CS207 Software Design

Agile & Scrum

What is software engineering?

- A systematic, disciplined, quantifiable approach to development
  - We attempt to formalize software development in order to develop best practices to ensure quality and consistency
  - Structural engineers are responsible if their bridge falls apart – we want the same accountability for software developers
- Recognizing problems that are already solved and applying known solutions can ensure a higher level of consistency
  - We do not always need to reinvent the wheel

Software Engineering Components

- Requirements analysis
- Design
- Construction
- Testing
- Validation
- Maintenance

UNIVERSITY OF TORONTO

The early days of software development

- In the early days, the cost of developing software was relatively small compared to the hardware cost
- The software was offered for free to the customers who purchased the computing equipment

Issues today

- Large software projects cost tens of millions (or even hundreds of millions) of dollars to produce, employing hundreds of developers, and other team members of all sorts
- Keeping this massive projects on-time and on-budget has become a huge priority, and for the most part a huge nightmare
- Companies are less willing to take risks and innovate since they are not guaranteed a success in the market
- Developers are expected to work long hours, putting their health and other factors at risk

From lone wolf to team member

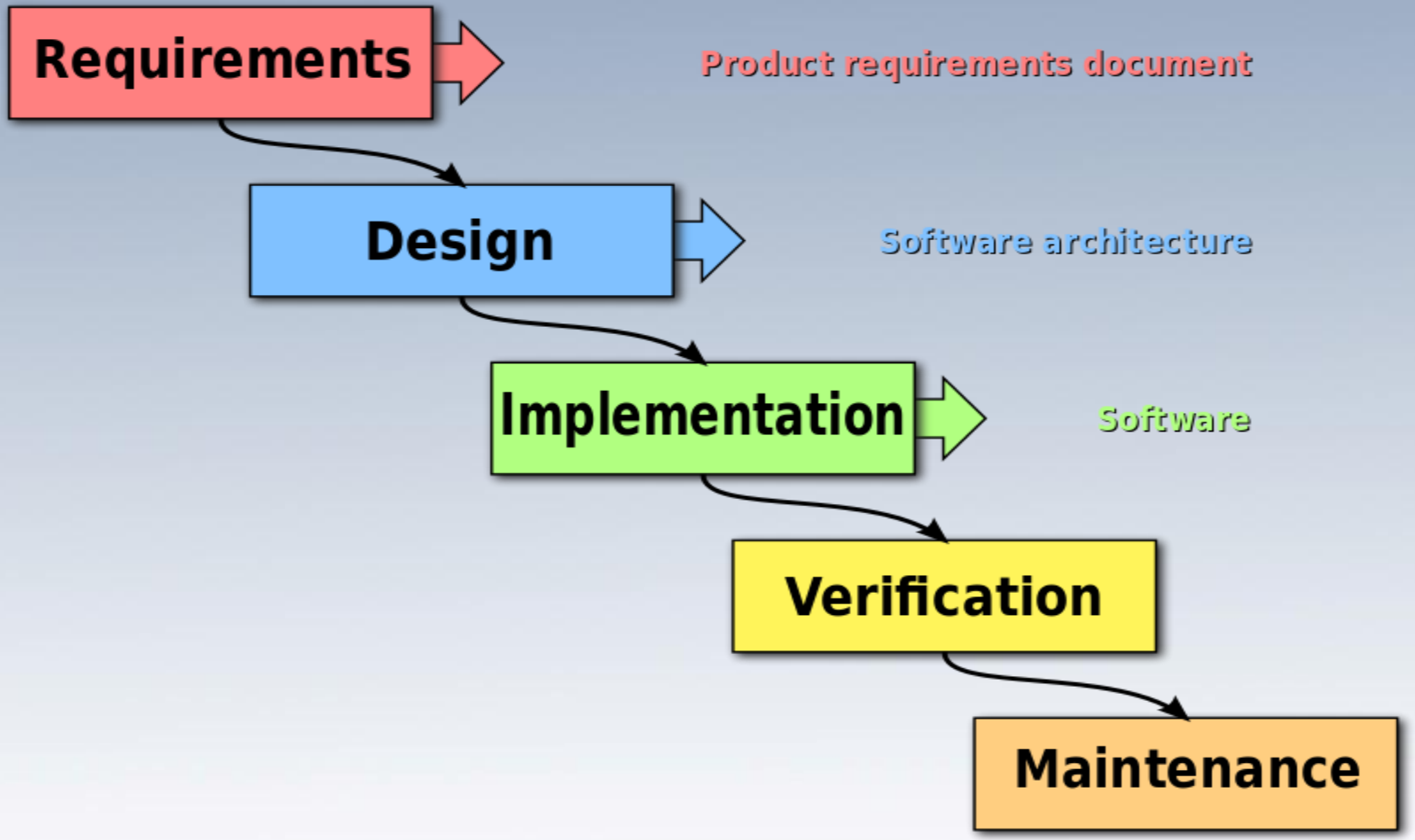
- Initially, software was often written by a single developer
- As software became more complex there became a need to scale and grow out teams of people to work on a single property
- Increased budgets / risk caused companies to start finding ways to try and minimize these potential pitfalls

Reducing the risk

- To reduce the risk, many companies adopted waterfall-style methodologies used by other industries.
- Waterfall methodology employed the idea of developing a large software project through a series of phases.
- Each phase lead to a subsequent phase more expensive than the previous.
- The initial phases consisted of writing plans about how to build the software.
- The software was written in the middle phase.
- The final phase was integrating all the software components and testing the software. Each phase was intended to reduce risk before moving on to more expensive phases.

UNIVERSITY OF TORONTO

The waterfall model



Waterfall continued

- Each phase is more expensive than the previous
  - More time spent in the earlier phases translates to savings later on down the line
- Projects should not return to a previous stage once they have moved on
  - Extremely linear and rigid
  - Does not allow for many revisions or change things not considered in the design phase

UNIVERSITY OF TORONTO

Why old methodologies failed

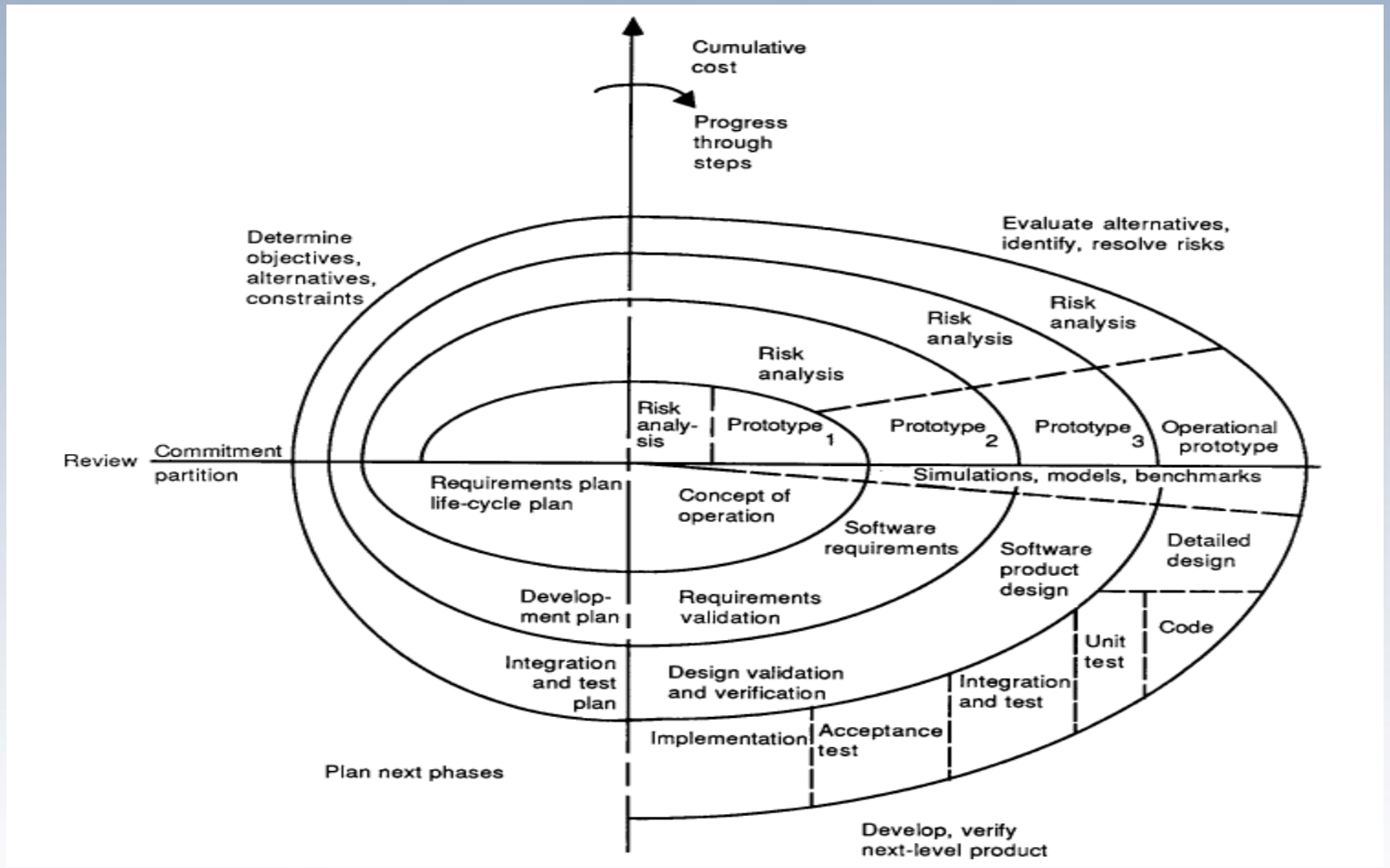
- The mains reason is the so-called “cognitive friction” – the resistance encountered by a human intellect when engaged with a complex system of rules that change as the problem changes.
- Example: violin is a complex instruments but never enters a “meta-state” in which various inputs make is sound like a piano or bell. Although violin’s behaviour is complex, it is predictable.
- On a computer screen, everything is filled with cognitive friction. Even a simple interface as www presents the user with a more intense mental engagement because all you can do is click on a hyperlink, but what the link points to can change independently of the pointer without any outward indication.
- So the hyperlink’s function is actually a hyperfunction!

What goes wrong?

- Unachievable ship date (features keep changing during development)
- Going into production too soon
- Adding people late to the project (count how much time do new people to come up to speed!)
  - Fred Brooks law
- Underestimating technical challenges

UNIVERSITY OF TORONTO

Early alternatives – spiral model



Alternative Methodologies

- As companies began to realize that the waterfall method was failing them (large projects were failing completely or going way over budget) alternatives were sought
- A gradual trend was in methods that used an incremental development of product using iterations (almost like smaller waterfalls)
  - Iterations could be only a few weeks, but still included the full cycle of analysis, design, coding, etc.
- These various lightweight development methods were later referred to as **agile** methodologies

Agile Manifesto

- We value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan
  - (there is value to the items on the right, but the left is valued more)

Agile frameworks

- Agile refers to a number of different frameworks that share these values:
  - Extreme programming (XP) (advocates frequent "releases" in short development cycles, which is intended to improve productivity and introduce checkpoints at which new customer requirements can be adopted; also programming in pairs)
  - Scrum (a flexible, holistic product development strategy where a development team works as a unit to reach a common goal)
  - Lean software development:
    - Eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build flexibility in, see the whole

Why use Agile?

- Demand for higher quality, with lower cost
- Post-mortems of various projects lead to a lot of knowledge gain about what went right / wrong during the development phase
  - With the waterfall model, we do not have a clear way to predict the future, so these lessons are always in hindsight
  - Smaller, iterative schedules mean problems can be identified / address much earlier in the cycle

Using iterative development

- Iterative design means we build a product in small steps
  - Incrementally add features
  - Software is in working condition at least every few weeks
  - Allows people to test earlier in the development cycle

UNIVERSITY OF TORONTO

UNIVERSITY OF TORONTO

Agile Team Management

- From the bottom up
- Teams are empowered to manage the smallest level of details, while leaving the higher levels to upper management
- Teams upon seeing the small amount of ownership they get from solving smaller problems take on responsibility for larger problems
  - Head off issues before they become major problems
  - Individuals solve problem with their colleagues

An Agile Project

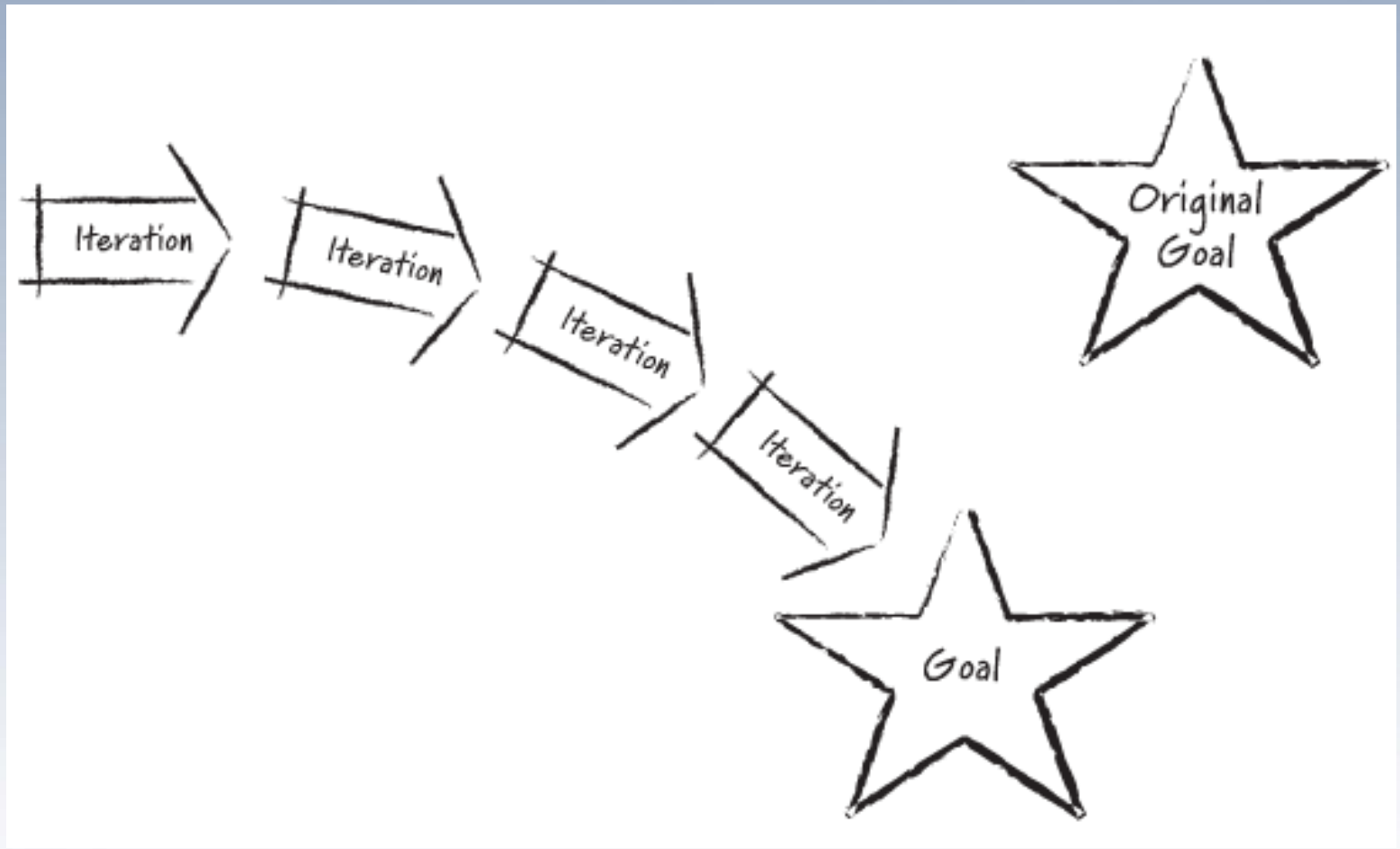
- An agile project consists of a series of iterations of development
- Each interval usually lasts only two to four weeks
- Developers implement features during each iteration that add value to the project
  - The features are called **user stories**

Iterations

- Each iteration contains a full development cycle
  - Analysis
  - Design
  - Coding
  - Testing
- The product is reviewed at the end of each iteration
  - Results are used to direct future iterations

UNIVERSITY OF TORONTO

Iterations

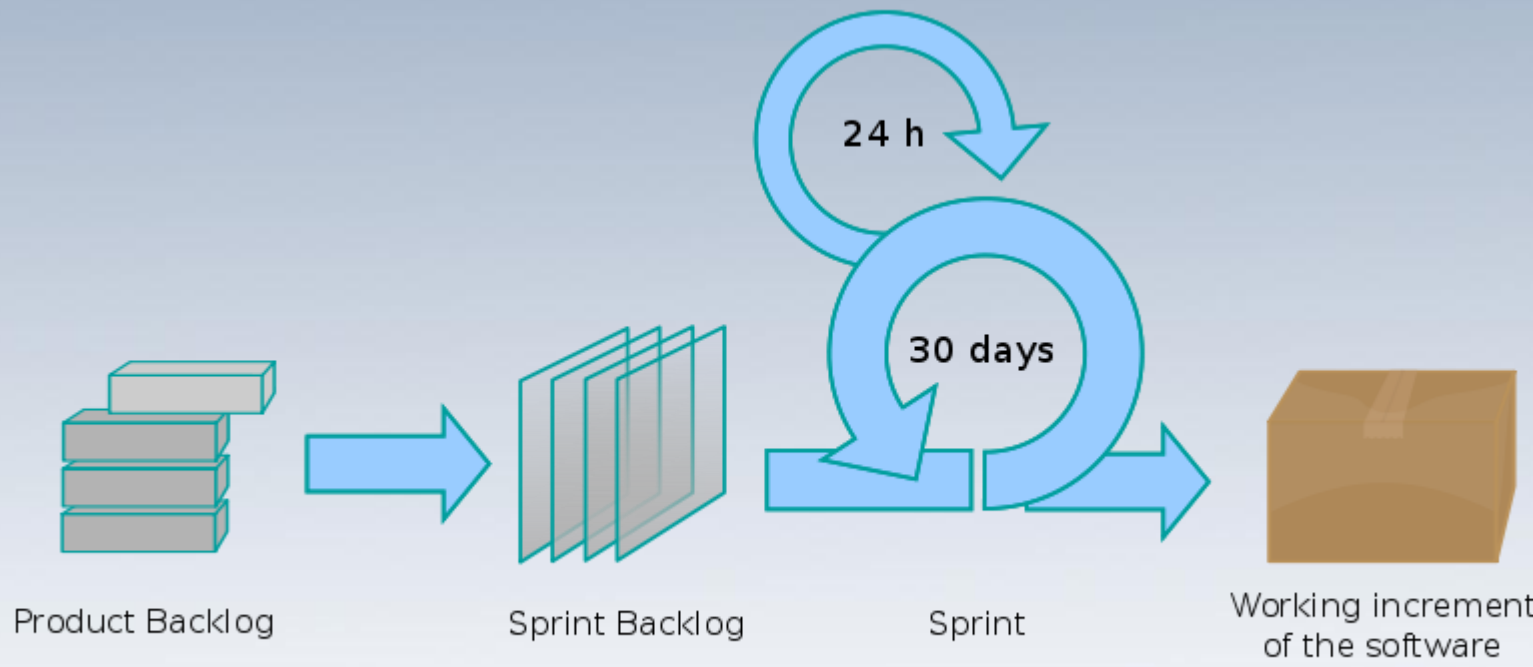


Sprints

- In traditional agile development (outside of the game industry) software is brought to release level every few months
  - They use releases, which are sets of sprints, to produce shippable versions of their products
- Agile projects use these sprints as milestones for deliverables such as “near shippable”

UNIVERSITY OF TORONTO

Scrum



Scrum

- Scrum is mainly about the management of software development projects
- Sprint
  - Basic unit of development in a scrum
  - Fixed-length – typically from one week to a month
  - Each sprint begins with a planning meeting to determine the tasks for the sprint and estimates are made
  - During each sprint a potentially deliverable product is produced
    - Features are pulled from a product backlog – a prioritized set of high-level work requirements

Scrum

- During a sprint, there are daily scrum meetings
  - Daily scrum or daily standup
  - No more than 15 minutes
  - Meeting must start on-time, and happen at the same location
  - Each member answers the following:
    - What have you done since yesterday?
    - What are you planning on doing today?
    - Any impediments or stumbling blocks?
  - A scrum master will handle resolving any impediments outside of this meeting

Key features of a scrum

- During a project a customer may change their minds about what they want / need
  - Accepts that the problem cannot be fully understood or defined, instead allows teams to delivery quickly and respond to changes in a timely manner

UNIVERSITY OF TORONTO

What is Scrum?

- A framework for creating complex products
- Creates an incremental and interactive development process
- Is self-managing
- Uses cross-disciplined teams
- An ever-changing, never-ending pursuit of improvement
- Consists of simple rules, but these rules vastly improve how teams work together

What scrum is not

- It is not a processor or methodology
- It is not specific enough to tell programmers, artists, QA, producers, etc. how to do their jobs
  - Companies merge their own practices into the scrum framework to create a new methodology

Process

- Software development, using scrum, progresses in two to four week iterations called sprints
  - Teams are generally 6-10 people in size and consist of a range of disciplines (not just programming)
- Sprints start with a **sprint planning meeting**
  - Each team selects a set of features from the **product backlog** – a prioritized list of features, each feature is referred to as a **product backlog item (PBI)**
  - The team creates time estimates for each PBI to create a **sprint backlog**
  - Teams only select features that they believe are achievable

Process continued

- Teams have daily 15-minute meetings (called the **daily scrum**)
  - Progress and any impediments to work are shared
  - Meetings are strictly limited to 15 minute **timeboxes**, regardless of whether or not the agenda is met
- At the end of a sprint, a **potentially shippable** version of the software has been created
  - Stakeholders (to be defined later) have a **sprint review meeting** to determine if the goals of the sprint were met and make adjustments as needed to the product backlog
- There may also be a **sprint retrospective** – sort of like a mini post-mortem

Scrum Principles

- Although scrum practices will evolve and be adapted to suit each individual team / project, a number of core principles should be upheld:
  - **Empiricism**: inspect and adapt – respond to emerging changes
  - **Emergence**: we can't know everything about a design at the start, so we can capitalize on a emerging features as they become apparent
  - **Timeboxing**: scrum delivers value on a regular basis, allows project to be synchronized and micro-steered throughout
  - **Prioritization**: some features are more important to stakeholders than others. Features are developed based on their value
  - **Self-organization**: small, cross-disciplined teams are given power to organize their membership, manage processes and create the best possible product within timeboxes

Product Backlog

- Prioritized list of requirements (called PBI's)
- The product backlog can change after each sprint
  - New PBI's that weren't thought of can be added
  - PBI's that are no longer needed can be removed
  - Priorities can be updated

Sprints

- The overall objective of a sprint is called the sprint goal
  - Sprint goals remains unchanged throughout the sprint
  - At the end of a sprint the stakeholders are shown a new version of the game which demonstrates the sprint goal
- Sprints produce vertical slices of functionality
  - Each sprint contains design, coding, testing, debugging and optimization
- Many features will require multiple sprints to develop
  - Each sprint must still demonstrate value
  - Allows for uncertainty or risk to be removed from the project as early as possible

Scaling Scrum

- Scrum teams should have less than a dozen members
- Larger teams are supported through scaling
  - A number of Scrum teams work in parallel
  - Work is coordinated through practices like the scrum of scrums

UNIVERSITY OF TORONTO

Sprint basic rules

- Sprint are timeboxed, usually two to four weeks in length
- The team commits to completing a print goal
- No additions or changes are made by anyone outside the team

UNIVERSITY OF TORONTO

Sprint Planning

- A the beginning of a sprint, teams meet with stakeholder to plan the next sprint over two meetings:
  - Sprint prioritization meeting
  - Sprint planning meeting