which is useful for tracking progress.

If new actions appear as a result of a card completion, contributors should coordinate with the Epic owner about creating new cards for the new actions to ensure there is always next step in the queue.

**Reporting issues**

Issues such as bugs or feature requests can be created by anyone in the Modularity issue tracker. This issue tracker is then processed by the team and new cards or even epics may be created as a result. Separating user feedback from the work lowers the pressure on the person who reports the feedback — they don’t have to understand our tracking system.
Project Tracking

The work and an overall progress is being coordinated and tracked on the Modularity WG Kanban Board.

User feedback such as bugs and feature requests is being collected in the Modularity issue tracker.

What is being tracked

1. Goals (epics) — A specific definition of a state we want to achieve, ideally a SMART goal. Sets the direction of the project.

2. Actions (cards) — Tasks that get us closer to a specific goal. Defines actionable work items contributors can complete.

3. Feedback (issues) — User feedback such as bugs, ideas, questions. Influences the project from outside.

How we operate
<table>
<thead>
<tr>
<th>Votes</th>
<th>Name</th>
<th>Assigned</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>▲ 0</td>
<td>#1 Mass rebuild process updates for modularity EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#2 Branch point process update for modularity EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#3 Module life cycles       EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#4 Stream branch and module ownership process EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#5 Content discovery       EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#35 Module defaults &amp; Fedora Change proposals EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#10 MBS enforces clean content release upgrade paths EPIC</td>
<td></td>
<td>in progress</td>
</tr>
<tr>
<td>▲ 0</td>
<td>#36 Module tagging service       EPIC</td>
<td></td>
<td>in progress</td>
</tr>
</tbody>
</table>
The security team and change owners need to be able to easily discover content that requires their attention. This includes packages affected by security issues or packages that need to be changed or rebuilt due to a change in Fedora.

Done when:

- tools to achieve simple content discovery exist
- archives with modular content SPEC files are available in some form (all, per platform, possibly more)
- repoquery supports disabled and non-default modules
- building block-only modules are supported
- the security team is aware of the new necessary steps and has adopted necessary changes to their processes
- documentation covers content discovery
• the security team is aware of the new necessary steps and has adopted necessary changes to their processes
• documentation covers content discovery
#104 (spike) Can we teach repoquery to work with modular content as well?

This User Story belongs to #5 Content discovery

△ Link to Epic

Created by Adam Samalik
10 Jan 2019 13:15

The `dnf repoquery` command only works with modular packages coming from an enabled or a default stream. However, to be able to do queries required by the identified scenarios we need to teach it to work with the modular content as well.

AC:

- Investigate how big of a change would that be to DNF — a small patch? rewriting half of DNF?
- Propose how the command line interface would look like (dnf repoquery command syntax and an example output)
Project Tracking

The work and an overall progress is being coordinated and tracked on the Modularity WG Kanban Board. User feedback such as bugs and feature requests is being collected in the Modularity issue tracker.

What is being tracked

1. **Goals (epics)** — A specific definition of a state we want to achieve, ideally a SMART goal. Sets the direction of the project.

2. **Actions (cards)** — Tasks that get us closer to a specific goal. Defines actionable work items contributors can complete.

3. **Feedback (issues)** — User feedback such as bugs, ideas, questions. Influences the project from outside.

How we operate
#122 Please make clear that Rawhide module builds don't need bodhi
Opened a month ago by praikup. Modified a month ago

#121 MBS listing should print Fedora version number per build (somewhere)
Opened a month ago by praikup. Modified a month ago

#120 Please document how to test `fedpkg module-build` results
Opened a month ago by praikup. Modified a month ago

#119 Modularity WG Charter (contd.)
Opened 2 months ago by nphillipp. Modified 3 days ago

#115 Discussion: Stream branch ownership for packages & modules
Opened 2 months ago by asamalik. Modified 8 days ago

#113 Please stop using DistTag in module packages as it conflicts with general usage of the tag
Opened 2 months ago by ngompa. Modified 2 months ago

#112 Discussion: Module lifecycles
Goals (epics)  

Actions (cards)  

Issues (feedback)
Agile is a mindset