Agile Hardware Development with Scrum

A Keynote Presentation to the Scrum4HW Gathering

Kevin Thompson, PhD

August 26, 2016

www.kevinthompsonphd.com

Kevin Thompson, Ph.D.

- Agile consultant and trainer
- Specializes in Agile development of hardware and integrated hardware / software products
- Develops Agile content, training, consulting standards
- Holds a Ph.D. in Physics from Princeton University
- Has background in classic (Project Management Institute) and Agile processes

Where It's Happening*

ThermoFisher SCIENTIFIC

Chromatography and Mass Spectrometry devices for laboratory and medical use

PICARRO

Isotope and Trace-Gas Analyzers for field use

Bird®

Radio-frequency telecom products and test equipment

GoPro Be a HERO.

Rugged video cameras for field and recreational use

plantronics.

Audio and communications equipment (headsets, speakerphones, etc.)

* My clients

What Is "Agile Hardware?"

Possibilities:

1. The construction of physical devices in an Agile way
   - Rapid prototyping, custom manufacturing

2. Techniques to shorten turnaround time for components
   - 3D printing, rapid circuit-board prototyping

3. An Agile process for developing devices to be manufactured
   - Scrum

Our focus is on Scrum

Hardware Variations

Hardware: Mass-produced electronic & electro-mechanical devices

- Pure Hardware Products
  - Example: Vacuum cleaner
- Hardware with Embedded Software
  - Hardware and software. Embedded software may evolve within the bounds of a static hardware environment
  - Example: Cell phone, network equipment, analytical equipment for chemistry and biotech
- Hardware with Associated Software
  - Hardware and software. Associated software may evolve over time, and may support multiple devices
  - Example: Laboratory equipment plus locally-hosted or Web-based software to control multiple devices

This is not Software Development!

- "Scrum" is a familiar and comfortable term. Don't let it fool you.
- Do not be misled into thinking that what you know about developing software with Scrum applies when developing hardware with Scrum
- Some aspects of hardware development with Scrum have no parallels in software development with Scrum
- Some aspects of hardware development with Scrum contradict or violate norms of software development with Scrum
- Be prepared to leave behind some cherished notions, or…
- Be prepared to fail

A Critical Distinction

- Software: Aggregation of deliverables yields usable features over time
  - Product has more working functionality at the end of the month than it did at the beginning of the month
  - Steady evolution of functionality enables frequent stakeholder review opportunities, to drive evolution of scope in ways that maximize value and minimize wasted effort
- Hardware: Aggregation of deliverables yields design for usable product at the end of development
  - Product's *design* has more components or design elements as time passes, but product does not work in any meaningful sense until all parts are completed and assembled
  - Stakeholder feedback loops are desirable, but more oriented towards concepts and mockups than working functionality

How to do Scrum for Hardware

- Retain all standard Scrum Ceremonies, Roles, Artifacts
  - Backlog Grooming, Sprint Planning, Daily Stand-Up, Review, Retrospective
  - Scrum Master, Product Owner, Team members
  - Stories for specifications, Burn-Up charts for tracking
- Do the following with a Hardware perspective
  - Organize people into Scrum Teams
  - Choose Sprint Length
  - Write Specifications
  - Plan Sprint
  - Track work

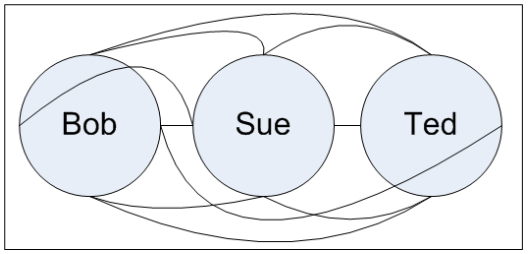
The (Big) Need for Teams

- Hardware development groups, on average, are substantially larger than software development groups
- Multiple Teams (potentially a large number) are more likely to be the norm than the exception
- Team Definitions are critical
- Agile Program Management is required

Team Definitions: Driven by Coupling

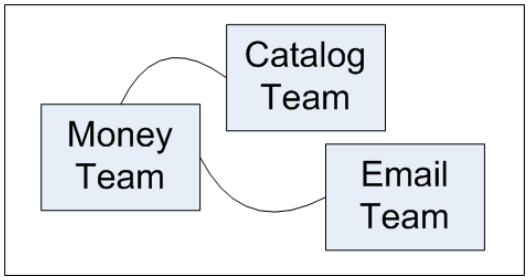
Coupling (noun): "A factor or relationship that connects one thing to another"

- Reflects frequency, number of dependencies
- Requires collaboration and synchronization of effort



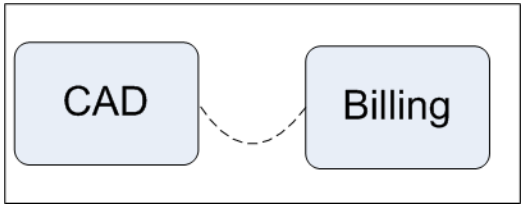
Work items of Team members are tightly coupled

➢ Collaboration happens constantly



Teams on a Project are moderately coupled

➢ Collaboration happens daily



Projects in a Program are loosely coupled

➢ Collaboration happens as needed

Interfaces make Collaboration more Difficult

- Collaboration within a Team is easy
  - The Team has its own plan and priorities
  - Members of the Team collaborate to execute their plan
  - Collaboration within the Team is easy and informal
  - Adjusting to changes in scope or schedule is relatively simple, and usually requires no more than a short discussion

- Collaboration between Teams is harder
  - Collaboration requires planning to ensure dependencies are addressed
  - Changes in scope or schedule that impact cross-Team dependencies require re-planning the schedules of both Teams

Cost of interaction increases with the number of interfaces involved!

Minimize Collaboration across Interfaces

Define Teams to minimize interfaces

Strategies

- Feature Teams: Divide one product into functional areas. Each Team owns one functional area, including User Interface, application logic, persistence, testing, etc.
- Client-Server Teams
- Component Teams

Feature Teams

- A Team should include all areas of expertise
- A Team often spans departmental boundaries (matrix organization)

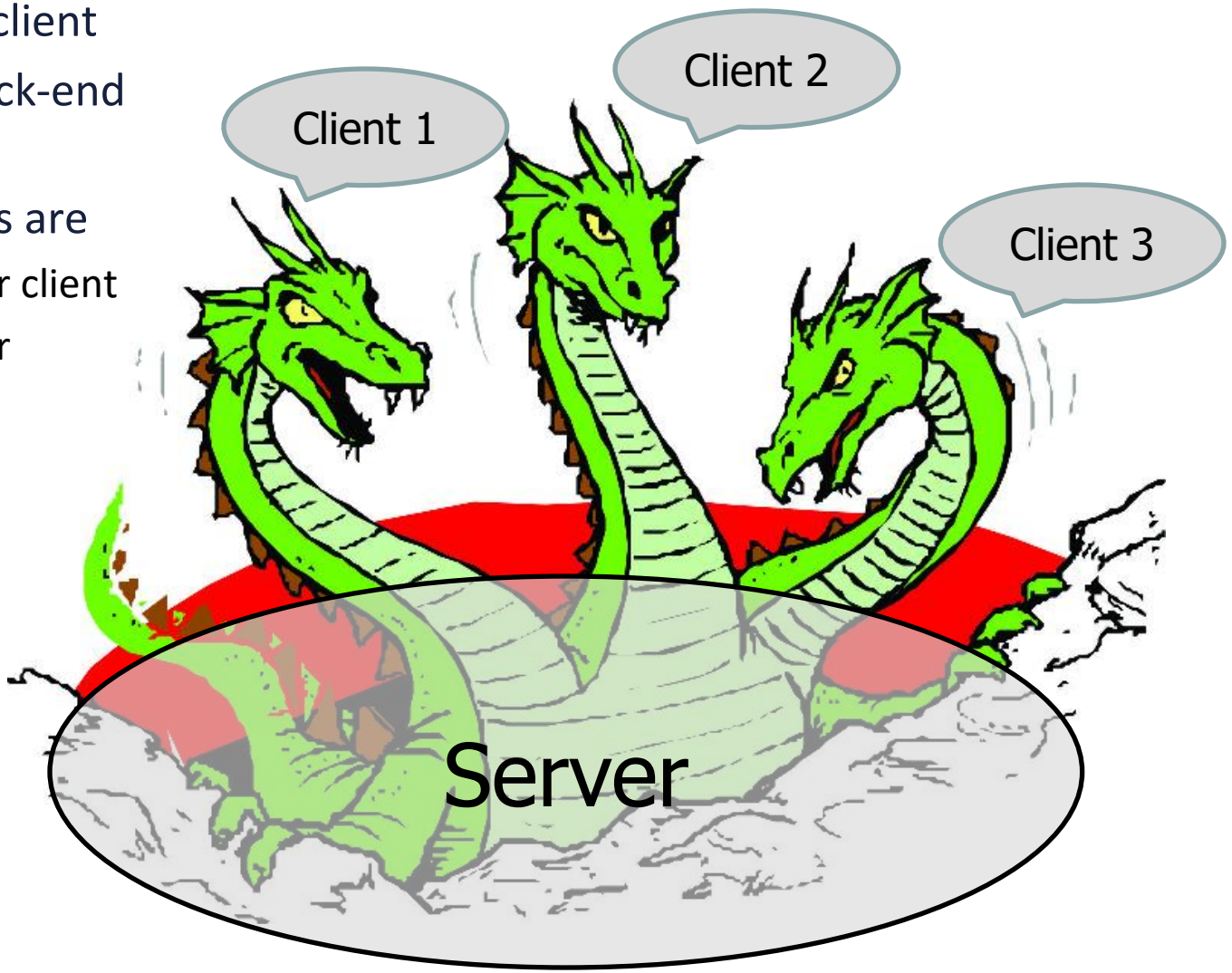
Department / Expertise	Billing	Catalog	Subscription
Engineering	Bob, Carson	Frank, Sanjay	Sandy, Jack
Quality Assurance	Sue, Carlos	Charles, Martin	Bernice, Henry
Database Design	Ted	Ted	Giles
User Experience	Mark	Julio	Mark
Technical Publications			Yvette

Bad

Good

Multiple Client and Server Teams

- More than one client
- One or more back-end servers
- Team definitions are
  - One Team per client
  - One Team per Server



Feature Teams or Multiple Client / Server Teams?

Key Points and Tradeoffs

Feature Teams

- Need at least one each Android, iOS, Web developer for each Team
- Workload may vary per client
- Teams share Back End server, must coordinate work to avoid conflicts

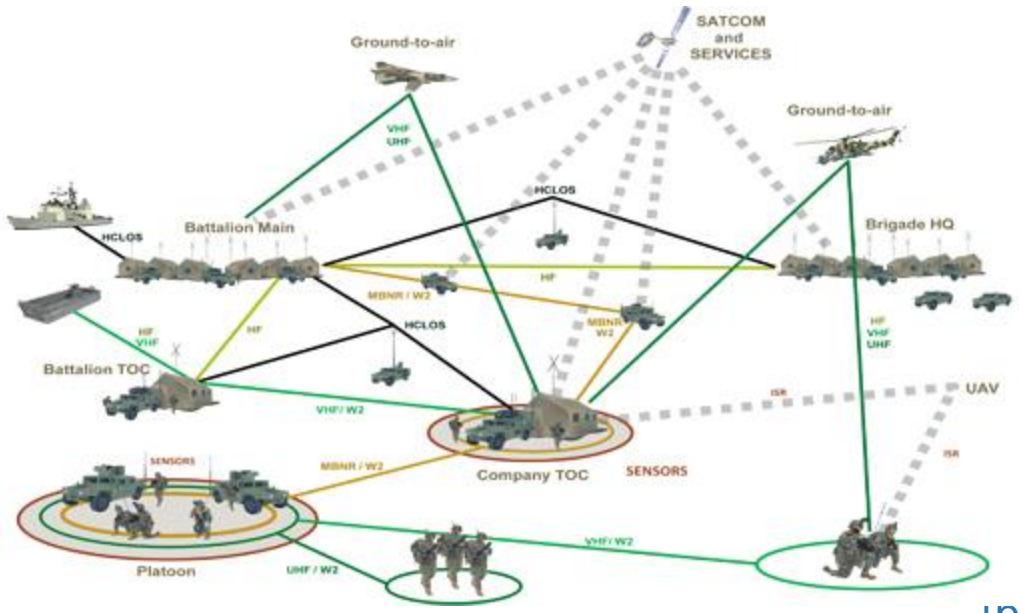
Multiple Client / Server Teams

- May need fewer client-specific experts
- Client Teams negotiate interfaces, priorities with Back End Team
- Less likelihood of one client's back-end work impacting another client by accident

The cost of resources and cost of collaboration across Teams is often lower for Multiple Client / Server Teams than for Feature Teams in this scenario

Component Teams

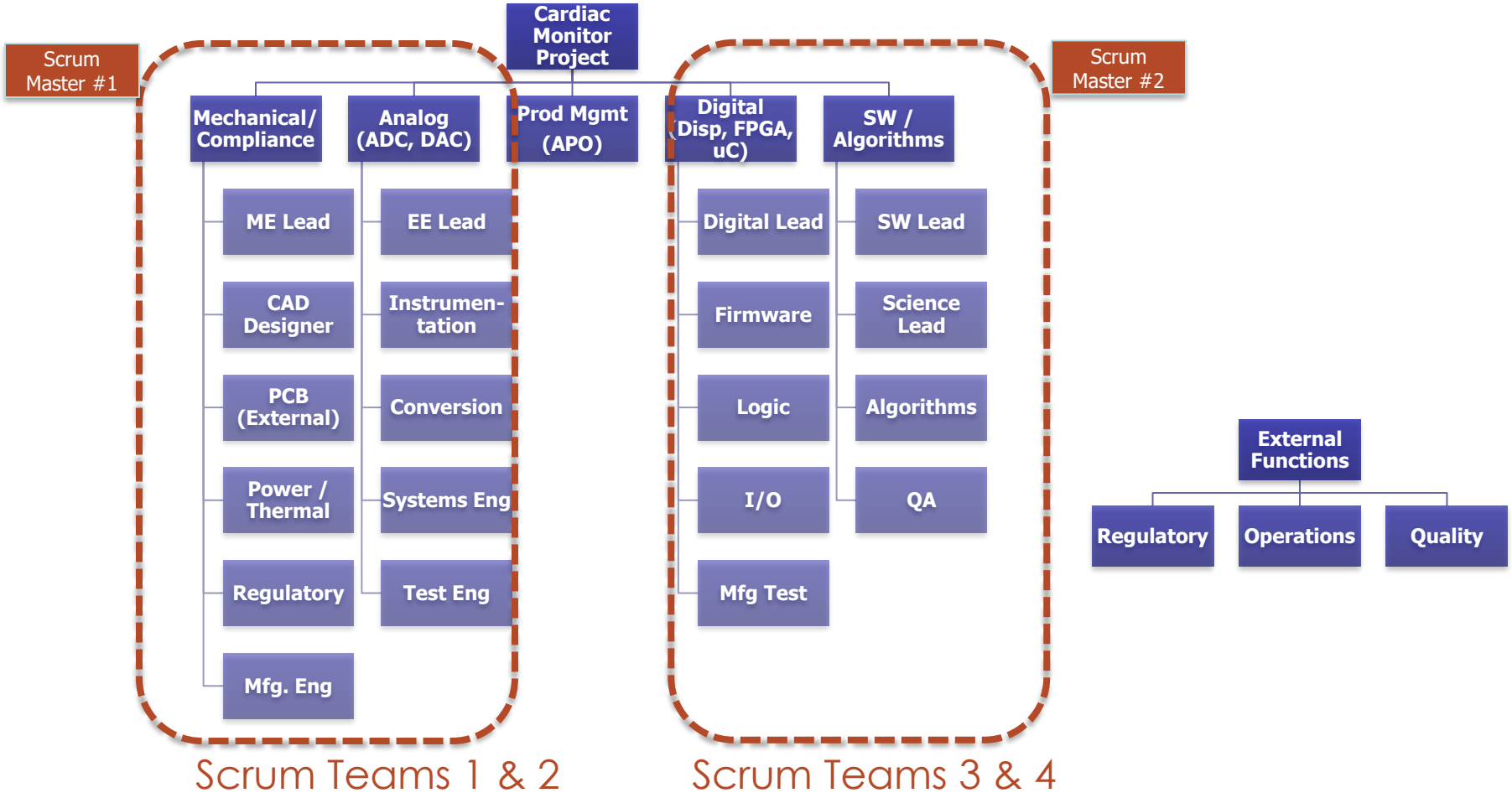
- Often for mixed hardware-software environments
- A “Solution” may not be equivalent to a product, but require that multiple products work together
- Products involve a number of specialized components, each of which involves a set of highly specialized skills, whose interface change less rapidly than the internals of the component
- Many components do not have human users



Final Thoughts on Team Definition

- Perfect solutions usually do not exist
- Examine multiple alternatives
- Pick the solution that has the fewest problems
  - If your solution works 80% of the time, consider yourself fortunate
- Component Teams are more likely to be appropriate in hardware development than is the case in software development
  - Design good modularity, interfaces

Example Scrum Organization for Cardiac-Monitor Development



Choose Sprint Length

- A two-week length has been working
- Three weeks is acceptable
- Other lengths have the same drawbacks as for software development

- Critical point: Hardware shops may not believe two-week Sprints are appropriate when several weeks might be required to build a component that works, but…
  - Work can still be decomposed into smaller deliverables (Stories) that can be built, tested, and validated in two weeks

Requirements Artifacts: Product Backlog Items

User Story

- Short narrative description of functionality, from user perspective
- Usually written by Product Owner

Technical Story

- Non-user facing (technical / infrastructure) requirements
- Often driven by technical needs of User Stories
- Usually written by Team members

Defect

- Bug report
- Usually written by support and development personnel

Stories are Mostly Technical

In Software, we implement a small increment of user-facing experience by writing code at many layers. The “User Stories” that define behavior incorporate technical work behind the scenes.

In Hardware, many user-facing behaviors are enabled by a fixed set of components with a range of capabilities. One cannot add behaviors incrementally.

- The User Interface then *becomes a component* whose details are driven by User Stories
- Most Stories are not about user interactions, but about component definitions. Thus relatively few User Stories, and lots of Technical Stories.

Product Owners Write Few Stories

In Software, Product Owners write many User Stories. Team members write Technical Stories, as needed. Bulk of writing effort tends to be on the Product Owner.

In Hardware, there are relatively few User Stories, and lots of Technical Stories. The Product Owner's focus is less on writing User Stories, and more on shepherding the Team members' writing of Technical Stories.

Sample Technical Story

Title	16-bit Digital-to-Analog Converter Motherboard Integration
Narrative	
Design into the motherboard a Digital-to-Analog converter (DAC), in order to provide the high-level analog voltage required to drive the analog display. Assume that the digital drive presented at the input of the DAC module is of sufficient amplitude to activate the unit properly, but not so large as to damage the input gates. Since no DAC has been chosen for this purpose yet, select an appropriate one for incorporation into the motherboard design.	
Acceptance Criteria	
• 16-bit output with +/- 0.5 bit nonlinearity • Single supply operation: 2.7 to 5.5V • The maximum current to drive the unit is 500mA. • The maximum clock rate is 3.4 MBit/sec. • The output will be a time-varying analog signal varying from 0V to 5V at steps of 7.5 µV, with a maximum output impedance of 1 Ohm. • Device meets the other requirements of the I2S serial bus	

Sprint Planning

Assume we have a list of ranked Stories that have been reviewed, and are understood, by all Team members

Basic Steps

1. Forecast Velocity
2. Estimate Top Stories
3. Select initial Sprint Backlog of Stories that satisfies the Velocity limit and other constraints
4. Create Task breakdowns for Stories, with hour estimates
5. Finalize Sprint Backlog based on Task estimates and sanity check

Farewell to Story Points?

- Relative sizing (Story Points) assumes that Team norms for Stories are possible
  - Requires enough skill overlap for Team members to understand areas outside their specialty to a useful extent
  - Skill overlap is common in software development, but not in hardware development
  - Multiple hardware clients state that they object to Story Points
- Time-Based units (Person-Days) do not require definition of Team norms
  - Do not require Team members to understand areas outside their specialty for estimation purposes
  - Can be summed across Teams for product-level Burn-Up charts

Specialization drives time-based estimation

Sprint Planning Meeting

Preparation

- Backlog Grooming meetings to prepare Stories, Team
- Scrum Master's Velocity forecast for the Sprint (Person-Days)

Sprint Planning Meeting

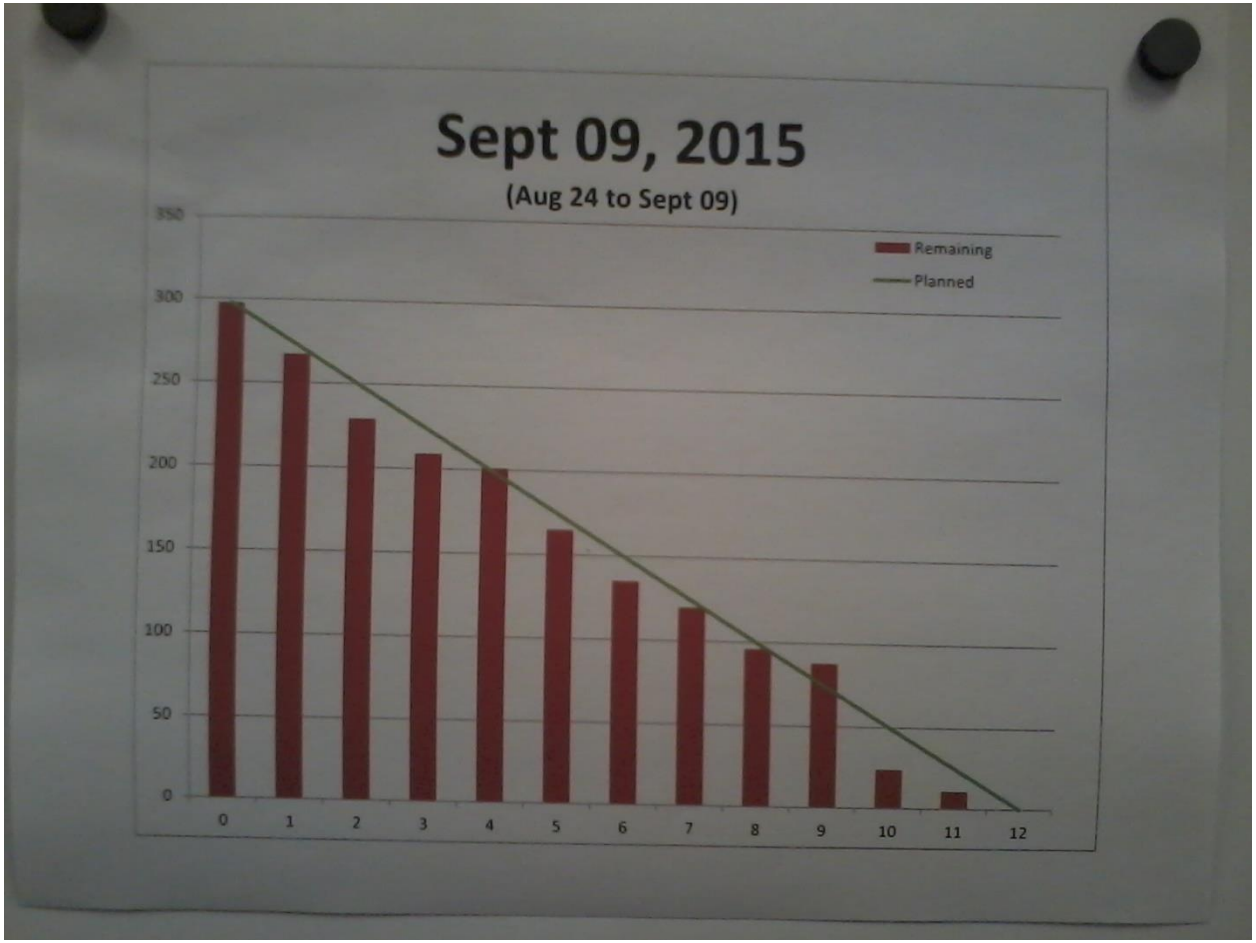
- Part 1: Estimate top-priority Stories (e.g., Planning Poker) and identify highest-value set that fits within Velocity ceiling
- Part 2: Team defines list of Tasks per Story, estimates Tasks in hours
- Final decision: Review for feasibility, adjust as needed

Impact of Specialization on Sprint Planning

In Software, we rank (sequence) Stories in a Sprint Backlog to ensure highest-value Stories are on top, and are done before lower-value Stories. This approach, combined with Swarming, ensures maximum value is completed in Sprint even when things go wrong.

In Hardware, ranking is useful for making Sprint scope decisions, but specialization of skills means that many Stories can be done by only one person, and a linear time sequence is not possible. Instead, expect several Stories in process at the same time, and selection of Stories in Sprint to be substantially constrained by availability of skills.

Tracking



Burndown Chart for First Sprint at Thermo Fisher Scientific

Conclusion

Dr. Michael Bedford, Senior Scientist and Product Owner had this to say at the five-month point of Thermo Fisher's adoption of Scrum:

"First and foremost, running this project with Scrum is an ongoing success. The team has formed nicely and we have proper buy-in from each team member. I make technical decisions with more input from the team than any other project that I have worked on. Grooming is a hassle because it takes so much time and requires us to stop doing and think, but the drawbacks I mention are also the clear benefits. We take time to plan. And we plan together….

Conclusion

"…The daily standup is a life saver. As is the board. I can quickly understand where we are and what needs to be done. Release planning has been beneficial because it really focused the team onto what our big goals are as opposed to smaller engineering goals. I feel that management here in San Jose is excited about Scrum. I have had many of the bosses and decision makers stop by and ask about the process. I am curious to see if another team pops up in the near future."

My Papers on Agile Hardware

Hardware

Case Study, “Eleven Lessons Learned about Agile Hardware Development,” by Kevin Thompson, Ph.D.

White paper, “Agile Processes for Hardware Development,” by Kevin Thompson, Ph.D.

“Scrum for Hardware? A Groundbreaking Experiment at Thermo Fisher Scientific Shows Promise.” AgileVox magazine, 1st edition, Spring 2016.

Agile Program Management and Beyond

“Recipes for Agile Governance in the Enterprise,” by Kevin Thompson

My Classes

Agile Hardware Development with Scrum

Learn the basics of Scrum for hardware at the Team level

Agile Program Management

Learn how to conduct large-scale product development with multiple collaborating Teams