Product Backlog(II)

Dr. Elham Mahmoudzadeh

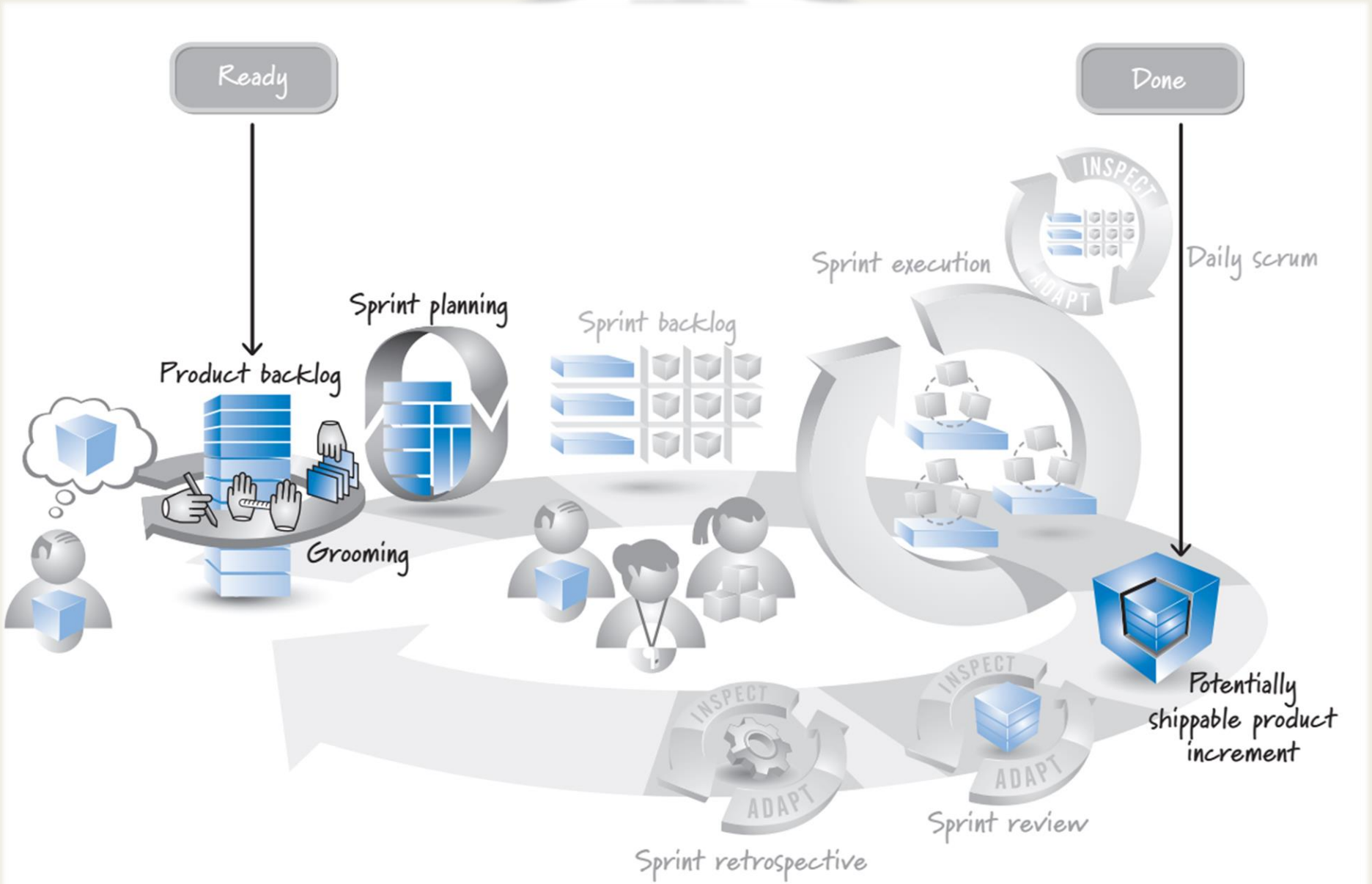
Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2022

Definition of Ready

- Grooming the product backlog should ensure that items at the top of the backlog are ready to be moved into a sprint so that the development team can confidently commit and complete them by the end of a sprint.
- Some Scrum teams formalize this idea by establishing a definition of ready.
- Think definition of ready and the definition of done as two states of product backlog items during a sprint.


Ready
Done
Sprint planning
Sprint backlog
Product backlog
Sprint execution
Daily scrum
Grooming
Potentially shippable product increment
Sprint review
Sprint retrospective
INSPECT
ADAPT

Definition of Ready (Cnt'd)

- Both the definition of done and the definition of ready are checklists of the work that must be completed before a product backlog item can be considered to be in the respective state.
- A strong definition of ready will substantially improve the Scrum team's chance of successfully meeting its sprint goal.

	Definition of Ready
❑	Business value is clearly articulated.
❑	Details are sufficiently understood by the development team so it can make an informed decision as to whether it can complete the PBI.
❑	Dependencies are identified and no external dependencies would block the PBI from being completed.
❑	Team is staffed appropriately to complete the PBI.
❑	The PBI is estimated and small enough to comfortably be completed in one sprint.
❑	Acceptance criteria are clear and testable.
❑	Performance criteria, if any, are defined and testable.
❑	Scrum team understands how to demonstrate the PBI at the sprint review.

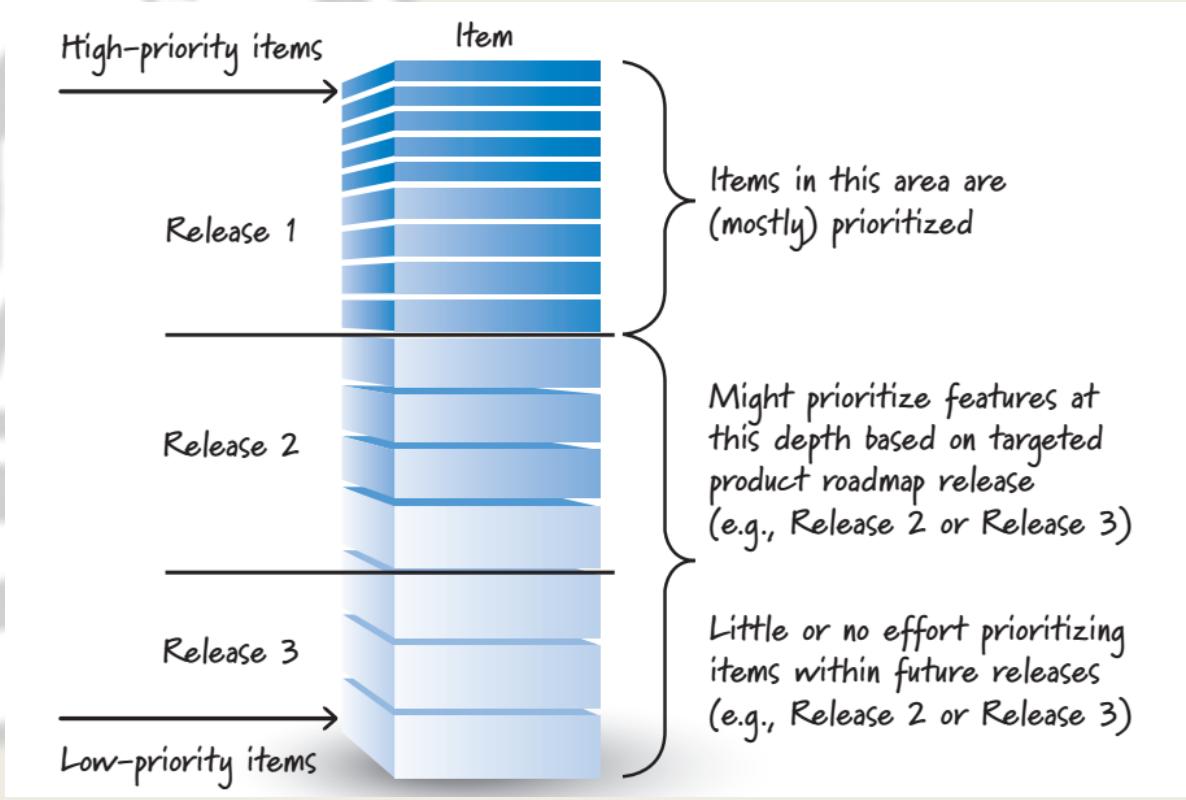
واضح و مشخص بیان شده

Flow Management

- The product backlog is a crucial tool that enables the Scrum team to achieve fast, flexible value-delivery flow in the presence of uncertainty.
- Uncertainty cannot be eliminated from product development.
- We must assume that a stream of economically important information will be constantly arriving and that we need to organize and manage the work (manage the product backlog) so that this information can be processed in a rapid, cost-effective way while maintaining good flow.

Release Flow Management

- The product backlog must be groomed in a way that supports ongoing release planning(the flow of features within a release).
- A release can be visualized as a line through the product backlog. All of the PBIs above the release line are targeted to be in that release; the items below the line are not.

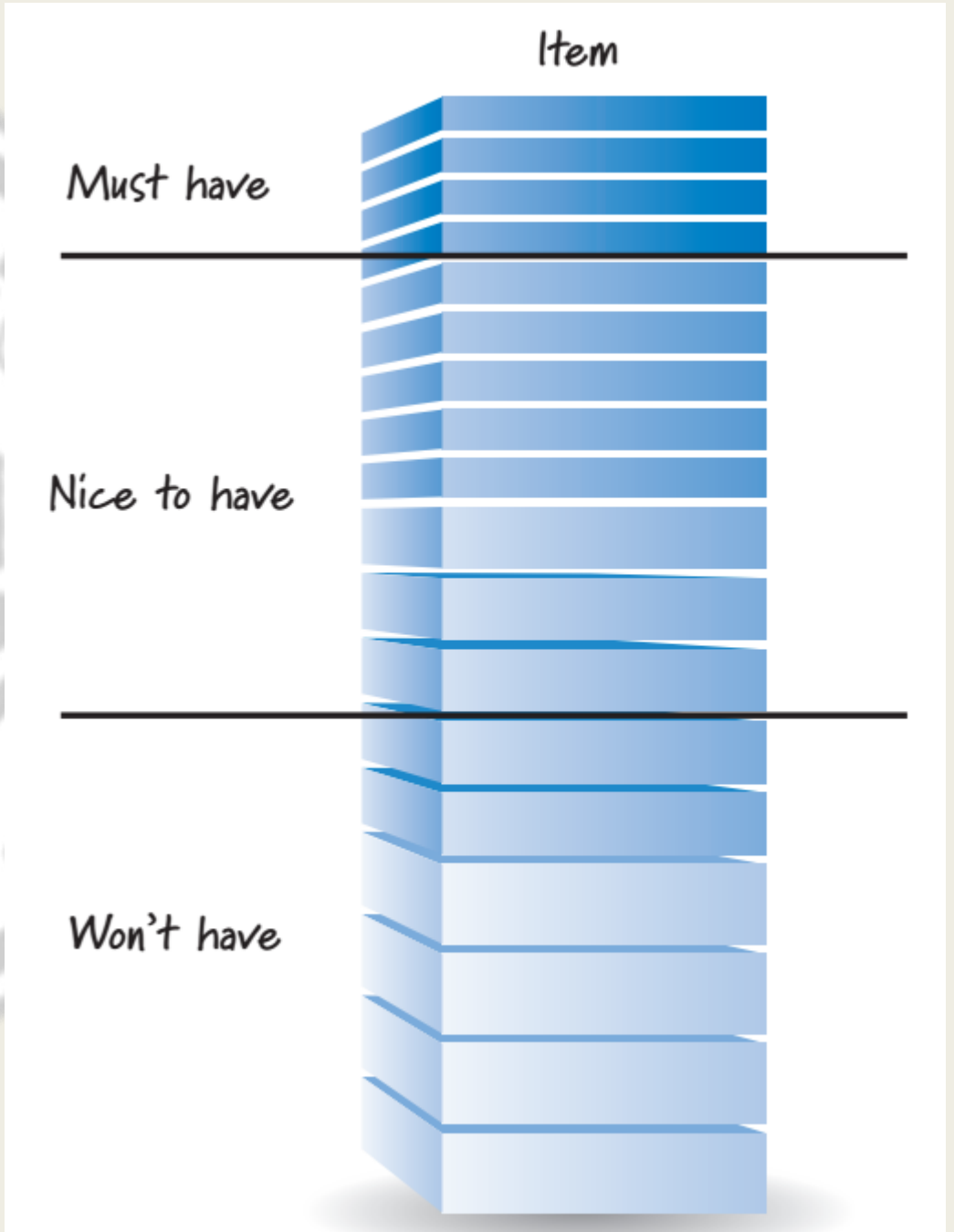


Release Flow Management

- It is useful to actually partition the product backlog using two lines for each release.
- These two lines partition the backlog into three areas: *must have*, *nice to have*, and *won't have*.
- The must-have features represent the items that we simply must have in the upcoming release or else we don't have a viable customer release.
- The nice-to-have features represent items we are targeting for the next release and would like to include. If, however, we run short of time or other resources, we could drop nice-to-have features and still be able to ship a viable product.
- The won't-have features are items that we're declaring won't be included in the current release.

Release-level view of the product backlog

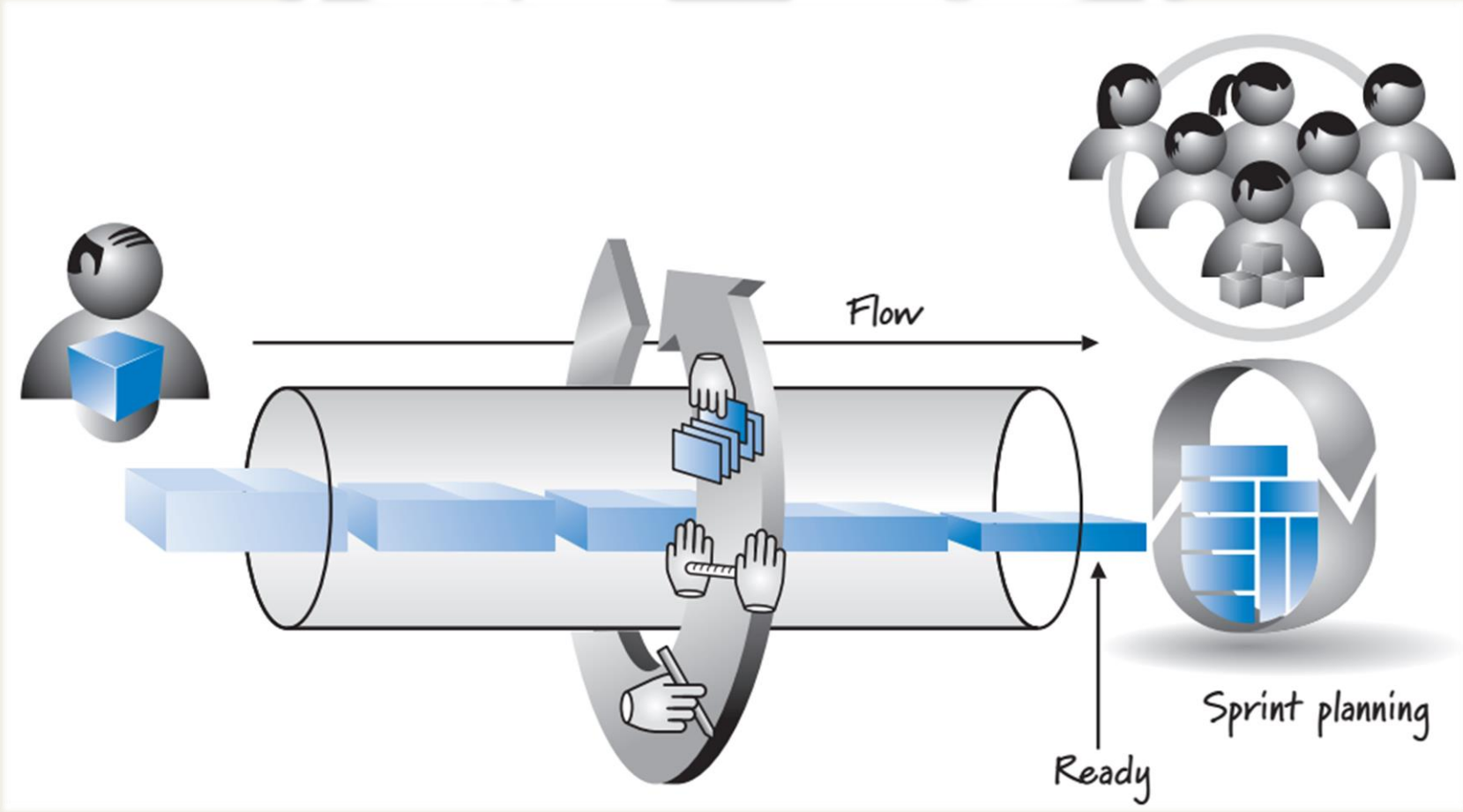
- The second line, the one that separates the won't-have items from the others, is the same as the Release 1 line .
- Maintaining the backlog in this fashion helps us better perform ongoing release planning.



Sprint Flow Management

- Product backlog grooming is essential for effective sprint planning and the resulting flow of features into a sprint.
- If the product backlog has been detailed appropriately, the items at the top of the backlog should be clearly described and testable.
- When grooming for good sprint flow, it is helpful to view the product backlog as a pipeline of requirements that are flowing into sprints to be designed, built, and tested by the team.

The product backlog as a pipeline of requirements



Sprint Flow Management (Cnt'd)

- Larger, less-well-understood requirements are being inserted into the pipeline.
- As they progress through the pipeline and move closer to the time when they will flow out to be worked on, they are progressively refined through the grooming activity.
- At the right side of the pipeline is the team. By the time an item flows out of the pipeline, it must be ready—detailed enough that the team can understand it and be comfortable delivering it during a sprint.

Sprint Flow Management (Cnt'd)

- If there is ever a mismatch or unevenness between the inflow and outflow of items, we have a problem.
- If the flow of groomed, detailed, ready-to-implement items is too slow, eventually the pipeline will run dry and the team won't be able to plan and execute the next sprint (a major flow disruption or waste in Scrum).
- On the other hand, putting too many items into the pipeline for refinement creates a large inventory of detailed requirements that we may have to
of waste).

rework or throw away once we learn more (a major source of waste).

Therefore, the ideal situation is to have just enough product backlog items in inventory to create an even flow but not so many as to create waste.

Sprint Flow Management (Cnt'd)

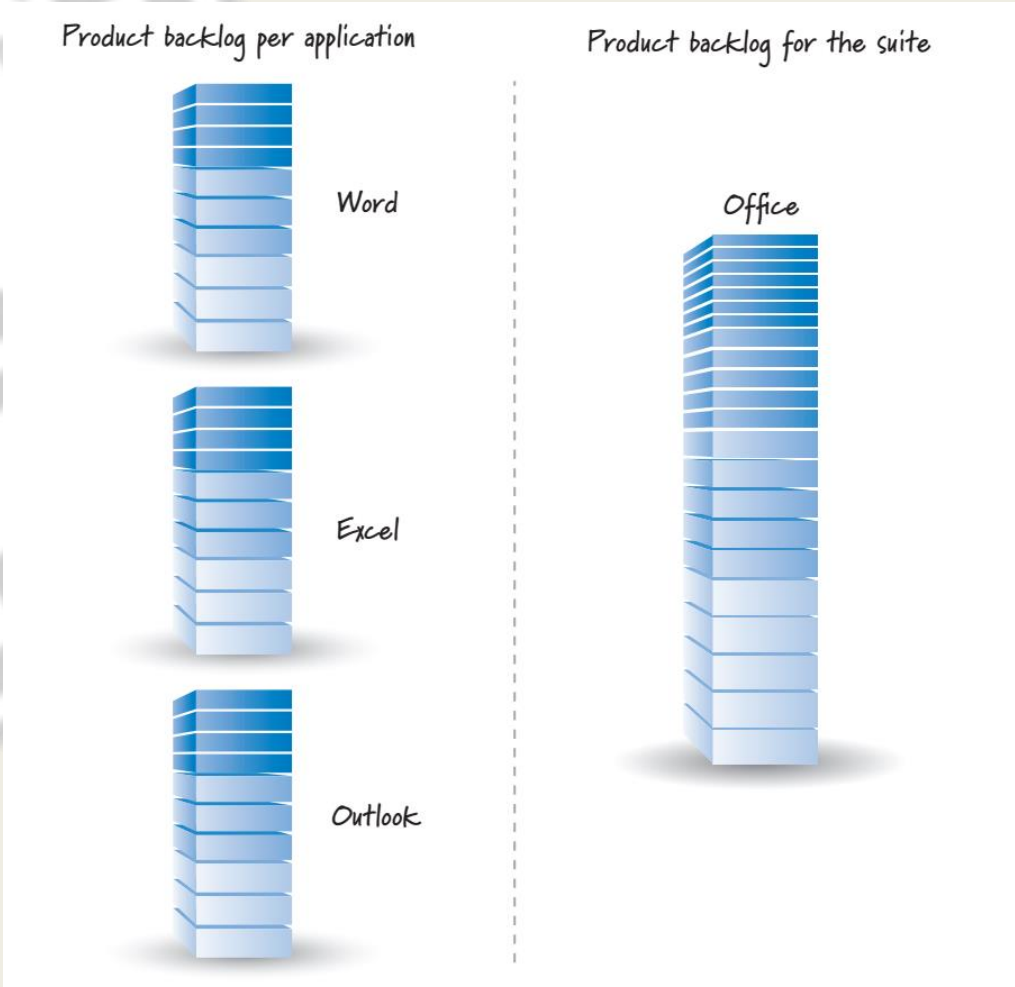
- One approach that Scrum teams use is to have an appropriate inventory of groomed and ready-to-implement items in the backlog.
- A heuristic that seems to work for many teams is to have about two to three sprints' worth of stories ready to go.
- So, for example, if the team can normally do about 5 PBIs per sprint, the team grooms its backlog to always have about 10 to 15 PBIs ready to go at any point in time.
- This extra inventory ensures that the pipeline won't run dry, and it also provides the team with flexibility if it needs to select PBIs out of order for capacity reasons or other sprint-specific constraints.

Which and How Many Product Backlogs?

- When deciding on which and how many product backlogs to form, I start with a simple rule: one product, one product backlog, meaning that each product should have its own single product backlog that allows for a product-wide description and prioritization of the work to be done.
- There are, however, some occasions when we need to exercise care when applying this rule to ensure that we end up with a practical, workable product backlog structure.

What Is a Product?

- Is Microsoft Word the product, or is it simply one facet of a larger product called Microsoft Office?



What Is a Product? (Cnt'd)

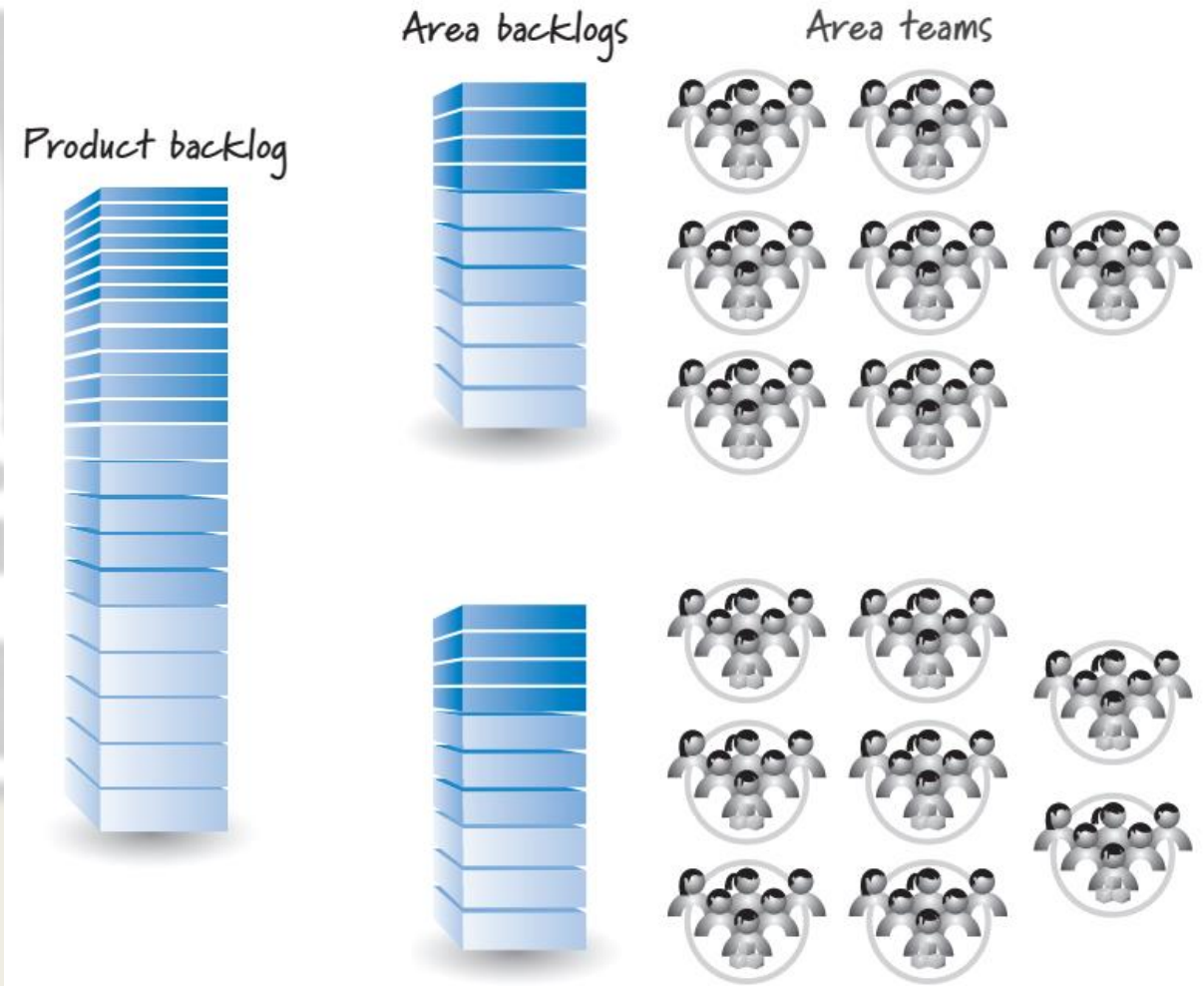
- IBM sold products from a catalog, so if you could put a PID on it, salespeople could include it on an order form and therefore it was a "product."
- A product is something of value that a customer would be willing to pay for and something we're willing to package up and sell.
- Think about what you create that is packaged, delivered, and adds end-customer value. Then align your product backlog with that offering.

Large Products—Hierarchical Backlogs

- Whenever possible, I prefer one product backlog even for a large product like Microsoft Office.
- However, we need to be practical when applying this rule. On a large product development effort to create something like a cell phone, we can have many tens or hundreds of teams whose work must all come together to create a marketable device. Trying to put the PBIs from all of these teams into one manageable product backlog isn't practical (or necessary).
- Not all of these teams work in related areas.

Large Products—Hierarchical Backlogs (Cnt'd)

- Most organizations address the large-product problem by creating hierarchical.



Large Products—Hierarchical Backlogs (Cnt'd)

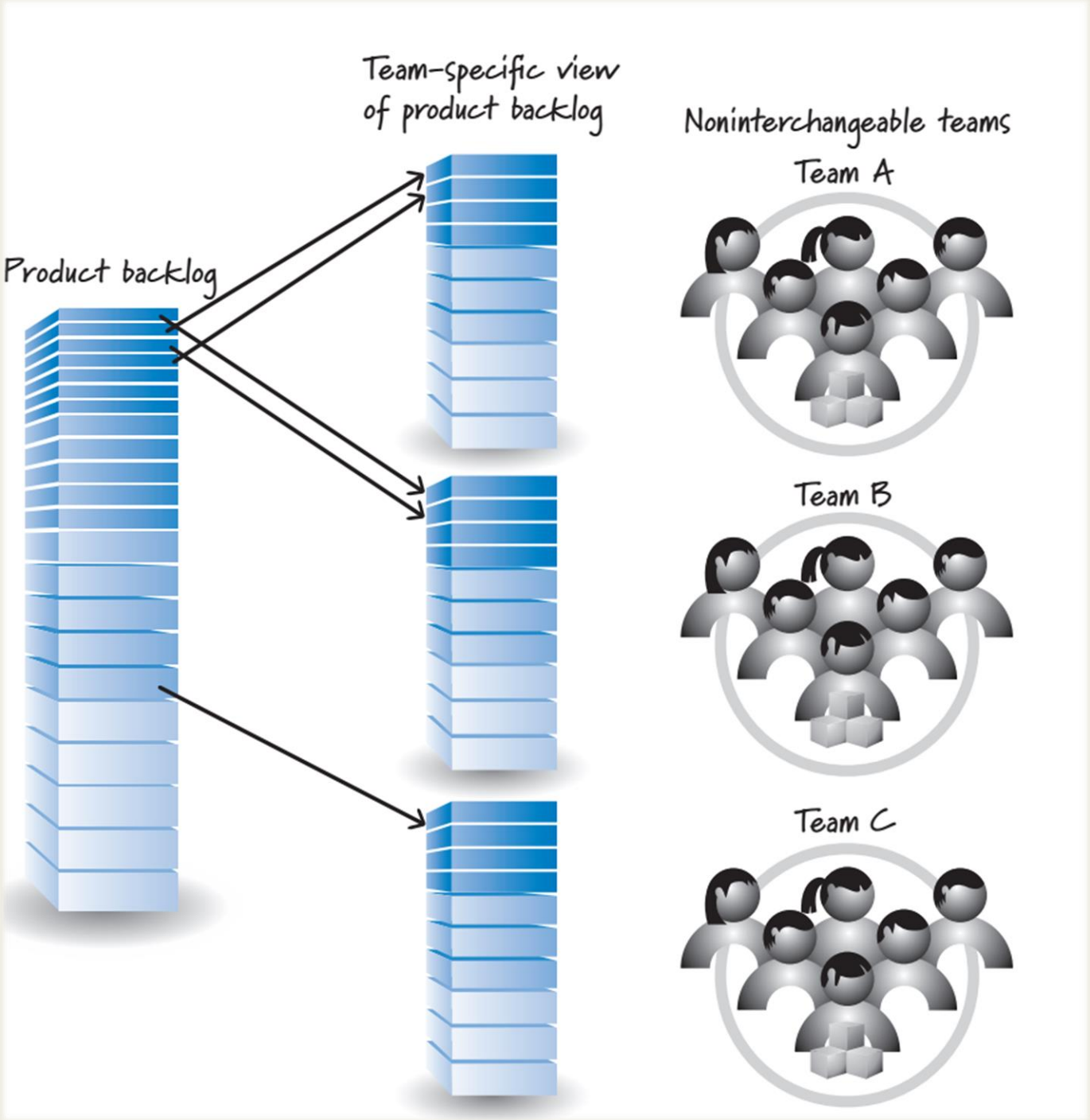
- At the top of the hierarchy we still have the one product backlog that describes and prioritizes the large-scale features (perhaps epics) of the product.
- Each of the related feature areas then has its own backlog. The PBIs at the feature-area level will likely be smaller in scale (feature or story size) than the corresponding items in the product backlog.

Multiple Teams—One Product Backlog

- The one-product-one-product-backlog rule is designed to allow all of the teams working on the product to share a product backlog.
- Aligning all of the teams to a single backlog enables us to optimize our economics at the full-product level.
- We get this benefit because we put all of the features into one backlog and make them compete for priority against all other features, ensuring that the highest-priority features from the full-product perspective are identified and prioritized to be worked on first.
- If all of our teams are interchangeable, so that any team can work on any PBI in the one shared backlog, we actually get to realize the prioritization benefit enabled by the single product backlog.

Multiple Teams—One Product Backlog (Cnt'd)

- But what if the teams aren't interchangeable?
- For example, a team that works on the Microsoft Word text-layout engine probably can't be assigned to work on the Microsoft Excel calculation engine.
- While not ideal, in some cases, not every team can work on every item in the product backlog.
- To work within this reality, we must know which items in the product backlog each team can work on.

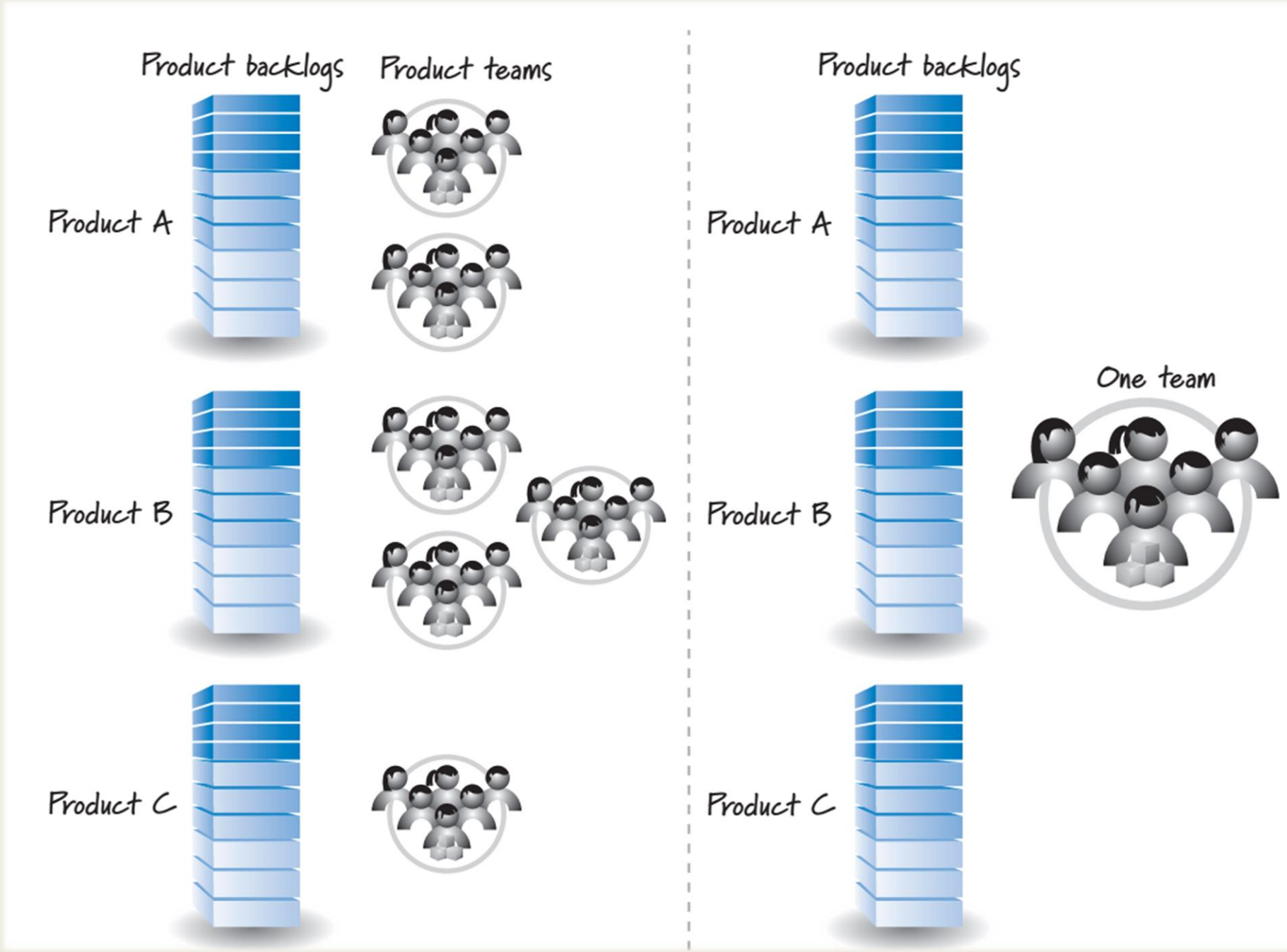

Team-specific view
of product backlog
Noninterchangeable teams
Team A
Product backlog
Team B
Team C

Multiple Teams—One Product Backlog (Cnt'd)

- We have team-specific views of the shared backlog.
- There is one backlog, but it is structured in such a way that teams see and choose from only the features that are relevant to their skill sets.
- The highest-level item in the team C backlog is derived from an item that is not a very high priority in the product-level backlog.
- If the teams were interchangeable, team C's backlog would correspond to much higher priority product-level backlog items.
- This lack of flexibility is why many organizations strive for a high level of shared code ownership and more interchangeable teams, so that they too can reap the benefits that come from having teams that can work on multiple areas of the product.

One Team—Multiple Products

- If an organization has multiple products, it will have multiple product backlogs.
- The best way to handle multiple product backlogs is to assign one or more teams to work exclusively on each product.


Product backlogs
Product teams
Product A
Product B
Product C
Product backlogs
Product A
Product B
Product C
One team

One Team—Multiple Products

- Our goal should be to minimize the amount of multi-projecting that teams or team members perform.
- The first, and often the best, solution is to have the team work on one product at a time. In each sprint the team works only on the items from one product backlog.
- However, if organizational impediments force us to have the single team work on multiple products concurrently, we might consider merging the PBIs for all three products into one product backlog. This would require that the product owners for the three products come together and reach a single prioritization across all of the products.

Reference

1- K. S. Rubin, “Essential Scrum, A Practical guide to the most popular agile process,” 2013.