Overview of Scrum.

This lecture is based on two SCRUM presentations:
Agile Software Development with SCRUM by Shveta Mehtani (http://www.scribd.com/doc/6578688/SCRUMAEG)
What is Scrum? by Richard Fennell (http://www.slideshare.net/businessquests/black-marble-introduction-to-scrum)
… as adapted by Michael Mateas
Upcoming deadlines

- **Wednesday (Jan. 9): release plan due**
  - A big effort: plans what you will do over the quarter
  - Will likely require at least two team meetings to complete
  - Developing a draft this quarter would be helpful
  - Sprint 1 begins this day

- **Thursday (Jan. 10): sprint 1 plan due**
  - Describes what will be accomplished in first 3 weeks of quarter

- **Thursday/Friday (Jan. 10 or 11): first daily scrum meeting**
  - Some tasks already completed
Game Lab Update (2)

- Each team should identify up to 2 Dell XPS machines or 2 Mac mini machines they will primarily use
  - Once this is done, we can start process of getting usernames on machines
  - Might be correlated with location you choose for scrum board

- If there is contention over machines, let me know
Game Lab

- Reminders
  - Only CMPS 17x students and associated project members (and, later in the quarter, game testers) are allowed in the lab
  - No P2P filesharing
    - All network traffic can be traced back to a specific port and machine
  - Treat ITS, BELS, and SOE facilities staff with respect when they are in the lab
    - They are very competent, and work hard, year-round, to make the lab a special place

- Other
  - Can enter lab via BE 364 (interior door), BE 368 (interior door) and BE 368 (exterior patio door). All doors use same omnilock codes.
  - Can enter BE 366 via interior door, using same omnilock code
  - Personal machines can access internet via wireless. If you need a fixed, wired connection, use one of the six open ports
Overview of Scrum
Lightweight Processes

- Small teams
- Incremental development
- Time-boxed scheduling
- Adaptive and agile
The Agile Manifesto: a statement of values

Individuals and interactions over Process and tools

Working software over Comprehensive documentation

Customer collaboration over Contract negotiation

Responding to change over Following a plan

Source: www.agilemanifesto.org
Scrum Characteristics

- One of the “agile processes”
- Small teams (< 10 people)
- Product progresses in a series of 2 to 4 week long “sprints”
- Visible, useful increments
- Requirements are captured as user stories in a list called the “product backlog”
- No specific engineering practices prescribed
Scrum History

- **1986**
  - Harvard Business Review paper by Hirotaka Takeuchi and Ikujiro Nonaka
  - Described new holistic approach to product development
  - Used game of Rugby as analogy
    - Team “tries to go the distance as a unit, passing the ball back and forth”

- **1991**
  - Book *Wicked Problems, Righteous Solutions* by Peter DeGrace, Leslie Stahl
  - Used “scrum” to refer to approach described in Takeuchi/Nonaka

- **Early 90’s**
  - Independent development of scrum methodology by Ken Schwaber (Advanced Development Methods) and Jeff Sutherland, John Scumniotales, and Jeff McKenna (Easel Corporation)

- **1995**
  - Sutherland and Schwaber presented paper describing Scrum at Business Object Design and Implementation workshop at OOPSLA ’95

- **2001**
  - Schwaber collaborates with Mike Beedle to write book *Agile Software Development with Scrum*

- **2002**
  - James Grenning develops planning poker
Releases: Multiple Sprints

- A Release occurs at the end of multiple Sprints
- In CS 171, there is one release, at the end of the quarter, and three Sprints
- CS 172 shifts to two week sprints, and two releases (at 6 and 10 weeks)
Scrum Process Overview

1. **Product Backlog**
   - As prioritized by Product Owner

2. **Sprint Backlog**
   - Expanded by team

3. **Daily Scrum Meeting**
   - 24 hours

4. **Potentially Shippable Product Increment**
   - 10 - 30 days

Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.
Sprints

- Scrum projects make progress in a series of “sprints”
  - Analogous to Extreme Programming iterations

- Typical duration is 2–4 weeks or a calendar month at most

- A constant duration leads to a better rhythm

- Product is designed, coded, and tested during the sprint
Sequential vs. overlapping development

Rather than doing all of one thing at a time...

...Scrum teams do a little of everything all the time

No changes during a sprint

Plan sprint durations around how long you can commit to keeping change out of the sprint

Sprints begin with a planning session, and end with a sprint review to evaluate the sprint. These provide points at which change can be accommodated.
Scrum framework

Roles
- Product owner
- ScrumMaster
- Team

Ceremonies
- Release planning
- Sprint planning
- Sprint review
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
Scrum framework

Roles
- Product owner
- ScrumMaster
- Team

Ceremonies
- Release planning
- Sprint planning
- Sprint review
- Daily scrum meeting

Artifacts
- Product backlog
- Sprint backlog
- Burndown charts
Product owner (in general)

- Define the features of the product
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results
- This role needs adaptation for CS 17x context…
The notion of Product Owner is tricky for this project class

- Each team “owns” their game design. There is no external customer, as in most software projects.

- But, the Product Owner must be a single person.
  - A team cannot be an effective Product Owner, since the Product Owner must be able to make decisions concerning feature priorities, features to include or cut, etc.

- The Professor/TA is a partial Product Owner
  - Since a grade is being assigned for how well the team does at creating their project, the Professor/TA is a stakeholder.
  - However, the Product Owner must be able to participate in Release Planning meetings, Sprint Planning meetings, and the Professor/TA cannot do this (not scalable).
Approach for this class

- Each team appoints one member as the Product Owner
  - This is typically the person in the team who “owns” the game design
  - They hold the game design vision, or, at the very least, are a person who is entrusted with the authority to make hard tradeoff decisions about the design.
  - This person will typically stay in the Product Owner role for the entire quarter
    - But, this can be changed at the start of a Sprint, if someone just doesn’t work out in this role
- The Professor/TA retain right to modify Product Owner decisions
  - For example, changing feature priorities, feature cut/save decisions
  - Same authority as Product Owner, but unlikely to exercise this authority often – authority is delegated with team’s Product Owner
The ScrumMaster

- Represents management to the project
- Responsible for enacting Scrum values and practices
- Removes impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences
Scrum Master in CS 171

- Each Sprint, one (possibly two) team members are appointed as Scrum Master
  - This role lasts for the entire Sprint
  - Each team member (except the Product Owner) must be a Scrum Master for at least one Sprint
    - On large teams, this role can be shared by two during a sprint
- Scrum Master is responsible for:
  - Maintaining scrum (task) board
  - Maintaining sprint burndown chart
  - Providing detailed feedback each week on activities of team members
  - Ensuring team follows correct Scrum practice
- Performance in this role is part of the individual performance grade for the class
  - Special team role performance – 8%
The team

- Typically 5-9 people
- Cross-functional:
  - Programmers, testers, user experience designers, etc.
- Members should be full-time
  - May be exceptions (e.g., database administrator)
- Teams are self-organizing
  - Ideally, no titles but rarely a possibility
- Membership should change only between sprints
story

A chicken and a pig are....
Scrum framework

Roles
• Product owner
• ScrumMaster
• Team

Ceremonies
• Release planning
• Sprint planning
• Sprint review
• Daily scrum meeting

Artifacts
• Product backlog
• Sprint backlog
• Burndown charts
Release

- A **release** is a major milestone in the development of a software project.
- A release contains a series of product **features**.
- Features are expressed in the form of **user stories**.

- The goal of **release planning** is to determine which user stories (features) will be included. This involves:
  - Taking the game concept and **decomposing** it into user stories.
  - **Estimating** the time required to perform each user story (using story points).
  - **Prioritizing** the user stories.
  - The release plan forms the input into the Sprint planning process.
A product feature is expressed in the form of a user story.
- This can be viewed as a specific technique for eliciting and writing software requirements.
- A user story is a software requirement.

User story format
- As a {user role}, I want {goal} [so that {reason}]
- Examples:
  - As a player, I need control over a laser pointer so that the cat will follow it.
  - As a player, I need to collect ammunition so that I can reload my weapon.
  - As a playtest manager, I need automated collection of gameplay metrics so that levels can be analyzed for areas that are too difficult.

Class exercise developing a few user stories for your game
INVEST in user stories

- What are the attributes of a good user story?
- **INVEST** conditions
  - Independent
    - Free of implementation dependencies on other stories
  - Negotiable
    - Useful as the basis for discussion between stakeholders and team
  - Valuable
    - Communicates value to player and to team
  - Estimatable
    - Possible to estimate effort to implement user story
  - Sized appropriately
    - Need to be small enough to fit into a Sprint
  - Testable
    - It must be possible to verify that a user story has been implemented.
Estimating size of user stories

- The relative size (implementation effort) of each user story is estimated using measure known as story points.
- Story points are **unitless**
  - Are not person-months, meters, hours, etc.
  - Key idea is to focus estimating effort on **relative** size
  - Use of unitless numbers avoids arguments
    - “That won’t take a week to implement – that’s easily a week and two days”
    - … but the point is trying to determine which tasks are $\mathcal{O}(\text{days})$, $\mathcal{O}(\text{weeks})$, and $\mathcal{O}(\text{months})$ – +/- a few days doesn’t matter!
- Story points are linear
  - A user story requiring 0.5 story points takes half the time to complete as one requiring 1 story point
  - Similarly, a user story requiring 3 story points is the same size as one requiring 1 story point and another requiring 2 story points
Using unitless points for estimation

Exercise using unitless points for estimation

- Let’s call the distance from Physical Sciences Building to the Science and Engineering Library “1 point”
- What is the distance from PSB to Engineering 2?
- What is the distance from PSB to McHenry Library?
Story Point Ranges

- When estimating, teams typically use a range of story points, as follows:
  - 0 points – Freebie, item already implemented, or ultra-trivial to do
  - ½ point - trivial
  - 1 – extra small
  - 2 – small
  - 3 – medium
  - 5 – large
  - 8 – extra large
  - 13 – double extra large
  - 20 – huge
  - 40 – exceptionally large
  - 100 – ginormous
  - $\infty$ - no conditions under which this is possible, technically impossible

- These values aren’t magic, and can be altered to fit a team’s needs
  - However, it is conventional to use these values

- Main value: spreads apart choices at high end, to avoid +/- 1 (or 2) kind of arguments
- The point range should agree with your planning poker deck (next slide)
Planning Poker

- A technique for teams to estimate sizes as a group activity
- Original article by James Grenning in 2002:
  - renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

Here’s how it works:

- Every team member is given a deck of cards with story point range
  - So, for range on previous slide, each person would have 12 cards
- The Product Owner picks a user story, and explains it to the team
- Team then discusses what is involved in implementing this item
- After discussion, each team member *privately* estimates the size of the item
  - Without making any assumptions about who might implement the item
  - Once estimate is done, take the card with the closest value, and place it **face down** on the table.
- Once everyone has played a card, they are all turned over **at the same time**
- If the estimates differ, the team members with the widest separation of estimates (low estimate, high estimate) **explain their reasoning**.
- All cards are returned, and the team plays another round.
  - Each person’s estimate may have changed, based on seeing the other estimates and listening to the rationale of the high and low estimates
- Repeat until **estimates converge**
  - Decision making rule is consensus; team should be comfortable with the estimate
Calibrating estimates

- Estimating user stories is difficult, especially when a team is in experienced
  - Accuracy improves over time, once many estimates have been performed, and a team can observe how well they have done

- For a team’s first estimate:
  - Pick a user story that all can agree is small, and estimate that first
  - Alternately, pick one that is small, large, and medium in size, and estimate those first, to get a sense of the range

- Once the team has estimated three or more items
  - Revisit the estimates, to ensure the team agrees with the relative size of the estimates of the items
  - This helps calibrate the scale used by the team

- Note that different teams might have different scales
  - That’s OK, so long as each team is internally consistent
Planning Poker exercise

- Break up into your teams.
- Pass out planning poker cards to four members
  - Others look on and watch, or can participate by writing card number on a scrap piece of paper
  - In real planning poker, everyone has a deck, we just don’t have enough decks
- Pick two members of your team who you think are good window washers
- Estimate how long it would take these two to wash the windows along the outside of the 2nd floor of Engineering 2, only where the patio is (i.e., where the stools and benches are)
Prioritizing user stories in a release

- During release planning, user stories must be **prioritized**
  - It’s a cop-out to say “everything is equally important”
  - You will implicitly prioritize things based on the order of implementation even if you don’t explicitly prioritize them up front
  - Better to be explicit about the order of implementation

- What do priorities mean?
  - A user story with highest priority is implemented first
  - A user story with lowest priority is implemented last
  - Lower priority items might **never be implemented**
    - If there is a feature you really want to see in the game, need to ensure it has a high priority

- Product Owner has ultimate authority over setting priorities
Assigning user stories to a Sprint

- During release planning, the team assigns user stories to a particular Sprint in which they can be implemented.
- This requires the team to guess how many story points they can implement during a Sprint.
  - Over time, a team will develop a good sense of their capacity. At the beginning, there will be a lot of uncertainty.
  - Make a good faith guess for now – this will be refined during Sprint planning.
- Important: Sprint goals identified during release planning are a forecast of work that can potentially be done by the team. They are not a commitment.
  - During Sprint planning, each user story will be broken down into tasks.
    - Task estimates are in ideal work hours, and are a commitment.
At the end of release planning:

- A prioritized list of user stories, with implementation time estimated in story points, organized into Sprints.

<table>
<thead>
<tr>
<th>Priority</th>
<th>User Story</th>
<th>Story Points</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sprint 1</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.</td>
<td>As {role} I ...</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>As {role} I ...</td>
<td>2</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sprint 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N.</td>
<td>As {role} I ...</td>
<td>15</td>
</tr>
<tr>
<td>N+1.</td>
<td>As {role} I ...</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Product Backlog

- All of the user stories that have not yet been implemented form the **product backlog**
- For a given release, some user stories will be grouped into planned Sprints. Others will not, but may be placed into future Sprints (or may be dropped).

- Product backlog = user stories assigned in current release + all unassigned user stories

- That is, the release plan is a subset of the product backlog intended for the current release
Product backlog

- The requirements
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint

This is the product backlog
# A sample product backlog

<table>
<thead>
<tr>
<th>Priority</th>
<th>Backlog item</th>
<th>Story Point Estimate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Allow a guest to make a reservation</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>As a guest, I want to cancel a reservation.</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>As a guest, I want to change the dates of a reservation.</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>As a hotel employee, I can run RevPAR reports (revenue-per-available-room)</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>Improve exception handling</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>...</td>
<td>30</td>
</tr>
</tbody>
</table>
Sprints

- Scrum projects make progress in a series of “sprints”
  - Analogous to Extreme Programming iterations

- Typical duration is 2–4 weeks or a calendar month at most

- A constant duration leads to a better rhythm

- Product is designed, coded, and tested during the sprint
Scrum framework

Roles
• Product owner
• ScrumMaster
• Team

Ceremonies
• Release planning
• Sprint planning
• Sprint review
• Daily scrum meeting

Artifacts
• Product backlog
• Sprint backlog
• Burndown charts
Sprint planning meeting

Sprint prioritization

• Analyze and evaluate product backlog
• Select sprint goal

Sprint planning

• Decide how to achieve sprint goal (design)
• Create sprint backlog (tasks) from product backlog items (user stories / features)
• Estimate sprint backlog in hours

Team capacity
Product backlog
Business conditions
Current product
Technology
Sprint goal
Sprint backlog
Sprint planning

- Team re-evaluates user stories from the release plan and product backlog they can commit to completing
- Sprint backlog is created
  - User stories are subdivided into tasks
  - Tasks are identified and each is estimated (~8 hours)
  - Collaboratively, not done alone by the ScrumMaster
- High-level design is considered

As a vacation planner, I want to see photos of the hotels so I can have a better idea of facilities
Priority 4 [10 Story Points]

- Code the middle tier (8 hours)
- Code the user interface (4)
- Write test fixtures (4)
- Code the foo class (6)
- Update performance tests (4)
User stories and tasks

1. As a dev, I want a level editor
   - Produce a level
   - Randomize levels
   - Save and load levels
   - 10 hrs
   - 2 people

2. As a moniker, I want to find the player
   - Pathfinding
   - 6 hrs
   - 2 people

3. As a dev, I want all work done so far in one Central location
   - Central location
   - Skeleton UMl
   - Code
   - 4 hours
   - 2 people

4. As a dev, I want to test familiarity by editing level editor
   - Test familiar by Phasm level editor
   - 2 people

5. As a dev, I want to teach team how to use lql editor
   - Lql editor
   - 6 hours
   - Everyone
Sprint planning (2)

- Task estimation
  - Performed as a group, using Planning Poker
  - Here, units of estimation are “ideal work hours”
    - The amount of work you can get done under ideal conditions
      - Full knowledge, no interruptions
    - Actual hours elapsed will be greater than ideal hours
  - Task estimates are a commitment to accomplish a development task in a certain period of time

- How many ideal work hours can each person perform?
  - Good question – so far, your group has no track record on this
  - For now, pick a conservative figure, such as 10-12 ideal hours/week
  - So, each group member can do 30-36 ideal hours of work per 3 week Sprint
Sprint planning (3)

- A likely scenario is that you team will find they don’t have enough time to implement all user stories in the release
  - In this case, need to assess user stories
    - Are the priorities all still the same?
    - If so, drop the lowest priority user stories until estimated work agrees with team’s work capacity
    - Can pick these up in later Sprints
  - What if the team finishes too soon (i.e., systemic over-estimation of task length)?
    - Very unlikely to occur – the opposite problem (under-estimation) is far more common
    - If it does happen, the team can add another user story midway through the Sprint
The sprint goal

A short statement of what the work will be focused on during the sprint

- **Database Application**
  - Make the application run on SQL Server in addition to Oracle.

- **Life Sciences**
  - Support features necessary for population genetics studies.

- **Financial services**
  - Support more technical indicators than company ABC with real-time, streaming data.
Managing the sprint backlog

- Individuals sign up for work of their own choosing
  - Work is never assigned
- Estimated work remaining is updated daily
- Any team member can add, delete or change the sprint backlog
- Work for the sprint emerges
- If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
- Update work remaining as more becomes known
Output of Sprint planning (for CS 171)

- Task listing (with time estimate), organized by user story (prioritized)
  - User story 1:
    Task 1 (time estimate)
    Task 2 (time estimate)
    ...
  - User story 2:
    Task 1 (time estimate)
    Task 2 (time estimate)
    ...
- Team roles
  - Team member 1: role
  - Team member 2: role
  - ...
- Initial task assignments
  - For each person, what is the first task they are working on?
- Initial task burndown chart
- Initial scrum board set up
- Schedule of Scrum meetings
  - When/where for 3 weekly face-to-face scrum meetings
Project Management During Sprints
Key project management challenges

- Awareness of the work of others
- Awareness of the current status of the project
- Clarity on what is your current task, and what is your next task
- Awareness of whether current sprint activity is completing tasks fast enough to meet sprint goals
- Making mid-course corrections if implementation activity is too fast or too slow.

Tools for addressing challenges:
- Scrum meetings
- Scrum board
- Burndown chart
The daily scrum

- **Parameters**
  - **Daily**
  - **15-minutes**
    - *Strictly* timeboxed
    - Can follow-up after meeting on bigger issues
  - **Stand-up**

- **Not for problem solving**
  - Whole world is invited
  - Only team members, ScrumMaster, product owner, can talk

- Helps avoid other unnecessary meetings
Everyone answers 3 questions

1. What did you do yesterday?
2. What will you do today?
3. Is anything in your way?

These are *not* status for the ScrumMaster
- They are commitments in front of peers
Scrum pitfalls

- Being late, missing the meeting
  - If you’re not present, the team doesn’t know what you’re doing
    - This is demoralizing – people assume nothing is happening
  - If someone needs information from you to move forward, they’re stuck
  - Disrespectful of other team members

- Grandstanding
  - Going into excessive levels of detail to make it seem like you’ve done more that you have (especially in front of TA)

- Going over time
  - Scrums are strictly 15 minutes, timeboxed.
  - Big issues are discussed by involved parties after the Scrum.
    - The Scrum just identifies the issues

- Failure to commit to work items
- Failure to update Scrum board

The scrum board

- A visual representation of **all** work that needs to be performed during the sprint
- Allows team members to clearly see tasks remaining
- Either put up on a wall, or put online (using a web-based scrum tool)
- A big chart
  - Rows are user stories and associated tasks
  - Columns are current status of tasks (To Do, In Progress, Done)
  - Tasks written on index cards or post-it notes

joshuahoover.com/2009/03/22/bitter-scrum-a-task-board-gone-wrong/
Sample task board
Updating the Scrum board

- During the scrum meeting, tasks are updated
- If a task is completed, it is moved from “In Progress” to “Done”
- If a task was “In Progress” at the last meeting, and is still “In Progress”, the time estimate for the task needs to be updated with remaining time
  - As well, if there is anything preventing completion of the task, this should be the answer to question #3 (“Is anything in your way?”)
- If a new task is assigned:
  - The name of the person working on the task is added to the task card
  - The task is moved from “To Do” to “In Progress”
- If a task is blocked (no further progress possible)
  - Move it back to “To Do” but mark it as obviously blocked (e.g., change the color of the card, add a sticker, etc.)
Keeping Scrum board up to date

- The primary value of the Scrum board comes from it being an accurate, up-to-date representation of the work of the team.
- If it is not kept current, its value diminishes quickly.
- It is the job of the Scrum Master to ensure the Scrum board is up-to-date.
  - The grade they receive for their role performance depends on this.
  - If someone misses a Scrum meeting, they need to proactively contact that person to find out what they have been doing, and update the board.
  - Scrum master also needs to ensure team updates task cards during daily Scrum.
Sprint burndown chart

- Burndown chart represents the total amount of work remaining in the sprint.
- As the sprint progresses, the remaining work should trend to zero.
- Typically posted on scrum board.
- Scrum Master maintains the burndown chart.
  - After each Scrum meeting, a new chart is created.
  - Sum the estimated time for all remaining tasks.
    - This is the data point (y-value) for that day (x-value).
- Ideal burndown trend.
  - Rate at which work is ideally performed so that all tasks are completed in sprint.

[Image of a burndown chart]

aydsoftware.blogspot.com/2009_01_01_archive.html
<table>
<thead>
<tr>
<th>Tasks</th>
<th>Mon</th>
<th>Tues</th>
<th>Wed</th>
<th>Thur</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code the user interface</td>
<td>8</td>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code the middle tier</td>
<td>16</td>
<td>12</td>
<td>10</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>Test the middle tier</td>
<td>8</td>
<td>16</td>
<td>16</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>Write online help</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Sample burndown chart
Sample burndown chart
When sprints go bad

- The burndown chart gives you early warning that your sprint will not achieve its objectives
  - Tasks clearly taking too long to complete, consistently
  - Need to take action

- How to adjust
  - Identify root cause
    - Under-estimation?
    - Impediments?
    - Flaky team members?
  - Get help
    - Contact TA/Professor
  - Reduce scope
    - Reduce number of user stories
    - Re-estimate tasks to ensure estimates reflect reality

scalingsoftwareagility.wordpress.com/2008/10/19/jeff-sutherland's-sprint-emergency-landing-procedure/
Evaluating the Sprint
The sprint review

- A sprint postmortem – occurs at the end of a sprint
- Team presents what it accomplished during the sprint
- Typically takes the form of a demo of new features or underlying architecture
- Informal
  - 2-hour prep time rule
  - No slides
- Whole team participates
- Invite the world
Sprint retrospective

- Periodically take a look at what is and is not working
- Typically 15–30 minutes
- Done after every sprint
- Whole team participates
  - ScrumMaster
  - Product owner
  - Team
  - Possibly customers and others
Whole team gathers and discusses what they’d like to:

- Start doing
- Stop doing
- Continue doing

This is just one of many ways to do a sprint retrospective.
Study questions

- Be able to describe role of Scrum Master, Product Owner in ideal Scrum, and for CS 171
- Define a user story, and know the template for a user story
- Describe how to play planning poker
- What is a story point? What is the value of having a story point range? Why are story points unitless?
- What are the INVEST criteria for story points?
- What are the outputs of release planning?
- What is the product backlog? How does this relate to the release plan?
- What is a sprint?
- What is the relationship of release to sprint?
- Why do we prioritize user stories? What does high priority and low priority mean?