CSC 171 – Module 01

Why Agile and Lean Approaches Work

The History of Software Development

- Before 1970
 - Software development was chaotic and unplanned
- 1970-1990
 - Software engineering field merging, borrowed things from physical engineering fields
 - Waterfall model become dominant.
- 1990s
 - "the application development crisis"
 - It was estimated that the time between a validated business need and an actual application in production was about three years
 - Within three years, requirements, systems, and businesses were likely to change
 - Thought leaders started looking for timely and more responsive ways to develop software

After 2000

- 2001, Snowbird meeting in Utah
 - Included representatives from Extreme Programming, SCRUM, DSDM, Adaptive Software Development, Crystal, Feature-Driven Development, Pragmatic Programming, etc.
 - Created Manifesto for Agile Software Development
- Agile software development became mainstream
 - · Iterative, incremental, lightweight

Traditional Development Data

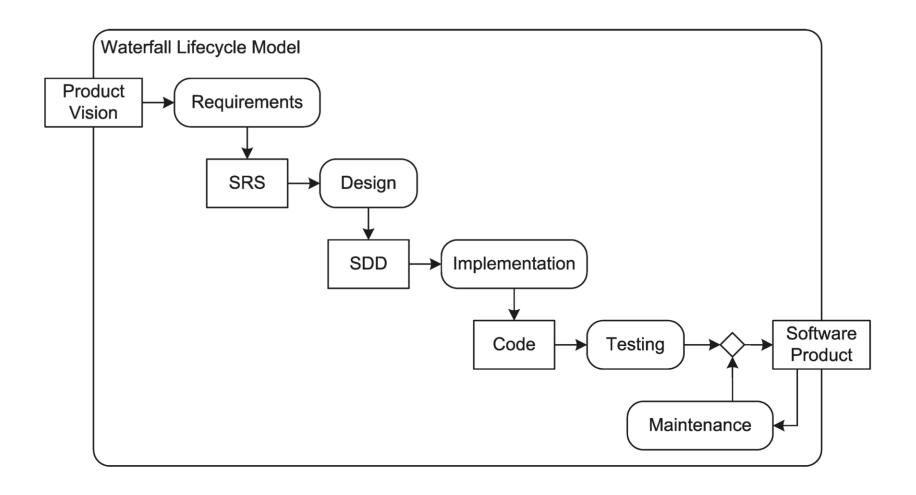
- Standish Group Chaos Report (1995)
 - Successful projects: 16.2%
 - The project is completed on-time and on-budget, with all features and functions as initially specified.
 - Challenged projects: 52.7%
 - The project is completed and operational but overbudget, over the time estimate, and offers fewer features and functions than originally specified.
 - Impaired projects: 31.1%
 - The project is canceled at some point during the development cycle.

Standish Group Chaos Report FY2011–2015

| Size | Method | Successful | Challenged | Failed |
|-------------------------|-----------|------------|------------|--------|
| All Size Projects | Agile | 39% | 52% | 9% |
| | Waterfall | 11% | 60% | 29% |
| | | | | |
| Large Size Projects | Agile | 18% | 59% | 23% |
| | Waterfall | 3% | 55% | 42% |
| Medium Size Projects | Agile | 27% | 62% | 11% |
| | Waterfall | 7% | 68% | 25% |
| Small Size Projects | Agile | 58% | 38% | 4% |
| | Waterfall | 44% | 45% | 11% |

Traditional definition of success: OnTime, OnBudget, and OnTarget Modern definition of success: OnTime, OnBudget, with a satisfactory result

The Waterfall Model



Traditional Development Approaches

- Sequential development
 - Assumes that everything can be predicted
 - Work can be handed off from one person to another and it would all "just work"
 - Only encourages one delivery, at the end
- Realities of software projects
 - We cannot predict everything about the project, we need to adapt quickly to changing realities
 - There are no one-person projects and there is need for interaction and collaboration
 - Software development is an "empirical process", which means assessment and learning is needed as development proceeds
 - Sequential development approaches do not provide enough learning opportunities

Lean Thinking

- Lean (or lean thinking) is the English name—popularized by MIT researchers—to describe the system now known as the Toyota Way inside the company that created it. [2]
- Resource efficiency vs flow efficiency
 - Resource efficiency: watch the runners
 - Flow efficiency: watch the baton
- Value
 - "The moments of action or thought creating the product that the customer is willing to pay for." [2]
- Waste
 - "All other moments or actions that do not add value but consume resources." [2]
- Value ratio = total-value-time / total-lead-time

The Lean Thinking House [2]

Sustainable shortest lead time, best quality and value (to people Goal and society), most customer delight, lowest cost, high morale, safety Respect Continuous Product Development for People Improvement long-term great engineers don't trouble - Go See - mentoring from manageryour 'customer' engineer-teacher kaizen cadence "develop people, spread knowledge cross-functional then build products" - small, relentless team room + visual mgmt retrospectives entrepreneurial chief - 5 Whys no wasteful work engineer/product mgr - eves for waste - set-based concurrent dev * variability, over- teams & individuals create more knowledge Pillar evolve their own burden, NVA ... -Pillar practices and (handoff, WIP, 14 Principles improvements info scatter. long-term, flow, pull, less delay, multi- build partners with variability & overburden, tasking, defects, stable relationships, Stop & Fix, master norms, wishful thinking..) trust, and coaching simple visual mgmt, good in lean thinking tech, leader-teachers from perfection challenge within, develop exceptional people, help partners be work toward flow develop teams (lower batch size, lean, Go See, consensus, reflection & kaizen Q size, cycle time) Management applies and teaches lean thinking, Foundation and bases decisions on this long-term philosophy

Sources of Waste [2]

Variability

varying cycle lengths, varying batch sizes of work packages, varying size of one work package, varying team members or size, varying delivery times, defects (these introduce much variability), interruption to handle hot defects, irregular arrival of requests

Resolution?

- leveling the work
- cadence; for example, timeboxed 2-week cycles
- decompose large work packages into many smaller ones, so that a more consistent amount of work is taken each cycle

Overburden

- overtime for arbitrary deadlines
- one Product Manager having to know hundreds of features in detail
- often seen with specialist bottlenecks and over-dependence on superspecialists

Resolution?

- develop "eyes to see" queues & bottlenecks and those who are doing too much
- take on less work in cycle; descope
- spread the work and skill—cross-train

3 Sources of Waste

NVA actions

- for example, handoff, waiting, scattered information, partially done work, task switching

Resolution?

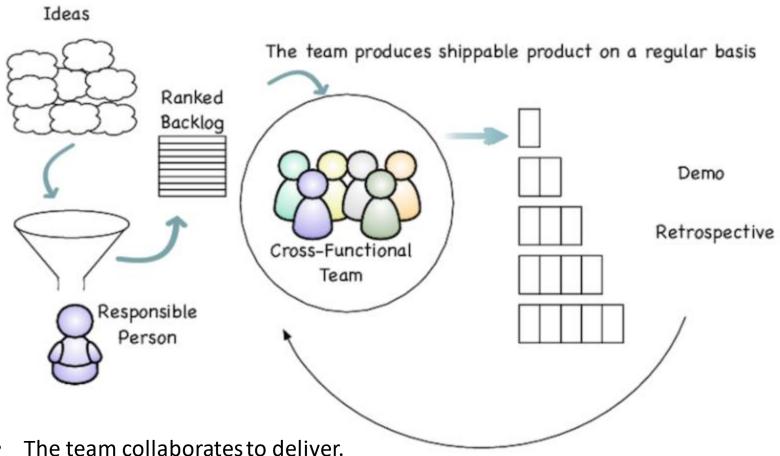
- kaizen events, to learn to see it and experiments to reduce

Agile Software Development

Manifesto

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan
- Principles
- Agile software development methods
 - Scrum, Kanban, Scrumban, lean software development, extreme programming (XP), feature-driven development (FDD), dynamic systems development method (DSDM), ...
- Agile software development practices
 - Backlogs, continuous integration, cross-functional team, daily stand-up, iterative and incremental development, pair programming, planning poker, refactoring, retrospective, test-driven development (TDD), timeboxing, user story, ...

Agile Development Approaches



- The team limits its wearly in manager (
- The team limits its work in progress (WIP).
- The team delivers releasable product often.
- The team reviews its work product and process on a regular basis.

Scrum

Sprint

- Time-boxed iteration, usually 2 to 4 weeks
- Each sprint includes planning, analysis, design, coding, testing, and getting feedback

Artifacts

- Product backlog
 - Ordered list of ideas for the product, typically begins short and vague, and becomes longer and more defined as time goes on
- Sprint backlog
 - The list of product backlog items chosen for development in the current sprint and plans for accomplishing the work
- Product increment
 - Be of high enough quality to be given to users, be acceptable to the product owner, meets the team's definition of done

The scrum team

- Product owner
 - Collaborates with stakeholders and team to clarify what to build
- Development team
 - Delivers product increment, self-organizing, cross-functional, normally 3 to 9 people
- Scrum master
 - Coaches the scrum team and organization, helps team to identify and remove impediments, ...

Activities

Product backlog refinement (grooming), sprint planning, daily scrum, sprint review, sprint retrospective

Scrum Activities

| Activity | Who | What | When |
|---------------------------------------|--|--|---------------------------------|
| Sprint planning | Whole scrum team | Select and understand the work to be done in the next sprint | At the beginning of each sprint |
| Daily scrum | Whole scrum team | Ensure team is on track for that sprint | Every day |
| Sprint review | Whole scrum team and interested stakeholders | Review the resulting product increment | End of each sprint |
| Sprint retrospective | Whole scrum team | Review and improve the process | End of each sprint |
| Product backlog refinement (grooming) | Whole scrum team | Prioritize and estimate product backlog items, split large items into smaller ones, merge smaller items, | Ongoing 13 |

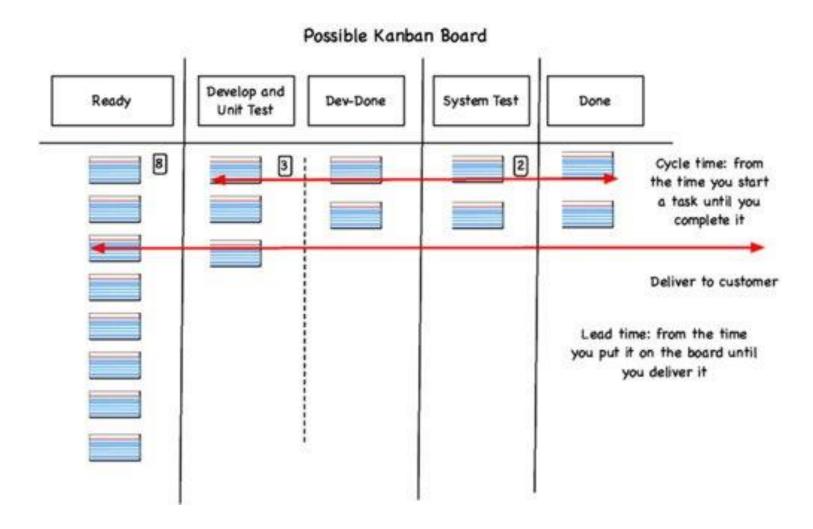
Kanban

- Is focused on process improvement, and is based on Lean values and Lean thinking
- Core practices
 - Visualize work
 - Limit work in progress (WIP)
 - Make policies explicit
 - Manage flow
 - Implement feedback loops
 - Improve collaboratively, evolve experimentally

Kanban board

- Kanban card, a visual representation of a work item, can include name, description, due date, assignee, tasks, ...
- Kanban column, represent a workflow stage, workflow can be mapped to meet the unique process of any team
- WIP limit, the maximum amount of cards (work items) can be put in each column (workflow stage)
- An example

Kanban – Cont.



Iteration-Based Agile Approach

- A team works in timeboxes of the same size for every iteration
- Each timebox
 - Is the same size
 - Results in running tested features
 - Includes these activities: requirements, analysis, design, build, test, release, and deploy.
- The product owner and the team manage the work in progress by estimating the number of stories (and other work) the team can commit to in a timebox
- Key metrics
 - Velocity: the amount of work a team can complete in an iteration
- Example
 - Scrum

| Requirements Analysis Design Build Test Release Deploy | Requirements Analysis Design Build Test Release Deploy | Requirements Analysis Design Build Test Release Deploy | Requirements Analysis Design Build Test Release Deploy | Repeat as needed | Requirements Analysis Design Build Test Release Deploy | Requirements Analysis Design Build Test Release Deploy |
|--|--|--|--|-------------------------|--|--|
|--|--|--|--|-------------------------|--|--|

Flow-Based Agile Approach

- Focuses on the continual pulling of work, there is no timebox
- The team limits the number of features active at any time with WIP limits for each activity
- Key metrics
 - Cycle time: how long each feature take on average
- After finishing some work, the team delivers and learns, retrospectives, and reviews what it wants to improve
- Example
 - Kanban

| Feature: Feature Clarify Clarify Req't, Requiren Analysis Design Build Build Test Release Deploy Deplo | Clarify Requirement, Analysis Design Build Test Release | Repeat as needed | Feature: Clarify Requirement, Analysis Design Build Test Release Deploy | Feature: Clarify Requirement, Analysis Design Build Test Release Deploy |
|--|---|-------------------------|--|---|
|--|---|-------------------------|--|---|

References

- [1] Create Your Successful Agile Project, Johanna Rothman, Pragmatic Programmers LLC, 2017. ISBN:9781680502602
- [2] Lean
 Primer, https://www.leanprimer.com/downloa ds/lean_primer.pdf