IC470, Software Engineering

Lab 6: Release Burndown Charts for tracking progress of Agile projects, and ScrumMaster Plan for the next sprint

Due: As per the course syllabus

Lab Focus: Building and evaluating a burndown chart. Developing a sprint plan for the next milestone delivery

Lab group: If there are other mids in this section from your capstone team, work together as a lab group. If not, pair up with other free mids from this section and work together as a lab group (and focus on your choice of just one of your groups capstone projects for this lab).

"I said I'd like to see the Burndown Chart, not burn down the chart!!!"

Tracking progress of an Agile project. A snapshot of an Agile project's progression can be visually represented via burndown charts. We will use a variation of a burndown chart tailored for use with your capstone project (and the academic calendar!) called a Release Burndown Chart. In terms of the Scrum approach to managing Agile software development, a Release Burndown Chart represents the entire project as a single sprint. A sample burndown chart is given at the bottom of this lab for use as a reference.

1. **Lab Requirements** (see part 2 for specific deliverables):
    a. Individual effort calculation (work **alone** on this part). For each acceptance test plan test case in your capstone team's Functional Requirements Trace Table, each lab group member is to work alone and assign a value representing the level of "effort" that they think it will take to design, implement, integrate and test that particular test case. Use a (unitless) number from the Fibonacci Sequence (0,1,1,2,3,5,8,13,21,34,…) as the relative level of effort you assign to each

acceptance test case. You can use each Fibonacci number more than once for test cases requiring similar amounts of effort.

i. Why the Fibonacci Sequence? Using the Fibonacci Sequence acknowledges the imprecision of such subjective valuations of effort. We are primarily interested in assigning a relative value of effort to accomplish each test case. The expanding spread of the Fibonacci Sequence numbers allows us to assign, to relatively hard to do things, relatively large effort values and vice versa.
ii. For example, putting operational login buttons on a GUI might be assigned an effort level of '1' while developing an algorithm for comparing color variations in images for detecting embedded steganographic messages might be a assigned a level of effort of '8'

b. **Effort consensus** (work as a lab group on this part). Once all lab group members are finished with part a:
   i. **Effort consensus**. Go through each acceptance test case as a group and reach consensus on what the group *collectively* thinks the effort valuation should be for each test case.
   ii. **Primary developer assignment.** Assign a team member as the primary developer to each acceptance test case (note that this may be different than the person assigned as the primary developer for the corresponding functional requirement).

c. **Effort refactoring** (work as a lab group on this part). For any acceptance test cases with effort levels of 13 or higher, review as a group whether the acceptance test case can be broken down into a set of smaller, testable, units that together meet the same requirement as the test case in question.
   i. If so, replace the original test case with the set of smaller test cases, and assign each of the smaller test cases an effort level and a primary developer.
   ii. These changes must be reflected in your Functional Requirements Trace Table for the next milestone.
   iii. Note that you will need to explain such changes to your Acceptance Test Plan with your Customer and get their concurrence.

d. Refactoring Analysis (work as a lab group on this part).
   i. Refactoring. Regarding any Acceptance Test Plan test cases with an effort level of 13 or higher that you refactored in c above:
      1. Give the original test case (include the effort value).
      2. Give the set of replacement test cases (include the effort values).
      3. Give a discussion of how you refactored the original test case (what discrete chunks did you identify that allowed the refactoring?).

ii. High effort level test cases remaining. Regarding any Acceptance Test Plan test cases with an effort level of 13 or higher that you chose not to refactor:
    1. Give the original test case (include the effort value).
    2. Explain why it is not possible to refactor this test case.

e. **Release Burndown Chart** (work as a lab group on this part – see the sample burndown chart at the bottom of this file). For your Release Burndown Chart, create a graph (neatly hand drawn is fine – see the course Resources page for a link to some printable graph paper) as follows:
    i. Title the chart "Release Burndown Chart for *Team X: Project Name*" (substitute your team's number and the name of your project for "Team X: Project Name," you may shorten the project name somewhat if needed).
    ii. The horizontal axis gives the time (use units of weeks) between when you started implementing your capstone (use 1 Nov) and when your capstone must be fully implemented (use the Wednesday *before* Spring Break). By "fully implemented," we mean:
        1. 100% of your acceptance test cases completed to your Customer's satisfaction,
        2. all system components integrated together and working with each other, and
        3. ready for your customer to begin alpha testing.
    iii. The vertical axis gives the amount of team-calculated, unitless, effort for each acceptance test case in your Acceptance Test Plan.
    iv. **Starting Backlog**: Add together all the effort valuations from your consensus results in part c. This is your total level of effort at the start of your project implementation and is known as your team's Starting Backlog.
    v. **Start datum**: Plot the Starting Backlog value as the leftmost (Start) datum on your burndown chart.
    vi. **End datum**: Plot zero as the rightmost (End) datum on your burndown chart.
    vii. **Ideal Work Remaining**: Connect your Start and End points with a straight line. This is your Ideal Work Remaining Line.
        1. Break and replot this line (see the sample burndown chart below) to represent extended periods such as Thanksgiving, winter break, and exam weeks in which you plan to make no progress on your capstone.
        2. Note that the slope of your Ideal Work Remaining Line increases when such breaks are included since no progress is planned during these breaks. Time, tide and formation wait for no one!
    viii. **Actual Work Remaining:** Plot the Actual Work Remaining as of today.

1. Do so by deducting from your backlog the sum of effort represented by the test cases you demo'd in your team's Milestone 4 (Part I - Progress Demo). This results in your Remaining Backlog.
2. Plot your Remaining Backlog value for the week in which these test cases were signed off on by your Customer as being completed to your Customer's satisfaction.

f. **Project Status:** Based on your Ideal Work Remaining vs Actual Work Remaining lines on your Release Burndown Chart:
    1. Give, both as an unreduced fraction as well as a percentage, your (Starting Backlog – Remaining Backlog) / Starting Backlog.
    2. Determine whether your capstone project is ahead, behind, or on schedule as of today. Explain your answer.

g. **Workload Distribution Table** (work as a lab group on this part). Give a table (see Table 0 below) with each team member listed in the left column (one per row).
    1. **Left Column**. In the left column, give each team member's name (one per row).
    2. **Center Column.** In the center column, give each team member's effort summation computed as the summation of the effort values of just the acceptance test cases for which the team member <u>is listed as the primary</u> developer.
    3. **Right Column.** In the right column, determine each team member's workload percentage as the team member's effort summation divided by your project's Starting Backlog. Show the team member's workload as both an unreduced fraction as well as a percentage. Note: If the right column were summed, you would get 100%
    4. **Workload Analysis:** Review your Workload Distribution Table and determine whether the project's total workload is reasonably distributed across the team members. Note that 'reasonably distributed' does not mean that everyone has the exact same workload percentage, rather, does the workload distribution make sense from the consensus perspective of the lab group.
    5. **Workload Re-Distribution**:
        a. If the workload is not reasonably distributed, identify changes to primary mids as needed to make the workload more reasonably distributed.
        b. If the workload cannot be reasonably distributed across the team members, explain why.

Team Member	Team member's total effort summation	Team member's workload
W.T. Door	35	35/83 = 42%
J. Gish	20	20/83 = 24%
J. Mid	28	28/83 = 34%
Workload Analysis: The workload is not reasonably distributed across the team members. MIDN Gish's workload needs to be increased so the workload is more evenly distributed across the project.		
Workload Re-distribution: Have MIDN Gish replace MIDN Door as the primary on the 'restart from last saved state' functional requirement's acceptance test cases.		

Table 0. Workload Distribution

h. **ScrumMaster's Plan** for the sprint to the next milestone (work as a lab group on this part). Scrum uses short, periodic, meetings where team members take stock of where they are on a project and reach consensus on what needs to be done next. Although the title of ScrumMaster sounds powerful, the ScrumMaster is not the project leader and is not held accountable for outcomes. The team as a whole is responsible for outcomes. For our purposes, the ScrumMaster is responsible for helping the team to reach consensus on:
   i. What can be achieved during the sprint to the next milestone?
   ii. Who will take the lead on accomplishing each item in the milestone sprint?
   iii. Where the team will be, relative to their burndown chart, presuming all items in the sprint are successful.
   iv. Who will take the lead on overcoming any known impediments?

Prepare a ScrumMaster's Plan for the sprint to the next milestone. Replace "X" in the below tables with the milestone number of the next milestone delivery. The filled out tables comprise your ScrumMaster's Plan for the sprint to the next milestone delivery.

ScrumMaster's Plan for Milestone 'X'. **ScrumMaster** ___Gish_____

Lead Developers *see notes below	**Sprint Backlog** (test cases being worked on)	**Expected "effort" value** for each test case

W.T. Door	1.1 Forgotten password and 1.7 FAQ page	2 and 3
J. Gish and S. Sam	3.2 Terrain map display - realistic shadow growth and motion at dusk.	6
J. Mid	5.7 Save gameplay to file for subsequent restart.	4

Table 1. Lead Developers and Sprint Backlog for Milestone "X"

*Notes on Lead Developers.

a) Every team member (including the ScrumMaster) must be assigned as a lead developer for at least one acceptance test case in each milestone sprint.
b) Team members may need to be assigned as the lead developer to more than one test case (especially ones with low expected effort values).
c) You may have up to two team members (one assigned as the lead developer and the other as the backup) to each acceptance test case in the milestone sprint (especially ones with high expected "effort' values).

Lead for addressing Unplanned Requirements/Test Cases	**Description** of Unplanned Requirements/Test Cases	**Steps** needed to address Unplanned Requirements/Test Cases
J. Gish	Customer wants to add a new functional requirement that shows a 30 second countdown timer. After the countdown timer elapses, player loses their turn.	Add new requirement to Functional Requirements Trace Table. Adjust Burndown chart to include replotting of Ideal Work remaining line. Adjust Starting Backlog to include effort values related to new functional requirement.

Table 2. Unplanned Requirements/Test Cases

Projected Burndown at next milestone (projection must include impact of any Unplanned Requirements/Test Cases)	**Projected Project Status:** Ahead, On, or Behind based on the requirements for the *next* milestone delivery and your projected burndown. Identify the amount for ahead or behind projections.
25%	Behind by 5%

Table 3. Burndown Projection for Milestone "X"

Lead for item	Description of non-test case items
W.T. Door	Arrange Customer meetings.
J. Gish	Prepare Action Items list emerging from Customer meetings, and ID who has the lead for each action item
J. Mid	Prepare Milestone Delivery Presentation

Table 4. Additional (non-test case) Items for Milestone "X"

Lead for resolving impediment	Description of impediment	Steps needed to resolve impediment
W.T. Door	AttackGoat character's movements on the game board are jerky.	Figure out how the GPU can be used to speed up AttackGoat character's refresh rate to produce a smoother display on the visual game board.

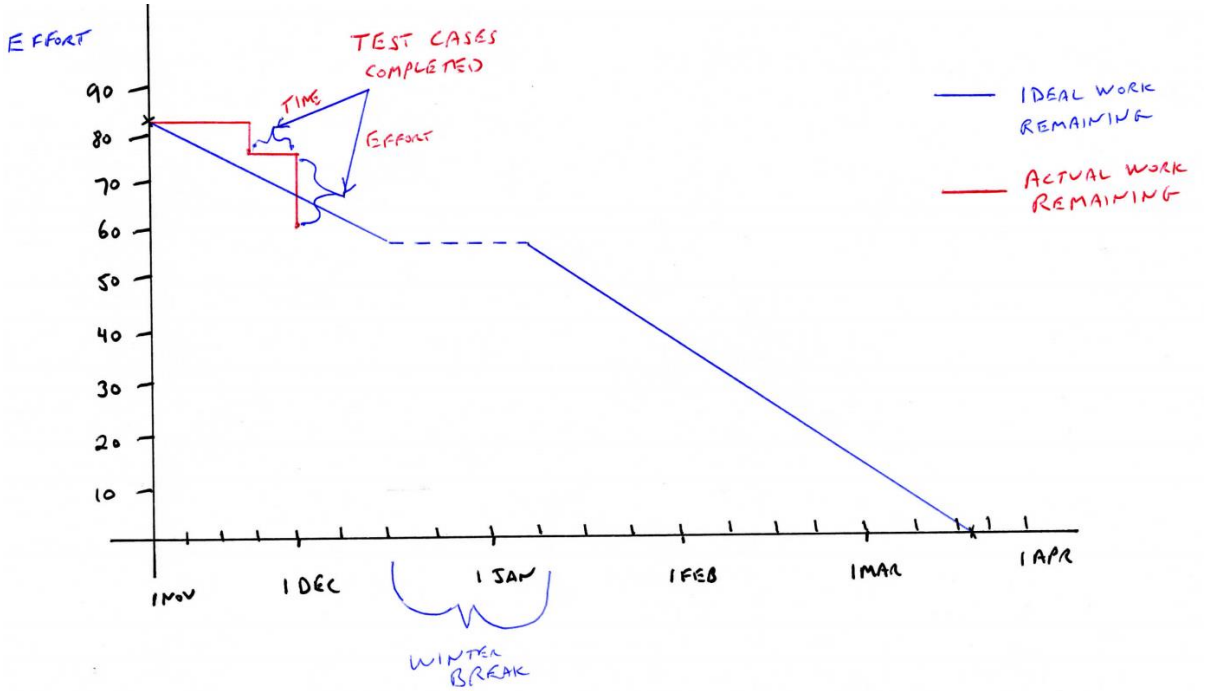
Table 5. Remaining Known Impediments to date (Cumulative, not milestone-specific)

2. **Lab Deliverables:** Each lab group, put your names at the top of your papers and turn in your:
    a. Refactoring Analysis,
    b. Release Burndown Chart,
    c. Project Status,
    d. Workload Distribution table and
    e. ScrumMaster's Plan (for the sprint to the next milestone).

Sample Release Burndown Chart

Assumes Starting Backlog is 83

Assumes Remaining Backlog is 60



% of Backlog Complete [as of 1st week of December] = (83-60)/83 = 23/83 = **27.7%**

Note that this chart shows the team to be slightly ahead of schedule.

Notes on Release Burndown Charts

X-Axis	Project timeline (units of weeks)
Y-Axis	The work (in terms of unitless relative effort) that still needs to be completed for the project for all the remaining Acceptance Test Plan test cases.
Project Start Point	This is the farthest point to the left of the chart and represents day 0 of the project.

Project End Point	This is the point that is farthest to the right of the chart and occurs on the predicted last day of the project.
Ideal Work Remaining Line	Straight line connecting the start point to the end point. At the start point, the ideal line shows the sum of the estimates for all the tasks (work) that needs to be completed. At the end point, the ideal line intercepts the x-axis showing that there is no work left to be completed. 1. This line may be broken to represent anticipated periods (such as Winter Break) during which no work will occur. 2. If functional requirements and/or acceptance test cases are added or removed after the start of the project, this line will need to be redrawn from that point forward to reflect the updated Ideal Work Remaining.
Actual Work Remaining Line	Shows the actual work remaining at various points in the project's timeline. At the start point, the actual work remaining is the same as the ideal work remaining but as time progresses, the actual work line fluctuates above and below the ideal line depending on the disparity between estimates and team effectiveness. This will look like a saw tooth as a project progresses due to the frequency of a Customer signing off on completed test cases - typically happen as part of a milestone delivery.
Percentage of Backlog Complete	Determined as (Starting Backlog – Remaining Backlog) / Starting Backlog.