SE-359/459 Agile Software Development
Fall 2019 – 2020
500 points

Clean Sweep Robotic Vacuum Cleaner & Sensor Simulator

Your company has assembled a small team of developers and business people to work on the implementation of the Clean Sweep robotic vacuum cleaner control system.

1 Control System

The control system integrates the Clean Sweep hardware components so that the vacuum can successfully navigate and clean a typical home.

1.1 Navigation

The Clean Sweep roams around the home sweeping up any dirt that it finds. Therefore it must be able to discover the basic floor plan of a home based on the input from its sensors. The Clean Sweep has four directional sensors around its perimeter that detect when it is about to encounter an obstacle. The control system must interpret the output from these sensors and stop the vacuum from running into any obstacles. Upon encountering an obstacle, the Clean Sweep must determine a new direction that prevents it from running into another obstacle. If the Clean Sweep is unable to move, it should shut itself down and wait for the customer to move it to a new location.

Under no circumstances should the Clean Sweep attempt to traverse stairs. A sensor on the bottom of the Clean Sweep can detect the edge of stairs or other declines. If this sensor detects anything, the control system should treat it as though it had encountered any other kind of obstacle.

The Clean Sweep can move in four directions. If you envision the Clean Sweep as traversing a grid, and assume that it is located at position (x, y), then it can move to positions: $(x+1, y)$, $(x-1, y)$, $(x, y+1)$, and $(x, y-1)$. There is no need for the vacuum to change its orientation or turn around in order to navigate.

1.2 Dirt Detection and Cleaning

As the Clean Sweep moves, a sensor determines if there is dirt to collect. If so, the control system should tell the vacuum to sweep up the dirt. The sensor is not able to tell how much dirt is present, only whether or not the current location is considered clean. Therefore multiple uses of the vacuum may be required to clean a particularly dirty area.

The Clean Sweep is equipped with a surface sensor that allows it to detect whether or not the surface it is currently traversing is bare floor, low-pile carpet, or high-pile carpet. It should automatically shift its cleaning mode between the surfaces accordingly.

Each use of the vacuum removes 1 unit of dirt. The Clean Sweep has a dirt-carrying capacity of 50 units. Once that capacity has been reached, it should return to its charging base and turn on its *Empty Me* indicator. The Clean Sweep doesn't stop cleaning until:

- It has visited every accessible location on the current floor.
- Its dirt capacity has been filled.
- It only has enough power remaining to return to its charging station. See *Power Management* for more details.

Each time the Clean Sweep begins a cleaning cycle, it automatically assumes that each location is dirty. A cleaning cycle is defined as the start of the next cleaning after all reachable areas have been cleaned. So, if the Clean Sweep is partially done cleaning a floor and has returned to its base station to recharge, it will assume that the current cleaning cycle is not yet complete.

1.3 Power Management

The Clean Sweep has a limited battery life of 250 units of charge. Each movement and vacuum operation requires units of charge depending on the surfaces being traversed:

- Bare floor - 1 unit
- Low-pile carpet - 2 units
- High-pile carpet - 3 units

The charge required for the Clean Sweep to move from location A to location B is the average of the required charge costs for the surfaces at the two locations. For example, if the Clean Sweep moves from a bare floor to low-pile carpet, then the charge required is 1.5 units. It costs the same amount of charge to clean the current location is it does to traverse that location.

The Clean Sweep should always return to its charging station before it runs out of power. Home-owners that come home and discover their robotic vacuum out of power in the middle of a room tend to be dissatisfied customers. As it returns to its charging station, the Clean Sweep will not perform any cleaning. Upon returning to its charging station, the Clean Sweep will automatically re-charge to full capacity. Once re-charged, it will resume cleaning until it detects that it has visited every accessible location on the current floor.

1.4 Diagnostics and Troubleshooting

Since the Clean Sweep is a new product, it is important that technical support personnel are able to troubleshoot its operation.

1.4.1 Activity Log

On request, the Clean Sweep should be able to dump a log of its activity for the latest cleaning. This log should provide relevant information including:

- Sensor checks.
- Movement.

- Cleaning.
- Power remaining after each activity.
- Recharging.

2 Sensor Simulator

The Clean Sweep must maintain an internal map of the home. In order to write the control software, it is necessary to simulate the input of the Clean Sweep's sensor array as it might respond within actual homes. The Sensor Simulator provides this capability and acts as a virtual sensor array. These simulations will act as test data against which you can test your control software.

2.1 Floor Plan Representation

You will need to define a persistent floor plan layout. The layout file will need to represent different attributes of floor plan.

2.1.1 Layout Files

To facilitate the testing of the Clean Sweep, the Sensor Simulator must be able to "play back" pre-defined floor plans. This means that the Sensor Simulator should be able to read a floor plan file and use that file to respond to requests from the Clean Sweep's control software.

The control software itself should have no knowledge of the floor plan layout. This file is only intended to allow the sensor simulator to emit controlled output based on interactions with the control software. The fact that the entire floor plan is available to the sensor simulator does not mean that the Clean Sweep has *a priori* knowledge of the home's layout. The Clean Sweep control system must discover the floor plan in real time just as the physical device would.

2.1.2 Layout

A floor plan is a grid comprised of a series of cells. Each cell is adjacent to at most 8 other cells.

The Clean Sweep can move 90° in any direction. Consider the figure to the right. If the Clean Sweep is on cell E, it can move to cells B, D, F and H. Similarly, if it is on cell A, it can only move to cells B and D.

A	B	C
D	E	F
G	H	I

2.1.3 Surfaces

The *surface sensor* detects the surface of the cell. This determines how much power is required to move into and clean that cell. The surface sensor can identify the following floor coverings:

1. Bare floor (e.g. wood, linoleum, etc.)
2. Low-pile carpet
3. High-pile carpet

The *dirt sensor* can tell whether or not a given cell must be cleaned. The simulator must be able to represent how many units of dirt are at a given location. While the sensor itself cannot tell exactly how many units are present, this will allow the simulator to continuously register a cell as dirty until the appropriate number of vacuum operations have been performed.

2.1.4 Navigation

The Clean Sweep has *navigation sensors* that allow it to move around the floor plan. The floor plan must identify where the Clean Sweep may navigate and when there are obstacles to its path. The floor plan must indicate whether there is a path from one cell to another or whether such navigation is blocked. Each path between two cells can contain the following:

1. Unknown. The Clean Sweep doesn't yet know what's in the specified direction.
2. Open. The sensor indicates that the path is open.
3. Obstacle. The sensor indicates that the path is obstructed.
4. Stairs. The sensor indicates that the path in that direction is blocked by stairs.

2.1.5 Charging Stations

Each floor plan may contain one or more *charging stations*. These charging stations are used by the Clean Sweep to replenish its battery when it runs out of power. The Clean Sweep always knows about the charging station where it begins its cleaning process. It can discover others if it is within 2 cells of the charging station. The detection of the charging station happens whether or not the path to the station is actually clear. For example, it is possible that the Clean Sweep could detect a charging station in another room, but not be able to get to that station if the room's door is closed.

The control system must identify cells that contain charging stations. As the vacuum sweeps the floor, it will continually track which charging station is closest to its current position and move to that station when necessary.

2.2 Sample Floor Plan

This section provides a more complex floor plan. Figure 1 is an example of a basic floor plan. In this plan there are 3 bedrooms and a main hallway that will be cleaned.

Notice that the doors, represented by the small rectangles, can be either open or closed. Each time the Clean Sweep encounters that door, it will need to determine whether or not it is open. This means that it is technically possible for the customer to accidentally (or deliberately) shut the Clean Sweep in a room.

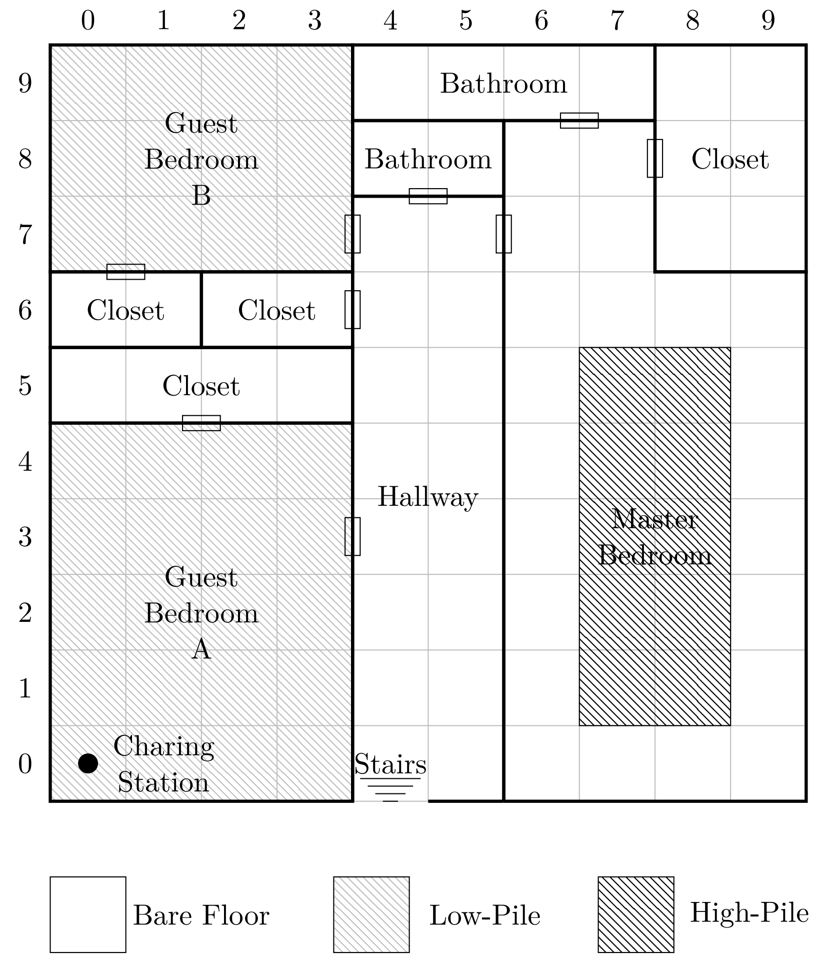

Figure 1: Sample Floor Plan

3 The Task

Your team will play the role of both "business" and "development". The instructor will play the role of "management". Your task is to develop the Clean Sweep's control system and sensor simulator.

3.1 Development

You are responsible for following Agile practices including working with the business to write user stories, estimating the development time, planning your releases, tracking your velocity and updating management's expectations accordingly, incorporating change requests, writing the various automated tests, and following development best practices including using source code control, continuous integration, constant code refactoring and automated build scripts.

If there are any questions about project functionality, management will provide the final determination.

3.1.1 Constraints

There are some constraints on how you may implement the solution. First, it must be written entirely in Java. Second, it must not require the installation of any other software like databases. Because the Clean Sweep's memory is limited, any third-party software or libraries must be carefully evaluated before they will be allowed. Management has the final say on the use of all such libraries.

There are no restrictions on the libraries that can be used for the Sensor Simulator. Since that software will not be installed within the actual Clean Sweep vacuum cleaner, there is greater flexibility.

You must develop the code as two separate modules: one for the Control System and the other for the Sensor Simulator. Remember that the control system must not have perfect knowledge of the floor plan; it derives all of its information from interactions with the sensors and those are driven by the information encoded in the floor plan layout document.

4 Activities and Ceremonies

You will use the basic activities and ceremonies of Scrum:

Backlog Management. You will need to keep your backlog groomed throughout the term. That means that you should always have a prioritized, estimated supply of stories. This will help sprint planning go much faster.

Sprint Planning. At the start of each sprint your team will carefully choose the stories that will add the greatest value for your minimum viable product (MVP). Plan on spending 1-2 hours per sprint for this activity.

Daily Scrum. This is the most challenging ceremony. Normally full-time teams meet daily to discuss their progress, but that may not be feasible for your team. You will need to arrange matters with your team members so that you can make solid progress on the project. This might be a combination of on-line or in-person meetings supplemented by useful technologies such as email, IM, and other collaborative work products.

Estimation. The team will need to agree on an estimation model for the various stories and tasks. Whatever model is selected, it must involve collaborative estimation. In other words, no one should be setting the estimate for a story or task without input from the other team members.

Review. At the end of each sprint, you will conduct a project review for the class that showcases what you've accomplished during the sprint. You would ordinarily plan on 30-60 minutes for this ceremony for a 2-week sprint, but since we're on a condensed schedule, plan for about 15-20 minutes instead.

Retrospective. At the end of each sprint you will, as a team, conduct a sprint retrospective. You should plan on 30-60 minutes for this if you intend to take it seriously, which you should.

5 Constraints

As with all projects, this one has a set of constraints that you need to follow:

1. Each student is limited to at most 12 hours per week outside of class to work on the project. You will work with your team to make your commitment for each sprint.

2. You must use only Java technologies for the implementation. This means the Java JDK as well as any additional libraries that you think might be useful.

3. You may only use freely available tools and frameworks.

4. You may be given other technologies to use during your project. No substitutions are allowed. These additions will be free.

6 Schedule

This project will run all term. You will deliver your project in phases using the Scrum and eXtreme Programming frameworks. The timeline and associated list of activities and deliverables is:

Sprint 0 (weeks 2-4, 2019-SEP-23 - 2019-OCT-07) — 125 Points

Goal: Bootstrapping.

Planning: During the first sprint, you have a lot of up-front work to do to get ready for the remainder of the project. This is the only sprint that the stories will be provided for you. You'll plan the remaining sprints around your own product backlog.

1. Devise your approach to decomposing the Clean Sweep project.
2. Prepare your product backlog.
3. Prepare a project plan for the term that describes which stories you think you'll complete in each of next three (3) sprints. You won't be held to that plan – after all, agile is all about adapting to change – but you should have at least thought through what should be delivered first, next, and last. Your goal for the end of the term is to have a MVP, whereas the goal of each sprint is to have demonstrable progress toward that product.

Review: Your initial product backlog in Trello.

Retrospective: A retrospective where you discuss how you can improve as a team. You will use the "stop", "start", "continue" format.

Evaluation: This sprint will be evaluated on:

- **0 pts.** **Technology**. There is no technology evaluation for this sprint.
- **25 pts.** **Backlog Management.** Your product backlog in Trello featuring epics and stories. You have defined priorities of these epics and stories in a way that leads you a minimum viable product at the end of the term and potentially shippable releases at the end of each sprint.
- **25 pts.** **Project Plan.** Your initial project plan identifying the product backlog items that you expect to complete in each sprint. This should be defined within Trello. Your project plan is based on the realistic availability of each team member.
- **25 pts.** **Review.** Since this is a planning phase, no review is expected.
- **25 pts.** **Retrospective.** The documentation and presentation of your team's lessons learned for the sprint and how you intend to apply those lessons to later sprints. You should use the "stop", "start", "continue" format.
- **25 pts.** **Contribution.** These are points that will be earned based on your peer evaluations.

Submit: A Powerpoint presentation that captures the major items described by the evaluation. You may include links to external systems like Trello. You only need to submit a single presentation for the team.

Sprint 1 (weeks 4-6, 2019-OCT-07 - 2019-OCT-21) 125 Points

Goal: Team's choice, but pick a goal that focuses on a specific theme. For example, you might choose a theme like "design the traversal algorithm" or "power management."

Planning: Based on your velocity from sprint 0, define your sprint plan.

Review: Your current project. Your overall project plan, accounting, and burndown. Your sprint plan, velocity, and burndown.

Retrospective: A retrospective where you discusses how you can improve as a team. You will use the "stop", "start", "continue" format. You should also review your retrospectives from the last sprint and discuss if the changes you made were beneficial.

Evaluation: This sprint will be evaluated on:

25 pts. **Technology**. You demonstrate good source code management practices by storing your code in a Github repository and by showing commits, with meaningful comments, by each team member. You have defined a sensible branching strategy and can articulate why you chose that strategy.

25 pts. **Backlog Management.** Your sprint backlog in Trello showing what your team completed and what you did not. Also, your product backlog showing new and removed stories.

25 pts. **Project Plan.** A burndown chart showing, by day, the progress your team made in closing the items from the sprint backlog. Also provide an accounting chart, like that provided in class, showing the story points at the start of the sprint, the number of points completed, any increases in scope, and any changes in estimates. Provide similar burndown and accounting charts that show the same information, by sprint, compared to your original project plan.

25 pts. **Review.** The current state of your project with an emphasis on what's changed since the prior sprint based on your sprint goal.

25 pts. **Retrospective.** The documentation and presentation of your team's lessons learned for the sprint and how you intend to apply those lessons to later sprints.You should use the "stop", "start", "continue" format. You should call out the changes that the team thought would be useful from the previous sprint, whether any progress was made on those changes, and the impact those changes have had.

Submit: A Powerpoint presentation that captures the major items described by the evaluation. You may include links to external systems like Trello and Github. You only need to submit a single presentation for the team.

Sprint 2 (weeks 6-8, 2019-OCT-21 - 2019-NOV-04) 125 Points

Goal: Team's choice.

Planning: Based on your velocity from sprints 0 and 1, define your sprint plan.

Review: Your current project, which shows demonstratable progress from the previous sprint. Your overall project plan. Your sprint plan, velocity, and burndown.

Retrospective: A retrospective where you discusses how you can improve as a team. You will use the "stop", "start", "continue" format. You should also review your retrospectives from the last sprint and discuss if the changes you made were beneficial.

Evaluation: This sprint will be evaluated on:

25 pts. **Technology**. On each commit to Github, your code is built under TravisCI using an automated build script.

25 pts. **Backlog Management.** Your sprint backlog in Trello showing what your team completed and what you did not. Your product backlog should also show new and removed stories.

25 pts. **Project Plan.** A burndown chart showing, by day, the progress your team made in closing the items from the sprint backlog. Also provide an accounting chart, like that provided in class, showing the story points at the start of the sprint, the number of points completed, any increases in scope, and any changes in estimates. Provide similar burndown and accounting charts that show the same information, by sprint, compared to your original project plan.

25 pts. **Review.** The current state of your project with an emphasis on what's changed since the prior sprint based on your sprint goal.

25 pts. **Retrospective.** The documentation and presentation of your team's lessons learned for the sprint and how you intend to apply those lessons to later sprints. You should use the "stop", "start", "continue" format. You should call out the changes that the team thought would be useful from the previous sprint, whether any progress was made on those changes, and the impact those changes have had.

Submit: A Powerpoint presentation that captures the major items described by the evaluation. You may include links to external systems like Trello, Github, and TravisCI. You only need to submit a single presentation for the team.

Sprint 3 (weeks 8-10, 2019-NOV-04 - 2019-NOV-18) 125 Points

Goal: Project Delivery.

Planning: Based on your velocity from sprints 0, 1 and 2, define your sprint plan.

Review: Your completed project along with final, project-level burndowns and reporting.

Retrospective: A team retrospective where you present the three (3) "do's" and "dont's" you learned about the agile frameworks while working on this project.

Evaluation: This sprint will be evaluated on:

25 pts. **Technology**. Your continuous integration environment demonstrates the running of meaningful unit tests and the team demonstrates the use of test-driven development using JUnit and Mockito.

0 pts. **Backlog Management**. There is no backlog management evaluation for this sprint.

25 pts. **Project Plan.** A burndown chart showing, by day, the progress your team made in closing the items from the sprint backlog. Also provide an accounting chart, like that provided in class, showing the story points at the start of the sprint, the number of points completed, any increases in scope, and any changes in estimates. Provide similar burndown and accounting charts that show the same information, by sprint, compared to your original project plan.

50 pts. **Review.** Your finished project. The team should plan on spending no more than 15 minutes presenting their finished product. If it's possible for you to make it to class for the evening, please do so. If your team is entirely DL, then please prepare and submit a recording of your Clean Sweep in action.

25 pts. **Retrospective.** The documentation and presentation of your team's top three (3) "do's" and "dont's" for the project and how you intend to apply those lessons to future agile projects.

Submit: A Powerpoint presentation that captures the major items described by the evaluation. You may include links to external systems like Trello, Github, and TravisCI. You only need to submit a single presentation for the team.

7 Peer Evaluation

Half of your project score for the term will be based on your team's evaluation of your contributions. At the end of the term, you will be asked to submit a peer evaluation of each of your teammates. This evaluation is intended to reflect your peers' understanding and ability to apply the agile practices, technologies, and techniques that make up this project.

The peer review determines how many of the project contribution points you will receive for your work during the term. Not everyone contributes equally to a project and thus not everyone should earn the same rewards for its successful completion. Your total peer evaluation score will be based on the median (middle value) of the peer evaluations provided by your team, including your own.

If you do not submit a peer evaluation form, then you will receive a "no show" (0%) for your self-evaluation, while each of your team members will receive "excellent" (100%) for your evaluations of them. Late evaluations will not be accepted.

An example. Suppose that your team receives full marks for each of the 4 sprints. There are 125 points per sprint for a total of 500 points earned by each member of the team. Suppose that your median contribution is evaluated at "Average." You would then receive 75% of the project points for your project contribution score for 375 points, for a total of 875 points for the project. If another individual on your team did not contribute as significantly and only received a median evaluation of "deficient" (25%), then they would only receive 25% of the project's 500 points for their contribution score, for a total of 625 points for the project.

Please fill out the form on the following page for yourself and each of your team members and submit it as a PDF to D2L for the "Peer Evaluation" assignment once it becomes available. Guidance on the evaluation criteria and scale is provided on the next page to help you in your assessment. You may provide additional comments below the table if you wish, but only your scores in the table will be used in the score.

Team Member Name:

DEPENDABILITY	**Always**	**Usually**	**Sometimes**	**Rarely**	**Never**
Comes to our meetings prepared.	5	4	3	2	1
Does the assigned tasks well.	5	4	3	2	1
Does the assigned tasks on time.	5	4	3	2	1
Makes timely responses to project communications.	5	4	3	2	1

TEAM PLAYER	**Always**	**Usually**	**Sometimes**	**Rarely**	**Never**
Seems invested in the team doing well.	5	4	3	2	1
Exhibits friendly professionalism.	5	4	3	2	1
Is respectful of others.	5	4	3	2	1
Did a fair share of the work.	5	4	3	2	1

PROJECT PARTICIPANT	**Always**	**Usually**	**Sometimes**	**Rarely**	**Never**
Proposes (but doesn't force) new ideas, suggestions, courses of action.	5	4	3	2	1
Builds on or extends others' proposals.	5	4	3	2	1
Expresses support for other team members' opinions or ideas.	5	4	3	2	1
Disagreement with other team members' opinions or ideas is done respectfully.	5	4	3	2	1
Accepts or respectfully negotiates when other team members disagree with his/her ideas.	5	4	3	2	1
Invites views or opinions from team members not actively participating in the discussion.	5	4	3	2	1

Based on your responses to the above questions, assign an overall rating you think is the fairest assessment of the team member's contribution to the overall project. Use the following scale:

Excellent	**Very Good**	**Satisfactory**	**Average**	**Marginal**	**Deficient**	**Unsatisfactory**	**Superficial**	**No Show**
100%	95%	85%	75%	50%	25%	15%	5%	0%

7.1 Scale

Each criteria should be evaluated on a scale of 0 to 5, with 5 being the highest score and 0 being the worst. Please use the following descriptions of each level as guidance.

Excellent	Consistently carried more than his/her fair share of the workload. They rarely relied on others and were often an individual on whom others relied.
Very Good	Consistently did what he/she was supposed to do, very well prepared and cooperative. Demonstrated a high degree of skill. Made few minor mistakes. Occasionally relied on others, but consistently contributed to the success of their peers.
Satisfactory	Usually did what he/she was supposed to do, acceptably prepared and cooperative. Demonstrated reasonable competence. They sometimes made mistakes, but recovered from them, learned from them, and did not make the same mistake again. Relied on others at the beginning, but by the end of the project were contributing as an equal.
Average	Often did what he/she was supposed to do, minimally prepared and cooperative. Did not volunteer for additional activities or to assist their team members who might have been over-committed.
Marginal	Sometimes failed to show up or complete assignments, rarely prepared. Did not demonstrate competence. Made frequent mistakes, sometimes learning from them, sometimes not. Relied heavily on others at the beginning but were able to gradually improve their own skills although they never contributed an equal share to the success of the project.
Deficient	Often failed to show up or complete assignments, rarely prepared. They often made mistakes but did not learn from them. They relied heavily on others and did not try to improve their own skills.
Unsatisfactory	Consistently failed to show up or complete assignments, unprepared. They were disruptive to the team. They made mistakes and failed to learn from them assuming they even tried at all. They relied heavily on others and made no attempt to improve their own skills.
Superficial	Practically no participation. Shows up at the last minute looking to help in the hopes of improving their peer evaluation score.
No Show	No participation at all.

8 Frequently Asked Questions

I've had some recurring questions about the group project for the class. I wanted to pull together some these questions and my responses to them.

Q: I'm not a strong coder so I'm not sure how much help I'll be as a developer.

A: This isn't a problem. While this is an SE class, and development is a key component, it's much more important that the group exercises the various agile practices than that they deliver software that completely addresses the project. This is one case where how you work is more important than the end result. You need to practice agile techniques to get better at using them. With that being said, it's certainly not acceptable for the group to have nothing to show for their efforts. The purpose of agile techniques is to deliver high-quality software. If you're not a strong developer, you'll need to be realistic during sprint planning when you commit to the tasks you'll be able to complete.

Q: You said in class that the classroom isn't like the real world. Then why are you making us do a group project?

A: I did indeed say that. However, that doesn't mean that we can't try to inject some realism into the course. Meaningful software development rarely happens as a solo project. The techniques you're going to learn are only meaningful when viewed through the lens of group-based software development and they will remain utterly abstract unless you get the chance to practice them in a team environment.

Q: Why should my grade depend on other people who might not have my work ethic?

A: This one comes up during every group project and it reflects an arrogance that has no place in either the classroom or the workplace. Why shouldn't your grade depend on other people? Contrary to the narrative many of us have grown up with, your success in most areas of endeavor is not due solely to your own efforts. We're each constantly receiving help from others. A wise person recognizes that and does not take more credit than is their due. An even wiser person will help others to be their best. This argument, however, is one of the reasons that you have the ability to provide a peer evaluation at the end of the term. That's your best means of holding your group mates accountable to their commitments.

Q: Why are you making it harder to collaborate by including DL students in the in-class groups and vice versa?

A: It's good experience for folks who haven't worked with a distributed group to get a chance to do so. You'll find that some of what we discuss in class needs to be adjusted once everyone isn't physically co-located. Look at this as an opportunity to practice and hone these skills since it's becoming more and more common in our industry.