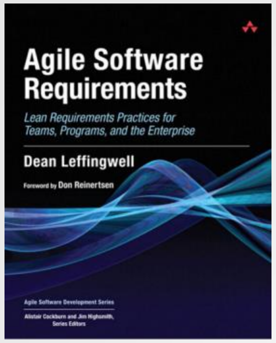
Agile Software Requirements
Lean Requirements Practices for Teams, Programs, and the Enterprise
Dean Leffingwell
Foreword by Don Reinertsen

Agile Software Requirements

Software Requirements Engineering – 40688

Computer Engineering department

Sharif university of technology

Fall 402

Chapter 9:

Iterating, Backlog, Throughput, and Kanban

Iterations

- Planning
- Daily
- Review
- Retrospectives

Little's Law

It states the following:

$$\omega_q = \frac{L_q}{\lambda}$$

- ***Wq*** is the average waiting time in the queue for a standard job.
- ***Lq*** is the average number of things in the queue to be processed.
- ***Lambda*** is the average processing rate for jobs in the queue.

$$\omega_q = \frac{6\ (Average\ Queue\ Length)}{1.33\ (\#\ Customers\ Processed\ per\ Minutes)}$$

An example

- A single agile/Scrum team, working in two-week iterations.
- The team averages about 25 to 30 story points per iteration, or a story completion rate of about 8 stories per iteration.
- The team is justifiably proud of how well they are maintaining their backlog, and the backlog averages about 100 stories, most of which are committed to near-term releases.

$$w_q = \frac{100\ Stories}{\frac{8\ Stories}{1\ Iteration}} = 12.5\ Iterations$$

Increase Agility and Decrease Time to Market

- If we are going to improve (decrease) time to market:
  - we have to either increase the denominator
  - or decrease the numerator (or both)

$$\omega_q = \frac{L_q}{\lambda}$$

Increasing Lambda(1)

- Of course, if we could simply add resources, we could probably increase Lambda.
- Let's assume that is impractical.
- It increases the cost of the value created and decreases return on investment (ROI).
- The primary mechanism for increasing the rate of story completion is the team's inspect and adapt process, whereby the teams review the results of each iteration and pick one or two things they can do to improve the velocity of the next.

Increasing Lambda(2)

- This is the long-term mission.
- It is a journey measured in small steps, and there is no easy mathematical substitute for such improvements.
- These improvements include better coding practices, unit testing and unit testing coverage, functional test automation, continuous integration, and other enhanced agile project management and software engineering practices.

Increasing Lambda(3)

In my experience, however, two primary areas stand out as the place where teams can get the *fastest* increase in Lambda:

1) First, **gaining a better understanding of the story itself before coding begins,** and

2) Second, **decreasing the size of the user stories contained in the backlog.**

Better Understanding of the Story

- Acceptance test driven development.
- Once a story has reached a priority whereby it will be implemented in the next iteration or two.
- Time spent in elaborating the story will pay dividends.
- Acceptance Test-Driven Development (ATDD)
- ATDD involves two things:
  1) Writing better stories
  2) And establishing the acceptance tests for the story before coding begins.

Increasing Lambda with Smaller Stories

- **Linear effect:**
  - Smaller user stories are just that, smaller.
  - They go through the iteration faster so teams can implement and test more small user stories in an iteration than large ones.
  - Although the total value of a small story can't be as big as a large story, the incremental delivery hastens the feedback loop, improving quality and fitness for use.
- **Exponential effect:**
  - Because they are smaller and less complex, small user stories decrease the coding and testing implementation effort.
  - The coded functions are smaller and less complex, and the number of new paths that must be tested also decreases exponentially with story size.
  - However, some of this is offset by the additional overhead of managing more stories.

Decreasing Lq
The Length of the Queue (1)

Decreasing queue size causes a directly proportional decrease in the wait time.

Reinertsen points out that there are a number of additional reasons why long queues are fundamentally bad:

- **Increased risk**

  Market or customer has changed their mind and the story is no longer valuable.

- **Increased variability**

  High variability decreases reliability, causes stress in the organization, and, perversely, drives even higher utilization.

Decreasing Lq
The Length of the Queue (2)

- **Increased costs**

  Team has to continue to account for it, prioritize it, and rearrange it.

- **Reduced quality**

  The longer the queue, the longer it is before we get feedback on new items from the customer.

- **Reduced motivation and initiative**

  If it's going to be a long time before a customer sees a story in the middle of the queue, there is little sense of urgency.

Software Kanban Systems

- Many teams have decided to place work-in-process limits on the size of the backlog.
- Which are sized and adjusted as necessary to create the desired time to market, response time, or internal feedback loop.
- Once the queue is full, they quit even thinking about new stories until there is room for more stories in the queue.

Kanban notes

- Kanban manages the flow of units of value through the use of work-in-process (WIP) limits.
- Kanban manages these units of value through the whole system, from when they enter until they leave.
- By limiting WIP, Kanban creates a sustainable pipeline of value flow.
- Limiting WIP provides a mechanism to demonstrate when there is capacity for new work to be added, thereby creating a pull system.
- Finally, the WIP limits can be adjusted and their effect measured as the Kanban system is continuously improved.

Kanban System Properties (1)

- **Visualize workflow:**

  Highlights the mechanisms, interactions, handoffs, queues, buffers, waiting, and delays that are involved in the production of a piece of valuable software.

- **Limit work in progress:**

  Implies the introduction of a pull system from a family of possible solutions.

- **Measure and manage flow:**

  Highlights a focus on keeping work moving and using the need for flow as the driver for improvement. A focus on flow rather than on waste removal is a higher mastery of lean and much less likely to lead to “Lean and Mean” anti patterns and dysfunction.

Kanban System Properties (2)

- **Make process policies explicit:**

  It's about holding up a mirror to the working reality and encouraging the whole team and its leadership to reflect on its effectiveness. Thinking of a process as a set of policies rather than a workflow is a very powerful technique.

- **Use models to recognize improvement opportunities:**

  Kanban is quantitative and takes a scientific approach to improvements. Focus on the theory of constraints, an understanding of variation and the system of profound knowledge, and the lean odes of waste and flow.

Classes of Service in Kanban

- **Expedite:** Unacceptable cost of delay
- **Fixed delivery date:** Step function cost of delay
- **Standard class:** Linear cost of delay
- **Intangible**: Intangible cost of delay