Agile Swarming, Pair Programming and Mob Programming

How to use Agile swarming techniques to get features done

Agile swarming is a means for a development team to narrow its focus, and quickly finish a feature or user story.

When developers work remotely, an Agile swarm can push a team to collaborate at high levels, not slip into independent, unsynchronized work.

What is swarming in Agile?

The Agile swarming technique directs team members with available time and appropriate skills to work in concert -- i.e., *swarm* -- and complete an in-progress story.

The goal is to deliver high-value features quickly by focusing the entire team on the product backlog items that make up that feature.

Aim is to move the Product backlog ➔ Sprint backlog

The team should select items appropriate to the feature and the skills of the team members.

A software organization can use **Agile swarming practices to maximize developer/DevOps/ DevSecOps/SRE/Software Engineers productivity** and produce a viable **product in an iteration**.

To succeed with Agile swarming, all team members must commit to a swarm's focus.

Agile swarming is not an individual practice.

Benefits of Agile swarming

Teams that concentrate on individual skills and tasks end up with some members far ahead and others grinding away at unfinished work.

For example, a back-end developer is still working on a feature, while the front-end developer for that feature has finished coding.

The front-end developer then starts coding the next feature. The team can design hooks into the code to let the front-end developers validate their work.

However, a feature is not done until a team completes the whole thing, fully integrates it, and tests it.

Letting developers move asynchronously (Individually through the project might result in good velocity metrics, but those measures do not always translate to the team delivering the feature on time.

If testers discover issues in a delivered feature, the entire team must return to already completed tasks.

Let this scenario play out in a real software organization, and you end up with partially completed work on many disparate tasks, and nothing finished.

The goal of Agile development is not to ensure the team is 100% busy, with each person grabbing new product backlog items as soon as they complete their prior task.

This approach to development results in extensive multitasking and ultimately slows the flow of completed items.

- **Stay goal focused, not task focused.**
- **Agile swarming helps the team accomplish a goal, not individual tasks.**
- **Working toward a goal delivers a viable product faster. While an Agile swarm means developers work on fewer items concurrently, it does not mean working on only one product backlog item at a time.**

How to succeed with Agile swarming

Product backlog meetings are the backbone of Agile teamwork.

Weekly meetings with the product owner and Agile team should cover:

- The stories that make up each feature.
- Story fitness -- discuss, refine, and size stories as needed.
- Story ordering; and
- Organization -- product backlog items should be tagged by component for tracking.

Weekly meetings set the team down the swarm path.

In conjunction, hold short, daily team meetings to identify any roadblocks when they arise and redirect resources as necessary.

During the daily meetings, review work status and ensure development is tracking across all components. Take any necessary action to realign team members.

Agile Scrum Practice

When working within an agile Scrum practice, **set a sprint goal and ensure it directly aligns to the feature that has the highest priority in the product backlog**.

That goal should inform what items a team moves from the product backlog into the sprint backlog.

Additional work can enhance Agile swarming:

- Use **timeboxing** techniques to make team meetings effective.
- When **grooming the user story**, have the entire team review the story and get all questions answered before development begins.
- Don't skip the daily check-ins, even if they are held via a virtual format, such as the team's chat platform.

Experiment with this approach, and then do **a retrospective** on it to identify what worked and where you can improve.

Agile swarms vs. mob and pair programming

Agile swarming is not the only way to apply the power of the group to development work. There are related but distinct practices called **pair programming and mob programming**.

Pair programming. Agile adopters often use pair programming, a technical practice where two developers work together to accomplish a task.

Unlike Agile swarming, pair programming is not necessarily about focusing all team members on the same feature.

Dev teams use this technique to improve reliability and reduce bugs.

Typically, with pair programming, the developers team up at one workstation, but it is also effective over video conferencing and screensharing tools.

One developer writes code while the other reviews each line on the spot.

Mob programming. Like swarming, mob programming involves the entire team creating a particular feature.

Unlike **Agile swarming**, the team stays together for a **post-feature grooming session** and collectively begins work on the next feature.

Developers achieve efficiency through greater collaboration, better code quality and less dependency on single individuals.

The team's collective competence constantly increases as part of this practice.

Team members alternate who writes code, who problem solves and who tests.

For co-located teams, mob programming traditionally means being in the same room and using a single computer to write the code and check it in.

With remote work and distributed teams, the togetherness required for mobbing can be a challenge.