Advanced Software Engineering

Dr. Elham Mahmoudzadeh

Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

2022

Agile Principles(V)

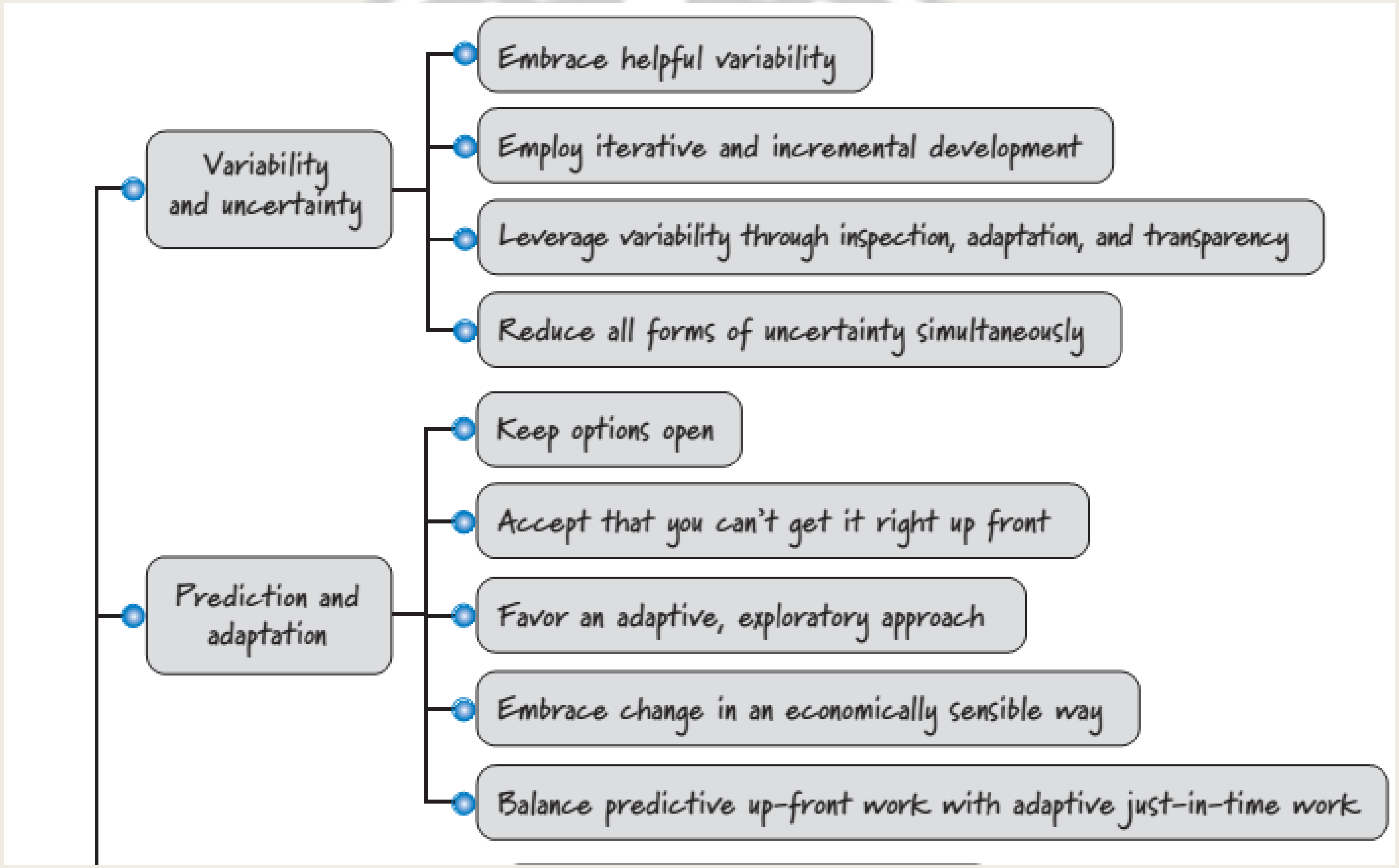
Dr. Elham Mahmoudzadeh

Isfahan University of Technology

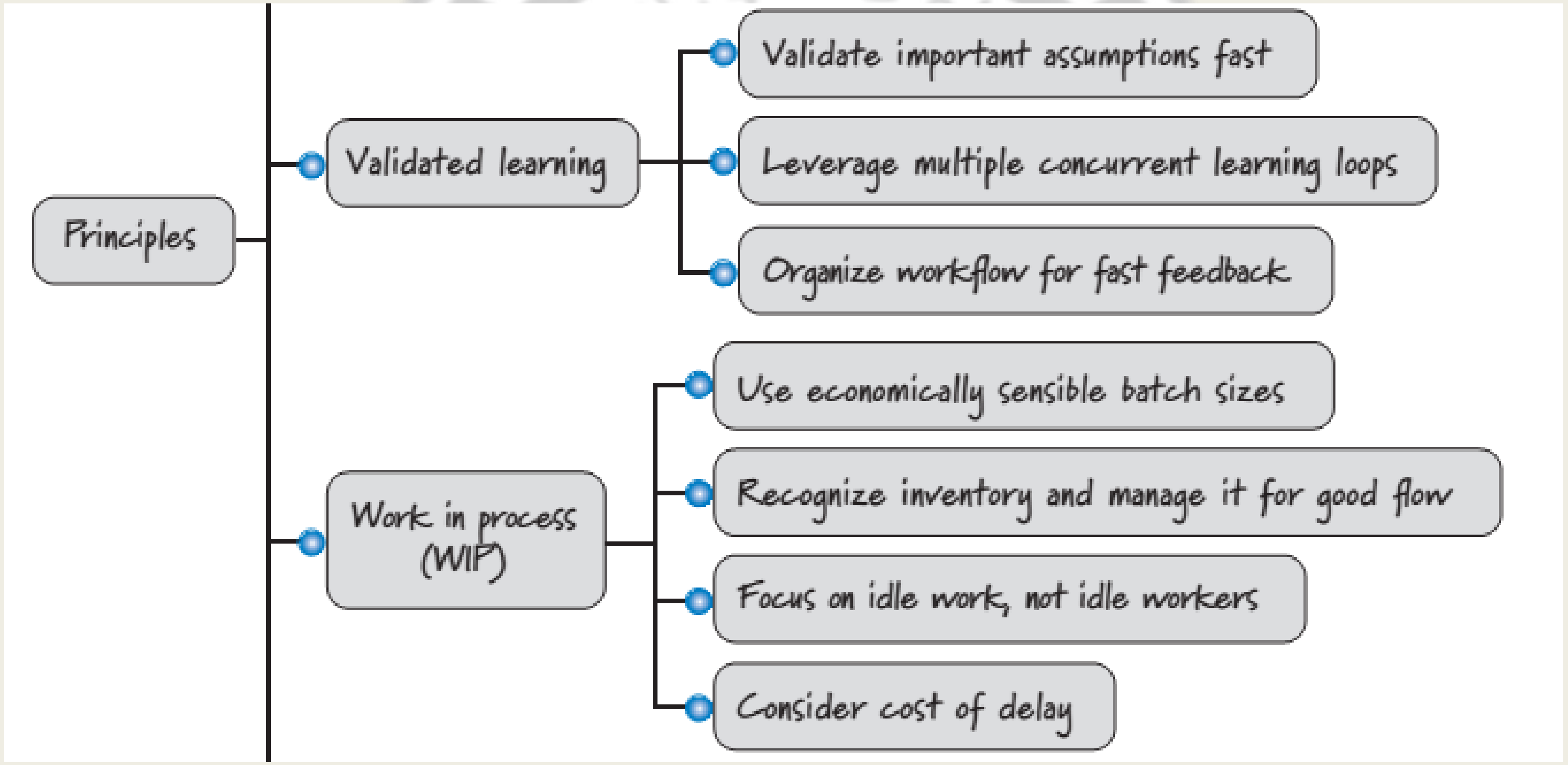
mahmoudzadeh@iut.ac.ir

2020

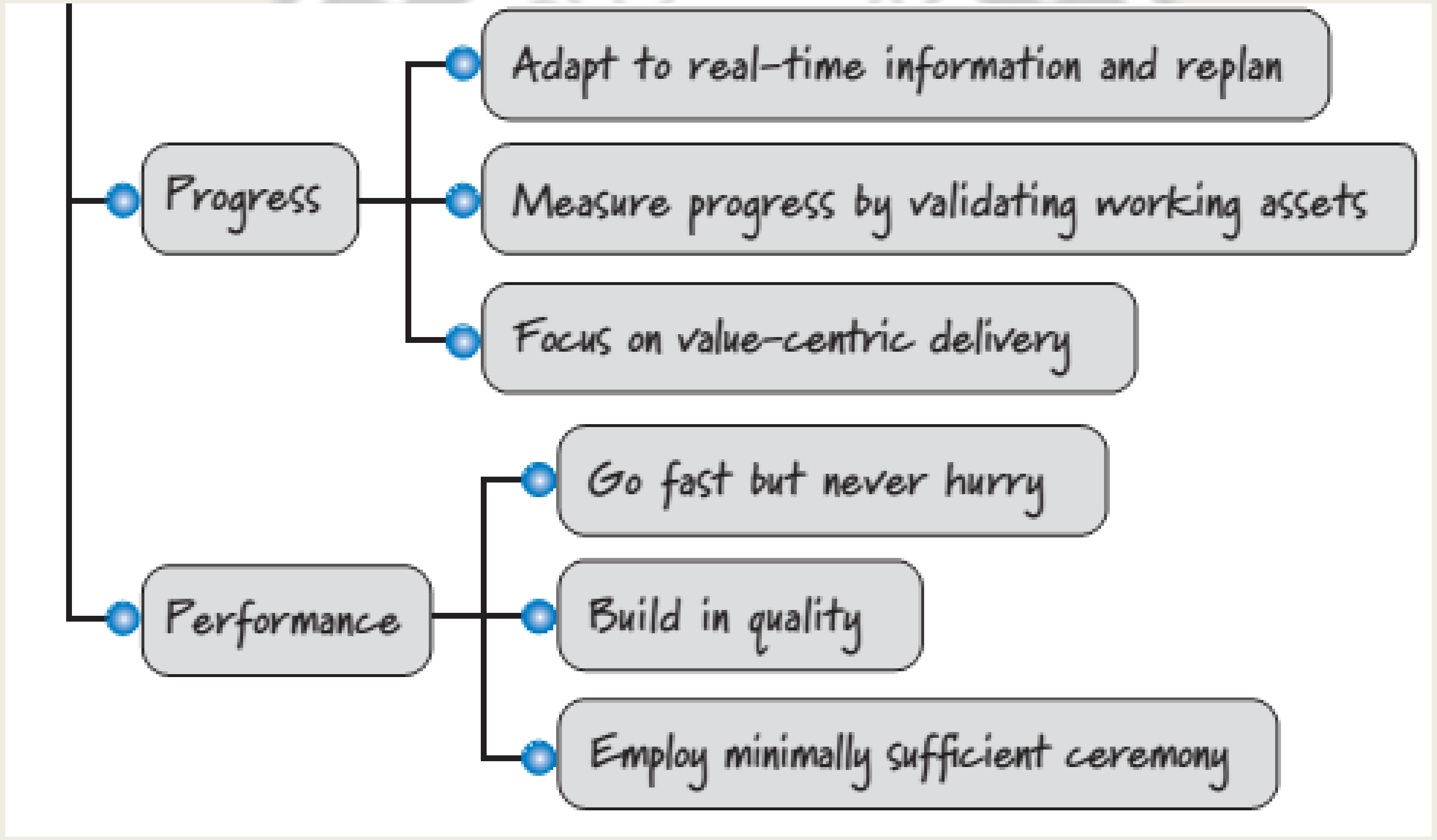
Categorization of principles (Up)


Variability and uncertainty
Embrace helpful variability
Employ iterative and incremental development
Leverage variability through inspection, adaptation, and transparency
Reduce all forms of uncertainty simultaneously
Prediction and adaptation
Keep options open
Accept that you can't get it right up front
Favor an adaptive, exploratory approach
Embrace change in an economically sensible way
Balance predictive up-front work with adaptive just-in-time work

Categorization of principles (Middle)


Principles
Validated learning
Validate important assumptions fast
Leverage multiple concurrent learning loops
Organize workflow for fast feedback
Work in process (WIP)
Use economically sensible batch sizes
Recognize inventory and manage it for good flow
Focus on idle work, not idle workers
Consider cost of delay

Categorization of principles (Bottom)


Progress
Adapt to real-time information and replan
Measure progress by validating working assets
Focus on value-centric delivery
Performance
Go fast but never hurry
Build in quality
Employ minimally sufficient ceremony

Work in Process (WIP)

- Refers to the work that has been started but not yet finished. During product development WIP must be recognized and properly managed.

- Four agile principles related to this topic.
  1. *Use economically sensible batch sizes.*
  2. *Recognize inventory and manage it for good flow.*
  3. *Focus on idle work, not idle workers.*
  4. *Consider cost of delay.*

Use Economically Sensible Batch Sizes(I)

- Another core belief underlying plan-driven, sequential development processes is that it is preferable to batch up all of one type of work and perform it in a single phase.
- Refer to this as the all-before-any approach, where we complete all (or substantially all) of one activity before starting the next.
- Let’s say we create all of the requirements during the analysis phase. Next, we move the batch of requirements into the design phase. Because we generated the complete set of requirements, our batch size in this example is 100%.

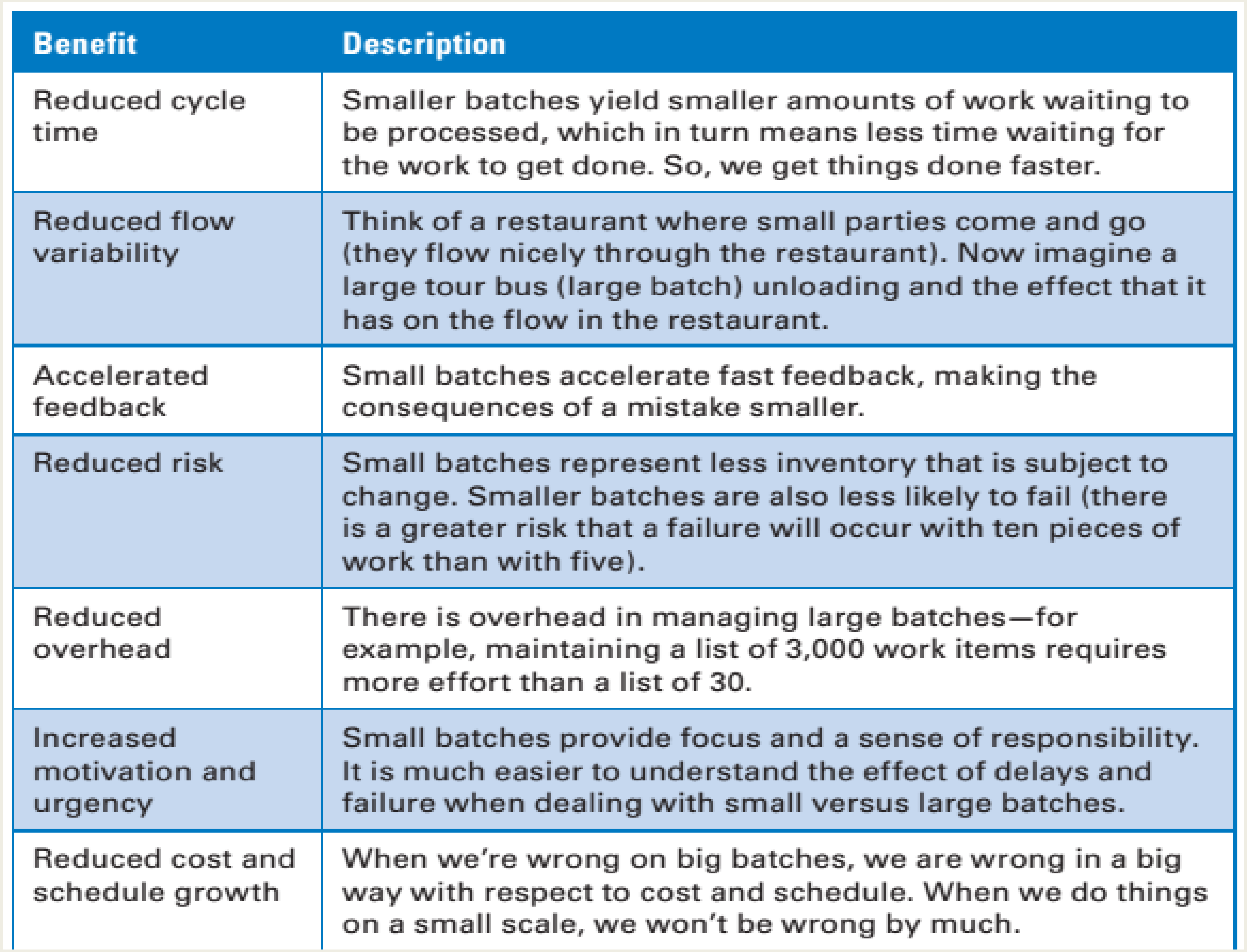
Use Economically Sensible Batch Sizes(II)

- The all-before-any approach is, in part, a consequence of believing that the old manufacturing principle of economies of scale applies to product development.
- This principle states that the cost of producing a unit will go down as we increase the number of units (the batch size) that are produced.

Use Economically Sensible Batch Sizes(III)

- In Scrum, we accept that although economies-of-scale thinking has been a bedrock principle in manufacturing, applying it dogmatically to product development will cause significant economic harm.

- Working in smaller batches during product development has many benefits.



Benefit	Description
Reduced cycle time	Smaller batches yield smaller amounts of work waiting to be processed, which in turn means less time waiting for the work to get done. So, we get things done faster.
Reduced flow variability	Think of a restaurant where small parties come and go (they flow nicely through the restaurant). Now imagine a large tour bus (large batch) unloading and the effect that it has on the flow in the restaurant.
Accelerated feedback	Small batches accelerate fast feedback, making the consequences of a mistake smaller.
Reduced risk	Small batches represent less inventory that is subject to change. Smaller batches are also less likely to fail (there is a greater risk that a failure will occur with ten pieces of work than with five).
Reduced overhead	There is overhead in managing large batches—for example, maintaining a list of 3,000 work items requires more effort than a list of 30.
Increased motivation and urgency	Small batches provide focus and a sense of responsibility. It is much easier to understand the effect of delays and failure when dealing with small versus large batches.
Reduced cost and schedule growth	When we're wrong on big batches, we are wrong in a big way with respect to cost and schedule. When we do things on a small scale, we won't be wrong by much.

Use Economically Sensible Batch Sizes(V)

- If small batches are better than large batches, shouldn’t we just use a batch size of one, meaning that we work on only one requirement at a time and flow it through all of the activities until it is done and ready for a customer?
- A batch size of one might be appropriate in some cases, but assuming that “one” is the goal might sub-optimize the flow and our overall economics.

Recognize Inventory and Manage It for Good Flow(I)

- There is, one lesson that manufacturing has learned that we should apply to product development and yet often do not. That lesson has to do with the high cost of inventory, also known as work in process or WIP.
- Manufacturers are acutely aware of their inventories and the financial implications of those inventories.
- Inventory quickly starts to pile up on the floor, waiting to be processed. Not only is factory inventory physically visible; It is also financially visible.

Recognize Inventory and Manage It for Good Flow(II)

- But, what happens if we purchase a truckload of parts, and then change the design of the product? What do we do with all of those parts?

- Maybe we rework the parts so that they fit into the new design.
- Or worse, maybe we discard the parts because they can no longer be used.
- Or, to avoid incurring waste on the parts we already purchased, are we going to not change our design (even though doing so would be the correct design choice) so we can use those parts—at the risk of producing a less satisfying product?

Recognize Inventory and Manage It for Good Flow(III)

- It's obvious that if we sit on a lot of inventory and then something changes, we experience one or more forms of significant waste.
- To minimize risks, competent manufacturers manage inventory in an economically sensible way.
- They keep some inventory on hand but use a healthy dose of just-in-time inventory management.

Recognize Inventory and Manage It for Good Flow(IV)

- In product development we deal with **knowledge assets that aren’t physically visible** in the same way as parts on the factory floor.
- Knowledge assets are far less intrusive, such as code on a disk, a document in a file cabinet, or a visual board on the wall.
- Inventory in product development is also typically not financially visible.

Recognize Inventory and Manage It for Good Flow(V)

- Inventory (WIP) is a critical variable to be managed during product development, and the traditional approaches to product development don't focus on managing it.

- By setting the batch size to be quite large (frequently 100%), traditional development actually favors the creation of large amounts of inventory.

- An important consequence of having a lot of WIP in product development is that it significantly affects the cost-of-change.

Recognize Inventory and Manage It for Good Flow(VI)

- Although we need some requirements if we are going to start development, we don’t need to have all of the requirements. If we have too many requirements, we will likely experience inventory waste when requirements change.

- On the other hand, if we don’t have enough requirements inventory, we will disrupt the fast flow of work, which is also a form of waste.

Recognize Inventory and Manage It for Good Flow(VII)

- In Scrum, our goal is to find the proper balance between just enough inventory and too much inventory.
- It is important to realize that requirements are just one form of inventory that exists in product development.
- There are many different places and times during product development where we have WIP. We need to proactively identify and manage those as well.

Focus on Idle Work, Not Idle Workers(I)

- **Idle work** is work that we want to do (such as building or testing something) but can’t do because something is preventing us.
- Perhaps we are blocked waiting on another team to do something, and until that team completes its work, we can’t do ours. Or maybe we just have so much work to do that it can’t all be done at once. In this case, some of the work sits idle until we become available to work on it.

Focus on Idle Work, Not Idle Workers(II)

- **Idle workers**, are people who have available capacity to do more work because they are not currently 100% utilized.
- In Scrum, we believe that **idle work** is far more wasteful and economically damaging than idle workers.

Focus on Idle Work, Not Idle Workers(III)

- Many product development organizations focus more on eliminating the waste of idle workers than on the waste of idle work.
- For example, in traditional thinking, if I hire you to be a tester, I expect you to spend 100% of your time testing. If you spend less than 100% of your time testing, I incur waste (you're idle when you could be testing).
- To avoid this problem, I will find you more testing work to do—perhaps by assigning you to multiple projects—to get your utilization up to 100%.

Focus on Idle Work, Not Idle Workers(IV)

- Unfortunately, this approach reduces one form of waste (idle-worker waste) while simultaneously increasing another form of waste (idle-work waste).

- Most of the time, the cost of the idle work is far greater than the cost of an idle worker.

Focus on Idle Work, Not Idle Workers(V)

- To illustrate the issue, let ’s apply the keep-workers-100 % -busy strategy to the 4 × 100-meter relay race at the Olympics.
- Based on the keep-them-busy strategy, this race seems highly inefficient. I pay people to run and they seem to be running only one-quarter of the time. The rest of the time they are just standing around.
- Well, that’s not right! I pay them 100% salary so I want them to run 100% of the time.
- How about if when they’re not carrying the baton, they just run up and down the stands or perhaps run another race on an adjacent track? That way they will be utilized 100% at running.

Focus on Idle Work, Not Idle Workers(VI)

- You don’t win the relay gold medal by keeping the runners 100% busy. You win the gold medal by getting the baton across the finish line first. So, the important takeaway is “Watch the baton, not the runners”.

Focus on Idle Work, Not Idle Workers(VII)

- In the context of product development, the baton sitting on the ground equates to work that is ready to be performed but is blocked waiting for necessary resources.

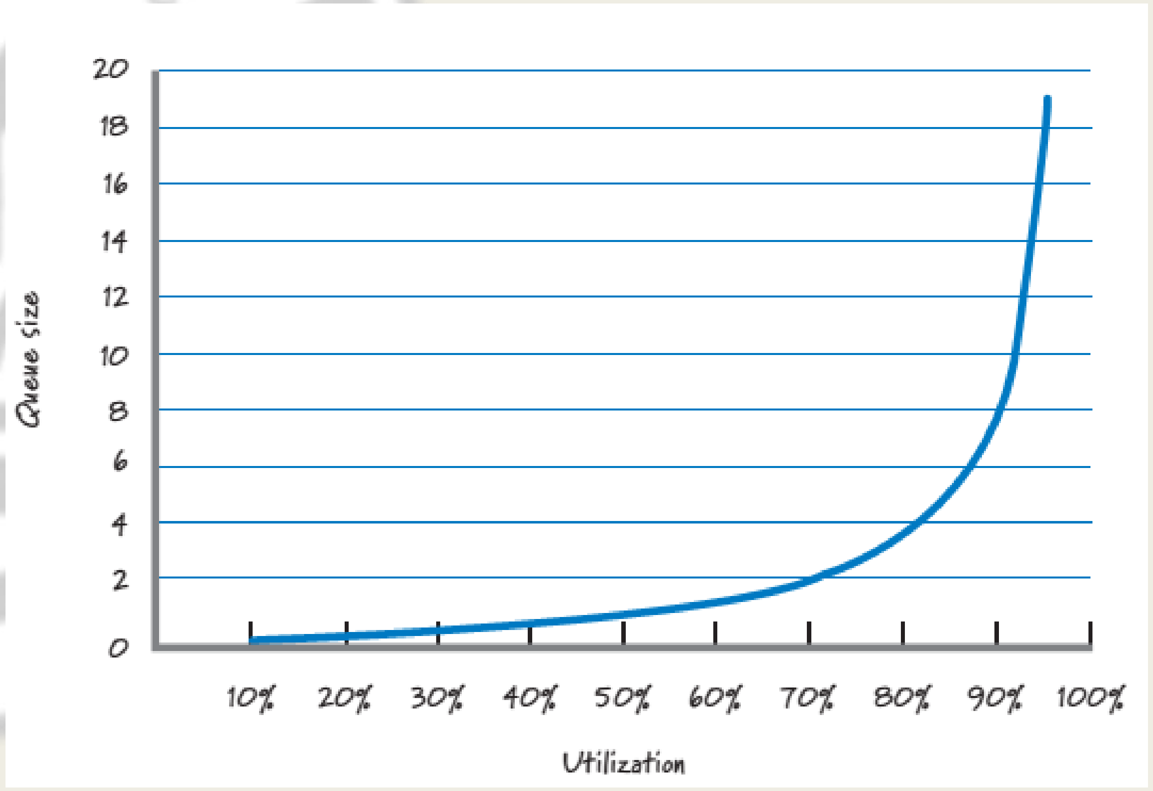
- You don’t win the race (deliver products) when the baton is on the ground.

Focus on Idle Work, Not Idle Workers(X)

- What happens if you run your computer at 100% (full processor and memory utilization)? It starts to thrash and every job on the computer slows down. In other words, the computer is working on more things and actually gets less productive work completed.
- Once you get into that state, it is very difficult to get out of it (you probably have to start killing jobs or reboot the machine).
- Your computer would be much more efficient if you ran it at closer to 80% utilization.

How utilization affects queue size

- obvious damage caused when striving for 100% utilization.
- As capacity utilization increases, queue size and delay increase.



Focus on Idle Work, Not Idle Workers(XI)

- The idle work (delayed work) grows exponentially once we get into the high levels of utilization.
- And that idle work can be very expensive, frequently many times more expensive than the cost of idle workers.
- So, in Scrum, we are aware that finding the bottlenecks in the flow of work and focusing our efforts on eliminating them is a far more economically sensible activity than trying to keep everyone 100% busy.

Consider Cost of Delay

- Cost of delay is the financial cost associated with delaying work or delaying achievement of a milestone.
- By reducing the waste of idle workers (by increasing their utilization), we simultaneously increase the waste associated with idle work (work sitting in queues waiting to be serviced).
- Using cost of delay, we can calculate which waste is more economically damaging.
- Combine that with the fact that most development organizations don’t realize they have accumulated work (inventory) sitting in queues, and it is easy to see why their default behavior is to focus on eliminating the visible waste of idle workers.

Reference

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