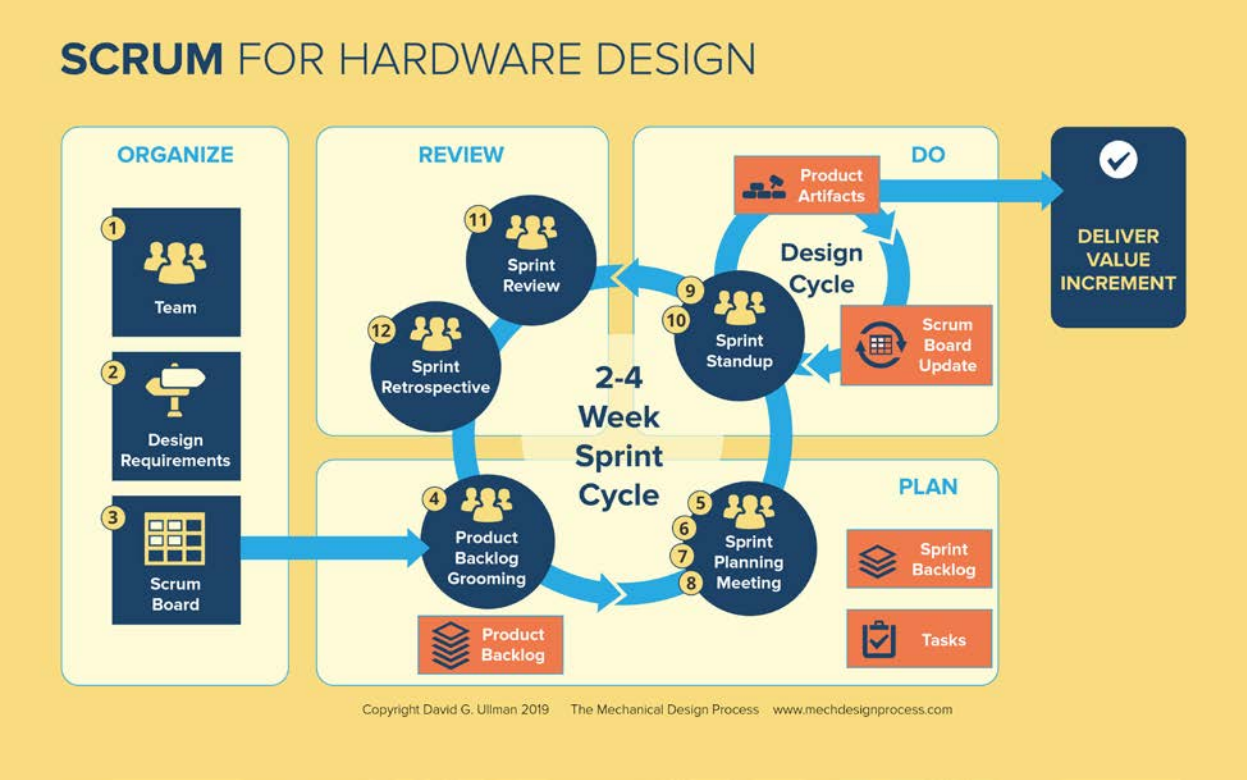
SCRUM FOR HARDWARE AND SYSTEMS DESIGN

- Instructor's notes –

Volume 2.0

David G. Ullman
March 2020
The Mechanical Design Process, 6th edition website: www.mechdesignprocess.com
Scrum website: www.mechdesignprocess.com/scrum
Personal website: www.davidullman.com
Email: ullman@davidullman.com



Preface

These notes support the online video course *Scrum for Hardware and Systems Design.* This course stands alone or can be used as supplemental material for the text *The Mechanical Design Process* 6th edition (hereafter referred to MDP6). This set of notes is intended to give background and approaches for teaching the scrum framework to hardware students (i.e., Mechanical Engineers, Systems Engineers, and Electronics Engineers).

This material is relatively unique. It is worth noting that as recently as 2016, Jeffery May of the Computer Information & Business Analytics Department at James Madison University noted:[i] "*Unfortunately, current System Analysis and Design textbooks provide cursory attention to Scrum*." This is certainly true in hardware texts. The material in this course and the material here are an effort to resolve this lack.

Where Scrum is a design process framework, it is discipline agnostic (to a great degree). However, some discipline-specific adaptations are included here. Also, Scrum for hardware design relies on many other best practices. *MDP6* is a compendium of best practices, and thus Scrum's success relies on methods already in the 6th edition.

Two case studies are available to support the material:

- *A Student Team Designs a Prosthetic Arm Using Scrum Methods*
- *Agile Design of an Agile Fighter at Saab Aerospace.*

They, along with other supplemental material at www.mechdesignprocess.com/Scrum.

Table of Contents

Syllabus for the online course: Scrum for Hardware and Systems Design

The course is composed of 30 Modules arranged into 7 Sections. The material in these instructor's notes will reference the Modules as needed.

1. Intro to This Course
    1. Introductions
    2. Course Content Overview
    3. Resources Available to You
    4. How to Best Experience This Course
2. Why Scrum for Hardware and Systems
    5. A Tale of Four Airplanes
    6. Why There Is Need for a Design Process Change
    7. Thirteen Reasons Why Designing Hardware and Systems Is Different from Designing Software
    8. The Agile Principles for Hardware and Systems Design: The Team and the Product
    9. The Agile Principles for Hardware and Systems Design: The Process
3. The Scrum Process
    10. The Two Cycles
    11. The Twelve Steps of Scrum
    12. The Scrum 3-5-3 Rule
4. Organizing for Scrum
    13. The Three Scrum Team Roles
    14. Product Requirements
    15. The Three Key Conversations
    16. Planed Hardware and Systems Requirements
    17. Agile Requirements
    18. Balancing Planned and Agile Requirements
    19. The Scrum Board
5. Scrum Planning
    20. Scrum Task Management: Two-Level Planning
    21. Product Backlog Grooming
    22. Manage Tasks: Sprint Grooming
    23. Estimate Task Time
    24. Choose Sprint Tasks
6. Doing the Sprint
    25. Do-The Design Cycle
    26. Reviewing the Sprint
7. Scrum in Your Organization
    27. Recap of The Process and The Important Points
    28. Mixing Scrum and Waterfall
    29. Scrum Pitfalls
    30. Make Scrum Happen in Your Organization

Why Teach Scrum Methods to Non-Software Engineers

All Agile methods are an effort to enhance the efficiency of the design process. Lean Design, often considered an Agile method, focuses on resource efficiency to maximize value. Many lean concepts are already covered in *MDP6*. Extreme Manufacturing (EM), an evolving agile methodology, is not yet very refined for hardware systems, yet many EM concepts are also already included in *MDP6*. The Scrum framework adds a new approach to hardware design as will be developed.

MDP6 and other engineering process books are built around the waterfall method of serially progressing from need, to concept, to product, to manufacturing. This is very important at the macro and planning level, but Scrum is important for doing work down in the trenches when information is uncertain and evolving.

Scrum was developed by and for software engineers. So, why teach it to hardware students? There many reasons:

1. Large, successful companies and organizations are using scrum methods for hardware design. These include Tesla, John Deere, Saab Aerospace, Raytheon, Oak Ridge National Labs, Bosch, Plantronics, SpaceX, and many others. This is recent.
2. Scrum methods are not taught at very many universities in mechanical, electrical, and systems engineering. Giving your students an agile experience will give them an advantage in the job market.
3. Students struggle with the traditional waterfall (aka stage-gate) as it forces an ability to predict the future, which is heavily tainted by uncertainty. This is not to say that waterfall should not be taught, but that a mix of methods is most robust.
4. Scrum forces teamwork, communication, transparency, and collaboration. All skills that are important to learn at the college level.
5. Scrum forces self-organizing teams with responsibility for decisions in a manner that ensures retrospection and improvement.
6. Scrum teaches students an approach to managing uncertainty. A typical engineering curriculum is rife with certainty; reality is not. Most problems that students solve during their education have one correct answer, which is simply not the case after graduation.
7. Scrum better allows for exploring the design space to find a good solution. In the words of Thomas Edison, "I have not failed. I've just found 10,000 ways that do not work." However, Scrum can also be seen as a license to develop a single concept. If Edison did this, we might never have had the light bulb or phonograph.
8. Scrum helps students decompose problems so that they can deliver value frequently.
9. Scrum helps students learn time estimation.
10. Scrum sprints provide a cadence for monitoring performance.

In the material below, I will more fully develop these points.

Do note that Scrum is more than a project management method; it is a state of mind. What is good about the scrum methodology is that helps students internalize agile principles and provide a framework for solving challenging design problems. What makes Scrum promising for university settings is that it relies on empowered, self-organizing teams to discover, implement, and evolve the best process that works for them to accomplish a shared goal.

Learning Objectives

The list below itemizes the learning objectives that can be realized by teaching Scrum. These are written in a format that can be used for ABET documentation. Tips on what to teach and how to teach it are in the following sections titled "Detailed Steps for Teaching Scrum".

1. **Team Organization**: Organize work in a team and manage the team tasks.
2. **Design Requirements Development**: Generate incremental design requirements in terms of user stories and engineering specifications. Show they are equivalent.
3. **Product Backlog Management**: Rank order stories to manage the product backlog.
4. **Task Identification, Measures, Targets, and Tests**: Identify the tasks needed to be done complete with measures, targets, and tests that define when each is done (i.e., test-driven development).
5. **Task Time Estimation**: Estimate the time needed to complete a task.
6. **Sprint Focus**: Use sprint planning to choose tasks to be completed during the Sprint and maintain a sprint backlog.
7. **Uncertainty Management**: Demonstrate the ability to identify and manage uncertainty.
8. **Modular Design Around Known Stable Interfaces**: Demonstrate the ability to design modules connected by known stable interfaces.
9. **Daily Meetings**: Hold sprint standup meetings and ensure colleagues are aware of progress, plans, and problems.
10. **Sprint Progress Tracking**: Create a scrum board (either physical or in software) and post stories, engineering requirements, and tasks on it. Track progress on a burndown chart.
11. **Sprint Review**: At the end of a sprint, have a sprint review (aka design review) where the progress on the product is demonstrated.
12. **Sprint Retrospective**: At the end of a sprint, have a sprint retrospective where the design process is reviewed, and improvements developed.

Teaching Experiences

Scrum must be taught in a project setting. The section, "Detailed Steps for Teaching Scrum" assumes that the students are designing a product concurrently to being introduced to the concepts.

There is very little experience in teaching Scrum to hardware students. The only classes I know of are at Olin and Bucknell. While there are over 40 papers on teaching Scrum

for software[ii], there are none (that I could find) focused on hardware. Hardware design differs from software design in many ways (see the video https://www.youtube.com/watch?v=qsdP6Se6Cig&t=1s), and these differences have an effect on how Scrum is taught. The material in this video is also covered in Module 7 of the course.

The items below are a combination of what has been learned in hardware design thanks to Aaron Hoover and Lawrence Neeley of Olin and Charles Kim of Bucknell, what is in the software education literature; and my experiences. References to the two case studies and the literature are given where appropriate.

- Students can achieve the learning objectives listed in the previous section.
- Best team size is 3-5 students with sprint lengths of 2 weeks. The short cycle allows students to do multiple iterations with feedback, both from instructors and retrospection.
- Where traditional Gantt charts for the waterfall process (aka stage-gate) are made in a design class and then ignored, using a scrum board gives the students more experience with estimation and planning. The case study, *Agile Design of an Agile Fighter at Saab Aerospace*, shows how Saab has integrated Scrum into the waterfall process. Waterfall is the macro construct in which Scrum operates. Robert Cooper, the father of stage-gate, refined his method in his latest edition of *Winning at New Products*, integrating Scrum into waterfall[iii].
- Daily Scrum standup meetings are nearly impossible for students. It is best to have them two or three times a week, depending on circumstances. In the case study *A Student Team Designs a Prosthetic Arm Using Scrum Methods,* the team did standup meetings in class twice a week and additionally 1-3 times a week outside of class.
- Estimating hours to complete a task is difficult since students are inexperienced and, therefore, cannot accurately predict how long it will take. However, the only way to tune estimation skills is by making estimates and then doing a retrospective to review the estimate accuracy. The scrum method, with built-in sequential sprints with retrospectives, give an opportunity to improve time estimation. To compensate for poor skills, one software instructor used an adjustment factor of 1.0 – 2.0 on the early estimates[iv].
- Students often assume that agile means "Jump right in and start to work with developing requirements and concepts.". This results in poor products. **Warning!** I spent 30 years trying to impress on students that if they did not do the up-front elements of the design process well, the odds of a good product were poor. This is why *MDP6* is so front-loaded with QFD (understand the voice of the customer) and concept development. In many ways Scrum can be seen as a license to ignore these early necessities. It was by the student team in the case study; *A Student Team Designs a Prosthetic Arm Using Scrum Methods* and even they realized this shortcoming in their retrospectives.

  Grabbing the first idea and developing it seems to be a natural tendency. In experiment I did in the late 1980s video taping engineers while they designed and

then dissembling their taped design process, we saw evidence of this behavior[v]. It led to poor products in terms of complexity.

- Teaching scrum can be a course on its own or part of the capstone experience. I am a strong believer that teaching design process should be a separate course and that the capstone should be just that, an experience applying what was learned in earlier courses. I make this case in my paper *The Value of Teaching the Design Process During the Junior Year* [vi] (link to paper on https://www.mechdesignprocess.com/mechanical-design-process-text).
- Scrum can be taught through lectures, practical work on student projects, or with educational games. The scrum process has enough new terms and steps that lecturing about it seems a waste of time. It is best taught through a hands-on project. The bulk of the material in this document is designed to help students through their first Sprint, one step at a time, regardless of the project.

  In Mahnic's 2015 survey of papers on teaching Scrum for software[ii], he found four games for teaching Scrum: Legos[vii], ball game[i], paper exercise[viii], and card game[ix].
- Grading needs documentation. While the scrum process does not require specific documentation, it does make work visible in two ways: 1) the scrum board with what is doing and done, and 2) tasks whose deliverables are documentation. Individual efforts can be hard to assess when the team is functioning as a team. Difficulties arise because of the self-organization of Scrum teams and the inequality of task assignments among students. However, progress on the scrum board is individual, giving a window into the accomplishments of each team member. Further, specific tasks can be defined or assigned that require documentation. Documentation tasks can be seen in the Saab case study.

Introductory Material

The first nine modules of *Scrum for Hardware and Systems Design* are all introductory. The first module introduces me; the second is a course content overview. The third introduces resources that are available to the student. These are available on the website for you to download and use.

The fourth module covers how best to experience the course. Here it is recommended that the modules be covered sequentially. However, they are named and tied to the pictograph in a manner that allows easy selection of specific topics.

The fifth module is the story of four airplanes. In it, I develop the history of the processes followed by the Wright Brothers, Samuel Langley, Lockheed's F-35, and Saab's Gripen. The Wright Brothers were quite agile in their approach, while their competitor Langley was not. This is clearly demonstrated in explaining the process each followed as they attempted to achieve manned flight early in the 20th century. One hundred years later, Saab used Scrum to design their successful Gripen jet fighter, whereas Lockheed used a semi-scrum method early in the F-35's development and more traditional methods later. This module serves to introduce many of the basic scrum concepts.

Module 6 itemizes the reasons that traditional methods fail and how Scrum attempts to over them. While Scrum has found great success in software design, it has only begun to be adopted in hardware and systems design. The reasons for this are itemized in Module 7.

Finally, Modules 8 and 9 present the fifteen agile principles for hardware and systems design. These last three modules are very important to understand why Scrum manages the design process as it does. It is suggested that they be referred back to periodically.

Detailed Steps for Teaching Scrum

Scrum can be managed in the twelve steps, which are itemized on a single sheet and in a pictograph shown on the title page of this guide (and available for download on the website) In this section, each of the steps is discussed in detail. Reference is made to sections in *MDP6* as needed. It is suggested that these steps be introduced one at a time. Once the team has completed their first Sprint, then they can be on their own.

To introduce students to Scrum, the material in Module 12 defines the two cycles that form the core of the process. In Module 11, the twelve steps mentioned above are introduced. Finally, to help the student understand what is required to do Scrum, the 3-5-3 rule is introduced in Module 12.

It is assumed that a sprint length has been chosen before beginning the steps. Two to four weeks is recommended depending on the academic calendar. Scrum is time-boxed

meaning that the end of a sprint is a hard stop. Saab and other companies live by this (see the case study: *Agile Design of an Agile Fighter at Saab Aerospace*).

1. Organize the Team

Step		Activity	Artifacts	Meeting
1	Organize Team (Module 13)	Choose team members and identify Product Owner (PO) and Scrum Master (SM).	Team Roles	Preparation Meeting

Scrum has a Product Owner (PO) and Scrum Master (SM) on each team, as developed in Module 13. The Product Owner is the voice of the customer on the team. The Scrum Master is the manager of the process. Further, the ideal technical team members (those not the PO or SM) should be capable of completing any of the tasks, swarming to do those needed to complete the Sprint. The academic realities sometimes force these roles to be modified. Some suggestions are:

- The instructor takes the role of scrum master, managing the process for the team while teaching it. The detailed steps in this document are intended to help the instructor fill this role.
- Use small, 3-5 student teams. Scrum teams in industry typically have 4 – 9 members, including the PO and SM. It is suggested that small teams are better for education as no one can get lost and do nothing.
- Have one team member assume the responsibility of the product owner, echoing the voice of the customer in the team. This role can also be played by the instructor. In the Olin case study, there was no voice of the customer, and even the students realized this lack.
- In multi-disciplinary teams, each student brings a different interest and expertise to the team. The tasks naturally fall to specific students. This is potentially problematic, with each student working independently. Ensure that multiple students address each task, so there is overlap, and the team does not disintegrate into a group of individuals.
- If all students have the same basic background, let them choose the tasks they want to do, being consistent with the priority developed in Steps 4-8.

2. Develop Design Requirements

Step		Activity	Artifacts	Meeting
2	Develop Product Requirements (Modules14-18)	Generate requirements, specifications and users' stories	Product Goals	Preparation meeting

Developing requirements is so important that the subject is covered in four modules (14-18). Module 14 focuses on how developing requirements is having conversations about:

- What is being designed?
- For whom?
- What does it need to do?
- How will you prove the need is met?

Module 15 shows how Quality Function Deployment (QFD) enables the conversations and more. Chapter 6 in *MDP6* has a very well refined presentation on QFD, a very strong method for translating the voice of the customer into measurable engineering specifications – measures that prove the requirements based on the voice on the customer are met. Using QFD is best at enabling the conversations and capturing requirements at the beginning of the design process.

Scrum bases requirements on user stories, the subject of Module 16. Stories are good for capturing evolving requirements are not as strong as QFD for hardware, as noted in the case studies. Scrum is somewhat weak on this very important step.

Module 18 presents how to use a mix of QFD and stories. The Saab case study well shows how to manage this mix.

Suggestions for teaching how to develop product requirements are:

- Use QFD at the beginning of the project. This will force the students to understand upfront:
  - Who are the customers?
  - What is it they want?
  - The importance of their wants?
  - How are the wants filled now?
  - How well does "now" meet the needs and where are the market opportunities?
  - How will we measure that the wants are fulfilled?
  - What are the targets for the measures?
- During the development of "what they want" encourage the use of stories in the scrum format: "As a < (customer role or system)> I want to <perform an action> so that I can <gain this benefit>". This format actually strengthens QFD by forcing the students to play the role of customer and imagining their needs.
- Show how QFD's first two rooms and scrum stories are synergistic.

3. Create Scrum Board

Step		Activity	Artifacts	Meeting
3	Create Scrum Board (Module 19)	Build Scrum Board (either physical or in software) with areas for Product Backlog, Sprint backlog (To Do), Doing and Done.	Scrum Board	Preparation meeting

Scrum boards are an excellent way for a team to keep track of stories and tasks. As shown in the Saab case study (*Agile Design of an Agile Fighter at Saab Aerospace*), these play a big role in managing the team activities. At Saab and many other companies, these boards are done on a whiteboard or wall, as shown in the figure here. This may not be realistic for a student team. In the case study *A Student Team Designs a Prosthetic Arm Using Scrum Methods,* the Olin College team used Trello, a whiteboard in the cloud. See below for other options.

Since secure common spaces are not generally available to students, have them use an online tool such as:

Trello: https://trello.com
- Free version

Pivotal tracker
https://www.pivotaltracker.com/
- Free for 3-person teams
- 30-day free trial for 5-person teams

Asana: https://asana.com/
- Limited basic version free

Jira: https://www.atlassian.com/software/jira
- Free trial offer

A comparison between them are at: https://project-management.zone/system/asana,pivotal-tracker,trello and https://technologyadvice.com/blog/information-technology/trello-vs-jira-choosing-an-agile-project-management-tool/

Just remember, with every new tool comes a new distraction.

4. Product Backlog Grooming

Step		Activity	Artifacts	Meeting
4	Rank Order Product Goals (Modules 20 & 21)	Rank requirements based on dependency, uncertainty, importance and lead time.	Product Backlog and Scrum board	Product Backlog Grooming

The two modules, 20 and 21, focus on the first half of Scrum's two levels of planning, project planning. The first of these introduces how Scrum breaks planning into what product requirements to focus on first (the macro level planning) and then what tasks to do during a sprint to meet the important requirements (micro level planning).

Rank ordering the requirements is needed to help decide what to work on next. A requirement's rank is a function of four things:

1. Dependency: What requirements are dependent on it and what is it dependent on?
2. Uncertainty: How certain is the knowledge needed to meet the requirement?
3. Importance: How important is it to meet the requirement?
4. Lead time: How much time is needed to get needed materials or information?

Part of the QFD process determines what requirements are important to which customers and explores requirement dependencies. While QFD is a very strong method, one major element that is left out is the level of uncertainty. One of the strengths of the scrum methodology is that it helps direct tasks to reduce uncertainty and thus, risk.

Students need to learn how to judge all four and decide which requirements are most important to tackle first. This is not formulaic.

Dependency: Order the requirements first by dependency. The roof of the QFD (Section 6.9 in *MDP6*) and the Design Structure Matrix (DSM) may be of help here (Section 7.9 in *MDP6*). DSM is really an architecture design tool (see Step 10 below), but since It is designed to help determine dependencies, it is useful here.

The goal here is to make sure that if system A is needed by system B then it is worked first. What is usually the case is that A and B are interdependent, and both need to coevolve, a topic of the next step.

Uncertainty: Then order by uncertainty. The most uncertain aspects need to be attacked early on and iteratively brought to a level where the team is confident it can develop a quality product.

Since, design is the evolution of information punctuated by decisions, sprints allow the information to evolve in a controlled and thoughtful manner. For students, most design uncertainty is a function of lack of knowledge. So early sprints may be more research and testing of concepts, rather than actual product development. This can be seen in the case study *A Student Team Designs a Prosthetic Arm Using Scrum Methods* from Olin College. With Scrum offering a window on the reduction of uncertainty and the decisions made, faculty can track the learning, a byproduct of the methodology.

Importance: The QFD captures the ranking based on what customers think is most important and also capture market opportunities. These are the final arbiters for ordering the goals.

5. Sprint planning

Step		Activity	Artifacts	Meeting
5	Identify Tasks (Module 22)	Identify the tasks that need to be done during the Sprint complete with measures, targets and tests that define when they are done (i.e. test-driven development).	Sprint Backlog on Scrum Board	Sprint Planning Meeting

For the product goals chosen for the Sprint, the students should itemize the tasks that need to be done to meet the requirements. Tasks are test-driven activities and should be of the form:

"A team member will <activity to achieve> and show that it is completed by <measurable deliverables at a target level>"

Activities are action verbs or verb clauses describing what needs to be done. A list of sample action verbs is given in the text box. Have the students break down what they need to do as fine as possible.

Activities:
- Research
- Analyze
- Develop
- Draw
- Model
- Order
- Specify
- Gather
- Organize
- Plan
- Other actions

Measures and targets fall into the following types:
- Y/N: either there is measurable proof that activity was completed or there is not.
- An integer: a count or number of instances that are achieved or not
- A measured value that is achieved or not
- A measured value with tolerances that are achieved or not
- A limit that is reached or not

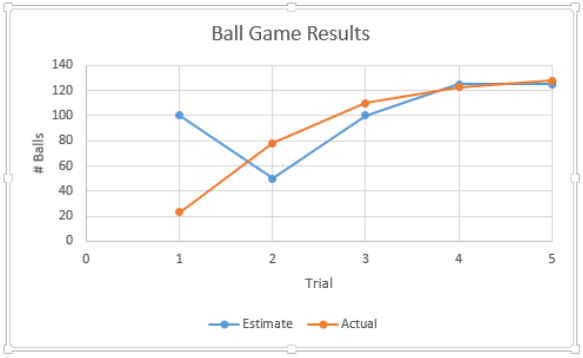
Some examples:
- A team member will research how to best grip a cup with a robotic hand and show that it is completed by presenting specs on at least five different end effectors.
- A team member will develop a written test specification and show that it is completed by delivering it to the test facility.
- A team member will analyze the energy needed to power a prosthetic hand for one day of use and show that it is completed by identifying at least three different batteries that a capable of delivering the energy and itemizing their important specs: weight, cycles etc.

Impressing test driven development (TDD) on the students may be one of the best gifts of teaching scrum. Each example clearly has a testable condition for showing it is completed.

6. Estimate Task Time

Step		Activity	Artifacts	Meeting
6	Estimate Task Time (Module 23)	Estimate the time each task will take.	Sprint Backlog on Scrum Board	Sprint Planning Meeting

There is some guidance on time estimation in Section 5.4.3.3 in *MDP6*. Time estimation[1] gets better with practice and it appears that the boxed sprint length helps improve the estimates, but this is unproven. Improvement with practice can be seen in the plot, results from the ball game exercise mentioned earlier. Here, as the task is repeated by a team their time estimates converge with the actual time taken. This game is a repeated task, a luxury that designers do not often get, but even with tasks that vary, estimation improves with practice.



One software instructor[iv] makes use of a time estimate "adjustment factor" for student estimates. He adjusts them by a factor of 1.0 – 2.0 depending on difficulty. See an example below.

ID	Product Backlog Item Description	Priority	Initial Estimate	Adjustment Factor	Adjusted Estimate
1-1	Maintenance of projects data (create/update/delete a project description)	1	20	1.50	30
1-2	Maintenance of developers data (create/update/delete developer's details)	1	15	1.00	15
1-3	Maintenance of a Product Backlog (create/update/delete a Product Backlog item)	1	60	2.00	120
1-4	Maintenance of a Sprint Backlog (create/update/delete a task)	1	60	2.00	120

One thing to realize about time estimation is that it greatly depends on how the information is requested. At a seminar some years ago, I did an experiment with150 person audience. I handed out a sheet of paper to each member of the audience that had a picture of sink full of dirty dishes and a list of how many plates, spoons glasses etc. were in the sink. On the sheet they were asked two questions: 1) to estimate of the time it will take to wash the dishes and clean up the sink afterwards and 2) the number of times they had done the dishes in the last 30 days. I then collected the sheets. The second question was to separate the experts (i.e. those who washed dishes more than 5 times in the last 30 days) from the novices.

What the subjects did not know was that there were four slightly different wordings of the first question about estimating the time. About 40 people got each of the options.

[1] Many in the scrum software community use story points rather than time as a basis. This has, in software, proven to be a good control of work effort. Story points are not covered in my material as they are not universally used in hardware and it takes additionally training to use. There are many good tutorials on the web to learn about the use of story points and how to get them using a technique called "planning poker".

The options and expert results were:

1. "**How many minutes will it take you to clean up the kitchen?**" – mean of 32 minutes with a standard deviation of 10 minutes.
2. **"How many minutes will it take for you to be 50% sure you can clean up the kitchen?"** - mean of 18 minutes with a standard deviation of 6 minutes.
3. **"How many minutes will it take for you to be 90% sure you can clean up the kitchen?"**- mean of 29 minutes implying that people with a standard deviation of 8 minutes.
4. **"How many minutes will it take for you to clean up the kitchen. We have to leave in 15 minutes?"** - mean 17 minutes with a standard deviation of 5 minutes.

Plots of this data are in Module 23. Comparing results 1 and 3 shows that, at least for washing dishes, the subjects were 90% sure they could meet their option 1 estimate, the simplest wording of the estimate request. When the estimate is anchored as in result 4, they come near the anchor. This is a well-known effect.

These results are both good and bad for task estimation in Scrum. As people become familiar with the work (they gain expertise) their estimates are for 90% surety that they can finish the task. This was also seen in the ball exercise above. This result supports the need for multiple sprints aiding student's tuning their ability to time estimate.

You can avoid anchoring by having students make independent estimates and then sharing the results. If they don't do it independently then, as soon as one person makes their estimate known to the others, all the rest are anchored to that estimate.

However, scrum tasks are time boxed, anchoring the total time available. So, there is naturally an anchoring effect here. I have not found any literature or discussion this effect. Using story points somewhat avoids this issue.

7. Choose Sprint Goals

Step		Activity	Artifacts	Meeting
7	Choose Sprint Tasks (Module 24)	Choose which tasks to complete during Sprint. Only start what you can finish.	Sprint Backlog and Product Backlog on Scrum Board	Sprint Planning Meeting

The objective is to define what is to be accomplished during the Sprint. This is the final step of micro-planning.

A key goal here is to generate the minimum viable product (MVP) during each Sprint. In the software world, this means releasable code that serves a core function for a set of users. Say you are developing code to help people book travel. The MVP might provide only the ability for frequent flyers to book on major airlines. The core function is booking on major airlines, and the set of users is frequent fliers. Then in a later sprint, other types of users and functions can be added to this minimal viable code.

For hardware, it is not that simple. The reasons for this are given in Module 7. This Module lists thirteen reasons that software and hardware differ. Important here is that for hardware systems you often cannot strip away functions as you can in software. For this and the other reasons listed in the Module, the MVP may be only a simulation, a prototype or an indication of learning (uncertainty elimination) about some of the functions.

Where waterfall generally produces a product at the end of the project, Scrum encourages working prototypes or simulations as early as is possible.

8. Begin Sprint

Step		Activity	Artifacts	Meeting
8	Begin Sprint (Also, Module 24)	Choose which team members are responsible for which tasks. Move tasks in the process to the "Doing" area of Scrum Board.	Scrum Board	Sprint Planning Meeting

At the beginning of a sprint, all the tasks are in the "To Do" area of the Scrum Board. As team members begin tasks, they move them to the "Doing" and then to the "Done" areas as they complete them by showing they have tested the results and met the targets.

By looking at the "Doing" section, you can see who is working on what tasks.

When a student moves a task to Done, it is ready for review. In the Scrum process, it is the Product Owner who must agree that the task is done. In an academic setting, it is the instructor who is the final arbiter of "done."

Since tasks are defined in terms of test-driven development (TDD), it is easy to assess if the task is indeed "done" and the quality of the result. It is like turning in homework, but the students have themselves predefined what the homework results should be. This seems a potent teaching tool, but there is no proof of its effectiveness.

9. Do Work

Step		Activity	Artifacts	Meeting
9	Do Work (Module 25)	Do the technical work moving tasks from "To Do", to "Doing" to "Done" on Scrum Board.	Scrum Board, Product Artifacts	Sprint Standup

Scrum gives no guidance about how to design anything. It is merely a framework for managing the process and techniques for keeping progress on course. During each sprint standup meeting, each member of the technical team needs to answer three questions:

- What did I do yesterday to help finish the Sprint?
- What will I do today to help finish the Sprint?
- What obstacles are blocking the team or me from achieving the sprint Product Goals?

If the meetings are not daily, then modify these to read:

- What did I do since the last meeting to help finish the Sprint?
- What will I do next to help finish the Sprint?
- What obstacles are blocking the team or me from achieving the sprint Product Goals?

Answering these questions is not a search for the guilty, but guidance on where to add resources or knowledge to get things done.

As far as what to do during the Sprint, Chapters 7-11 in *MDP6* are focused explicitly on engineering best practices needed to do quality work. While written for mechanical engineering students, this material is very helpful for other disciplines, and the reading is not difficult. Specific engineering topics in the chapters are:

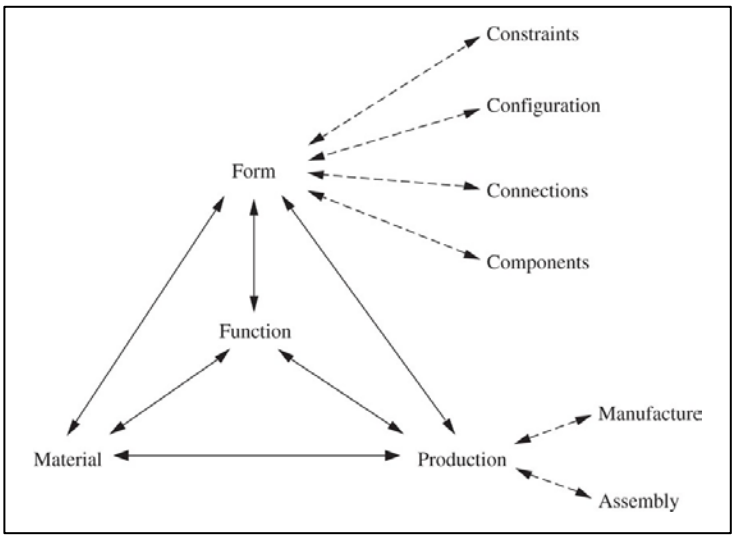
Chpt	Title	Topics
7	Concept Generation	• Designing with function • Concept generation methods • Using a morphology • TRIZ introduction
8	Concept Evaluation and Selection	• Feasibility evaluation • Technology readiness • Decision matrix • Product, project and decision risk • Robust decision making
9	Product Generation	• Designing constraints • Designing configurations • Designing connections • Designing components
10	Product Evaluation for Performance and the Effects of Variation	• Goals of performance evaluation • Trade-off management • Accuracy, variation, and noise • Factors of safety • Modeling for performance evaluation • Tolerance analysis • Sensitivity analysis • Robust design

11	Design for Cost, Manufacture, Assembly, and Other Measures	• Design for Cost (DFC) • Design for Manufacture (DFM) • Design for Assembly (DFA) • Design for Reliability (DFR) • Design for Test and Maintenance (DFT and DFM) • Design for Sustainability (DFS)

One feature that Joe Justice of Scrum Inc. and Wikispeed rightfully considers a significant best practice for hardware design is modular design around known stable interfaces. What this means is that a product should be divided, as best as possible, into discrete modules, and the flow of information, energy, and materials in and out of each module at its interfaces with others should be designed first. In other words, design should be from the interfaces to the modules. Designing from function to form aids in this effort (see Section 7.3 in *MDP6*). This approach solidifies the architecture of the product early on.

Further, the interfaces, the locations where energy, information, and material flow between the modules in the architecture, should be designed, as much as possible, so that they are unchanging – stable. In this way, teams can design modules independently – without affecting each other.

This design approach is enhanced in Chapter 9 in *MDP6*, titled "Product Generation", where form design (i.e., the physical product) progresses from constraints (i.e., the physical limitations) to configurations (i.e., the architecture of interrelated modules) to connections (i.e., interfaces) and finally to components (i.e., parts and assemblies) as shown in Fig. 9.3 from the text, reprinted here. Additionally, the Design Structure Matrix (Section 7.9) is an excellent tool for designing product architecture.



This hardware design sequence can be seen in the Saab case study.

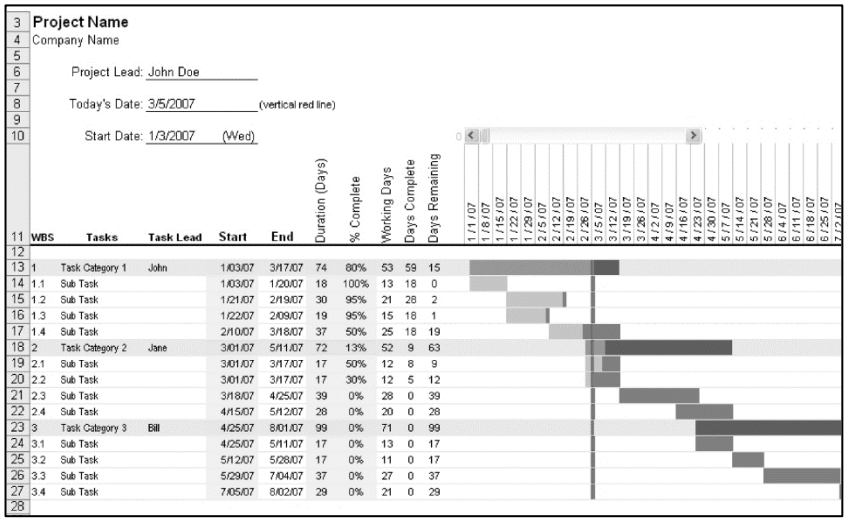
Students should be encouraged to design in the same manner, with early sprints developing the architecture and the interfaces. They must be discouraged from designing parts and assemblies (the components) until they have the architecture and the interfaces defined.

Note that this is directly contradictory to what is encouraged by CAD and solid modeling systems that build from part to assembly with little concern for architecture and interfaces. As early as 1990, I was chiding the CAD/Solid Modeling industry on the disconnect with good design practice[x] , and in a 2002 paper,[xi] I made the call to:

> "*Extend CAD systems to allow the designer to develop the architecture of parts and assemblies to fulfill needed function. They must allow the designer to work from the architecture to the shape and fit of the components themselves. This will require work with abstractions of parts and assemblies and building the geometry of objects from their architecture and interfaces with other objects.*"

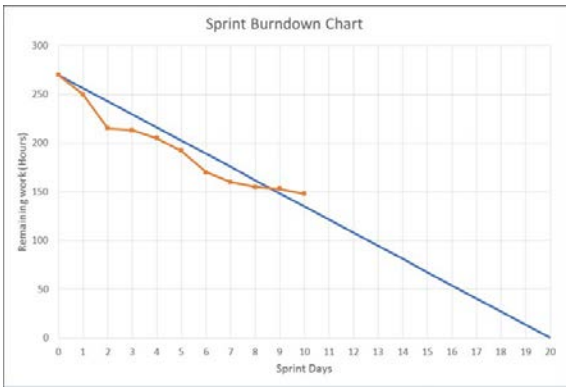
Also, part of the doing is tracking progress. Of all the steps, this seems to be the one most ignored by scrum teams. Sprint burndown charts are useful for visualizing progress. To best understand how to use them (or if to use them) compare Gantt charts and burndown charts.

Fig. 5.22 from *MDP6* shows a typical Gantt Chart used for planning. It itemizes the tasks with estimates for start and end dates and progress tracked by changing the color of the bars.



Limitations of Gantt charts are:

1. They are not often updated. They are done at the beginning of a project and then, for the most part, ignored. Since tasks change and dates slip, A Gantt chart is always out of date.
2. If too detailed, then hard to read and even more likely to be out of date. The management overhead is just too high.
3. Gantt charts are typically associated with a waterfall process. While work on multiple tasks can be seen (the vertical bar – the current state – shows five tasks in process), the actual progress toward done can be lost, especially if the chart is not very detailed.



Burndown Charts do not show the relationship between all the tasks and who is doing what and when. However, they clearly show the amount of work currently estimated to complete the project and how this estimate is changing over time. It is simple and easy to update.

Probably the best approach is to use both charts: the Gantt Chart to create the initial project plan to understand dependencies and the ordering of work at a macro level, and the Burndown Chart can track the work throughout each Sprint.

Burndown Charts are optional for student-teams until they have the sprint basics working. Then, tracking progress in each Sprint on a Burndown Chart becomes important.

10. Review Sprint

Step		Activity	Artifacts	Meeting
11	Review Sprint (Module 26)	Hold a sprint review (aka design review) where the progress on the product is demonstrated.		Sprint Review

This is a review of what was accomplished on the product during the Sprint. Step 13 is a review of the process followed. The sprint review is essentially a design review, and as such, the audience can be any stakeholder that should hear about the progress made.

In the software world, each Sprint ideally results in code deliverable to the customer. Reality shows that "deliverable" software is often not the sprint result.

In this material for hardware design, it has been suggested that the deliverable be the result of test-driven design (see steps 6 and 9). This way, the deliverables – the sprint progress – are whatever has been defined in the task's tests. This makes the sprint review very straight forward by examining the "test" results.

If multiple teams working on the same project, the sprint reviews are also an excellent opportunity for the student teams to learn from each other. Hearing how another team tried to solve a problem is often enlightening.

11. Review Design Process

Step		Activity	Artifacts	Meeting
12	Review Design Process (Also, Module 26)	Have a Sprint Retrospective where the design process is reviewed, and improvements developed.		Sprint Retrospective

This is possibly the most significant learning opportunity for students in the scrum process. Being able to reflect on what worked and what did not usually enlightening. The retrospective should create a safe space for people to share their honest feedback on what is going well, what could be improved, and generate a discussion around things that should change next time around – with actionable items documented. The underlined terms are key.

The sprint retrospective offers the following benefits[xii]:

- It creates a safe, blameless space for team members to share their valuable feedback.
- It allows the team to document wins and areas of opportunity.
- It provides an actionable list of the next steps and identifies who is owning which item.
- It identifies small, incremental changes that can lead to larger waves of improvement.
- It allows teams to iterate on their process to amplify results.
- It allows opinions to be heard.
- It helps the team mature.
- It makes each Sprint better than the last.

One tool that can help here is the *Team Health Inventory* available as one of the course's resources on the web site.

Applying Scrum

The last four modules of the online course give helpful hints for how to be successful when applying Scrum to hardware and systems. This material was initially designed for practitioners, but it applies to and should be shared with students.

Module 27 features the benefits and key points of Scrum. The key benefits as found through a survey of product designers using Scrum for hardware and systems found:

Measures	Reason for Adopting Scrum	Benefits of Adopting Scrum
Accelerate product delivery	75%	61%
Manage change	64%	71%
Increase productivity	55%	62%
Better business/engineering alignment	49%	65%
Increased product quality	46%	47%
Enhanced delivery predictability	46%	49%
Improve project visibility	42%	66%
Reduce project risk	37%	47%
Improve team morale	28%	61%

100% = great value, 75% = strong value, 50% = some value, 25% weak value, 0% no value

The shaded findings are important for both students and educators as they indicate what to expect when applying Scrum. Module 29 (I will come back to Module 28 in a moment) gives a list of pitfalls. While these are directed at industry, as said before, they can help highlight areas where student teams might struggle.

Module 28 addresses how to mix waterfall and Scrum. There has not been much work done on this mixing in n academic setting. However, as with industry, developing the high-level requirements at the beginning and having them drive the product backlog (the macro planning) and then following the scrum methodology for micro-planning the sprint and detailed requirements seems best. This is well discussed in Module 18, where Saab is used as an example.

Finally, Module 30 has two important checklists. The first is a general checklist for starting Scrum, and the second a suggested order of adoption reprinted right. This list gives you an order to present to your students. While item 2, team colocation is impossible in an academic setting, following the first nine items is a minimum introduction to Scrum.

- ❑ Internalize Scrum principles
- ❑ Co-locate the cross-functional team members
- ❑ Hold daily Scrum meetings
- ❑ Hold frequent product review and team retrospectives
- ❑ Establish Product Owner and Scrum Master team roles
- ❑ Establish Scrum boards
- ❑ Develop backlog lists for product
- ❑ Refine the task creation and format
- ❑ Do time boxed sprints complete with sprint backlog
- ❑ Use burn down charts
- ❑ Measure sprint velocity

References

[i] J. May et al, "Teaching Tip Play Ball: Bringing Scrum into the Classroom", Journal of Information Systems Education, Vol. 27(2) Spring 2016

[ii] Mahnic V. et.al., "Scrum in software engineering courses: an outline of the literature", Global Journal of Engineering Education, Vol 17, Number 2, 2015.

[iii] Cooper R. G. *Winning at New Products*: Creating Value Through Innovation, 2017. Basic Books.

[iv] Mahnic V., "Teaching Scrum through Team-Project Work: Students' Perceptions and Teacher's Observations", International Journal of Engineering Education, January 2010. https://www.researchgate.net/publication/257895006_Teaching_Scrum_through_Team-Project_Work_Students'_Perceptions_and_Teacher's_Observations/citations

[v] Ullman D.G., Dietterich T. G. and Stauffer L., "A model of the mechanical design process based on empirical data", AI EDAM, Volume 2, Issue 1 February 1988, pp. 33-52.

[vi] Ullman D. G., "The Value of Teaching the Design Process During the Junior Year" https://docs.wixstatic.com/ugd/20f020_e024ce31643a47379c3bcf2390f07ebf.pdf

[vii] Paasivaara, M., Heikkilä, V., Lassenius, C., and Toivola, T., "Teaching students Scrum using LEGO blocks", Companion Proc. 36th Inter. Conf. on Software Engng., Hyderabad, India, 382-391 (2014)

[viii] Von Wangenheim, C.G., Savi, R. and Borgatto, A.F., "SCRUMIA - an educational game for teaching SCRUM in computing courses". J. of Systems and Software, 86, 10, 2675-2687 (2013).

[ix] Fernanades and Sousa Fernandes, J.M. and Sousa S.M., "PlayScrum - a card game to learn the Scrum agile method". Proc. Second Inter. Conf. on Games and Virtual Worlds for Serious Applications (VS-GAMES), Braga, Portugal, 52-59 (2010).

[x] Ullman, D. G. "The Importance of Drawing in The Mechanical Design Process", Computers and Graphics Special Issue on Features and Geometric Reasoning, Vol. 14, No. 2, 1990, pp. 263-274.

[xi] Ullman, D. G., "Toward the Ideal Mechanical Engineering Support System", Research in Engineering Design 13(2), March 2002.

[xii] Huston A. "How to Run A Sprint Retrospective That Knocks Your Team's Socks Off", 2018, https://thedigitalprojectmanager.com/how-run-sprint-retrospective/