Scrum for Hardware Design
- Instructor's Notes -

David G. Ullman
January 2019
The Mechanical Design Process, 6th edition website: www.mechdesignprocess.com
Personal website: www.davidullman.com
Email: ullman@davidullman.com

Table of Contents

Preface

These notes support *Scrum for Hardware Design* a supplemental chapter to *The Mechanical Design Process* 6th edition (hereafter referred to *MDP6)*. They are intended to give background and approaches for teaching the Scrum framework to students studying hardware design (i.e., mechanical, systems, electronic, mechatronic engineers).

The material in *Scrum for Hardware Design* and these notes are fairly unique. 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. *Scrum for Hardware Design* and these notes is an effort to resolve this lack.

Where Scrum is a design process framework, it is idealistically discipline agnostic. However, there are some discipline-specific adaptations needed for hardware design that are included here. Further, hardware design using Scrum relies on many mechanical design best practices. Since *MDP6* is a compendium of these best practices, many of the Scrum activities described here reference material in *MDP6.*

Two case studies included in *Scrum for Hardware Design* support the material:

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

In the future, they may be included in the book *The Mechanical Design Process Case Studies* which already contains thirteen other cases supporting the methods in *MDP6*.

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*. Scrum is often described as an agile framework and adds a new approach to hardware design.

Yet, *MDP6* and other engineering process books are built around the waterfall method of serially progressing from need to concept, to product, to manufacturing. It is not very agile in its ability to respond to change or manage uncertainty. Waterfall is very important at the macro and planning level, but Scrum is important for doing work on a micro level, down in the trenches when information is uncertain and evolving.

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

1. Large, successful companies and organizations are using Scrum methods for hardware design including Tesla, John Deere, Saab Aerospace, Raytheon, Oak

Ridge National Labs, Bosch, Plantronics, SpaceX, and many others. This move toward Scrum 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 during the college experience.
5. Scrum forces teams to self-organize with responsibility for decisions and has built-in 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 just not the case after graduation.
7. Scrum better allows 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 would never have had the light bulb or phonograph.
8. Scrum helps students decompose problems, so they can solve small segments and 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 it helps internalize agile principles and provides a framework for solving challenging design problems. What makes Scrum promising for the 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.

1. **Team Organization**: Organize work in a team and manage team tasks.
2. **Design Requirements Development**: Generate incremental design requirements in terms of user stories and engineering specifications.
3. **Product Backlog Management**: Rank order stories to manage the product backlog.

4. **Task Identification and Associated Measures, Targets and Tests**: Identify the tasks needed to be done; complete with measures, targets, and tests that define when they are done (i.e., Test-Driven Development (TDD)).
5. **Task Time Estimation**: Estimate the time needed to complete a task.
6. **Sprint Focus**: Use sprint planning to choose which 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 answer the three Scrum progress questions.
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 Activities for Teaching Scrum" assumes that the students are designing a product concurrently to being introduced to the concepts.

There is minimal experience in teaching Scrum to students studying hardware design. 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 Appendix B itemizing these differences in *Scrum for Hardware Design*) and these differences affect how Scrum is taught.

The list below is 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.

- Best team size is about five students with sprint lengths of 2 weeks. The short cycle allows students to do multiple sprint iterations with feedback both from instructors and retrospection.
- Where traditional Gantt charts for the waterfall process (aka stage-gate) are made 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 stand-up meetings are nearly impossible for students. It is best to have them three or more times a week, depending on circumstances. In the case study *A Student Team Designs a Prosthetic Arm Using Scrum Methods* the team did stand-up 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 can not 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 with the exact fudge-factor a function of task difficulty[iv].
- Students often assume that agile means "*Jump right in and start to work without developing requirements and concepts.*". This approach can result 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 an experiment my students and I did in the late 1980s we videotaped engineers while they designed and then dissembled their design process, we saw evidence of this behavior[v]. It led to poor results.
- Teaching Scrum can be a course on its own or part of the capstone experience. I am a firm 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]
- Scrum can be taught through lectures, practical work on student projects, or with educational games. The Scrum process has enough new terms and activities 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 activity 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: using Legos[vii], a ball game[i], a paper exercise[viii], and a card game[ix]. I have not personally tried any of them although a conversation with one of the developers of the ball game indicated that it had been used successfully over many years and in many settings.
- Grading needs documentation. While the Scrum process does not require specific documentation, there are two ways to see student progress: 1) the Scrum board with the status of the tasks, and 2) tasks whose defined 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 assignment 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.

Detailed Activities for Teaching Scrum

The activities in the following sections are condensed on one sheet near the end of this document and also in Table 1 in *Scrum for Hardware Design*. Here, each of the activities is discussed in detail. Reference is made to sections in *Scrum for Hardware Design* and also sections of *MDP6* as warranted. It is suggested that these activities be introduced one at a time. Once the team has completed their first sprint, then they can be on their own.

It is assumed that a sprint length has been chosen before beginning these activities. 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

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

Orthodox Scrum has a Product Owner (PO) and Scrum Master (SM) on each team. 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 activities in this document are intended to help the instructor fill this role. This role can be eliminated and the introduction of Scrum process act as a de facto Scrum Master.
- Have the instructor act as the Product Owner. The Product Owner is the role that must be convinced that the deliverable at the end of each sprint is acceptable which is what a teacher needs to do.
- Use small, 3-5 student teams. Scrum teams in industry typically have 4 – 9 members counting the PO and SM. It is suggested that small teams are better for education as no one can get lost and do nothing.
- 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 Activities 4-8.

2. Develop Product Goals

Process Objective		Activity	Artifacts	Meeting
2	Develop Product Goals	Generate Product Goals in terms of user stories or requirements.	Product Goals	Preparation meeting

Orthodox Scrum is built on user stories (see section 5 in *Scrum for Hardware Design)*. However, stories do not work well in hardware as noted in the case studies. Scrum is somewhat weak in this critical activity. However, Chapter 6 in *MDP6* has a very well refined presentation on QFD, a powerful method for translating the voice of the customer into measurable engineering specifications. This method pushes hard to capture the Product Goals at the beginning of the design process.

Suggestions for teaching how to develop Product Goals:

- Use QFD at the beginning of the project. This will force the students to answer the following questions up front:
  - Who are the customers?
  - What is it they want?
  - The importance of their wants?
  - How are the wants are filled now?
  - How well does “now” meet the needs and where are the design opportunities?
  - How will they measure that the wants are fulfilled?
  - What are the targets for the measures?
- During the development of “what is it 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 strengthens QFD by forcing the students to play the role of customer and imagining their needs, or even better, seeking out the customers and hearing their voice first hand.
- Show how QFD’s first two rooms and Scrum stories are synergistic.

3. Create Scrum Board

Process Objective		Activity	Artifacts	Meeting
3	Create Scrum Board	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 significant 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. Walls or boards 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.

It is suggested that, if the students have a secure common space, they build the Scrum board on a white wall or whiteboard. Figures 18,19 and 20 in *Scrum for Hardware Design* show different Scrum Board options. The figure on the right shows a Saab Aerospace team at their Scrum board during a daily standup meeting.

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

A comparison between some of these is at https://project-management.zone/system/asana,pivotal-tracker,trello

Do keep in mind that: *With every tool comes yet another distraction*.

4. Rank Order Product Goals

Process Objective		Activity	Artifacts	Meeting
4	Rank Order Stories	Rank order stories based on dependency, uncertainty, and importance.	Product Backlog	Product Backlog Grooming

Rank ordering the stories is needed to decide what to work on next. A story's rank is a function of four measures:

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: Does the goal need to be addressed early because there are outside factors that require a lead time?

Part of the QFD process determines what requirements are important to which customers and explores requirement dependencies. While QFD is a powerful 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 goals 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 an architecture design tool (see Activity 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 B needs system A, then it is worked on first at least to the point that system B can proceed. What is usually the case is that A and B are interdependent, and both need to coevolve, a topic of the next activity.

Uncertainty: Second, 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 focused on research about similar problems and testing of concepts, rather than actual

product development as 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 o the decisions made, culty 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 captures market opportunities. These are the final arbiters for ordering the goals.

Lead Time: Sometimes issues need to be addressed early because of needed lead time for manufacturing, material availability, outside information availability or other uncontrollable factors.

5. Choose Product Goals for Sprint

Process Objective		Activity	Artifacts	Meeting
5	Choose Stories for Sprint	Choose Stories to be addressed in Sprint.	Sprint Backlog on Scrum Board	Sprint Planning Meeting

The Scrum method is one of continuous refinement. The objective here is to choose the goals to address in a future sprint. According to Scrum orthodoxy, these should be goals that can be completed during the sprint. However, with mechanical systems often a goal will need several iterations to resolve, and each iteration may be tackled in a different sprint.

To manage this and to direct the work, the next activity will focus on defining the tasks that need to be accomplished to fulfill the goal. Then the sprint backlog will be further groomed with some of the tasks addressed in this sprint and others put back in the Product Backlog for a future sprint.

6. Identify Tasks

Process Objective		Activity	Artifacts	Meeting
6	Identify Tasks	Identify the tasks that need to be done to meet Product Goals 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 finely as possible.

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

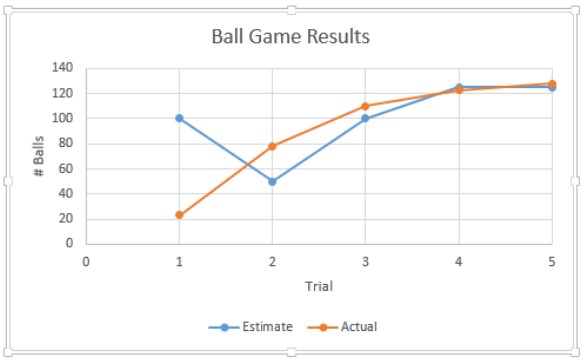
Measures and targets fall into the following types:
- Y/N: either there is measurable proof that activity was completed or there is not.
- An integer: a count or number of instances that are achieved or not
- A measured value that is achieved or not
- A measured value with tolerances that are achieved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

7. Estimate Task Time

Process Objective		Activity	Artifacts	Meeting
7	Estimate Task Time	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 set 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.[i] [abovei] Here, as a team repeats the task, 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

[1] Many in the Scrum community use story points rather than time as a basis. Story points have proven to be a good method control of work effort. Story points are not covered in *Scum for Hardware Design* as they are not universally used in hardware and it takes additional training to use. There are many good tutorials on the web to learn about the use of story points and how to assign them using a technique called "planning poker."

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 experimented with a150 person audience. I handed out a sheet of paper to each member of the audience that had a picture on it of a 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) Estimate of the time it will take to wash the dishes and clean up the sink afterward; and 2) How many times have you washed 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 for the first question about estimating the time. About 40 people got each of the options.

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 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.

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 most straightforward wording of the estimate request. When the estimate is anchored as in Option 4, they come near the anchor. Anchoring is a well-known cognitive psychology bias.

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 result 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.

Anchoring can be an issue in Scrum for two reasons. First, sprints are time-boxed which sends the message that all tasks can be done within the box, regardless of what the task is. I have not found any literature or discussion about this anchoring, but it must surely exist. Using story points somewhat avoids this issue.

Second, as soon as the first person on a team makes an estimate known to the others, they are anchored. You can avoid this anchoring by having students make independent estimates and then sharing their results. Have the high and low estimates justify their reasoning, then repeat the independent estimations. This cycle made need two or three iterations. If not converging, then the task is not understood the same by all and needs to be rewritten.

8. Choose Sprint Goals

Process Objective		Activity	Artifacts	Meeting
8	Choose Sprint Tasks	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. There is a good description of how to do this for the project goals in Section 5.1 of *Scrum for Hardware Design.* Here it is done at the sprint level.

An objective 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 for the first sprint 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 asitemized in Appendix B of *Scrum for Hardware Design*. This Appendix 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 Appendix, the MVP may be a simulation, a prototype or an indication of learning (uncertainty elimination) about some of the functions.

9. Begin Sprint

Process Objective		Activity	Artifacts	Meeting
9	Align Team Members with Tasks	Choose which team members are responsible for which tasks. Move tasks in-process to "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. The students have themselves predefined what the homework results should be, a seemingly powerful teaching tool but there is no proof of its effectiveness.

10. Do Work

Process Objective		Activity	Artifacts	Meeting
10	Do Work	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 beneficial for other disciplines, and the reading is not difficult. Specific engineering topics in the chapters are:

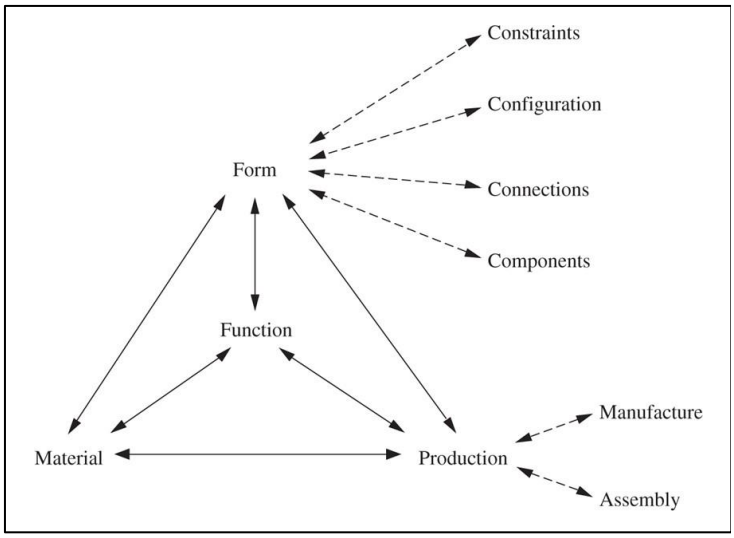
Chpt	Title	Topics
7	Concept Generation	• Designing with function • Concept generation methods • Using a morphology • TRIZ introduction
8	Concept Evaluation and Selection	• Feasibility evaluation • Technology readiness • Decision matrix • Product, project and decision risk • Robust decision making
9	Product 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 other modules 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. If one team wants to modify an interface, others affected will need to be a part of the modification decision.

This design approach is the focus of 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 this disconnect with good design practice[x] and in a 2002 paper[xi] I made the call to:

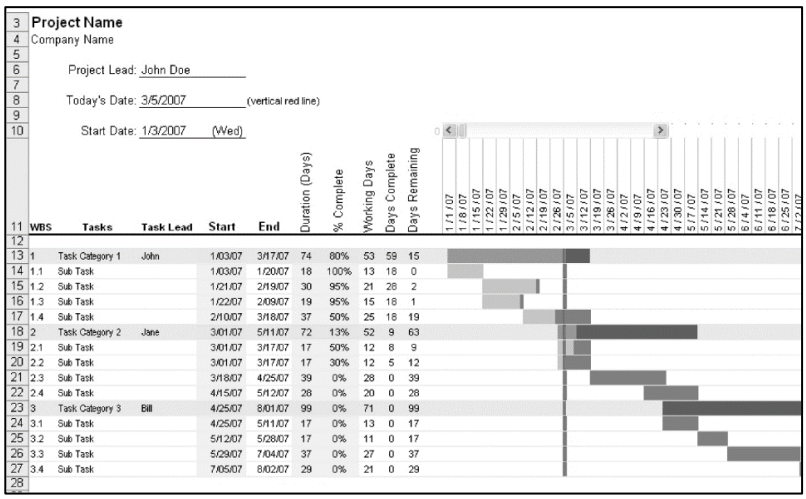
> “*Extend CAD systems to allow the designer to develop the architecture of parts and assemblies to fulfill needed function. They must allow the designer to work from the architecture to the shape and fit of the components themselves. This will require work with abstractions of parts and assemblies and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

11. Track Progress

Process Objective		Activity	Artifacts	Meeting
11	Track Progress	Use Burndown chart to track progress.	Sprint Burndown chart	

Of all the thirteen activities 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.

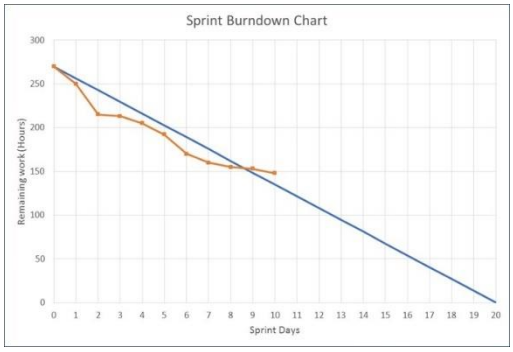
Fig. 5.22 from *MDP6* shows a typical Gantt Chart used for planning. It itemizes the tasks with estimates for start and end dates and progress tracked by changing the color of the bars.



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 sprint, 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.

12. Review Sprint

Process Objective		Activity	Artifacts	Meeting
12	Review Sprint Product	Hold a sprint review (aka design review) where the progress on the product is demonstrated.		Sprint Review

This meeting is a review of what was accomplished on the product during the sprint. Activity 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 Activities 6 and 10). This way the deliverables – the sprint progress – are whatever has been defined in the task's tests making the sprint review very straight forward by examining the test results.

If multiple teams are working independently 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 is solving the same problem is often enlightening.

13. Review Design Process

Process Objective		Activity	Artifacts	Meeting
13	Review Design Process	Have a Sprint Retrospective where the design process is reviewed, and improvements developed.		Sprint Retrospective

This meeting is possibly the most significant learning opportunity for students in the Scrum process. Being able to reflect on what worked and what did not is usually enlightening. The retrospective should create a safe space for people to share their honest feedback on what's 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 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 next activities and identifies who owns 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 a form that helps measure the health of the team. One such form is available as Template 4, *Team Health Inventory* on the *MDP6*'s website https://www.mechdesignprocess.com/templates

Acknowledgments

These notes were compiled after conversations with Aaron Hoover and Lawrence Neeley of Olin College, Charles Kim of Bucknell and Joshua Tarbutton of the University of North Carolina Charlotte.

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" 2018 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] Fernandes 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/

SCRUM FOR HARDWARE DESIGN

	Activity		Activity	Artifacts		Meeting
Organize	1	Organize Team	Choose team members and identify Product Owner (PO) and Scrum Master (SM).	Team Roles		Preparation Meeting
	2	Develop Product Goals	Generate Product Goals in terms of user stories or requirements.	Product Goals		
	3	Create Scrum Board	Build Scrum Board (either physical or in software) with areas for Product Backlog, Sprint Backlog (To Do), Doing, and Done.	Scrum Board		
Plan	4	Rank Order Stories	Rank order stories based on dependency, uncertainty, and importance		Product Backlog	Product Backlog Grooming
	5	Choose Stories for Sprint	Choose stories to be addressed in the Sprint.		Sprint Backlog	Sprint Planning Meeting
	6	Identify Tasks	Identify the tasks that need to be done to meet Product Goals complete with measures, targets, and tests that define when they are done (i.e., test-driven development).		Tasks for Sprint Backlog and Product Backlog	
	7	Estimate Task Time	Estimate the time each task will take.			
	8	Choose Sprint Tasks	Choose what tasks to complete during the Sprint. Only start what you can finish.			
	9	Align Team Members with Tasks	Choose which team members are responsible for which tasks. Move tasks in-process to "Doing" area of Scrum Board.			
Do	10	Do Work	Do the technical work moving tasks from “To Do” to “Doing” to “Done” on Scrum Board.		Product Artifacts	Sprint Standup
	11	Track Progress	Track progress on Scrum Board Burndown Chart.	Update Scrum Board		
Review	12	Review Sprint Product	Hold a Sprint Review (aka design review) where the progress on the product is demonstrated.	Updated Product Intent and Tasks		Sprint Review
	13	Review Design Process	Have a Sprint Retrospective where the design process is reviewed, and improvements developed.	Team Process Changes		Sprint Retrospective

A Brief History of Hardware Design Process Texts

In 1982, while teaching mechanical engineering at Union College in Schenectady NY, I began to transform the traditional nuts, bolts, gears, and bearings course - the common machine element design curriculum - into one that also covered the process of design. This approach was not original with me as it was greatly influenced by Walter Starkey of The Ohio State University who taught a graduate level design process course. I was fortunate enough to take his class the year before he retired.

In 1984 I joined the faculty at Oregon State University and, with the help of Bob Paasch, spun the process course off from the machine elements course. Oregon State has required these two courses - Design Process and Machine Elements - of their junior mechanical engineering students ever since.

Until the translated publication of Pahl and Beitz's *Engineering Design: A Systematic Approach*[2] (1988), there were no design process texts in English. This text is very sequential, what today is referred to as a "waterfall" or "stage-gate" process.

By the time I read Pahl/Beitz, the course at Oregon State was fairly mature and I had started to write the first edition of *The Mechanical Design Process.* McGraw Hill published this book in 1992. Over the years this text matured and the 6th edition is now self-published - released in 2018 and hereafter referred to as *MDP6*. *MDP* was greatly influenced by the English translation of Pahl/Beitz.

Since 1990 virtually all mechanical design process texts have been based on the waterfall methodology. While it is absolutely essential for hardware systems to be planned and be sequential from needs, to concepts, to product, to manufacturing, the real-world forces much non-sequential effort that cannot be ignored.

Like the others, *MDP6* features a waterfall process. In 2018 I began to work to integrate Agile methods into the text. I found that Scrum was widely adopted in software development and, increasingly used in hardware design to accommodate what cannot and should not be forced to be sequential. I now believe that the Scrum method should be taught in mechanical and systems engineering education.

A computer search of "scrum for hardware" and similar terms identified Joe Justice and Kevin Thompson as leaders in the effort to broaden Scrum's application in hardware.

Joe is the founder of Wikispeed[3] and President of Scrum Inc[4]. Scrum Inc. has a brief presentation on Scrum for hardware on its site[5] and offers training in all aspects of Scrum. I was able to take the one-week "Scrum in Hardware: Train the Trainer Course" offered by Joe in. Sept 2018.

---

[2] G. Pahl and W. Beitz, Engineering Design: A Systematic Approach, Springer, 1988.
[3] http://wikispeed.org/
[4] https://www. Scruminc.com/
[5] https://www. Scruminc.com/ Scrum-in-hardware-guide/

Kevin Thompson with cPrime Inc[6] has published “Eleven Lessons from Agile Hardware Development”[7] and “Agile Processes for Hardware Development”[8]. I was fortunate enough to take his two-day course on Scrum for Hardware in October 2018.

With this background and numerous conversations with Joe, Kevin and other consultants; with Jörgen Furuhjelm of Saab Aerospace (contributor to the case study *Agile Design of an Agile Fighter at Saab Aerospace*); and with Professor Aaron Hoover of Olin College (contributor to the case study *A Student Team Designs a Prosthetic Arm Using Scrum Methods*) and his colleagues Lawrence Neeley (also at Olin) and Charles Kim of Bucknell, I gained the knowledge needed to write this material.

[6] https://www.cprime.com/
[7] https://www.cprime.com/resource/white-papers/11-lessons-learned-from-agile-hardware-developemnt/
[8] https://www.cprime.com/resource/white-papers/agile-processes-for-hardware-development/