Agile Principles(II)

Dr. Elham Mahmoudzadeh

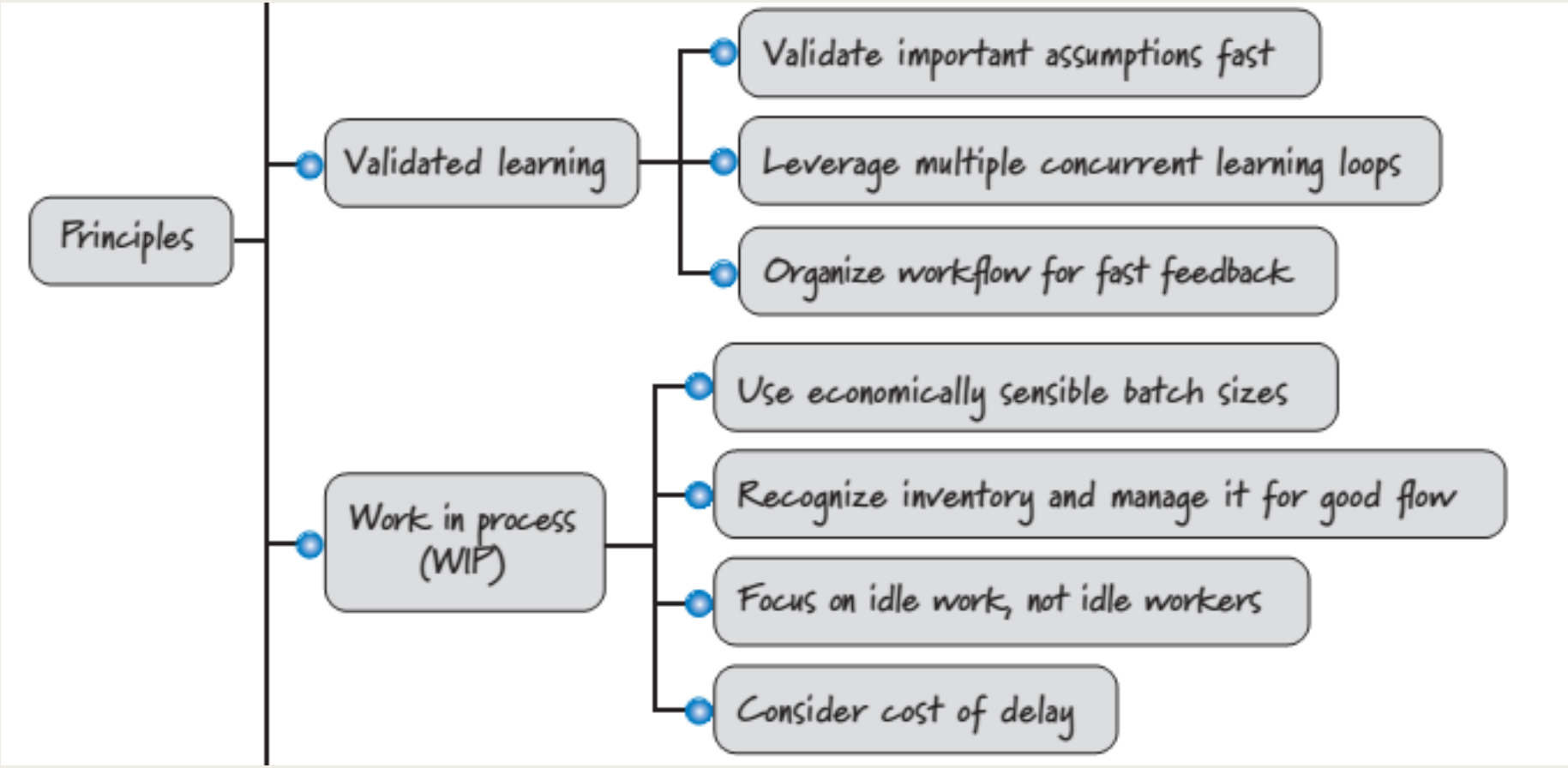
Isfahan University of Technology

mahmoudzadeh@iut.ac.ir

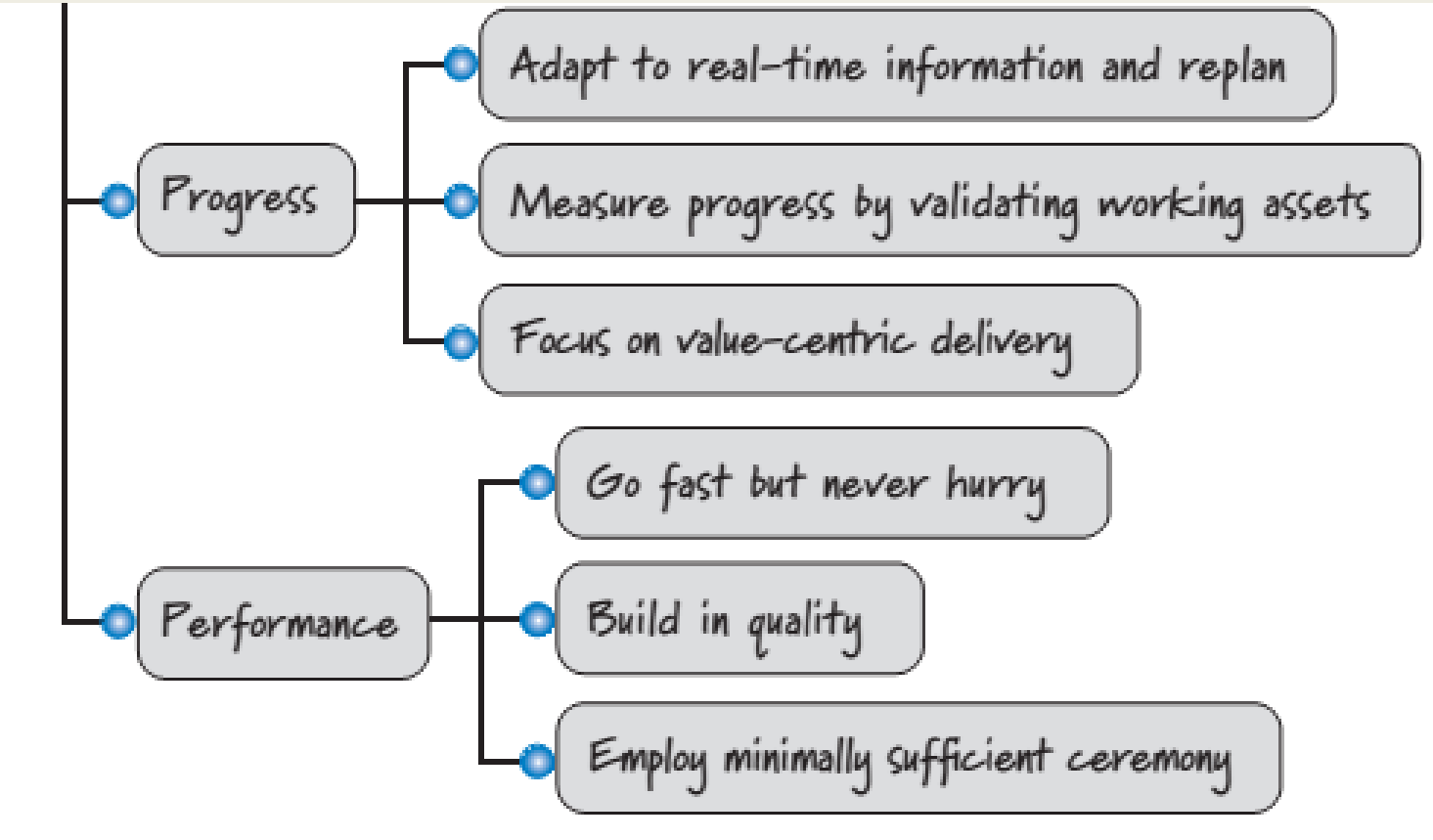
2020

Categorization of principles (Up)

Categorization of principles (Middle)



Categorization of principles (Bottom)

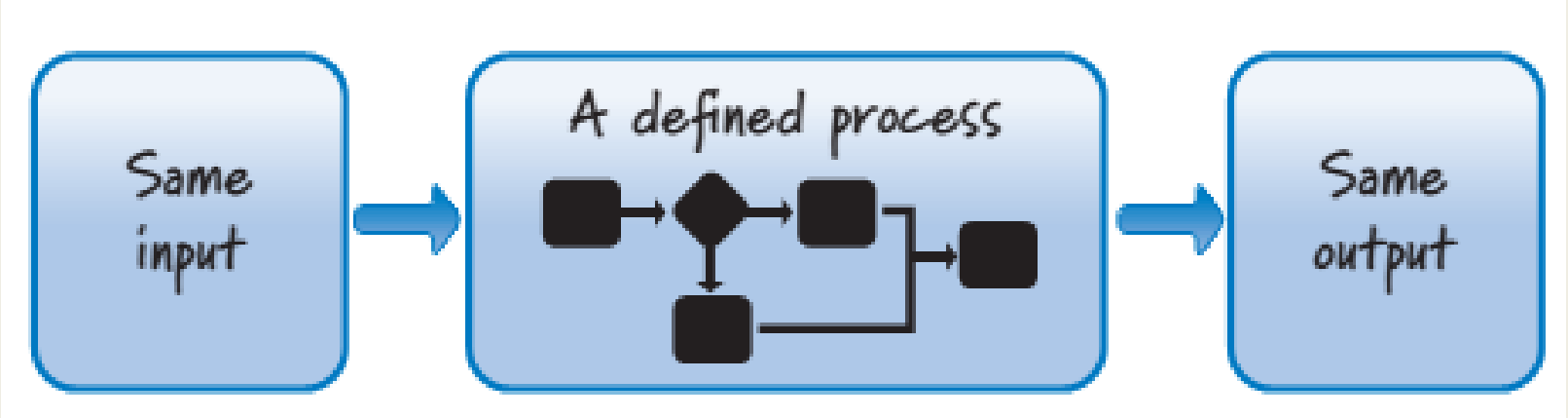


Variability and Uncertainty

- To create innovative solutions.
- Four related principles
  1. *Embrace helpful variability.*
  2. *Employ iterative and incremental development.*
  3. *Leverage variability through inspection, adaptation, and transparency.*
  4. *Reduce all forms of uncertainty simultaneously.*

Embrace Helpful Variability (I)

- Plan-driven processes treat product development like manufacturing. In manufacturing our goal is to take a fixed set of requirements and follow a sequential set of well-understood steps to manufacture a finished product that is the same (within a defined variance range) every time.
- In product development, the goal is to create the unique single instance of the product, not to manufacture the product. This single instance is analogous to a unique recipe.



Embrace Helpful Variability (II)

- We don’t want to create the same recipe twice;
- Instead, we want to create a unique recipe for a new product. Some amount of variability is necessary to produce a different product each time.
- In fact, every feature we build within a product is different from every other feature within that product, so we need variability even at this level.

Employ Iterative and Incremental Development (I)

- Plan-driven, sequential development assumes that we will get things right up front and that most or all of the product pieces will come together late in the effort.

- Scrum, on the other hand, is based on iterative and incremental development.

Iterative Development (I)

- Acknowledges that we will probably get things wrong before we get them right and that we will do things poorly before we do them well (Goldberg and Rubin 1995).
- Is a planned rework strategy. We use multiple passes to improve what we are building so that we can converge on a good solution.
- Iterative development is an excellent way to improve the product as it is being developed.
- The biggest downside is that in the presence of uncertainty it can be difficult up front to determine (plan) how many improvement passes will be necessary.

Iterative Development (II)

- For example, we might start by creating a prototype to acquire important knowledge about a poorly known piece of the product. Then we might create a revised version that is somewhat better, which might in turn be followed by a pretty good version.

- In the course of writing this book, for example, I wrote and rewrote each of the chapters several times as I received feedback and as my understanding of how I wanted to communicate a topic improved.

Incremental Development (I)

- Based on the age-old principle of “Build some of it before you build all of it.” We avoid having one large, big-bang-style at the end of development.
- Instead, we break the product into smaller pieces so that we can build some of it, learn how each piece is to survive in the environment in which it must exist, adapt based on what we learn, and then build more of it.
- Incremental development slices the system functionality into increments (portions).
- Incremental development gives us important information that allows us to adapt our development effort and to change how we proceed.
- The biggest drawback to incremental development is that by building in pieces, we risk missing the big picture (we see the trees but not the forest).

Incremental Development

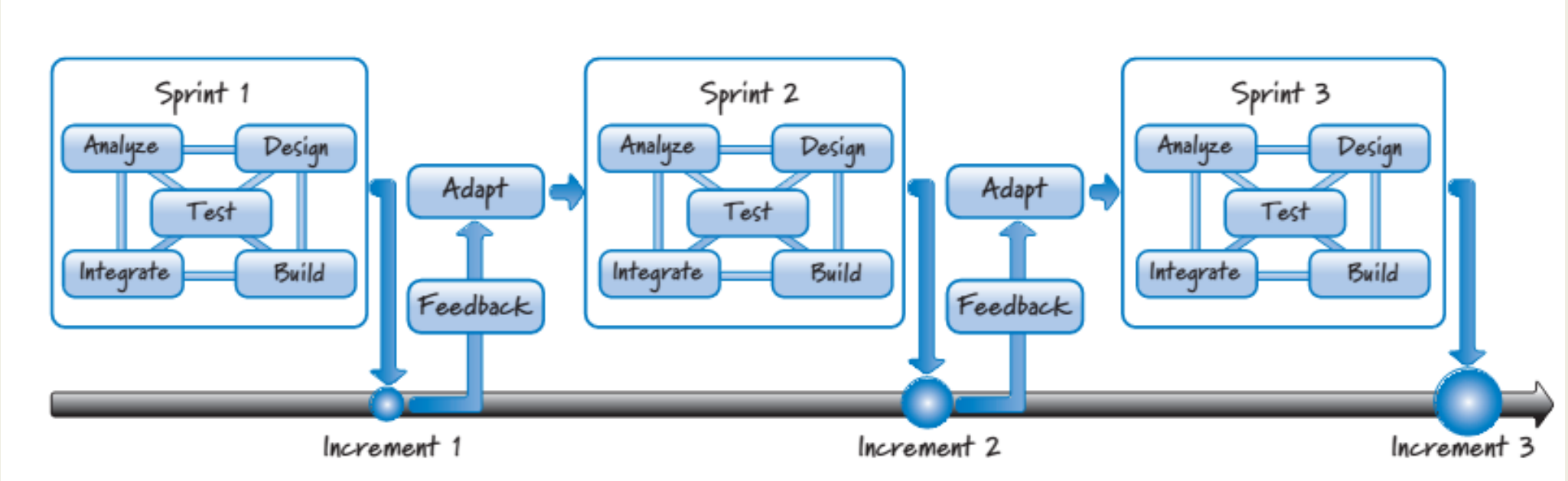
- For example, while writing this book, I wrote a chapter at a time and sent each chapter out for review as it was completed, rather than trying to receive feedback on the entire book at once.
- This gave me the opportunity to incorporate that feedback into future chapters, adjusting my tone, style, or delivery as needed.
- It also gave me the opportunity to learn incrementally and apply what I learned from earlier chapters to later chapters.

Iterative and Incremental Development in Scrum (I)

- Scrum leverages the benefits of both iterative and incremental development, while negating the disadvantages of using them individually.

- Scrum does this by using both ideas in an adaptive series of time boxed iterations called sprints.

Iterative and Incremental Development in Scrum (II)



Iterative and Incremental Development in Scrum (III)

- During each sprint we perform all of the activities necessary to create a working product increment (some of the product, not all of it).
- This all-at-once approach has the benefit of quickly validating the assumptions that are made when developing product features.
- For example, we make some design decisions, create some code based on those decisions, and then test the design and code—all in the same sprint. By doing all of the related work within one sprint, we are able to quickly rework features, thus achieving the benefits of iterative development.

Iterative and Incremental Development in Scrum (IV)

- In Scrum, we don't work on a phase at a time; we work on a **feature** at a time.
- So, by the end of a sprint we have created a valuable product increment (some but not all of the product features). That increment includes or is integrated and tested with any previously developed features; otherwise, it is not considered done.
- At the end of the sprint, we can get feedback on the newly completed features within the context of already completed features. This helps us view the product from more of a big-picture perspective than we might otherwise have.

Iterative and Incremental Development in Scrum (V)

- We receive feedback on the sprint results, which allows us to adapt.
- We can choose different features to work on in the next sprint or alter the process we will use to build the next set of features.
- In some cases, we might learn that the increment, though it technically fits the bill, isn't as good as it could be. When that happens, we can schedule rework for a future sprint as part of our commitment to iterative development and continuous improvement.
- This helps overcome the issue of not knowing up front exactly how many improvement passes we will need.
- Scrum does not require that we predetermine a set number of iterations.
- The continuous stream of feedback will guide us to do the appropriate and economically sensible number of iterations while developing the product incrementally.

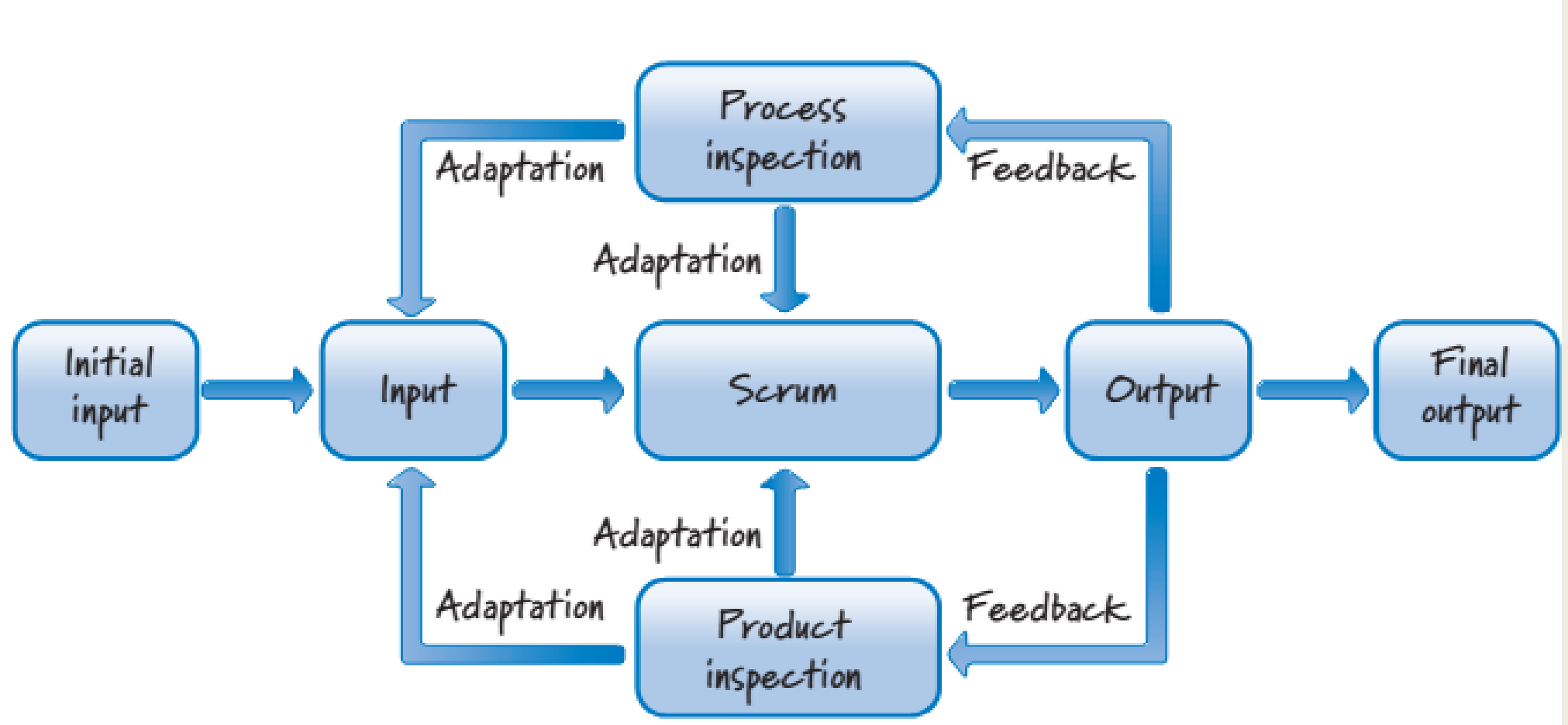
Leverage Variability through Inspection, Adaptation, and Transparency

- A plan-driven, sequential development process assumes little or no output variability. It follows a well-defined set of steps and uses only small amounts of feedback late in the process.
- Scrum embraces the fact that in product development, some level of variability is required in order to build something new.
- Scrum also assumes that the process necessary to create the product is complex and therefore would defy a complete up-front definition. Furthermore, it generates early and frequent feedback **to ensure that the right product is built and that the product is built right.**

Comparison

Dimension	Plan-Driven	Scrum
Degree of process definition	Well-defined set of sequential steps	Complex process that would defy a complete up-front definition
Randomness of output	Little or no output variability	Expect variability because we are not trying to build the same thing over and over
Amount of feedback used	Little and late	Frequent and early

Scrum Process model



Inspect, Adapt and Transparency

- Are heart of Scrum.

- In Scrum, we inspect and adapt not only what we are building but also how we are building it

Transparency

- All of the information that is important to producing a product must be available to the people involved in creating the product.
- Transparency makes inspection possible, which is needed for adaptation.
- Transparency also allows everyone concerned to observe and understand what is happening.
- It leads to more communication and it establishes trust (both in the process and among team members).

Reduce All Forms of Uncertainty Simultaneously (I)

- Developing new products is a complex endeavor with a high degree of uncertainty.
- Three categories of uncertainty.
  1. *End uncertainty (what uncertainty)—uncertainty surrounding the features of the final product.*
  2. *Means uncertainty (how uncertainty)—uncertainty surrounding the process and technologies used to develop a product.*
  3. *Customer uncertainty (who uncertainty) —who the actual customers of their products will be.*
     - This uncertainty must be addressed or they might build brilliant products for the wrong markets.

Reduce All Forms of Uncertainty Simultaneously (II)

- Traditional, sequential development processes focus first on eliminating all end uncertainty by fully defining up front what is to be built, and only then addressing means uncertainty.
- This simplistic, linear approach to uncertainty reduction is ill suited to the complex domain of product development, where our actions and the environment in which we operate mutually constrain one another.

Reduce All Forms of Uncertainty Simultaneously (III), An Example

- We decide to build a feature (our action).
- We then show that feature to a customer, who, once he sees it, changes his mind about what he really wants, or realizes that he did not adequately convey the details of the feature (our action elicits a response from the environment).
- We make design changes based on the feedback (the environment's reaction influences us to take another unforeseen action).

Reduce All Forms of Uncertainty Simultaneously (IV)

- In Scrum, we do not constrain ourselves by fully addressing one type of uncertainty before we address the next type.
- Instead, we focus on simultaneously reducing all uncertainties (end, means, customer, and so on).
- Of course, at any point in time we might focus more on one type of uncertainty than another. Simultaneously addressing multiple types of uncertainty is facilitated by iterative and incremental development and guided by constant inspection, adaptation, and transparency.
- Allows to opportunistically probe and explore our environment to identify and learn about the unknown unknowns(the things that we don't yet know that we don't know) as they emerge.

Reference

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