Objectives

- Review Schedule
- Lean Thinking

Reminder

- Quiz 1 will be held, in-class on September 23
    - Cumulative
    - Staff will post some sample questions to the moodle this week

Managing Chaos

- According to Brechner[1], project management is about *limiting chaos*
  - Helpful way to differentiate between different software processes
- Waterfall: chaos is managed by a detailed plan and milestones
- Scrum: all work done in *time-boxed* sprints
  - A *timebox* is a period of 2-3 weeks where all requirements are frozen
- Kanban: limit the current work in progress

[1] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

A Simpler Time

A RON HOWARD FILM
STARRING MICHAEL KEATON
GUNG HO
THE COMEDY WITHOUT BRAKES.
PARAMOUNT PICTURES PRESENTS A RON HOWARD FILM · MICHAEL KEATON · GUNG HO
MUSIC SCORE BY THOMAS NEWMAN · DIRECTOR OF PHOTOGRAPHY DON PETERMAN, A.S.C.
EXECUTIVE PRODUCER RON HOWARD · SCREENPLAY BY LOWELL GANZ & BABALOO MANDEL
STORY BY EDWIN BLUM AND LOWELL GANZ & BABALOO MANDEL
PRODUCED BY TONY GANZ AND DEBORAH BLUM
DIRECTED BY RON HOWARD
A PARAMOUNT PICTURE
FILMED IN PANAVISION
DOLBY STEREO
PG-13

Figure: Movie Poster for ‘‘Gung Ho’’ (1986)

- According to wikipedia “*Gung Ho* received mixed to negative reviews”

NUMMI[2]

Figure: `New United Motor Manufacturing, Inc. Fremont CA`

- Joint venture between General Motors and Toyota
  - Closed in 2010, now a Tesla plant
- Using the *Toyota Production System*, transformed the “worst workforce in the industry” into one of the most productive

[2] Mike Parker and Jane Slaughter. “Management-by-stress: The team concept in the US auto industry”. In: *Science as Culture* 1.8 (1990), pp. 27–58.

NUMMI[2]

Figure: New United Motor Manufacturing, Inc. Fremont CA

- Joint venture between General Motors and Toyota
  - Closed in 2010, now a Tesla plant
- Using the *Toyota Production System*, transformed the "worst workforce in the industry" into one of the most productive
- The ideas and philosophy of the TPS are known as *lean*

[2] Mike Parker and Jane Slaughter. "Management-by-stress: The team concept in the US auto industry". In: *Science as Culture* 1.8 (1990), pp. 27–58.

Lean Thinking

- *Lean* is an idea that has been applied in manufacturing for many decades
- Adapted for software development by the Poppendiecks[3] in a series of influential books
- Lean is a “mindset”
  - Way of thinking about problems, but does not prescribe practices like Scrum or XP
  - Includes values and principles and provides tools for thinking
- XP and Scrum also include their own sets of values (overlap across all agile methodologies)
  - XP Values: *simplicity*, *communication*, *feedback*, *respect*, *courage*

[3] Mary Poppendieck and Tom Poppendieck. *Lean software development: an agile toolkit*. Addison-Wesley, 2003.

Push vs. Pull

Figure: Piggly-wiggly Grocery Store

- Piggly-wiggly was one of the first grocery stores to implement the self-service model that we have all become familiar with

Lean Values[4]

- The Lean values for software development are
  - Eliminate Waste
  - Amplify learning
  - Decide as late as possible
    - Defer decisions until the last responsible moment
  - Deliver as fast as possible
  - Empower the team
  - Build integrity in
  - See the whole
- Lean shares many of these values with other agile processes
- In fact, creators of popular processes such as Scrum already knew about Lean ideas as applied in manufacturing

[4] Mary Poppendieck and Michael A Cusumano. "Lean software development: A tutorial". In: *IEEE software* 29.5 (2012), pp. 26–32.

Eliminating Waste

- **Waste** is any work the team does that does not actively contribute to building better software
  - Example: creating a meticulous plan with detailed milestones at the beginning of a project; spending a lot of time writing documentation or comments that no one uses or quickly become out-of-date
  - Can you think of anything from your projects that you would consider waste?
- But before we can eliminate waste we have to **see waste**

Seven Wastes of Software Development

- This list of wastes in software was adapted from a list of seven manufacturing wastes (Shigeo Shingo, one of the creators of the TPS)[5]
- Seven wastes in software
  - Partially done work
  - Extra processes
    - Status meetings
  - Extra features
  - Task switching
  - Waiting
  - Motion
  - Defects

[5] Mary Poppendieck and Tom Poppendieck. *Lean software development: an agile toolkit*. Addison-Wesley, 2003.

Value Stream Map

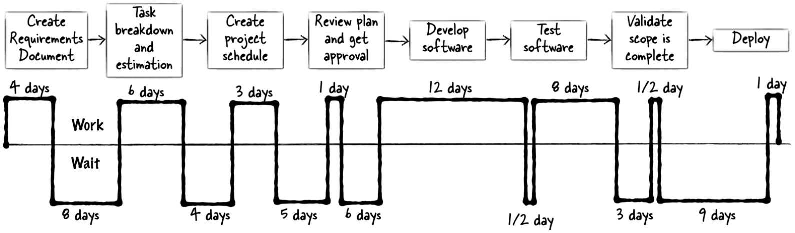


Figure: Value stream map in waterfall process

- A *Value Stream Map* is a tool for helping to identify when time is wasted in a project
- You can create a value stream map for a *minimal marketable feature* that you have already completed
- Real history of something you did

Amplify Learning

- **Use feedback from the project to improve how the software is built**
- In Scrum we introduce feedback to development by engaging in short time-boxed iterations
- A thinking tool from Lean is *set-based development*

Tool: Options Thinking

- Why does a *satisfaction guaranteed* clause make a potential purchase more attractive?

Tool: Options Thinking

NAZARBAYEV UNIVERSITY
SCHOOL OF ENGINEERING AND DIGITAL SCIENCES

- Why does a *satisfaction guaranteed* clause make a potential purchase more attractive?
- Delay decisions until uncertainty is reduced
- Anecdote: HP Printers
- **Set-based development** pursue several solutions at once

Commitment

- A commitment is something that your obligated to do
  - Example: in Scrum you are committed to delivering the sprint goal
- **Options thinking** is being able to distinguish between what you are committed to and what you can do
- **Set-based development** is the idea of pursuing several solutions at once
  - A/B Testing is also an example of set-based development
- Unrealistic commitments can lead to *technical debt*

Technical Debt

- Lowering quality can shorten development in the short term but lengthen development in the long term
  - You will eventually have to face the consequences of cutting corners in many places
- Patterns of technical debt: schedule pressure, duplication, getting things right the first time

Causes of Schedule Pressure

- Scope Creep: Features added to the project without removing scope
- Outside estimates
- Change in composition of the team
- Late integration

Magical Thinking

- One management style: set aggressive goals and expect team members to individually rise to the occasion
  - Rewards “heroic” behavior by individuals
- Management should rather focus on teamwork
- It is easy to fall in the trap of *magical thinking*
  - Thinking that anything is possible

Empower the Team

- **Establish a focused and effective work environment, and build a whole team of energized people.**
- XP and Scrum both have practices that align with this value
    - Scrum work is not assigned
    - XP has the idea of *energized work*: achieving sustainable pace with good work/life balance, only working when you can be effective

Building Integrity In

- A team with a lean mindset thinks about how to build integrity into their product
  - Both *external* and *internal* integrity
- Lean thinking tools for internal integrity are refactoring and testing
- **Perceived Integrity**: how well the software meets the needs of the user
- **Conceptual Integrity**: how the features of the software work together

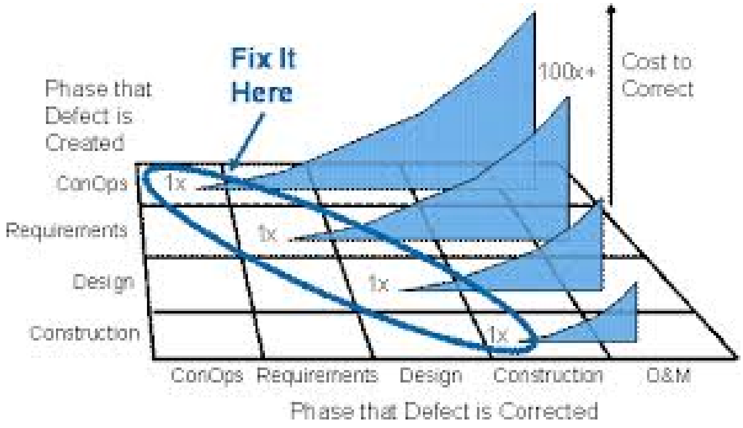
Acceptable Rates of Error

- Bugs (Defects) are an inevitable part of software development
- Simply not possible to entirely eliminate them
- Perfect (v.) your software don't try to make it perfect (adj.)
- Try to increase the mean time from one failure to the next to an acceptable level
  - Acceptable in terms of economics, human constraints
  - Compare an airplane system to a high-traffic website
  - "Five Nines"

Testing in XP

- *Test-Driven Development* (TDD) is an XP practice that brings testing into the inner loop of programming
- Test harness run frequently against the code base (for example, during every build)
- The idea that unit tests should be written *before* the implementation
- Lower rate of bugs $\rightarrow$ growth of trust in the team
    - Don't hide errors to protect yourself

Cost of Fixing a Bug



- The total cost to correct a bug (time spent, monetary, lost business, etc.) increases the later the bug is detected (image credit[6])
- XP uses early testing as a way to double-check all work

[6] https://ops.fhwa.dot.gov/publications/seitsguide/section3.htm

See (Optimize) the Whole

- Understand, objectively, how the team works, including any flaws
- *Measurements* as a thinking tool
    - Lead time: average time between when a feature is requested and when it is delivered
- Understand root causes
    - Use the technique of the *Five Whys*

Tool: The Last Responsible Moment

- Breadth-First vs. Depth-First Search
- The moment when “failing to make a decision eliminates an important alternative”[7]
- In software, good OO design allows you to delay decisions
  - Design with low coupling and high cohesion
  - Encapsulate what varies
  - SOLID Principles

[7] Mary Poppendieck and Tom Poppendieck. *Lean software development: an agile toolkit*. Addison-Wesley, 2003.

Using Kanban (1)

NAZARBAYEV UNIVERSITY
SCHOOL OF ENGINEERING AND DIGITAL SCIENCES

- Kanban means *signboard* in Japanese
- Method to visualize work by placing cards or sticky notes on a board. As work progresses, the cards flow across the board
- Many guides to implementing Kanban available and also online tools such as *Trello*
    - Our discussion based on Brechner[8]

[8] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Using Kanban (2)

- The most important work done by a software team is producing new and improved products, features, and infrastructure...this is the focus of Kanban
  - Team may do other work such as fixing bugs, talking to the client, ansering emails etc.
- From the waterfall methods, we know the basic activities that are involved in developing software: planning, design, implementation, testing, maintenance

Work Routine in Kanban

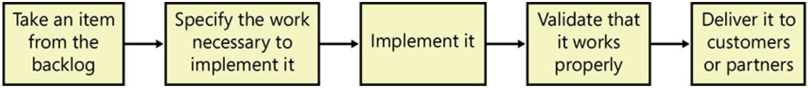

Take an item from the backlog
Specify the work necessary to implement it
Implement it
Validate that it works properly
Deliver it to customers or partners

Figure: `Image source`[9]

- This diagram shows a basic process that can be used in Kanban
- As a feature is complete it moves from left to right across the board until it is finally *done*

[9]Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

The Signboard

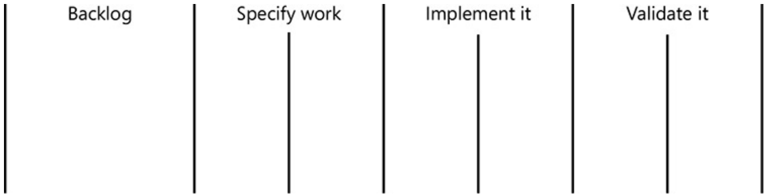


Figure: `Image source`[10]

- The signboard is the main artifact that is used to manage the project in Kanban, an example is shown in the image
- The middle columns are divided in half with left representing active work and right representing items that are completed
- The backlog basically has the same meaning as the *product backlog* in Scrum

[10] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Setting Limits

- In Scrum, how do we limit work or keep things from getting out of control?

Setting Limits

- In Scrum, how do we limit work or keep things from getting out of control?
  - Work is limited by *timeboxing*, changes are only allowed at the beginning of the next sprint
- Likewise, in Kanban, work is controlled by setting limits on the amount of work that can be in progress at any one time
  - Only so many note cards can be placed at each step
  - *Work in Progress* limit
- You should try to set the work-in-progress limits as small as possible to keep the team engaged
- Clearly, the limits will depend on the size of your team

Example Work-in-Progress Limits

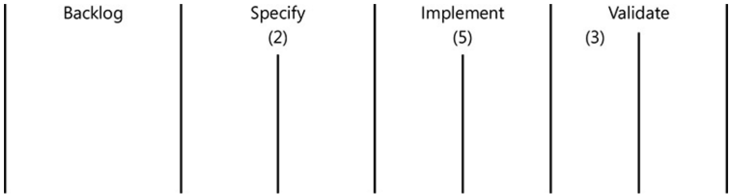


Figure: Image source[11]

- Cards in both columns of the specify or implement steps count against the limit
- For the final step (validation), done items do not count against the limit
- Notice that this system distinguishes between finishing one step and beginning the next

[11] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Done Rules

- Can define what *done* means for each step
- For example,
  - *Specify*: Backlog item broken down into smaller tasks (each can be finished in around 1 day)
  - *Implement*: All unit and acceptance tests pass (JUnit green bar for example)
  - *Validate*: Feature/Software tested by customer in production environment

Comparing to Scrum

- There are fewer defined roles in Kanban compared to Scrum
  - Only have the development team and the product owner
  - Responsibility of the product owner similar to Scrum (prioritize backlog items)
- There are no milestones or retrospectives, once the signboard and backlog are in place, Kanban flows continuously
- Like Scrum, Kanban also has the notion of a *daily stand-up*
  - Opportunity for teams members to say that they are blocked
  - Rearrange the cards on the signboard

Flow Example (1)

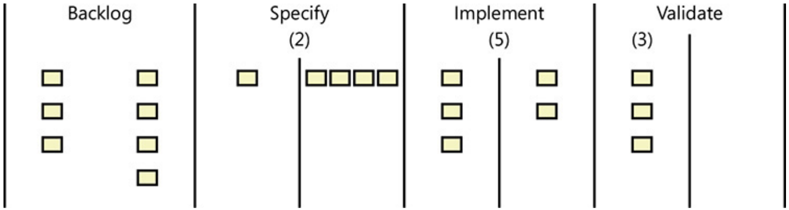


Figure: Image source[12]

- This shows an example of what the signboard for a project might look at a particular point in time
- Cards can only move to the right if there is room (*pull system*)
- Suppose that two items are validated, how would the cards move?

[12] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Flow Example (2)

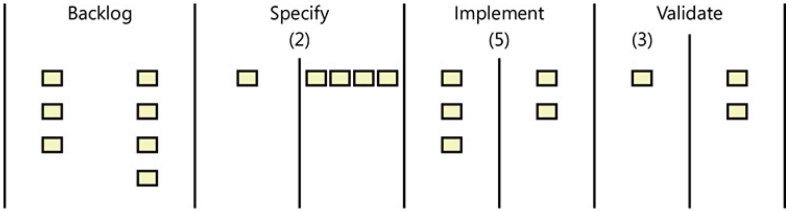


Figure: `Image source`[13]

- Two cards in the validate column are moved to done
- Now there are free slots for validation (notice that two items are done with implementation)

[13] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Flow Example (3)

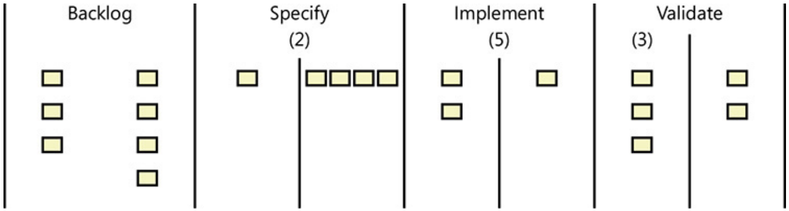


Figure: Image source[14]

- This image shows the state of the board after
  - The done implementation items are moved to the validate step
  - The team informs that one of the implementation cards is finished
- Next, the top items from specify will move over to implementation. Also, assume that the active specify task has been broken down into 2 smaller tasks

[14] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Flow Example (4)

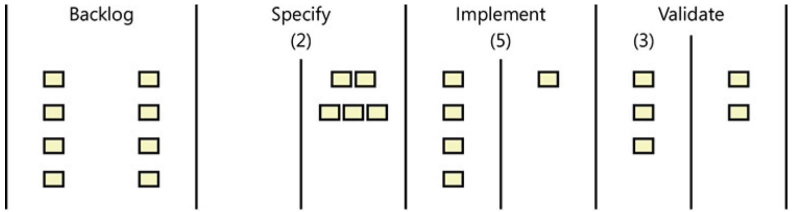


Figure: Image source[15]

- Notice that no items have moved from the backlog to specify
- Implementation is blocking progress. For any manager this reveals an area to focus on

[15] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.

Problems

- In the example above the specification work was blocked because all items were done, and the implementation step was full
- There are other reasons that work may become blocked...
  - Prior step has no items done, nothing to pull
  - Item might need some substantial design work. One option would be to create a card for the design and define each step's done for that card

Dealing with Dependencies

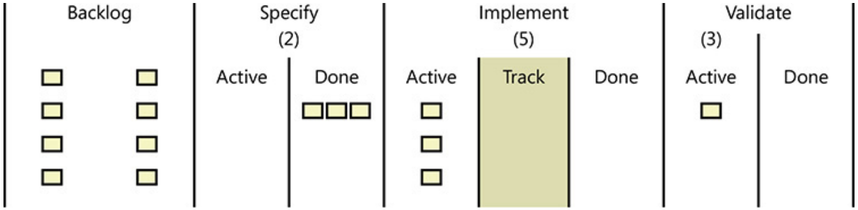


Figure: `Image source`[16]

- Oftentimes, progress on one item might require the completion of some other task or of some external input
- In such cases we can introduce a *Track* column to the relevant step
- Tracked items do not count against the limit and the items moved back to active once their dependency is met

[16] Eric Brechner. *Agile project management with Kanban*. Pearson Education, 2015.