Overview

The purpose of the term project is to follow key phases of software product development. It is a chance to pilot, adopt and implement the best practices covered in the course. In today's environment, collaboration is imperative as it drives the overall effectiveness of a software development project. Peer reviews are a critical activity that allows teams to collaborate on finding defects in each other's work products. After participating in this activity in a controlled environment, you will be able to apply your skills later to a real situation.

Weekly project submissions correspond to the lecture material for that week. For example, the first module covers requirements. So the deliverable for the second module is to write project requirements in Pivotal. It is advisable to start coding as early as possible. A project team is expected to meet weekly. There is a demo expected at a weekly meeting of a project team. A partially-working system with coded modules and prototypes - as early as the second week. The last week is dedicated to testing and bug fixing. Hence coding should be pretty much done by the week before last.

How to select a project scope

When defining a scope for your project, you should feel confident that the specified functionality will be completed on time. Since the timeframe is short, you are discouraged from deep-diving into a new language or a framework concurrently with your weekly submissions. Implementations and languages can vary from a fairly simple website to a complex Java application. The application must be easy to understand. Each stated requirement has to be tested and accepted against specific criteria.

If you decide to add to a project that you have been involved previously, make sure your enhancements are clearly delineated, e.g. new versus old - requirements, screens, db tables, etc. During final demo you will be asked to demonstrate the old features versus new ones.

A Defect Tracking System is a common choice for a term project. A variety of several dozen reports from previous projects are attached to the Discussions tab. See complete list below, including the following,

- PoolMyRide is an app offering to sign up for a carpool between certain destinations
- CMS (Content Management System) publishing and tracking various posts and documents
- Code review repository - maintaining snippets of code along with corresponding comments
- Performance measurement app that creates a graph with a response time, while comparing several options to select the fastest path
- Specialized search engine delving into government site that offers grants to small businesses
- Pet Book site similar to Craig's List that matches pet owners with prospective buyers of pets
- Career Hub site matching qualifications of job seekers with descriptions of open positions

- Shopping cart app optimizing budget, given a list of products with prices and priorities; also provides access to Amazon API Gateway
- K-Know Portal - a knowledge management system similar to a PAL (process assets library)

How to select a team

It is better if folks belonging to the same team have complementary backgrounds. The initial composition of a team is based on a Self Intro form completed by each student before the class starts. As an example from previous class, see table below, reflecting on two projects.

To spread the load, each team member is focusing on a certain type of deliverables.

- Team Lead facilitates meetings and tracks actions
- Senior Developer guides resolution of merge conflicts
- Usability Lead creates wireframes
- Test Expert creates and executes test cases

Teams are small, 4-5 students.

				RE	CM	DESIGN	UX	TEST
				30-Jan	6-Feb	13-Feb	20-Feb	27-Feb
		Team Lead	Component	Personas(RASCI)	CI List	Use Cases	Wireframes	Test cases
				Requirements	Estimation Record	State Transitions		Data-driven table
				PivotalTracker	Tools Connectivity	Components Interraction		
		Project 1 - (XXX)						
1	student 1							
2	student 2							
3	student 3							
4	student 4							
5	student 5							
		Project 2 - (YYY)						
8	student 6							
9	student 7							
10	student 8							
11	student 9							
13	student 10							

Note that creating a team with folks of a complementary backgrounds has many benefits. It facilitates an environment when best practices bubble up and are actually being used. As an example, an engineer of one team has been using the Cloud Foundry at his work. Hence, the whole team was introduced to the cloud-native software development with a huge number of unit tests automatically created. A member of another team happens to be an expert in agile process. Needless to say that his team applied all advanced methods offered by Pivotal Tracker (that are far beyond course material) as tracking velocities of each iteration, and graph reporting on epics' progress.

An additional consideration when selecting team members - it is easier to collaborate with folks who are within a close proximity of each other.

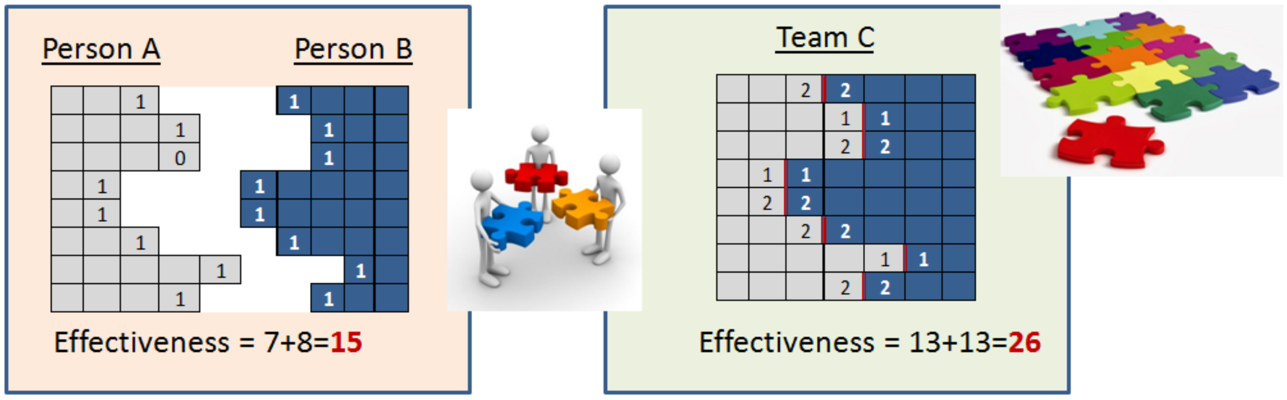
I was thinking how to provide a visual representation of the Aristotle's notion … *the whole is greater than the sum of its parts….* The goal is to follow mechanics of a team collaboration.

First pictorial shows Person A and Person B, with each cell representing a certain activity. Effectiveness of an activity is equal to (1). Although, when they form a Team C and start working together, their effectiveness double.

A good example of such increase is removal defects. Individual DRE (Defect Removal Effectiveness) is about 30%, but one of a group is well-over 60%. Since folks are commonly help each other through complementary skills, alternative angles, long-forgotten associations, etc. It is always easier to find someone else's defects than your own defects.

- Note that collaboration does not improve some activities and their effectiveness will remain equal to (1). An example of such activity is examining several paths of a performance-sensitive routine. Where a focus with a deep concentration could be easily disturbed by other folks. For many software developers, coding is surely an individual activity, as pair programming is out of the question.
- In another scenario, a person is simply unable to accomplish a task by himself. See his effectiveness is marked as (0) . However , collaboration brings effectiveness of both, his and his partner to (2) .

Second pictorial shows cells **touching each other** and their effectiveness… surprise … jumps from (15) to (26).



Schedule and Deliverables

It is best for each team to have a brief weekly demo. Leaving all deliverables to the very end proves to be a flawed strategy. During the weekly demo, all participants should provide comments to the presenter. The last demo during module 6 should be planned with your instructor to make sure all deliverables in the table below are covered.

Module	Deliverable / Objective
1	Register with Pivotal (tutorial); setup project with weekly iterations Create GIT account (tutorial) Start discussing/collaborating using Confluence Propose project scope Propose team's composition Distribute initial report that will be appended with weekly deliverables
2	Document five users (personas) Document Tools Connectivity Diagram Converge on five requirements in canonical form in Pivotal Start coding
3	Provide estimation record Compile the list of configuration items
4	Document four use cases using UML notation Converge on a components interaction diagram Document state transitions

5	Transform mock-ups into wireframes Complete coding
6	Document two (positive and negative) test cases in a standard format Reduce data-driven combinations using Allpairs Submit draft report with all deliverables Final team's presentation of app's functionality

a) To facilitate convergence and cross-pollination among teams, several tools were pre-selected. Each tool has a support function. Boston University IT does not support tools for term projects.
   - Pivotal Tracker is a simple tool used to maintain requirements/stories. You are expected to go through a tutorial first, to become familiar with basics (labels, epics, releases). Pivotal Tracker is web-based-only tool and, so far, does not offer an off-line mode. https://www.pivotaltracker.com/help/gettingstarted
   - Git is the source control tool that received a wide-spread adoption throughout the industry. It facilitates auto / manual merges and code reviews. https://guides.github.com/activities/hello-world/
   - Confluence is a great tool for collaboration, user-friendly discussions, and doc control. Note that there are many other tools available with similar functionality, e.g. Google Hangouts, Slack
   - BalsamIQ is a fitting tool to design and review wire frames. Below is the license information,
     License Name: Boston University 2018
     License Key: eJzzzU/OLi0odswsqnHKLy7Jz1MIzcssSy0qziypVDAyMLSoMTQ1MTazNDExAIGakBpDAOKaEMo=
     For questions refer to : https://support.balsamiq.com/desktop/classroom
     HCD (Human Centered Design) is the methodology we use in our term projects. UI design is inherently iterative process. It is a good idea to demo several mock-ups, to trace and confirm their morphing into screenshots of the final system.
   - Vacava offers a cloud-native environment simplifying your application development. Its product called RapidBIZ helps a variety of customers to rapidly build custom apps with easy to use pre-built components that does not require extensive programming background. There is a Saturday online meeting to respond your questions about RapidBIZ, led by Josh Hanson, who is MET CS facilitator. If you are not proficient in any programming language, e.g. Java, C#, then RapidBIZ might be a perfect fit for you. http://vacava.com/rapidbiz/

b) For this course, you are expected to maintain a flexible project scope. When stories in Pivotal are advanced from Icebox into Current Backlog and then into Done - the scope of the project is adjusted. Note that dragging stories into Current and marking them Accepted is a manual activity, although the advancement of stories into Done is automatic. Pivotal moves stories into those iterations where they have been accepted. The tool is intentionally designed to prevent folks from prematurely advancing stories into Done.

c) It makes sense to start a project with identifying users/personas, which is a sign of a profound respect for those who will ultimately evaluate the outcome of a project.

d) After documenting stories in Pivotal in canonical form, you accept these stories within certain sprints. It is prudent from the very beginning to write stories in such a way so to be able to complete them within the first sprint, second sprint, third sprint, etc. Contrary to such plan, while getting acquainted with a software development process, folks commonly write far too many stories and accept them at the very end of the project. This ultimately results in a lot of rework.

e) While creating use cases, following exact UML notation is not required. Make sure system boundaries have been identified. Provide an example of a use case that is outside of your term project scope.

f) Estimation Record is done using several complementary techniques. Averaging SME's estimates is not recommended; make sure assumptions are in fact resolved. The final report should include actuals.

g) Configuration Items list is the mechanism used to communicate the state of each deliverable. Make sure versions are assigned only to those items that need to be controlled.

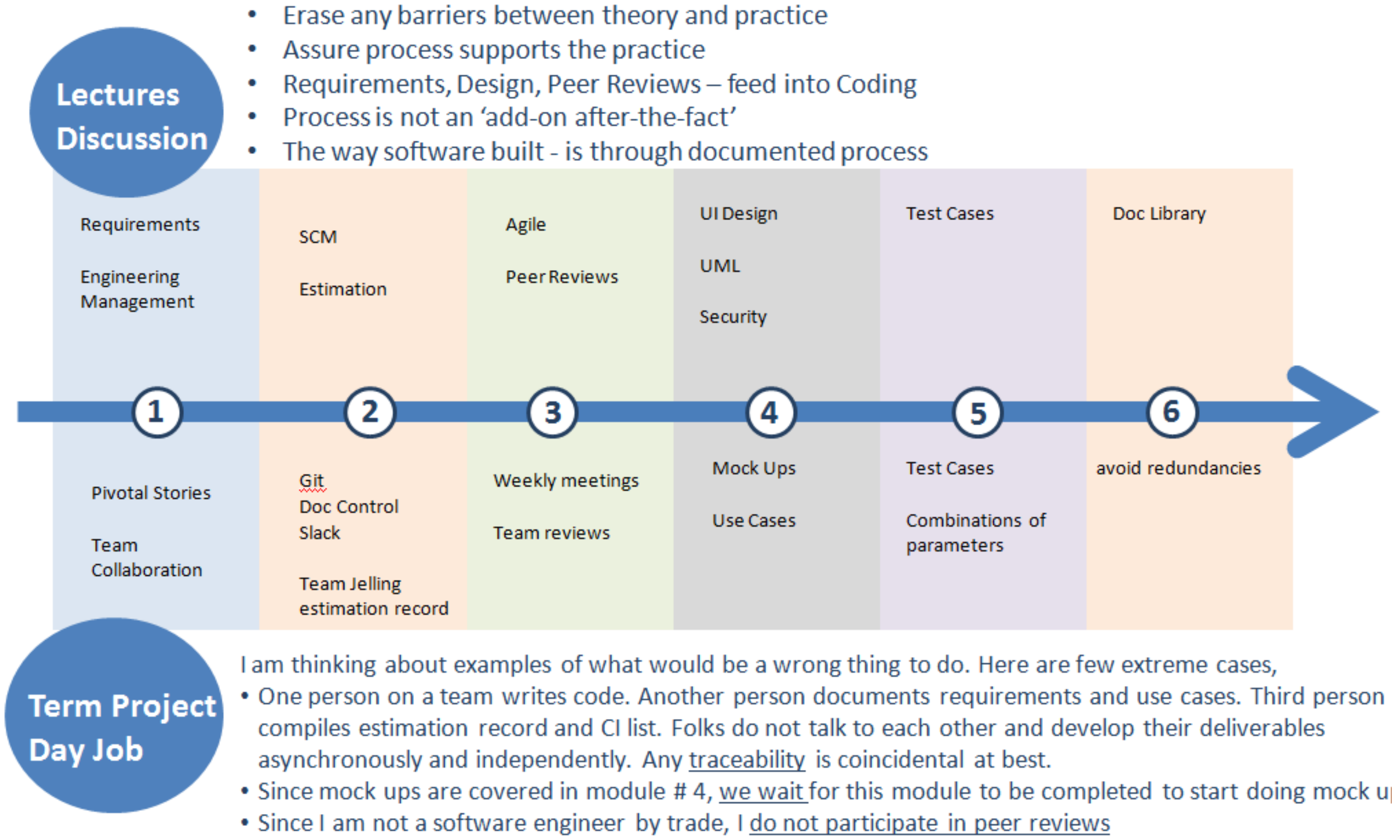
h) Component diagram has many benefits. On one occasion, folks started their final presentation with the component diagram stating who on a team is responsible for which component. It served a dual purpose, first, as a design tool and second, as an agenda for their talk, since each team member covered a certain component per the diagram.

i) Documenting state transitions is part of the system design. It is common that folks compile a far too elaborate set of transitions, although only a part of them find a way into implementation.

j) Data-driven test design is a key concept of this course. During the final demo, you shall examine the logic behind the selection of test cases with the goal of providing adequate coverage.

It is your responsibility to immediately apply, whatever was learned during lectures

- Erase any barriers between theory and practice
- Assure process supports the practice
- Requirements, Design, Peer Reviews – feed into Coding
- Process is not an 'add-on after-the-fact'
- The way software built - is through documented process



I am thinking about examples of what would be a wrong thing to do. Here are few extreme cases,

- One person on a team writes code. Another person documents requirements and use cases. Third person compiles estimation record and CI list. Folks do not talk to each other and develop their deliverables asynchronously and independently. Any traceability is coincidental at best.
- Since mock ups are covered in module # 4, we wait for this module to be completed to start doing mock ups.
- Since I am not a software engineer by trade, I do not participate in peer reviews
- Since peer reviews are covered in module # 3, we wait to learn the perfect process for our first team meeting.
- Since code is developed based on requirements, we wait for requirements to be finalized so to start coding
- We examine ten requirements in Pivotal; then document ten redundant use cases with one-to-one mapping
- When documenting the project's estimation record I did not talk to anyone, the only source of our estimation – are the examples from previous projects

Final Presentation Format

Here are several points toward your final presentation and report.

a) The key part of the presentation is to demo some useful functionality

b) Each team member presents a part of the report focusing on his/her specific tasks

c) The target audience for the final report is C-Level executives, who need to quickly confirm traceability and to make sure all pieces tell the same story so-to-speak.

d) In a spirit of openness of a peer review and sharing best practices - deliverables of term projects are made available to future students. Let us know if you do not want to share your final report.

e) Fifty minutes should be enough for a team presentation, with each member taking 10 minutes and an additional 10 minutes allocated to Q&A.

f) It is important to link the working system with the key concepts covered in the lectures. Here are several examples of statements made during the final demo.

- Looking at the Pivotal, one could see stories in every category. As we started with the wish list (Icebox) and then accepted stories every week. This concept is discussed in the course as "wisdom of vertical slices".
- We used peer reviews to reveal discrepancies between requirements and system behavior. Each of us prepared for the discussion and made a deliberate attempt to clearly communicate our findings. See an example of a comment provided, accepted and incorporated. Apparently, there were no managers among us, so each of us put the best effort that he/she could possibly contribute to the team.
- CI List was quite useful. As we established the list of deliverables at the beginning of the project. It is good to reflect all versions in one place, although items have been stored in different repositories.
- Results of a unit test are not a mandatory deliverable. Still, I would like to show you the defects found with our Python unit test.
- I found the tools connectivity diagram discussed in the class to be most useful for our course project. I also adopted such diagram at my work.

Grading

You will be graded based on the quality and timeliness of your submissions for each phase (requirements, configuration management, estimation, design, implementation, and test) and comments you provide to your peers' work products, on-line and during Adobe Connect meetings.

You shall submit your very final report by the last open day of the course, the day before the exam opens. This submission shall include all completed project artifacts, source and implementation files with comments from peer reviews addressed and test defects are fixed.

Here is an example of grading matrix.

	D	C-	C+	B-	B+	A	Grade
Clarity	Disorganized or hard-to-understand		Satisfactory but some parts of the submission **are** disorganized or hard to understand	Generally organized and clear	Very clear, organized and persuasive presentation of ideas and designs	Exceptionally clear, organized and persuasive presentation of ideas and designs	A-/92
Technical Soundness	Little understanding of, or insight into material technically		Some understanding of material technically	Overall understanding of much material technically	Very good overall understanding of technical material, with some real depth	Excellent, deep understanding of technical material and its interrelationships	A-/92
Thoroughness & Coverage	Hardly covers any of the major relevant issues		Covers some of the major relevant issues	Reasonable coverage of the major relevant areas	Thorough coverage of almost all of the major relevant issues	Exceptionally thorough coverage of all major relevant issues	A/95
Relevance	Mostly unfocused	Focus is off topic or on insubstantial or secondary issues	Only some of the content is meaningful and on topic	All of the content is reasonably meaningful and on-topic	All of the content is entirely relevant and meaningful	All of the content is entirely relevant and meaningful	A-/90
Utilization of resources	No useful use of notes, text(s), Web, or tools, with incorrect details or applicability		Some useful use of notes, text(s), Web, or tools, with mostly correct details or applicability	Fairly good use of notes, text(s), Web, and tools, with correct details or applicability	Very good use of notes, text(s), Web, and tools, with correct details or applicability	Excellent use of notes, text(s), Web, and tools with entirely correct details or applicability	A-/92

Additional Checklist for Final Presentation

In a spirit of the principle discussed in the class of 'staying close to reality,' final demo is where the 'rubber hits the road' and all in-process artifacts, requirements, use cases, and state transitions are expected to match the actual system functionality.

1. The draft report is distributed before the demo and attached to Discussions, giving reviewers enough time to prepare. The link to the working system is distributed, so reviewers are able to play with features. The link to codebase repository is distributed along with the history of commits and merges. The report in Word format that has been submitted during the course, with additional deliverables appended weekly - is now complete. For the last week, all deliverables are assembled. The power point slides are optional and not required.
2. The final report is submitted as an assignment, enabling each team member to receive an individual grade. Final report incorporates those suggested changes that have been accepted. Submitting a single report for the whole team is sufficient.
3. Each in-process deliverable has a version number, e.g. in the right bottom corner. This version number is above <1.0>, meaning that the asset has been released. The asset's version matches the version stated in the CI List.

4. These are eleven configuration items, along with an elaboration on each item
    1) Personas
    2) Requirements
    3) Use cases
    4) Estimation record
    5) Configuration Items List
    6) Tools Connectivity Diagram
    7) Components Interaction Diagram
    8) State Transition Diagram
    9) Wireframes
    10) Test Cases
    11) Data-driven combinations

5. Personas include humans and non-humans. These are the same roles as in Pivotal stories and screenshots. <Question ... what are the advantages of starting a project with definition of users versus starting it with another activity, e.g. screen mockups ? >
6. Requirements in Pivotal in canonical form that have been implemented are marked as 'done'. The acceptance date for each story matches the iterations and project weeks. The 'ice box' is longer ... and includes requirements planned for the next release'. <Question ... was it helpful to use Pivotal and mark accepted stories as 'done', as you advance a project every week, instead of leaving it to the end ?>
7. Use cases are not expected to match requirements one-to-one. < Question ... what new information do use cases provide that is in addition to requirements?> . Looking at the diagram, we should observe several use cases that are outside of boundaries. These scenarios are ...'planned for next release'.
8. Estimation record has been updated to match the actual effort spent. It is best to include two versions of estimation record, showing initial estimate and then the final record that matches actual. There should be meaningful Size Measures. Sizes should not have a total, e.g. adding 'screens' to 'test cases' makes little sense.
9. CI List includes only objects (thingies) and does not include activities (verbs). Peer review comments and test defects do not have a version number.
10. Tools Connectivity Diagram reflects the set of tools you are using on your term project. The assignment # 2 of module # 4 has a connectivity diagram for a project team. Course notes of module # 4 have several examples. A number of reports for completed projects show tools that were used.
11. Components Interaction diagram utilizes MVC pattern, as 'screens' are separated from 'databases' and 'business logic.'
12. State transition diagram has a minimum possible number of states. It should be clear, which states are implemented and which states are outside of scope. We should be able to follow each line of the diagram during the demo. It is best to name states as "Fixed" or "Browsing". Instead of an imperative "Fix" or 'Browse" . There should be a stated condition that enables the transition from one state to another state.
13. Wireframes follow the principles of UI design (HCD) covered in the course.
14. Question .. .what is your rationale when you reduced the number of test combinations? Give an idea of test coverage ... whether all requirements are covered? and all use cases are covered? What is not covered?

15. Demonstrate several defects that were discovered during testing and eventually were fixed.
16. If you were engaged with unit tests, demonstrate those defects originated and fixed. If you were not using unit test, describe how you would do it, if additional project time permitted.
17. Demonstrate several comments from a peer review or static testing - that were provided and corresponding changes incorporated.
18. What are the lessons learned from team collaboration? What collaboration tools were used and whether these tools were effective? Note the common lesson aired during demo ... *we should have not waited but should have jumped into coding from day one*.

Not Required Items

As we are discussing in the sixth module, it is an explicit responsibility of a software engineer and project manager to find the most effective path to delivery. Here is the list of items that are not required, although are commonly found in the final presentation.

- An elaborate description of personas, including color photographs of their children and dogs might be useful to get a warmNfuzzy toward your future users, although is considered - an over-the-top, unnecessary - in the context of our project.
- A Gant chart, WBS with project activities is redundant to Pivotal user stories
- A scope statement or project objective is not required. Again, Pivotal should 'tell the story'.
- To have both, PPT presentation and Word report is redundant. Should select one or another
- Multiple use cases and test cases in a detailed format are not required . One use case and one test case in a detailed format is sufficient. The goal is to demonstrate that you are familiar and could justify these details. The target audience (C-Level executive) should be able to easily navigate and trace through all deliverables.

A Start Up

The term project is a miniature start up. Imagine, you just joined a company that is small now, but ultimately is bound to grow into a sizable enterprise. By the end of the class, you will have a functioning product prototype to be accepted by VCs.

To this end, the whole course could be interpreted as How-To-Start-A-Real-Software-Company. This must be exciting doing it for the first time!

Alex Elentukh experience with several successful start ups is reflected throughout course notes.

- Jupiter Technology, a manufacturer of data communication equipment and protocol converters; sold to Intel.

- Reveal Imaging, manufacturer of explosive detection devices, has been certified by TSA and installed in many airports; sold to SAIC.

List of examples

You are encouraged to learn from previous projects. Here is the content of the tab 'Term Project, Examples from previous classes'.

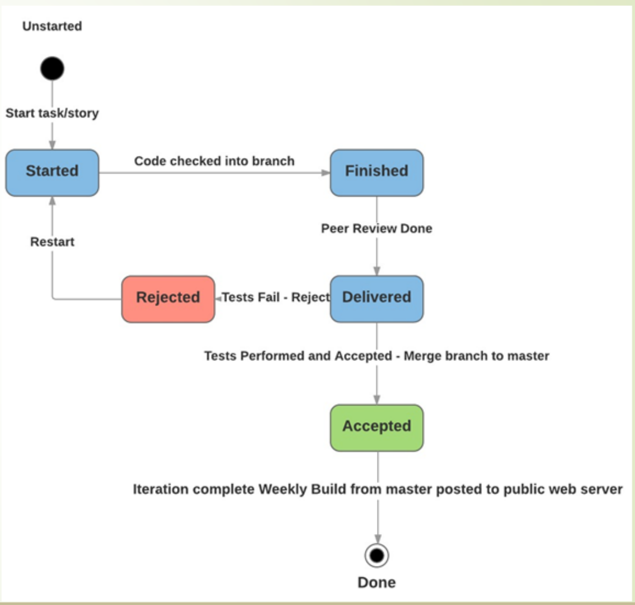
	Topic	Course	Date
1	Asset Management System	CS633 EL	Fall 2017
2	Water Bottle Filling Station	CS 633 EL	Fall 2017
3	Health and Wellness	CS 633 OL	Spring 2017
4	sEATs	CS 633 OL	Spring 2017
5	ShipIT	CS 633 OL	Spring 2017
6	GroupDate	CS 633 OL	Spring 2017
7	Excusinator	CS 633 OL	Spring 2017
8	BudgetShop	CS 633 OL	Spring 2017
9	BUStudentPlanner	CS 633 OL	Spring 2017
10	Magic8Ball	CS 633 OL	Spring 2017
11	Refrigerator Raider	CS 633 OL	Spring 2017
12	ToolShare	CS 673 HAFB	Spring 2017
13	REDDIT_Clone	CS 673 HAFB	Spring 2017
14	Budget Buddy	CS 633 EX	Spring 2017
15	Listus	CS 633 EX	Spring 2017
16	BU Park Everywhere	CS 633 EL	Fall 2016
17	PoolMyRide	CS 633 EL	Fall 2016
18	Study Session Planning	CS 633 OL	Summer 2016
19	The Barking Topia	CS 633 OL	Summer 2016
20	TeeTime	CS 633 OL	Summer 2016
21	Sips	CS 633 OL	Summer 2016
22	GymBuddy	CS 633 OL	Summer 2016
23	K-Now Portal	CS 633 OL	Summer 2016
24	Shopsterfield	CS 473 EX	Spring 2016
25	Football Squares	CS 473 EX	Spring 2016
26	Career Hub	CS 633OL	Fall 2015
27	Insult Generator	CS 633OL	Fall 2015
28	Defect Database	CS 633OL	Fall 2015
29	Business Directory	CS 633OL	Fall 2015
30	Real Estate CMS	CS 633OL	Fall 2015
31	Defect Tracking	CS 633OL	Fall 2015
32	Grading App	CS 633 EL	Fall 2015
33	PetBook	CS 633 EL	Fall 2015
34	HR Docs	CS 633 OL	Summer 2015
35	E2E Performance Meas	CS 633 OL	Summer 2015
36	Grant Search	CS 633 OL	Summer 2015
37	CMS	CS 633 OL	Summer 2015
38	CMS	CS 473 EX	Spring 2015
39	Peer Reviews	CS 473 EX	Spring 2015
40	Quiz Grader	CS 473 EX	Spring 2015
41	Defect Tracker	CS 633 OL	Fall 2014

Examples from Term Projects

Slide below is taken from the term project "ToolShare" Spring 2017. It shows how stories in Pivotal are transformed into code ... that is peer reviewed, tested and merged into master. Pivotal's states of a story (Started, Finished, Delivered, Rejected, and Accepted) constitute a coherent workflow. The team demonstrated the application of techniques we covered during lectures.

Programming Workflow

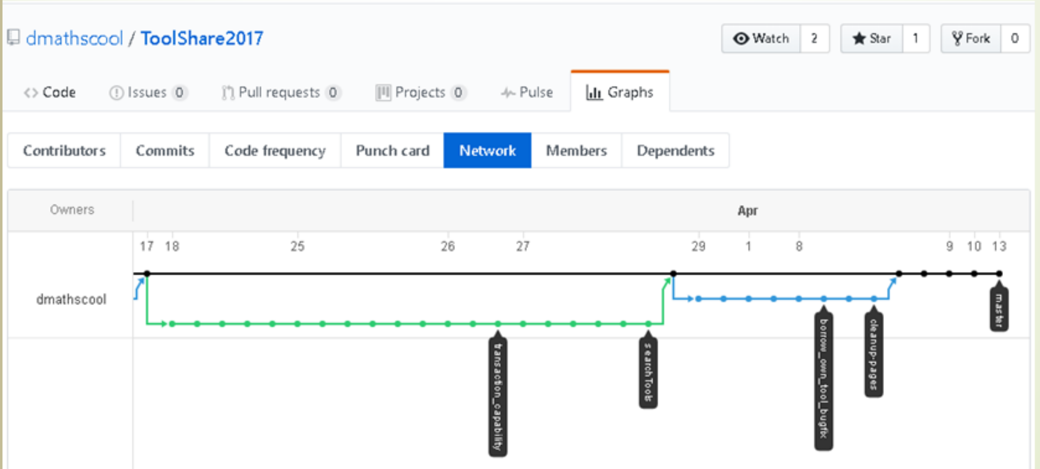
- 1-week iterations
- Used Pivotal's task states
- Working changes to code tested peer review done and merged to master
- User stories accepted/rejected during weekly meetings
- New version from master posted on production site



The next slide shows the code branches along with merges over time, which is the standard Git chart. Each weekly merge corresponds to a story that is transitioning into Accepted state.

Source Control and Branching

- Github source control
- Features added on branches and merged into master weekly


dmathscool / ToolShare2017
Watch 2
Star 1
Fork 0
Code
Issues 0
Pull requests 0
Projects 0
Pulse
Graphs
Contributors
Commits
Code frequency
Punch card
Network
Members
Dependents
Owners
Apr
17 18 25 26 27 29 1 8 9 10 13
dmathscool
transaction_capability
search Tools
borrow_own_tool_bugfix
cleanup-pages
master