AGILE DEVELOPMENT FRAMEWORKS: THEORY

CMU SE 17604

EDUARDO MIRANDA

ASSIGNMENT 1: DEFINING, PLANNING AND EXECUTING A SCRUM PROJECT

VERSION 1.0

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

CONTENTS

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

THE ASSIGNMENT

This is a running group assignment, which consist of four deliverables and a class activity. Due dates for each deliverable and the class activity are provided in the syllabus. The objectives of the assignment are for students to practice the writing of user stories and story maps, estimation using the Planning Poker technique, planning a Sprint, track the project execution using Sprint and Release burndown charts and conduct a retrospective session in which suggestion for improvements are made.

The expectation is that each submitted presentation will be of professional quality. Each report should contain at least the output of the activities described for each of them. Besides its evaluation in terms of content, the deliverables and the class activity will be graded on the basis of its overall organization, grammar, spelling and readability. Submissions consist of a single PDF file named with the name of the group and the deliverable number.

Students must structure the content so that is readable as a report. Image, html, disconnect files and zooming in or out of a gigantic one-page document are examples of things that will not be accepted. it important to revise and proofread the reports before they are submitted. At all times you must apply the reasonable person principle. If you find yourself wondering if something is right or wrong it is safe to assume the latter.

THE TIME REPORTING SYSTEM

INTRODUCTION

Your company has won a contract to develop a Time Reporting System for nonexempt employees. The system is going to replace a legacy mainframe application with a more modern web based one.

The system will be developed using Java and a yet undetermined relational database. It must also be able to execute seamless on Internet Explorer and Firefox. The system will be deployed in AWS.

SYSTEM DESCRIPTION

The purpose of the Time Reporting System is to allow nonexempt employees of a firm report the hours worked in different projects so that customers could be billed and the employees paid. The client has offices in four differ locations across the country. The system's users and its interface to other systems are depicted in Figure 1.

The hours worked by employees are charged to standard categories called "pay codes". Employees can only charge to those pay codes and projects they have been authorized to do so by the supervisors. Employees must report the hours worked weekly which for historical reasons is called a "time sheet". See Figure 2.

As many of the firm's employees work at their customers' premises, the system must be able to be remotely accessed, preferably through a web interface to eliminate the need to install software on the employees' machines.

To minimize the risk of accidental misuse or intentional tampering with the data, access to the system will be password protected and the creation of pay codes separated from the authorization to charge to them.

Worked hours can be only approved by those supervisors who created the authorization to charge or their delegates. A supervisor cannot authorize himself to charge to a pay code.

To minimize the risk of accidental misuse or intentional tampering with the data, access to the system will be password protected and the creation of pay codes separated from the authorization to charge to them.

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

Worked hours can be only approved by those supervisors who created the authorization to charge or their delegates. A supervisor cannot authorize himself to charge to a pay code. The initial list of user stories is given in Table 1.

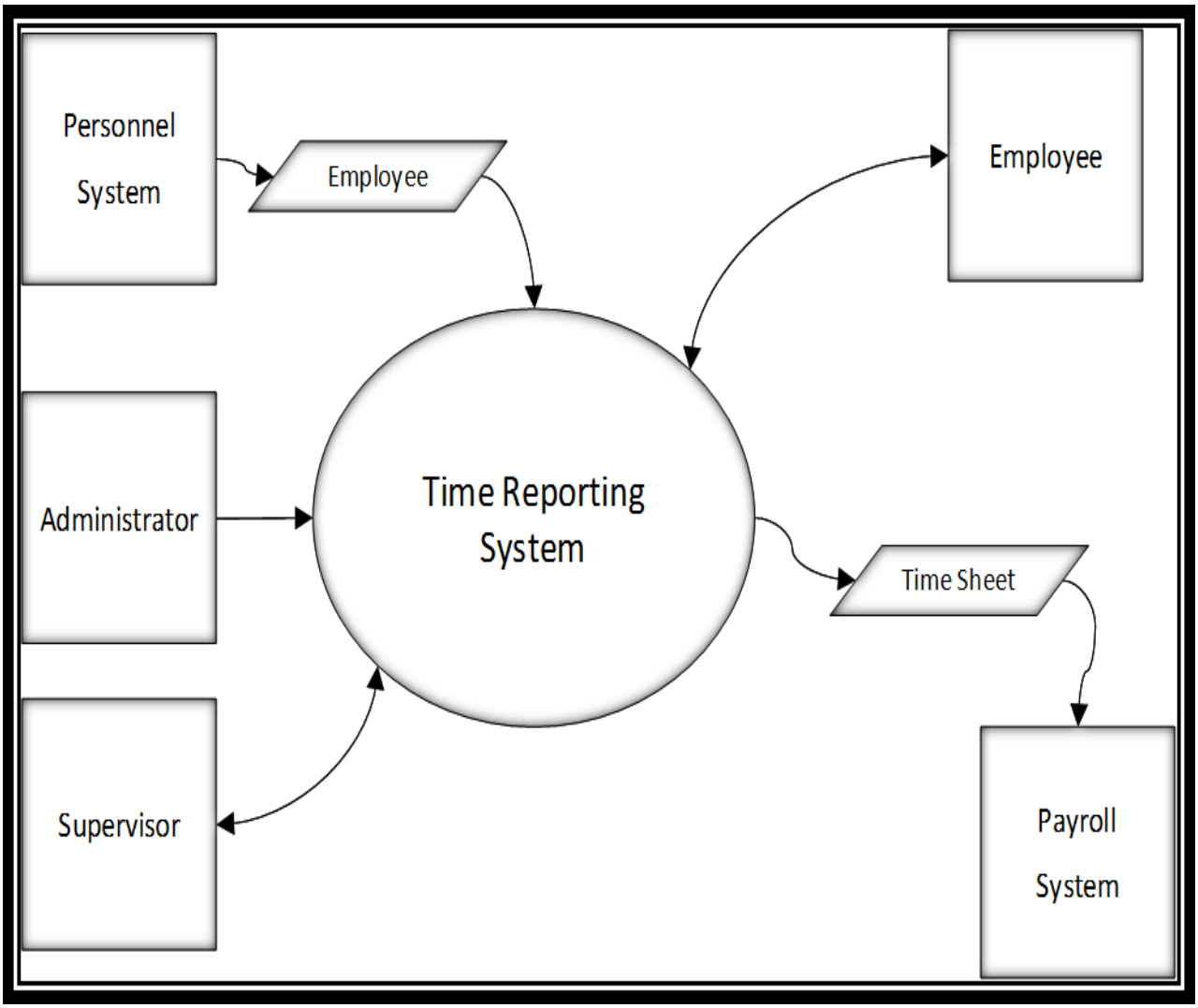


Figure 1. Context Diagram for the Time Reporting System

Weekly Employee Time Sheet for Week Ending: MM/DD/YY

Employee Name:

Employee No:

Supervisor:

			Regular hours		Overtime			
Day	Project	Task	From	To	From	To	Total	Comments
Week								

Submitted: MM/DD/YY

Approved by xxxxxxxxx on MM/DD/YY

Figure 2. Existing time sheet

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

Table 1. List of user stories identified during the initial meetings with the project sponsor

User Story Id		Role		Action		Benefit
5	As a(n)	System Administrator	I would like to	Authorize employees	so that	I can assure the integrity of the data reported
10		System Administrator		Block employees from temporarily use the system		Assure the integrity of the data reported
15		System administrator		Reset an employee password		Help people that forgot their password
20		System administrator		Create pay codes		So that the company can capture hours into appropriate categories while preserving the integrity of the data
25		System administrator		Delete pay codes		So that the company can capture hours into appropriate categories while preserving the integrity of the data
30		Any user		Log in into the system		So that he/she can use the system
35		Employee		Report working hours		So that I can log the hours worked
40		Employee		Submit reported hours		So that I will be paid
45		Employee		Recall reported hours		So that I can correct a mistake that was not detected at the time of data entry
50		Employee		Produce a report of worked hours		So I can have a record of the hours worked in a given period
55		Supervisor		Create a charge authorization		So that employees only charge hours to those projects/tasks they are authorized to
60		Supervisor		Approve time sheet		So I can control the numbers of hours reported by my subordinates
65		Supervisor		Reject time sheet		So I can control the numbers of hours reported by my subordinates
70		Supervisor		Delegate approval power		So I can designate an alternate to approve time sheets in case I am unavailable
75		Supervisor		Revoke a delegation		To terminate the authorization for somebody to approve time sheets on my behalf

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

DELIVERABLE 1: USER STORIES AND STORY MAP

This part of the assignment focus on documenting the needs and want of the user. After the initial discussions that led to the information provided in the introduction. The team needs to confirm their understanding and identify gaps and overlaps in the required functionality as well as to establish the look and feel of the new system.

a. Draw a story map that includes the given user stories

b. Identify between three and five missing user stories

c. For two of the user stories above, create the conversation part of the story by interviewing a member of another team, your pick (In the report provide the name of the student you interviewed, and the number of the group that interviewed you). The conversation may include for example, specific data you want to capture, or some policy the use story needs to comply with

d. Write the confirmation part of each user story (all of them)

e. Create a wireframe diagram for two key user stories that could be used as a guideline for the look and feel of the other user stories

f. Create a notional architecture for the system. Introduce technical user stories and knowledge acquisition stories

g. Create the product backlog

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

DELIVERABLE 2: PLANNING POKER ESTIMATION

This part of the assignment focuses on learning a well-known agile estimation technique: The planning poker.

a. Estimate all user stories in story points. Provide a sheet with the final estimate for each user story by each team member

b. Estimate all technical and knowledge acquisition stories in ideal hours

c. Evaluate the internal consistency of the user stories' estimation

d. Given a budget of 900 hrs. decide on the feasibility of the project taking in consideration all the meetings prescribed by Scrum, all the technical and knowledge acquisition stories and that 10% of the iteration effort ought to be allocated to grooming the backlog

e. Update the backlog with the estimates

DELIVERABLE 3: SPRINT PLANNING

Given the set of product owner preferences shown in Table 2 and assuming a sprint duration of two weeks, select a number of stories (user, technical, knowledge acquisition) to complete during the same.

a. Prepare a Sprint resource availability plan: discount time devoted to meetings, backlog grooming and other on-going activities mandated by the process

b. Discuss the initial velocity to be used for planning

c. Prepare a detailed design (e.g. class diagram, sequence diagram) corresponding to the selected stories. It should provide enough understanding for the team to build it

d. Create a dependency matrix at the class or method level to identify the development sequence

e. Map the detailed design onto independent, individually assignable specific tasks, e.g. develop class Customer, develop mock object DB, develop test fixture for the Customer class, prepare user documentation, etc.

f. Estimate each task above in ideal hours. Tasks shall be such that they can be completed by an individual in between 4 and 16 hours. Shorter tasks should be consolidated, longer tasks should be broken down.

g. Iterate until the work the team will be committing to is doable within the Sprint time box

h. Create the Sprint backlog

Table 2 Product owner preferences

Login
Create Pay Code
Create Charge Authorization
Submit reported hours
Approve time sheet
Reject a time sheet

CLASS ACTIVITY: EXECUTE THE SPRINT

The purpose of this activity is for participants to learn how to use the task board and the burndown charts to manage a Sprint. Secondarily the activity will provide some insight into the things that need to be considered when deciding whether to expedite, multitask or put more than one person to work in a given task. Students must preserve all material and take notes to use them in the retrospective deliverable.

The professor will provide the team with self-adhesive paper charts before the class; students need to procure their own sticky notes.

A simulation session, with a two-week sprint and four participating teams of four people each, can be run in approximately 1.5 to 2 hours.

BEFORE THE CLASS

a. Students must read the instructions for performing the Sprint simulation detailed in Appendix A

b. Students will bring to the class:

   i. A list of all work items: user stories, technical stories, etc., the team has committed to complete during the sprint, each individually consigned to a sticky note, as shown in Figure 3-a

   ii. For each work item in (i), all the tasks needed to realize it, each individually consigned to a sticky note as shown in Figure 3-b

   iii. A task board as shown in Figure 4, drawn in a self-adhesive paper sheet

   iv. A team log to register significant events, see Table 3

   v. A Sprint burn down/burn up chart as shown in Figure 6, drawn in a self-adhesive paper sheet

   vi. A Release burn down/burn up chart as shown in Figure 7, drawn in a self-adhesive paper sheet

   vii. The dependency matrix drawn in a self-adhesive paper sheet

c. Charts must be sizeable enough for the team to work together standing in front of it and for the instructor to see it

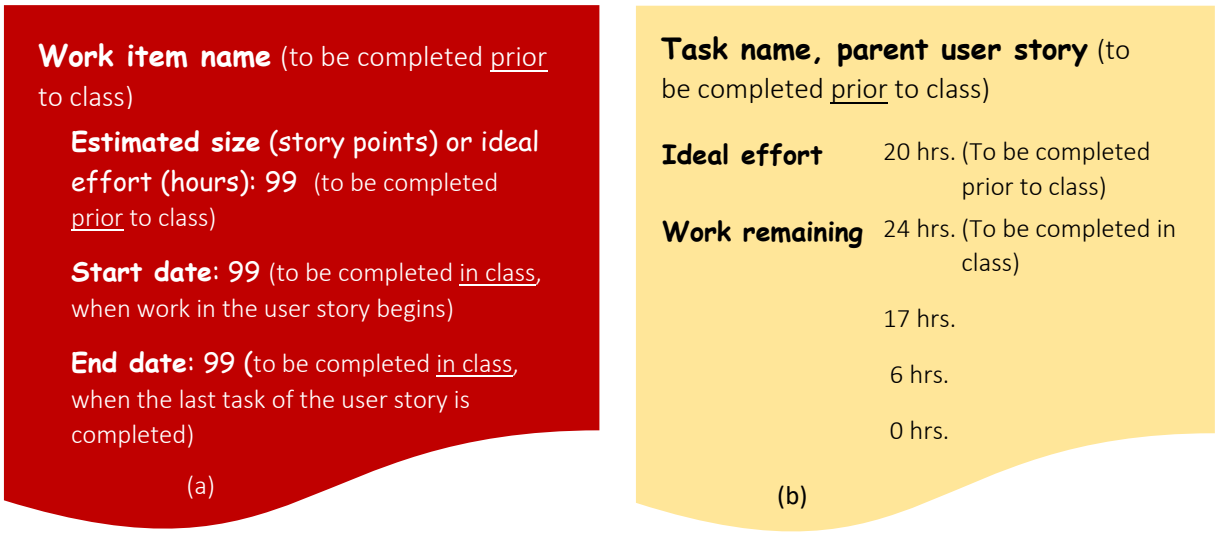


Figure 3 User story (a) and task card (b)

d. Leave some empty rows at the end of the task board to accommodate unplanned work discovered during the execution of the Sprint.

DURING THE CLASS

a. Team members will "execute" the sprint by following the dictates of two lotteries telling them whether to add or drop new and existing work and whether the scheduled work has been completed ahead of time, on time, blocked or delayed. See Figure 4. Notice, that although level of effort tasks such as the daily standup meetings and backlog grooming will not be included in the task board, the time they take is real and for that reason they must be accounted in the availability chart and in the Sprint burndown chart.

   i. Team members report the status of their tasks in a Scrum meeting

   ii. Team members having completed their tasks select the next task in sequence

   iii. Scrum master updates burndown charts & logs significant events

b. At the end of the Sprint:

   i. Unfinished stories are returned to the backlog with their estimated remaining time

   ii. Calculate the actual team velocity & efficiency

   iii. Discussion of the results and Q&A

Table 3 Team log (To be completed in class)

Day	Event	Decisions/Consequences
1		
2	Team member absent 3 days	Decide to reallocate tasks
3		
…	Team is falling behind schedule	Team goes into overtime
10		

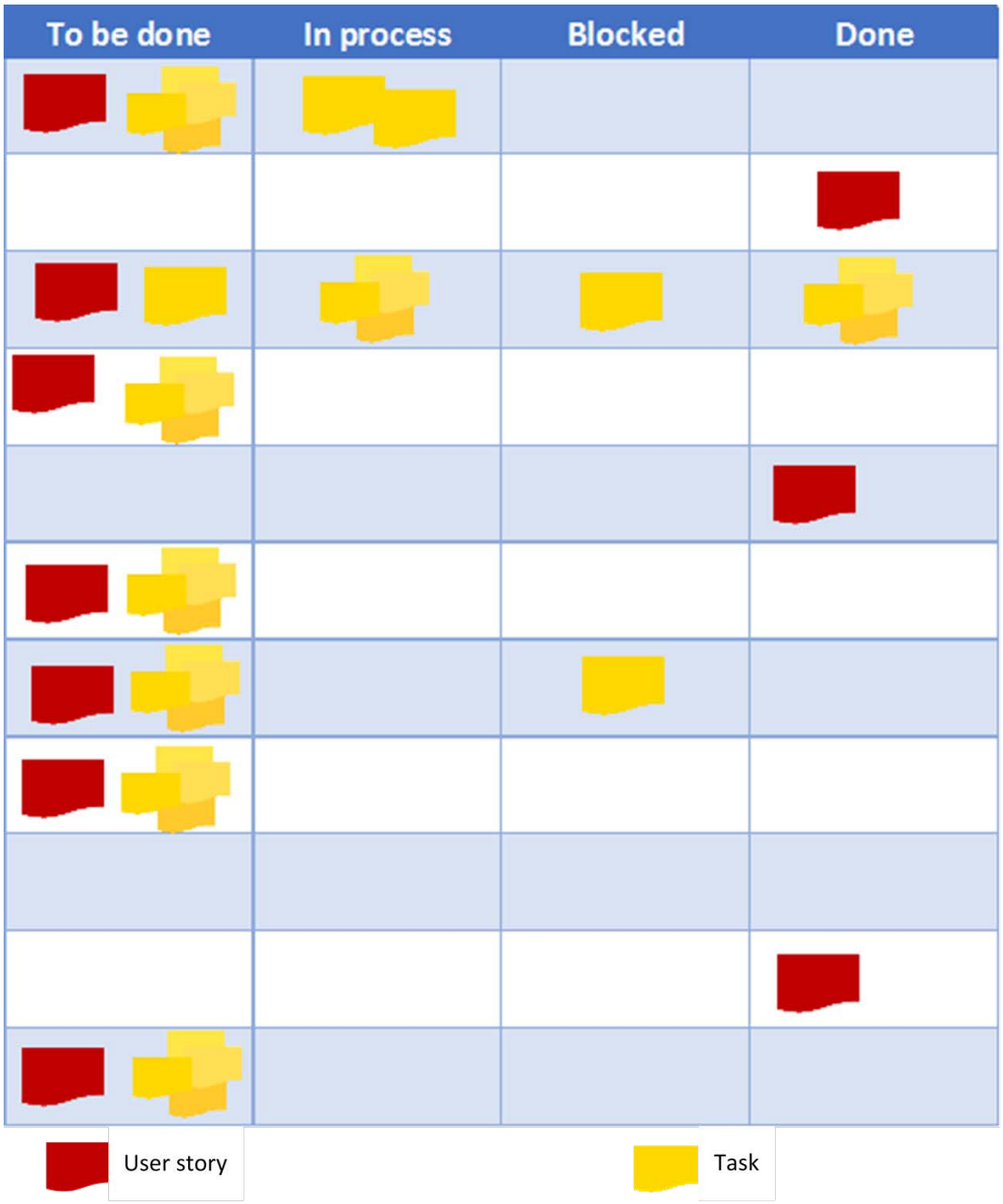
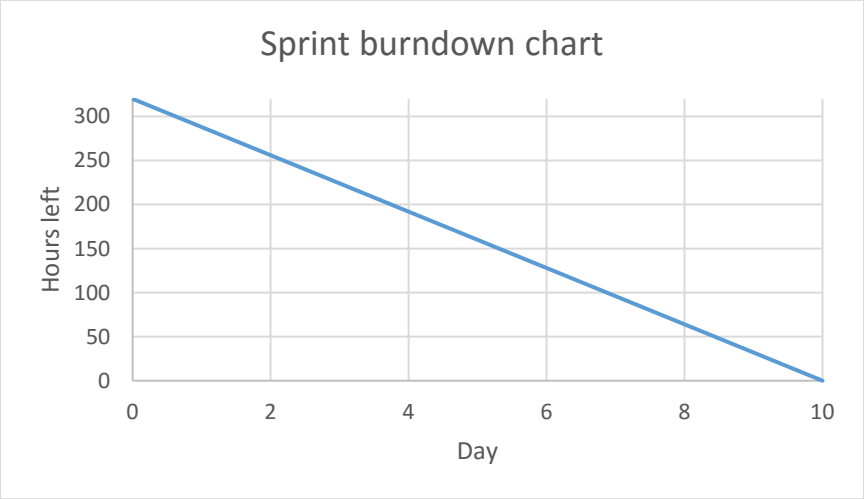


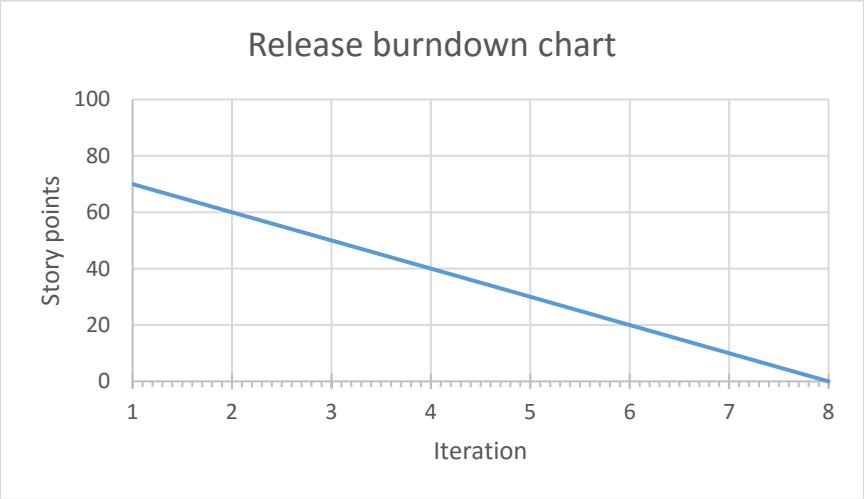
Figure 4 Scrum task board



The initial number of hours left will correspond to the total effort required to execute the sprint backlog in its entirety plus all other schedule activities such as daily meetings, sprint review meeting, retrospective meeting, backlog grooming meetings, etc.

Level effort tasks such as the daily standup meeting will burn down hours at a fixed daily rate equal to the duration of a meeting times the number of members in the team.

Figure 5. Sprint burndown chart



Story points is total for all items in the product backlog. Convert ideals hours into story points to use a single set of curve or use two curves, one for user stories expressed in story points and the other for the technical and knowledge acquisition stories and any other work estimated in hours.

Figure 6. Release burndown chart

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

DELIVERABLE 4: RETROSPECTIVE REPORT

Retrospectives help continuously improve the way of working. In the Scum framework, retrospectives are conducted at the end of each sprint. The purpose of this part of the assignment is to give the team the opportunity to practice this activity. The result of the retrospective should be submitted as a report that includes the following items:

a. Release burndown chart

b. Sprint burndown chart

c. Velocity, Efficiency & quantity of unfinished work

d. Insights. Think about the difficulties you experienced and your "aha!" moments. Were the difficulties the result of logistics or lack of knowledge, or was something intrinsic to the methods? How could you overcome those? What did you learn? When? What were your beliefs before the exercise? What is the significance?

e. Recommendations for the team

f. Recommendations to the instructor

APPENDIX A: SPRINT SIMULATION

INTRODUCTION

This appendix describes the materials and rules necessary to simulate the execution of a Scrum Sprint. Depending on the educational goals and the available course time, the sprint duration can be set to between two and four weeks. shorter Sprints are not recommended, as it is unlikely team members will be exposed to the full range of situations captured in the simulation. A simulation session, with a two-week sprint and four participating teams can be run in approximately 1.5 to 2 hours.

The exercise consists in the selection, by members of the team, of tasks for execution from a previously planned sprint backlog and their completion according to the dictates of two prize wheels: the *Events Wheel* and the *Progress Wheel*, see Figure 7, and the rules of the simulation.

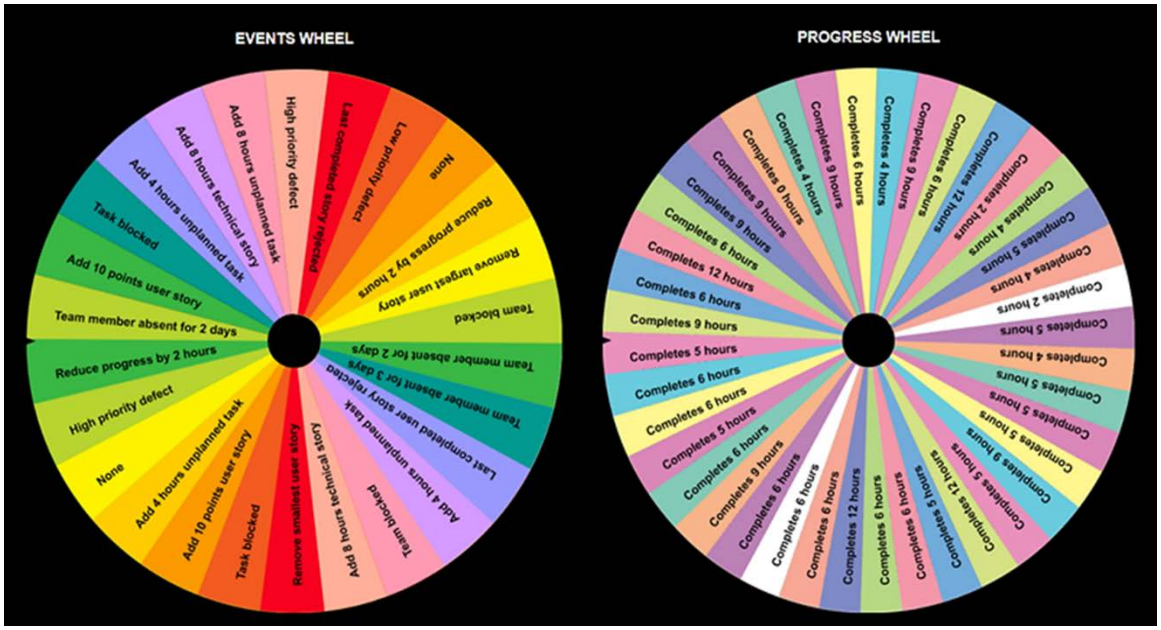


Figure 7 The Events Wheel determines events to which the team must respond while the Progress wheel dictates how much progress each team member accomplished in one day of work

Figure 8 illustrates the basic workflow. Each simulation cycle corresponds to one day in the life of the sprint. Each cycle starts with the spinning of the Events Wheel and concludes with the updating of the burndown charts by the Scrum Master.

Each team's objective is to complete as many story points as planned at the minimum cost over the duration of the Sprint. The *Velocity* and *TeamEfficiency* metrics, see Table 4, will be used to determine the wining team.

Each team will bring its own task board and burndown charts, see figures 3 to 6, to manage its work and report progress.

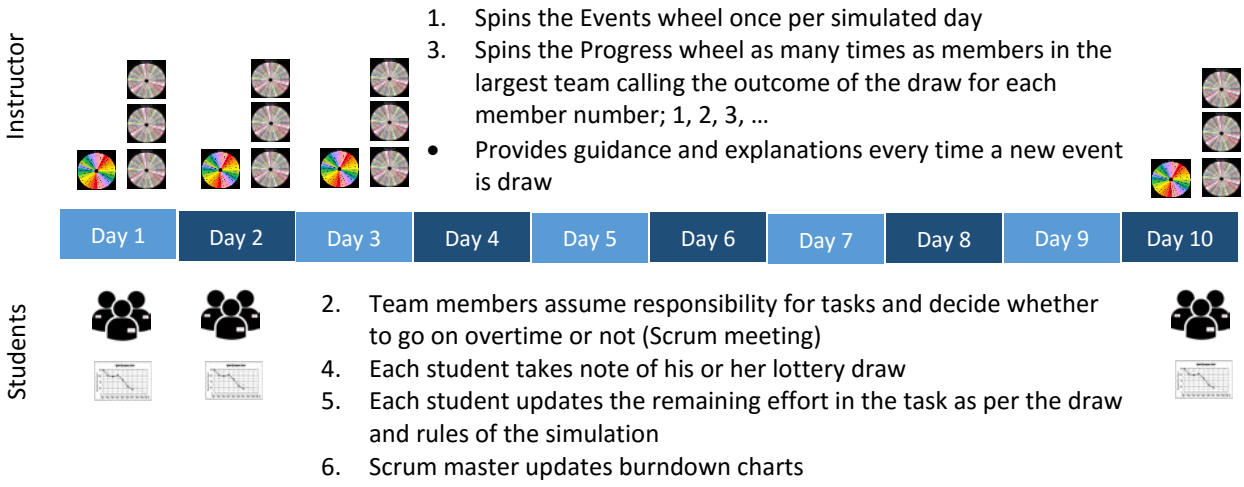


Figure 8. Simulation workflow for a two weeks Sprint

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

DEFINITION OF TERMS

This section defines terms used in the rest of the text. Please be sure you are familiar with them.

Table 4. Definitions

IdealEffort	The effort estimated by the team during the sprint planning meeting or, in the case of technical stories, before it. It is the effort required to complete a task by a single, fully dedicated developer
AllocationAdjustment	The amount of effort by which the *IdealEffort* or the *WorkRemaining* should be adjusted to reflect factors that affect the original estimate such as : familiarity of the assignee with the task, the number of people involved in its execution, ramp-up efforts caused to a late assignee. These adjustments are applied only when a new assignee joins a task, whether at its beginning or halfway through it
Progress	A measure of the degree of advance achieved in the realization of one or more tasks, expressed in terms of the budget planned for them. Indicated by the draw of the Progress Wheel. The Progress achieved in any one day is measured in hours and could be higher, lower or equal to the number of actual hours worked. For example working in a task that resulted simpler than expected, will result in a higher progress. By the contrary, working in a task that ended being more complex than anticipated result in less progress than the number of hours put in
ProgressAdjustment	The amount by which the Progress effort draw should be adjusted to compensate for particular circumstances of each individual task assignee, such as multitasking and overtime working. These adjustments, if applicable, are performed every simulation cycle
AdjustedProgress	The sum of *Progress* and *ProgressAdjustment*
WorkRemaining	The effort left in the execution of a task $WorkRemaining_0$ <- *IdealEffort* + *AllocationAdjustments* $WorkRemaining_k$ <- $WorkRemaining_{k-1}$ − $AdjustedProgress_k$
HoursWorked	The number of hours worked by the team. Includes normal and overtime hours
Velocity	The number of *story points* completed by the team in the iteration
TeamEfficiency	A measure of the team efficiency in terms of the output achieved in a Sprint in relation to the resources put into it. Defined as the sum of all IdealEffort estimates for the stories completed during the iteration divided by the number of HoursWorked

Absent team member	A team member who draw an absent status in the Events Wheel. Team members with an absent status cannot take on tasks or contribute to their progress until their absent status expires.
Active team member	A team member who doesn't have an *Absent* status
Overtime mode	A state in which team members contribute longer work hours. The team makes the decision to enter or exit the Overtime mode at the daily Scrum meeting. If the decision is made, it applies to all team members. Overtime is specially expensive since it incurs significant process losses

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

INSTRUCTOR SCRIPT

1) Be sure to bring markers, self-adhesive charts, self-adhesive name tags and printed simulation instructions for the teams
2) Teams organize themselves in the classroom and put the task board and charts on the wall
3) Have a dry run to verify students understanding on how to play the simulation
4) For each simulated day in the spring
    a. Spin the Events Wheel and announce the result which applies to all teams equally
    b. Spin the Progress wheel as many times as students in the larger group. The outcome of each draw applies to all team members with the same identification number across teams, e.g. the first draw applies to all team members whose identification number is one, the second to those whose identification number is two and so forth
5) For each event that has not been draw before the instructor asks the team what should they do and why, clarifying if necessary
6) At the end of the class, the instructor facilitates discussion of the experience

SAMPLE QUESTIONS TO USE AT THE END OF THE CLASS TO PROMOTE CRITICAL THINKING

Some of the questions below might not be applicable, as the corresponding event was never draw in the simulation.

- Did you use overtime? Why? Why not?
- How did you show the cancelled user stories in the release burndown chart? Did this cause any problems?
- In hindsight, would you handle blocking tasks in a different way?
- How did you handle the request for a technical story? Did you include it on the product backlog? Did you include it in release burndown chart? How did you go from ideal effort to story points? What would be the argument for doing this?
- Should we include the technical story in the calculation of the velocity?
- Should the hours of work invested in a rejected user story included in the Efficiency calculations?
- How did you handle the unplanned tasks?
- Why do we need to adjust the ideal effort when more than one person is allocated to a task?

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

STUDENT SCRIPT

1) Each team member adopts a sequential identification number within his team starting with one, writes it into a self-adhesive nametag and attaches it to his clothes for others to see. One of the members is designated Scrum Master by the team. See Figure 9.

2) Only active team members participate in the activities below. Team members with an absent status cannot take on tasks or contribute to their progress until their absent status expires

3) For each day in the sprint

   4) The instructor spins the *Events Wheel.* Team members take note of the draw. The outcome applies to all teams according to the rules defined in the *Event Definition List*

   5) If the outcome of the draw was *"Team Blocked"*, the team loses a day, go to step 10

   6) ***Scrum meeting***. Each team holds its own daily stand-up meeting to discuss work status, decide what tasks tackle next, who should work on what and whether or not, to go into *overtime mode*. In making these decisions, the teams must consider the impact on productivity of different task allocation schemes, as described by the *Effort Adjustment Rules for Task Allocations*. After doing this, the *WorkRemaining* for the tasks to be worked on is calculated per formulas below and newly started tasks are moved from the *"To Be Done"* to the *"In Process"* column.

      a. Tasks starting in this cycle: $WorkRemaining_0 \leftarrow Max(IdealEffort + AllocationAdjustment, 1)$

      b. Tasks whose allocation has changed in this cycle: : $WorkRemaining_k \leftarrow Max(WorkRemaining_{k-1} + AllocationAdjustment, 1)$

   7) The instructor spins the *Progress Wheel as many times as there are members in the largest team. After each draw, the instructor announces the number of the team members to whom it applies. Each* **Team member takes note of the** *Progress* that corresponds to him and waits until all draws have been called out.

   8) The draw of the *Progress Wheel* dictates how much progress a team member accomplished in one day of work. To account for special circumstances, such as whether team members are working overtime or assigned to multiple tasks, each team member will adjust his *Progress* in accordance with the *Effort Adjustment Rules for Progress*

$$AdjustedProgress \leftarrow Max(\ Progress + ProgressAdjustment, 0)$$

   9) Each team member will now choose how to allocate his *AdjustedProgress* to the task or tasks he is working on. Team members cannot apply their *AdjustedProgress* opportunistically, i.e. to tasks they were not allocated to during a Scrum meeting. If the team members decide to apportion some of the *AdjustedProgress* towards the completion of group tasks, all members should apply the same number of hours

      a. Working on Single task: $WorkRemaining_k \leftarrow Max(WorkRemaining_{k-1} - AdjustedProgress, 0)$

      b. Working on multiple tasks: $WorkRemaining_{i,k} \leftarrow Max\big(WorkRemaining_{i,k-1} - W_i \times AdjustedProgress, 0\big),\ \sum W_i = 1$

      The weights W_i are chosen by each team member at his discretion.

10) ***Tracking***. At the end of the simulated day, the Scrum Master looks at what happened during the day and adjusts the Release and the Sprint burndown charts in accordance with the *Reporting Rules*. It is a good practice to keep a log of significant decisions such as going into overtime or extraordinary events such as a team member being absent for a couple of days to use in the reflection meeting

11) ***Closing***. After the Sprint is over, the team calculates the team's *Velocity* and the *TeamEfficiency*

12) The Scrum master presents the results to the class

13) The winner team is the one that achieves maximum *TeamEfficiency*

14) All collected information must be preserved to be used in the reflection meeting

Team A

Team B

Figure 9 All participants with the same identification number, irrespective of the team to which they belong, are affected by the same wheel draw

SIMULATION RULES

GENERAL

All rules are cumulative. This means, that when more than one rule applies, they all should be applied in succession.

The values adopted in the simulation are chosen for convenience and do not reflect any particular study. This being said, they are better than "0", which in the words of J. Forrester, is the only value known to be wrong[1]

[1] *There seems to be a general misunderstanding to the effect that a mathematical model cannot be undertaken until every constant and functional relationship is known… This often leads to the omission of admittedly highly significant factors…because these are unmeasured or unmeasurable. To omit such variables is equivalent to saying they have zero effect—probably the only value that is known to be wrong!* Forrester, J. W., Industrial Dynamics, Cambridge, MA: MIT Press, 1961

EVENTS DEFINITION LIST

The following rules affect the team as a whole. They reflect, for example, changes in scope, events outside the control of the team or in some cases bad planning.

Rule 1	**Team blocked**	
Condition	*Events Wheel* draw was *"Team blocked"*	
Exceptions	None	
Rationale	The team makes no progress on any of the current tasks	
Examples	• The team gets called to an unscheduled offsite meeting • The discovery of a design limitation forces an unplanned refactoring • A system meltdown prevents the team from working or forces it to recover the software from backup devices	
Calculation	None	
Actions	None	

Rule 2	**Member absent**	
Condition	*Events Wheel* draw was *"Team member absent for 2 days"* or "*Team member absent for 3 days"*	
Exceptions	None	
Rationale	People gets sick or has unexpected problems	
Examples	• A team member gets the flu • The daughter of one of the team members needs to be cared for	
Calculation	None	
Actions	The team member with the largest current task will not contribute to the progress of the team for the next *n* days. After that, he or she returns to where it left or picks up a new task. Each team member is responsible for keeping track of his or her absentee days.	

Rule 3	**Task blocked**	
Condition	*Events Wheel* draw was *"Task blocked"*	
Exceptions	None	
Rationale	Some work cannot be completed until other work is done or an external supplier provides us a needed good or service	
Examples	• A library needs to be installed • A software module needs to be completed before others can be integrated	
Calculation	None	
Actions	All current tasks, but the largest are put on hold. Team members can decide to swarm to the blocking task, take on new tasks or wait. The blocked tasks will be returned to the "in process" state once the blocking task has been completed or the next day, whatever is longer	

Rule 4	**Requirements change**
Condition	*Events Wheel* draw was one of the following: *"Add 10 points user story", "Remove the smallest user story", "Remove the largest user story"*
Exceptions	None
Rationale	Practice updating the product and sprint backlog in reaction to changes. The team response depends on whether the smallest or largest user stories are scheduled for the current iteration or not. Adding user stories to the product backlog will not affect the sprint backlog since one should not introduce changes to it during its execution, with the exception of stopping work that is no longer needed
Example	Self-explanatory
Calculation	None
Actions	Update the product and sprint backlog as necessary

Rule 5	**Story rejected**
Condition	*Events Wheel* draw was *"Last completed user story rejected"*
Exceptions	None
Rationale	The product owner rejects the last completed user story
Example	Self-explanatory
Calculation	None
Actions	The application of this rule is delayed until the last cycle of the sprint to coincide with the timing of the Demonstration Meeting. The last completed story is returned to the product backlog

Rule 6	**Defect reported (Low and high priority)**
Condition	*Events Wheel* draw was *"Defect reported"*
Exceptions	None
Rationale	A user detects a defect after the system went into production
Example	• High priority, the system rejects payments made with the "Discovery" credit card • Low priority, several employees detected that the font employed in two different screens is not the same
Calculation	None
Actions	Fixing either defect will require six hours. The action to take will depend on the defect urgency. High priority defects are likely to preempt other work being performed. Low priority work will be registered in the product backlog and prioritized against other work

Rule 7	**Technical stories**
Condition	*Events Wheel* draw was *"Add 8 hours technical story"*
Exceptions	None
Rationale	Practice dealing with mixed units in the product backlog
Example	It is becoming clear the current design is not scaling up and a refactoring of the database would be needed before implementing the user stories planned for the next Sprint
Adjustments	None
Actions	Update as necessary

Rule 8	**Unplanned work**
Condition	*Events Wheel* draw was *"Add X hours unplanned task"*
Exceptions	None
Rationale	Practice updating the sprint burndown chart in response to the realization by the team it needs to perform some work it did not plan for. Unplanned work does not result in changes to the release burndown chart, since it does not contribute "additional value" to what was already planned. It just reflects poor planning or lack of knowledge on the part of team
Example	• The functionality provided by a third party library is not what was assumed • It is necessary to modify a data structure to increase the performance of an application
Calculation	None
Actions	An X hour task is added to an *"In process"* user story. It doesn't matter which one, as long as it is one which is being worked on

EFFORT ADJUSTMENT RULES FOR TASK ALLOCATION

These rules affect either the *IdealEffort* estimated for each task or the *WorkRemaining*, depending on whether somebody joins a task at its beginning or after it was started. The rules capture things like the extra coordination effort required when you have more than one person working on a task or the need to catch-up with the work already done, when somebody joins a task at a time other than the start.

Rule 9	**Familiar work**	
Condition	Team member takes on a new task within a parent story in which he or she had already worked	
Exceptions	This rule can only be applied once per task	
Rationale	If a team member takes on a new task within a parent story in which he has been working, there will be some productivity increase because of the familiarity with the task and in consequence, the originally estimated effort ought to be reduced by some amount. The influence of the experience factor will be higher if applied at the beginning of the task	
Example	The realization of a user story requires executing tasks, A, B and C. Initially Tom works on task A. Later in the Sprint, Tom takes on task C, whose *IdealEffort* is 20 hours. Since the task has not yet been started, we use the first rule and the *Adjustment* would 4 hours to reflect Tom's experience Anna, another team member, has been working on task B for a while but still, has 16 more hours of work to do. In order to expedite the task, Tom who had experience working on A, joins Anna. In this case, because the task has already started we apply the second rule. As *WorkRemaining* is 16 hours, the *Adjustment* would be 1.60 hours Finally, if Anna and Tom had both been familiar with the task, the adjustment would have been 4 hours in both of the above scenarios because the rule can only be applied once per task	
Calculation	$\begin{cases} -0.2 \times IdealEffort \\ -0.1 \times WorkRemaining_{k-1} \end{cases}$	
Actions	None	

Rule 10	**More than one person**	
Condition	A task is allocated to two or more team members	
Exceptions	Group tasks such as meetings or backlog grooming are except from this adjustment, as their estimates shall already account for process losses	
Rationale	When more than one person works in a task, we could expect some productivity loss due to things such as extra communication and social loafing. Assuming the estimated effort for the task corresponded to the effort of one person working alone, this ought to be increased by a certain amount	
Examples	Anna and Dave start working on task D, whose IdealEffort is 30 hours. The adjustment would be 6 hours To speed up the task, with a *WorkRemaining* of 20 hours; Tom joins Anna and Dave and the adjustment is 2 hours.	
Calculation	$\begin{cases} +0.1 \times IdealEffort \times \#Persons \\ +0.1 \times WorkRemaining_{k-1} \times \#AdditionalPersons \end{cases}$	
Actions	None	

Rule 11	**Late joining**	
Condition	A team members take over or helps with a task already started by another team member	
Exceptions	None	
Rationale	If one or more team members take over or help with a task already started by another team member, there will be an extra effort associated with the ramp-up period of the new contributors. That extra effort is proportional to the work already done.	
Example	Anna and Dave decide to help Tom with task D, whose cumulative progress on it has been 10 hours and in consequence the adjustment will be 1 hour total	
Calculation	$+0.1 \times (WorkRemaining_0 - WorkRemaining_{k-1})$	
Actions	None	

These rules capture particular situations that affect the amount of progress achieved by each member of the team. The situations considered include an event that affects all members of the team, multitasking and the use of overtime.

Rule 12	**Common problem**
Condition	*Events Wheel* draw was "*Reduce progress by 2 hours"*
Exceptions	None
Rationale	This is a team event, like a system crash, a general meeting called by the division head or any other external event that affects all members of the team
Example	Each team member will reduce the amount of Progress indicated by the wheel draw by 2 hours in addition to any other adjustment that might apply
Calculation	$-2\ hours$
Actions	None

Rule 13	**Multitasking**
Condition	Team member is working on multiple tasks
Exceptions	Blocked tasks and group tasks do not count as additional tasks
Rationale	If the team member is working in multiple tasks, other than group activities, we need to account for the productivity losses that occur when switching context between tasks
Examples	• A team member that is working on two tasks draws a *"Progresses 9 hours"*, the *AdjustedProgress* will be 8.5 hours • A team member that is working on an task and also allocated to a group ask, draws a *"Progresses 9 hours",* the *AdjustedProgress* would be 9 hours as group tasks do not count as additional tasks
Calculation	$-0.5\ hours \times (\#tasks\ he\ is\ working\ on - 1)$
Actions	None

Rule 14	**Overtime**
Conditions	Team has made the decision to work overtime
Exceptions	None
Rationale	It is well known that overtime hours are not as productive as normal hours, not only because as fatigue settles in we do less but also because we are likely to introduce more errors which need to be fixed later at a higher cost. So in general sustained overtime should be used sparingly. To simplify the simulation we assume that if the team has chosen to employ overtime, it will do 4 hours of overtime per day per team member and that the effective contribution will be around 75% of a normal hour and proportional to the complexity of task as reflected by the amount of progress draw. If one wanted to make the return of overtime independent of the complexity of the task one would need to add a certain number of hours to *Progress* rather than multiplying for a factor
Examples	• A team member draws a *"Completes 9 hours"* in the *Progress Wheel* while the team is working overtime, the amount of extra progress to be applied against the task or tasks he is working on would hours $\lceil 9 \times 0.25 \rceil = 3 \text{ hours}$ • A team member draws a *"Completes 5 hours"* in the *Progress Wheel* while the team is working overtime, the amount of extra progress to be applied against the task or tasks he is working on would be $\lceil 5 \times .25 \rceil = 2 \text{ hours}$
Calculation	$Ceiling(Progress \times 0.25)$
Actions	None

REPORTING RULES

These rules applies to the updating of the release and sprint burndown charts.

Rule 15	**Task completion**
Condition	A task is completed, i.e. its *WorkRemaining* reaches 0
Exceptions	None
Rationale	Practice updating the task board
Examples	A task whose IdealEffort was 8 hours is completed in 5. The number of hours left in the sprint burndown chart is reduced by 8 hours. A task whose *IdealEffort* was 10 hours took 15 hours to complete. The number of hours left in the sprint burndown chart is reduced by 10 hours.
Calculation	None
Actions	The completed task is moved to the *"done"* column in the task board

Rule 16	**User story completion**
Condition	All tasks necessary for the realization of a user story have been completed
Exceptions	None
Rationale	Practice updating the task board
Examples	N/A
Calculation	*None*
Actions	The completed user story is moved to the *"done"* column in the task board. In a real project, this would involve checking all marks included in the Definition Of Done

Rule 17	**Level of effort**
Condition	All level of effort tasks
Exceptions	None
Rationale	Practice updating the sprint burndown chart in response to work that is performed for as long as the sprint lasts, e.g. Scrum meetings, quality assurance and other non-discrete activities.
Examples	During the Sprint planning meeting the team allocated 10 hours for Scrum meetings and 5 hours for backlog grooming
Calculation	$\sum_{All\ level\ of\ effort\ tasks} EstimatedEffort \Big/ Length\ of\ the\ Sprint$
Actions	None

Rule 18	**Product backlog changes**
Condition	User stories additions, rejections, completions, modifications and deletions
Exceptions	None
Rationale	Discriminate between changes to the scope of the project as perceived by the product owner and the developers. Practice updating the release burndown chart.
Examples	N/A
Calculation	*None*
Actions	Update the release burndown chart

Rule 19	**Sprint backlog changes**
Condition	Tasks additions, deletions, completions, and depletion of level of effort tasks
Exceptions	None
Rationale	Discriminate between changes to the scope of the project as perceived by the product owner and the developers. Practice updating the sprint burndown chart
Examples	N/A
Calculation	*None*
Actions	Update the sprint burndown chart

CLOSING

These rules define the calculations to be applied at the end of the sprint

Rule 20	**Velocity**
Condition	End of the sprint
Exceptions	None
Rationale	Velocity is a key metric in Scrum. It has many uses, e.g. determine how much work the team can tackle in a sprint, measure team stability, learning curves, etc.
Examples	N/A
Calculation	$\sum_{All\ user\ stories\ completed\ during\ the\ Sprint} EstimatedSize$
Actions	None

Rule 21	**Efficiency**
Condition	End of the Sprint
Exceptions	None
Rationale	Analyze the use of resources that went into achieving whatever was achieved
Examples	A team of three people completes tasks for 200 hours of planned effort over a two weeks Sprint without using overtime. $$Efficiency = \frac{200hrs}{240hrs} = 83\%$$ A team of three people completes tasks for 200 hours of planned effort over a two weeks Sprint using 3 days of overtime. $$Efficiency = \frac{200hrs}{240hrs + 36hrs} = 72\%$$
Calculation	$\begin{cases} No\ overtime, & hours\ worked = 8\frac{hrs}{day} \times team\ size \times Sprint\ lenght \\ Overtime, & hours\ worked = \left[\left(8\frac{hrs}{day} \times Sprint\ lenght\right) + \left(4\frac{hrs}{day} \times Number\ of\ overtine\ days\right)\right] \times team\ size \end{cases}$ $Efficiency = \frac{\sum_{All\ completed\ tasks} IdealEffort}{hours\ worked}$
Actions	None

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

EVENTS DEFINITION

This table defines the events that affect the team as a whole. The relative frequencies of the events do not correspond to real data; they are designed for the team to experience events of different kind in the course of single simulation run.

Event	Relative frequency	Rationale for inclusion	Rule
Team blocked	2	No progress in any of the current tasks. The purpose of this event is to simulate an event that prevents all team members for completing any work	1
Task blocked	2	Sometimes work cannot be completed until some other work is done or an external supplier provides us a needed good or service	3
Add 10 points user story	2	New work is added to the product backlog	4
Remove the smallest user story	1	Scope reduction	4
Remove the largest user story	1	Scope reduction	4
Low priority defect reported	1	Defect detected after software is deployed	6
High priority defect reported	2	Defect detected after software is deployed	6
Add 8 hours technical story	2	Add no-direct user value work	7
Reduce progress by 2 hours	2	The purpose of this event is to simulate correlated task durations, e.g. every team member is affected by the same cause	12
Add 4 hours unplanned task	3	The team realizes they need to perform a task they did not plan for	8
Add 8 hours unplanned task	1	The team realizes they need to perform a task they did not plan for	8
Customer rejects last completed user story	2	The team got it wrong	5
Team member is absent for 2 days	2	An employee gets sick or needs to take some unplanned time-off. The frequency of this event is not realistic but designed to create some havoc in the project	2
Team member is absent for 3 days	1	An employee gets sick or needs to take some unplanned time-off. The frequency of this event is not realistic but designed to create some havoc in the project	2
None	2	Everything according to plan, no changes, no fights	N/A
Total	26		

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

PROGRESS DEFINITION

This table defines the events that affect individual team members, such as the amount of progress in a task or taking days off. The relative frequencies of the events do not correspond to real data. They are designed for the member to experience one or two events of a kind without being annoying.

Progress achieved	Relative frequency	Rationale for inclusion	Rule
0 hours	1	Progress achieved represents the proportion of work done in terms of the budget established for the task independently of the number of hours on task. Cero progress indicate the team member, for whatever reason did not get anything done on the work he was supposed to do	9-14
2 hours	2	These values mean that the team member was either sidetracked by other work or the work was more difficult than anticipated and so the pace of progress was slower than what would be normal in an 8 hour work day	
4 hours	5		
5 hours	9		
6 hours	12	This value corresponds to a realistic estimate of what is achievable in an 8 hours work day	
9 hours	7	These values correspond to a situation in which the task was less complex than anticipated and so the pace of progress was faster than what would be normal in an 8 hour work day	
12 hours	4		
Total	40		

APPENDIX B. DATA FOR THE CREATION OF THE PRIZE WHEELS – STUDENTS DO NOT NEED TO WORRY ABOUT THIS

Data to create Events Wheel (wheel diameter= 800, light colors)	Data to create Progress Wheel (wheel diameter = 800, pastel colors)
Team member absent for 2 days	Completes 9 hours
Add 10 points user story	Completes 0 hours
Task blocked	Completes 4 hours
Add 4 hours unplanned task	Completes 9 hours
Add 8 hours technical story	Completes 6 hours
Add 8 hours unplanned task	Completes 4 hours
High priority defect	Completes 9 hours
Last completed story rejected	Completes 6 hours
Low priority defect	Completes 12 hours
None	Completes 2 hours
Reduce progress by 2 hours	Completes 4 hours
Remove largest user story	Completes 5 hours
Team blocked	Completes 4 hours
Team member absent for 2 days	Completes 2 hours
Team member absent for 3 days	Completes 5 hours
Last completed story rejected	Completes 4 hours
Add 4 hours unplanned task	Completes 5 hours
Team blocked	Completes 5 hours
Add 8 hours technical story	Completes 5 hours
Remove smallest user story	Completes 9 hours
Task blocked	Completes 5 hours
Add 10 points user story	Completes 12 hours
Add 4 hours unplanned task	Completes 5 hours
None	Completes 6 hours
High priority defect	Completes 6 hours
Reduce progress by 2 hours	Completes 12 hours
	Completes 6 hours
	Completes 6 hours
	Completes 6 hours
	Completes 9 hours
	Completes 6 hours
	Completes 5 hours
	Completes 6 hours
	Completes 6 hours
	Completes 5 hours
	Completes 9 hours
	Completes 6 hours
	Completes 12 hours
	Completes 6 hours
	Completes 9 hours

You can use the urls below to instantiate the Events and Progress wheels ready to use. Good as of July 15, 2017

Agile Development Frameworks: Theory

CMU SE 17604, Eduardo Miranda

Assignment 1: Defining, planning and executing a Scrum project

Due: Dates shown on syllabus

EVENTS WHEEL

HTTP://WHEELDECIDE.COM/INDEX.PHP?C1=TEAM+MEMBER+ABSENT+FOR+2+DAYS&C2=ADD+10+POINTS+USER+STORY&C3=TASK+BLOCKED&C4=ADD+4+HOURS+UNPLANNED+TASK&C5=ADD+8+HOURS+TECHNICAL+STORY&C6=ADD+8+HOURS+UNPLANNED+TASK&C7=HIGH+PRIORITY+DEFECT&C8=LAST+COMPLETED+STORY+REJECTED&C9=LOW+PRIORITY+DEFECT&C10=NONE&C11=REDUCE+PROGRESS+BY+2+HOURS&C12=REMOVE+LARGEST+USER+STORY&C13=TEAM+BLOCKED&C14=TEAM+MEMBER+ABSENT+FOR+2+DAYS&C15=TEAM+MEMBER+ABSENT+FOR+3+DAYS&C16=LAST+COMPLETED+STORY+REJECTED&C17=ADD+4+HOURS+UNPLANNED+TASK&C18=TEAM+BLOCKED&C19=ADD+8+HOURS+TECHNICAL+STORY&C20=REMOVE+SMALLEST+USER+STORY&C21=TASK+BLOCKED&C22=ADD+10+POINTS+USER+STORY&C23=ADD+4+HOURS+UNPLANNED+TASK&C24=NONE&C25=HIGH+PRIORITY+DEFECT&C26=REDUCE+PROGRESS+BY+2+HOURS&COL=LIGHT&T=EVENTS+WHEEL&TIME=10&WIDTH=800

PROGRESS WHEEL

http://wheeldecide.com/index.php?c1=Completes+9+hours&c2=Completes+0+hours&c3=Completes+4+hours&c4=Completes+9+hours&c5=Completes+6+hours&c6=Completes+4+hours&c7=Completes+9+hours&c8=Completes+6+hours&c9=Completes+12+hours&c10=Completes+2+hours&c11=Completes+4+hours&c12=Completes+5+hours&c13=Completes+4+hours&c14=Completes+2+hours&c15=Completes+5+hours&c16=Completes+4+hours&c17=Completes+5+hours&c18=Completes+5+hours&c19=Completes+5+hours&c20=Completes+9+hours&c21=Completes+5+hours&c22=Completes+12+hours&c23=Completes+5+hours&c24=Completes+6+hours&c25=Completes+6+hours&c26=Completes+12+hours&c27=Completes+6+hours&c28=Completes+6+hours&c29=Completes+6+hours&c30=Completes+9+hours&c31=Completes+6+hours&c32=Completes+5+hours&c33=Completes+6+hours&c34=Completes+6+hours&c35=Completes+5+hours&c36=Completes+9+hours&c37=Completes+6+hours&c38=Completes+12+hours&c39=Completes+6+hours&c40=Completes+9+hours&col=pastel&t=Progress+Wheel&time=10&width=800

ACKNOWLEDGMENTS:

Thanks to Nicholas Jurgens and Yazid Hamdi, MSE 2017 @ CMU for their review of this document and their valuable comments.