

FINAL MILESTONE

Team 5

Team Lead: Brock Brown

Azure Dev Ops Lead: Brad Allred

Lead Programmer: Myron Burton

Programmer: Tanner Chamberlain

Programmer: Reid Kuttler

Lead Architect: Mariah Bleak

Scrum/QA Lead: Heather Hyer

Scribe: Linda Jensen

Software Engineering (CS 2450) - Fall 2018

10/2/18 – 11/6/18

Table of Contents:

Software Development Plan	2-42
Introduction	2
Process Model	3
Organization of the Project	4
Standards, Guidelines, Procedures	6
Management Activities	7
Staffing	7
Risks	8
Methods and Techniques	9
Quality Assurance	12
Work Packages	16
Resources	17
Budget and Schedule	25
Changes	27
Delivery	33
Meeting Logs	43

Introduction

Background/History

The Abacus has been used for calculations for thousands of years. It has proven to be an effective means of visualizing numbers and the way they interact. It is proposed that we implement a virtual abacus to teach kids how to visualize and work with numbers.

Aims

The abacus aims to teach children how to think of numbers visually and kinesthetically, thus helping many different kinds of minds to understand numbers and math.

Deliverables

- Software Development Plan
- AbracadAbacus App Prototype
- All Baseline Requirements
- Chapter expert explanations
- Meeting logs

Persons Responsible

Team Lead:	Brock
Administrator:	Dr. Sharp
IT Department:	IT Department
Teacher:	Pam Hyer
Student:	Joslin Kuttler

Summary

The AbracadAbacus (name subject to change) app will be used on computers in a classroom setting. It will be used by children in grades K-5 as a learning tool. There will be different levels for each grade, which will change the number of the columns in the abacus. This app will have the functionality of an abacus with counting, addition, and subtraction. It will be available for Windows computers versions 7 and newer and macOS versions 10.10 and newer. Both systems also need to be running Java 1.8.

Process Model

Process Model

We are using Kanban. See reasons in support of Kanban in Milestone 1 Documentation - page 11. We used Kanban by using the board of tasks to be completed in each sprint and making sure our tasks were completable inside each sprint.

Activities

Planning, coordination, programming, system/GUI design, documentation.

Identifiable Milestones

- Milestone 1
 - o Meetings w/ Administrator, Teacher, and Student to determine requirements specifications
- Milestone 2
 - o Approval of concept and proposals with Administrator, Teacher, and Student
- Milestone 3
 - o Approval of Prototype Design (GUI)
- Milestone 4
 - o Proposal of change order
- Milestone 5
 - o Approval of change order concept
- Milestone 6
 - o Approval of finished product (including documentation, prototype, and chapter expert explanations)

Critical Paths

The most critical path is the documentation. The documentation is what drives the project and tells us what we're doing.

Organization of the Project

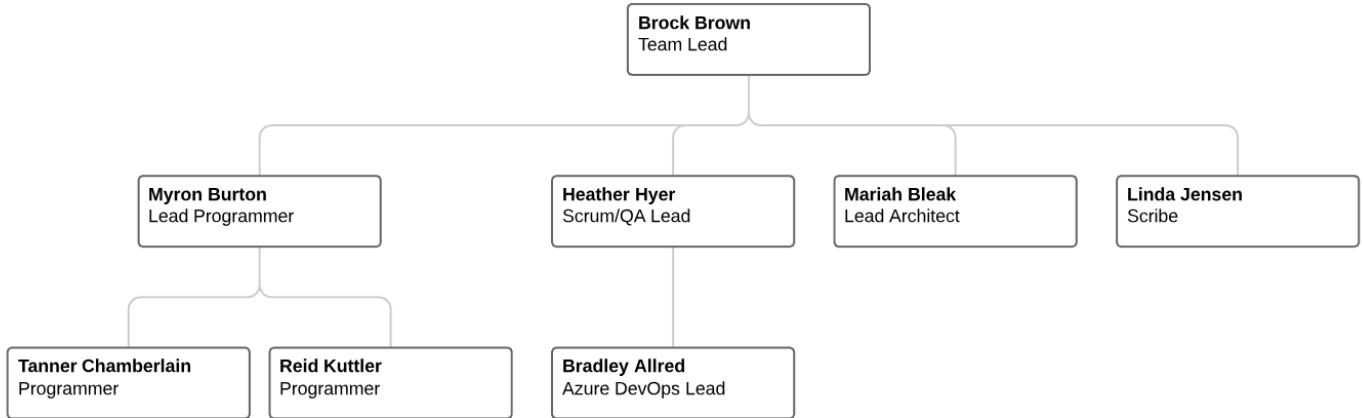
Relationship with User Organization

- Information needed
 - o Grade requirements, planned use, current teaching methods
- Services Required
 - o People to test the program on (students and teachers)

Identified Team Member Roles

- Team Lead – Brock Brown
 - o Looks at the big picture and makes sure all team members have feasible, clear, and complementary tasks assigned to them.
- Azure Dev Ops Lead – Bradley Allred
 - o Cost estimation, task time management, and Azure DevOps management. Role in Azure DevOps will allow all team members to add time estimation to their tasks and actual time.
- Lead Programmer – Myron Burton
 - o Design and develop the structure of the code. See where development needs to be focused know the skills of the programming team. Adhere to software quality standards agreed upon so work may be done to the highest possible quality.
- Programmers – Tanner Chamberlain/Reid Kuttler
 - o Assist in creating the abacus prototype. Create supporting documents to plan and design the project, and work with the programming team to identify and manage the risks involved in the project. Program the secondary views for the Abacus app.
- Lead Architect – Mariah Bleak
 - o As software architect for this project, my duties relate a lot to the liaison between stakeholders and the product development. It is my job to ensure the graphical representation of the project, which consists of but not limited to: UML diagrams, class diagrams, use cases, and prototypes. Requirements and specifications set by the stakeholders influence the design and implementation of the product, and as it develops I use the architecture items to share with the stakeholders and get their feedback about further functionality and development. I take that feedback and relate it to the programmers and developers to further cultivate a product that satisfies the needs of the client while remaining within the financial means laid out in the contract with the client.
- Scrum/Quality Assurance (QA) Lead – Heather Hyer
 - o Oversee the team's agile development process. Decompose milestones into sprints and backlogs. Oversee task assignment and management. Create a quality assurance plan and document test cases.
- Scribe – Linda Jensen
 - o Take meeting notes, compile documentation, and help in all other areas as necessary. Ensure that each meeting has a detailed meeting log and mark any questions and action items where follow-up is needed. Work to assure that all documentation meets requirements.

Org Chart



Chapter Expert Explanation (Chapter 5- Brock Brown)

We will use section 5.2.5 (The Agile Team) frequently. We will also use the hierachal organization style and adhocracy because our project is new to everyone. As the manager, I will use the relation management style and I will identify goals at an early stage and communicate them clearly.

It's really important to decide what the project goals are, i.e. fastest, least memory use, etc. so everyone knows and can comply with said goals. Along with these goals must come the explicit expectation of clear notes/pseudocode in the file. As we progress we will further clarify the standards for both in-code documentation and out-of-code documentation. I think that the ideal communication is an informal and open dialogue, which I believe supports my goal of an integration-style leadership and team.

Chapter Expert Explanation (Chapter 11- Mariah Bleak)

Software architecture concerns the large-scale structure of software systems. This large-scale structure reflects the early, essential design decisions. This decision process involves negotiating and balancing of functional and quality requirements on one hand, and possible solutions on the other hand. Software architecture is not a phase strictly following requirements engineering, but the two are intertwined

Three purposes:

- Vehicle for communication among stakeholders
- Captures early design decisions
- Transferable abstraction of a system

Forces that influence architecture:

- Developmental organization
- Background and expertise of the architect
- Technical and organizational environment

In the software architecture, the global structure of the system has been decided upon. This global structure captures the early, major design decisions. Whether a design decision is major or not really can only be ascertained with hindsight, when we try to change the system.

Stakeholders speak with architect back and forth over requirements and quality, when they reach an agreement that feeds into the development.

Standards, Guidelines, Procedures

Documentation procedures

Documentation is to be delivered once a week before the client meeting each Thursday afternoon at 2:30 p.m. The quality of the documentation is to be assessed through the Quality Assurance (QA) plan (see section marked “Quality Assurance”). The documentation is kept up to date through the Azure DevOps wiki, that uses versioning and updates immediately when changes are made.

Chapter Expert Explanation (Chapter 4 – Mariah Bleak)

Configuration management comprises of careful procedures that are necessary to manage every element over the lifespan of a large software system or a distributed development project. Throughout the lifetime of a system or project, there may at any given time exist different versions of the software, depending on what updates have been given to the client and if any programmers are working on older versions.

Baseline –official version of the complete set of documents related to the project.

Configuration items are the items contained in the baseline.

Ex:

- Source code components
- The requirements specifications
- The design documentation
- The test plan
- Test cases
- Test results
- The user manual

Configuration management takes care of controlling the release and change of CRs throughout the software life cycle.

CCB (Configuration control board) ensures that any change to the baseline is properly authorized and executed. When a change request is submitted to the CCB, they need info on how the change will affect the product and the development process. If approved, the change request becomes a work package which will need to be scheduled. All changes and status of changes need to be recorded, this ensures that if needed to go back to a previous version or to monitor what changes have been made there is a clear record.

Management Activities

Management Priorities

The priorities for the project are documentation, usability, and responsiveness

- Documentation should be clear, meet all requirements, and thoroughly communicate the work done on this project.
- The prototype delivered should work as expected, making it usable.
- The program should be quick to load and respond to user input.

Report Regularity

- Reports will be every Thursday afternoon at 2:30 p.m.

Staffing

Milestone 1

Allred, Brad	Azure DevOps Lead
Bleak, Mariah	Lead Architect
Brown, Brock	Team Lead
Burton, Myron	Lead Programmer
Chamberlain, Tanner	Programmer
Hyer, Heather	Scrum/QA Lead
Jensen, Linda	Scribe
Kuttler, Reid	Programmer

Final Milestone

Allred, Brad	Azure DevOps Lead
Bleak, Mariah	Lead Architect
Brown, Brock	Team Lead
Burton, Myron	Lead Programmer
Chamberlain, Tanner	Programmer
Hyer, Heather	Scrum/QA Lead
Jensen, Linda	Scribe
Kuttler, Reid	Programmer

Risks

Hardware Risks

We have the hardware available for the scope of creating documentation and a functional prototype. We may not have hardware to test our prototype on all platforms, but that may be impossible given our current scope. Documents and data are backed up in a Microsoft server to prevent data loss.

Unstable Requirements

Requirements may be subject to frequent change or review, delaying the project. Remedied by creating a firm requirements specification document to confirm all requirements, then creating a change order protocol to handle all changes during production. Users, client, and developers must all be included during the design phase to minimize the need to change requirements.

Product is too complex

It is possible to become over-ambitious with our project. Product should meet minimum requirements, and the design phase should consider our time frame and resources before committing to a requirements specification.

Personnel Shortfall

Personnel may become sick or quit unexpectedly. In that event, tasks assigned to that person will be redistributed throughout the group.

Unrealistic Schedule or Budget

With a hard deadline for the project, an unrealistic schedule is possible as the deadline approaches. Planning, task assignment, and timeboxing are critical to ensure that tasks will be completed on time.

Wrong Functionality

Creating the wrong prototype or providing the wrong documents. Proper requirement definitions and clear communication with the client before and during the project will minimize this risk.

Wrong User Interface

It is possible to make a technically correct program that is difficult for people to use. The goal of our project is to enhance classroom learning, not detract from it with a difficult or confusing UI. Gather information and feedback from teachers and students to minimize this risk.

Gold Plating

Creating unnecessary or "nice" features outside of the requirements may make it hard to meet our deadline. The requirements outlined should be met first, then any additional features checked off by the client and submitted through a change order if time permits.

Capability Shortfalls

Complications with using tools or software may arise and delay the project. It's important that everybody learn and become familiar with project tools like ADO and Eclipse. Team members with experience can teach members that are less familiar with the software.

Chapter Expert Explanation (Chapter 8- Tanner Chamberlain)

8.1 talks about identifying variables, might not be relevant to the requirements.

8.2: 3 types of uncertainty: Product (what we're making), Process (how we're doing it), Resource (people and tools). May not be relevant to requirements

8.3: Techniques for mitigating risk during development. Not included in requirements, but to be used afterwards.

8.4: Techniques and systems for keeping the project on track. Might be useful for planning requirements. Includes task assignment and project diagrams such as Gantt charts and PERT diagrams.

Methods and Techniques

Requirements Engineering

- Meet with the administrator and the client to discuss the requirements. We will use the interview method and will read the state requirements for students in each grade.

Non-functional requirements our team would recommend for consideration

- Write name in app and when you complete the abacus task it sends the results to a remote database for storage.
- Party Parrot hidden Easter egg
- Change of colors
- Reverse-engineered tutorial
- Fun music

Chapter Expert Explanation (Chapter 9 – Heather Hyer)

We need to specify requirements from the perspective of:

- IT
- Student
- Teacher
- Team
- Admin (Prof. Sharp)

9.1: The end result of this should be the requirements specification document (baseline). We need this.

9.1.4: The Delphi process (MoSCoW). After this process is complete, we will build tasks bases on the requirements specification acquired.

9.2: This will be continuously updated by various team members but mainly the scribe.

9.2.1: This is where you can find requirements traceability techniques and formats.

9.3: Use if you need help with creating this document.

9.3.1: Non-functional requirements will be continuously expanding.

Four types of non-functional requirements:

- External interface requirements
- Performance requirements
- Design constraints
- Software system attributes.

These non-functional requirements can be viewed as constraints placed upon the development process or the products to be delivered.

9.4: Tracking track progress and completion. I suggest we track mainly in ADO and everyone needs to update their progress regularly.

Design

- Prototype-based design
- UML diagrams
- Use-case diagrams
- Storyboard

Chapter Expert Explanation (Chapter 12- Reid Kuttler and Tanner Chamberlain)

12.1: Code must be designed by using modules.

Modules are units of code that do one specific function.

Examples: a method, a class, or a related set of classes.

Good module usage has lots of benefits and makes clean code.

12.1.1: Modules must only have the data necessary to carry out their task. Accomplished through proper return and method passing channels.

12.1.2: Modules must strive for high cohesion: all code in a module is devoted to one purpose.

Modules must strive for low coupling: modules must only communicate essential information, nothing extra.

12.1.3: NO GLOBAL VARIABLES. Use "get" and "set" methods, keep data within modules private.

12.1.4: Modules must strive for low complexity, while still being complex enough to complete their functions. Complexity is measured in a variety of ways.

12.1.5: Correct module set up should be able to be shown in a graph, such as a UML diagram for example.

12.1.6: Not important.

12.2.1: Decompose Functions in code into a number of sub functions.

Use a combination of Top-down and bottom-up decomposition

12.2.2: Draw a context diagram which is a data flow diagram representing the whole system in one process.

Go from a set of data flow diagrams to a hierarchical model for how you are going to implement this.

12.2.3: A good program reflects the structure of both the input and the output.

Perhaps use Jackson Structured Programming. This involves combining the input and output diagrams and merging them together to form the global picture of what the program does.

12.3: Object oriented involves identifying objects, determining their attributes, and determining relationships between objects.

This is a middle-out design.

12.3.1: A rich set of notations

A poor process model as it is difficult to decide when to iterate or when to do a specific iteration.

12.3.2: Large attention paid to the design phase

The version of the method hinges on the availability of a good requirements document.

As a method, Fusion is very prescriptive

12.3.3: It is very complete iterative model that goes further than just analysis and design.

It uses the unified Modeling Language to represent artifacts and views.

Use cases play a central role.

12.4

	Problem-oriented	Product-oriented
Conceptual	ER modeling Structured Analysis OO Analysis	Structured Design OO Design
Formal	JSD VDM	Functional Decomposition JSP

12.4.1: Object oriented methods usually only cover the specification subprocess. We still need the other three processes discussed in chapter 9, Elicitation, specification, validation and negotiation.

12.5: Good Design pattern is MVC

Model, View Controller

12.6: There are seven user roles for design documentation

1. The project manager
2. The configuration manager
3. The designer
4. The programmer
5. The unit tester
6. The integration tester
7. The maintenance programmer

12.7: Errors made at an early stage are difficult to correct, therefore it is necessary to pay extensive attention to testing and validation issues during the design stage.

Implementation

- Java

Chapter Expert Explanation (Chapter 13- Myron Burton)

13.2.5: Test driven development is an effective way of assuring that we cover our requirements. TDD is done by first designing tests and then making the implementation. We can use this to create basic functionality tests that would show that an abacus can do the math asked.

13.3: Testing is documented in what IEEE calls a verification and validation plan. This report covers the design specification, the procedure specification, and an item transmittal report showing how each feature will be tested, what sequence testing takes and what items will be tested respectively. After testing is done a test log is composed, an incident report with incidents that need further investigation is written, and a summary report shows the overall evaluation of the testing and any findings that should be reported. Documenting our bug fixes will allow us to track progress and note where new issues may have been presented.

13.5: We will use this to make sure that all of our code is relevant and that our documentation actually matches what is made. If we cover all of the requirements, both in documentation and in code, then we will provide a more complete solution

13.5.3: With this we can make sure that no part of our requirements specification is lost. We can assure a complete solution for the customer.

13.6: We will be able to test robustness if an invalid input is given in code. By seeing how our code reacts to artificial errors we can pin down issues with code and also adjust the way we handle input.

13.7: Theoretical joints in a program can be an issue. Places where information has to be transferred from one system to another may cause issues. Making sure that we define potential areas where errors may occur, we can stress those joints to assure proper functionality

Version Control

- Git Hub
- Azure DevOps Wiki

Quality Assurance

Quality Assurance Plan

Manual testing:

- We will implement peer reviews of code
- The programmers will perform regular walkthroughs of one another's code
- Correctness proofs are unnecessary for a project of this scope
- Stepwise abstraction is a moot point, since we designed our product around requirements

Coverage-based testing:

- This will make sure the code is actually executed
- This will help measure development efficiency
- Our primary focus will be on coverage-based testing of Requirements Specifications

Fault-based testing:

- Error seeding doesn't make sense since we're not dealing with database manipulation or statistical analyses
- Mutation testing will be useful in testing different values for calculation on the abacus

Error-based testing:

- This will be useful for testing boundaries and edge cases

Requirements-based acceptance testing:

- Validation: Are we building the right system?
 - This is more applicable in the earlier stages of development
- Verification: Are we building the system right?
 - Acceptance testing for each feature will help to verify functionality

User testing:

- Often the users encounter problems the programmers never consider. In this particular project, since children can be unpredictable, having them test a product designed for them would help ensure its quality.
- It would be beneficial to have actual users go through use cases such as the following:

Primary Actor: Elementary school student

Scope: School district

Level: User goal

Stakeholders and Interests:

User---wants to have fun at school

Teacher---wants their students to learn

Department---wants the technology they pay for to be used

Precondition: None

Minimum Guarantee: Student accesses desired lesson

Success Guarantee: Student accurately completes desired lesson

Main Success Scenario:

1. Student selects their grade level
2. Student selects activity type
3. Student is given a number and an operation to perform
4. Student moves the correct number of beads
5. Student completes the activity

Extensions:

- 4a. Student never moves the correct number of beads
 - 4a1. After a certain number of tries, student is shown the correct answer

Chapter Expert Explanation (Chapter 6- Linda Jensen)

Correctness: The extent to which a program satisfies its specifications and fulfills the user's mission objectives.

Reliability: The extent to which a program can be expected to perform its intended function with required precision.

Efficiency: The amount of computing resources and code required by a program to perform a function.

Integrity: The extent to which access to software or data by unauthorized persons can be controlled.

Usability: The effort required to learn, operate, prepare input, and interpret output of a program.

Maintainability: The effort required to locate and fix an error in an operational program.

Testability: The effort required to test a program to ensure that it performs its intended function.

Flexibility: The effort required to modify an operational program.

Portability: The effort required to transfer a program from one hardware and/or software environment to another.

Reusability: The extent to which a program (or parts thereof) can be reused in other applications.

Interoperability: The effort required to couple one system with another.

These are the quality charts that we should be using to model our prototype after. Everyone should be familiar with these terms as we go into production. We should be mostly focusing on Usability and Correctness for this project. It is very important that we work with the client to produce a prototype that best matches their specifications. The output should be able to be interpreted by someone that doesn't know the program already, hence usability for an elementary school aged audience.

Product operation:	
Correctness	Does it do what I want?
Reliability	Does it do it accurately all of the time?
Efficiency	Will it run on my hardware as well as it can?
Integrity	Is it secure?
Usability	Can I run it?
Product revision:	
Maintainability	Can I fix it?
Testability	Can I test it?
Flexibility	Can I change it?
Product transition:	
Portability	Will I be able to use it on another machine?
Reusability	Will I be able to reuse some of the software?
Interoperability	Will I be able to interface it with another system?

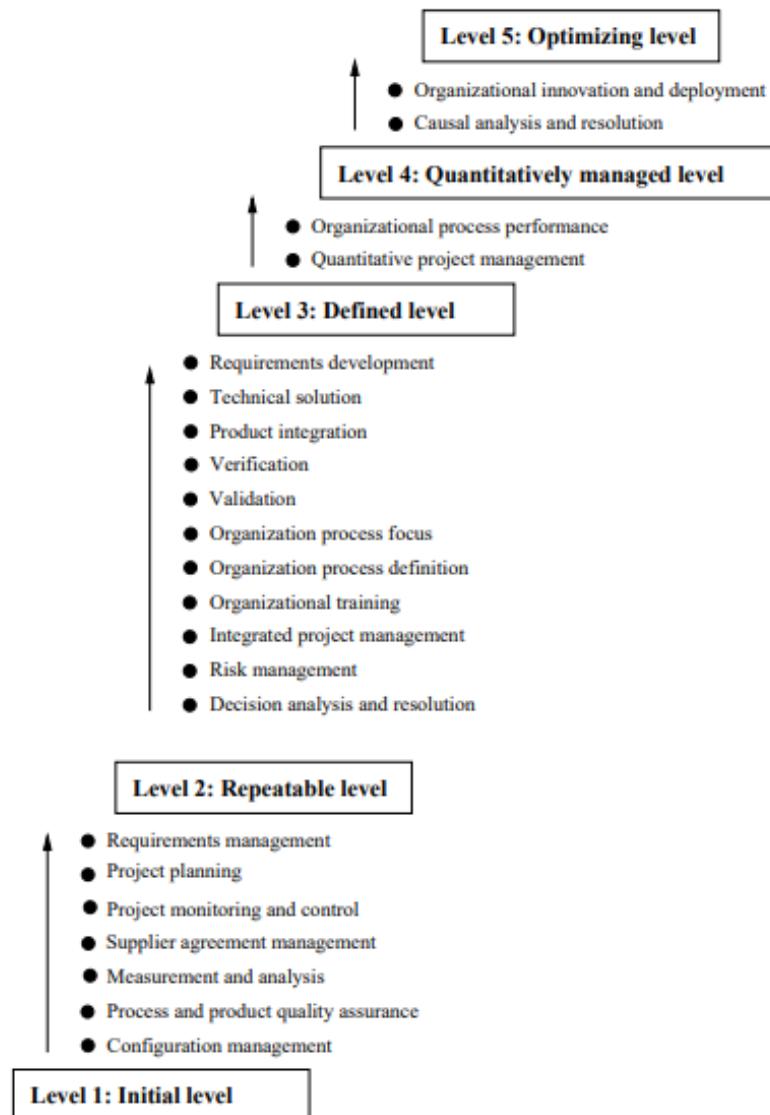
Throughout the process of creating, testing, and editing the product, we need to be able to answer all of these questions

Characteristic	Subcharacteristics
Functionality	Suitability Accuracy Interoperability Security Functionality compliance
Reliability	Maturity Fault tolerance Recoverability Reliability compliance
Usability	Understandability Learnability Operability Attractiveness Usability compliance
Efficiency	Time behavior Resource utilization Efficiency compliance
Maintainability	Analyzability Changeability Stability Testability Maintainability compliance
Portability	Adaptability Installability Co-existence Replaceability Portability compliance

These are the quality characteristics that we're looking for in a final product. We can't focus on all at once, so I suggest we focus on Functionality and Usability as the most appropriate for the Abacus app.

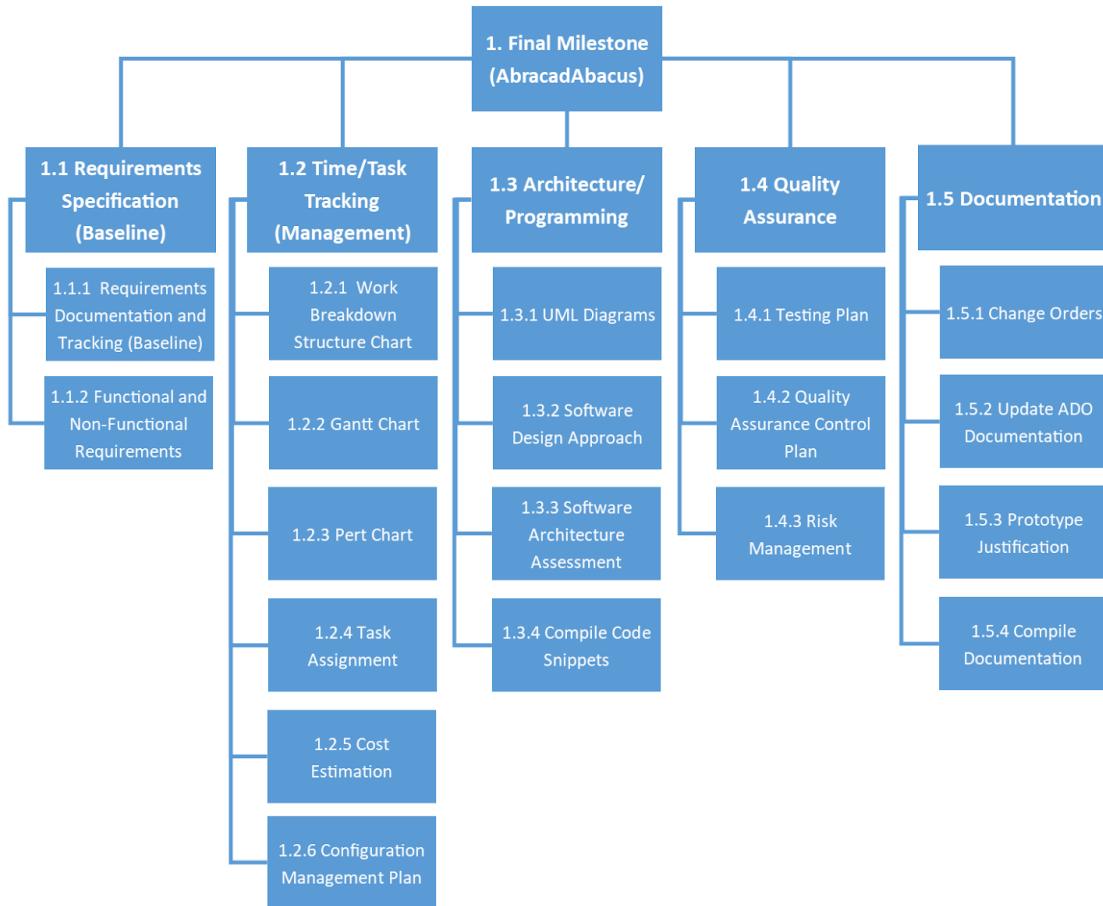
The Total Quality Management (TQM) concept is about applying principles of quality to each aspect of the organization and is the responsibility of each employee. TQM has three cornerstones: customer value strategy, organizational systems, and continuous improvement. The basic idea of these cornerstones is that the product should benefit what the customer actually needs (always work with the customer to determine their needs), people are willing and able to create a quality product, and errors are opportunities for learning.

The Capability Maturity Model (CMM) was derived from TQM. The software process is characterized into 5 maturity (evolutionary) levels: Initial, Repeatable, Defined, Quantitatively managed, and Optimizing. Below is a chart detailing the CMM.



Work Packages

Work Breakdown Structure



Resources

Required Tools

- Azure DevOps
- Google Docs
- Microsoft Word
- OneDrive
- Eclipse
- GitHub
- Java FX
- Java 10,
- Trello
- Slack.
- Windows computer versions 7 and newer or macOS versions 10.10 and newer

Azure DevOps

Our use of ADO incorporated the use of Epics. Features, Tasks and User Stories. We used Epics and the foundation for each of our milestones.

Work Items

ID	Title	Assigned To	State	Area Path	Tags	Comments	Activity Date
21	Milestone 2	Unassigned	New	Abacus		1	11/6/2018 6:14:15 PM
184	Final Milestone	Heather Hyer	Active	Abacus			11/6/2018 6:14:12 PM
20	Milestone 1	Unassigned	Resolved	Abacus		1	9/26/2018 12:26:00 PM
22	Milestone 3	Unassigned	New	Abacus		1	9/13/2018 2:45:56 PM

Inside each Milestone were links to Features which were all the objectives needing fulfillment

20 Milestone 1

State	Reason	Area	Iteration
Resolved	Features complete	Abacus	Abacus

Description

Click to add Description

Discussion

Add a comment. Use # to link a work item, ! to link a pull request, or @ to mention a person.

Bradley Allred commented 3 months ago
Copied from Epic 1

Planning

Priority: 2
Risk:

Development

+ Add link
Development hasn't started on this item.

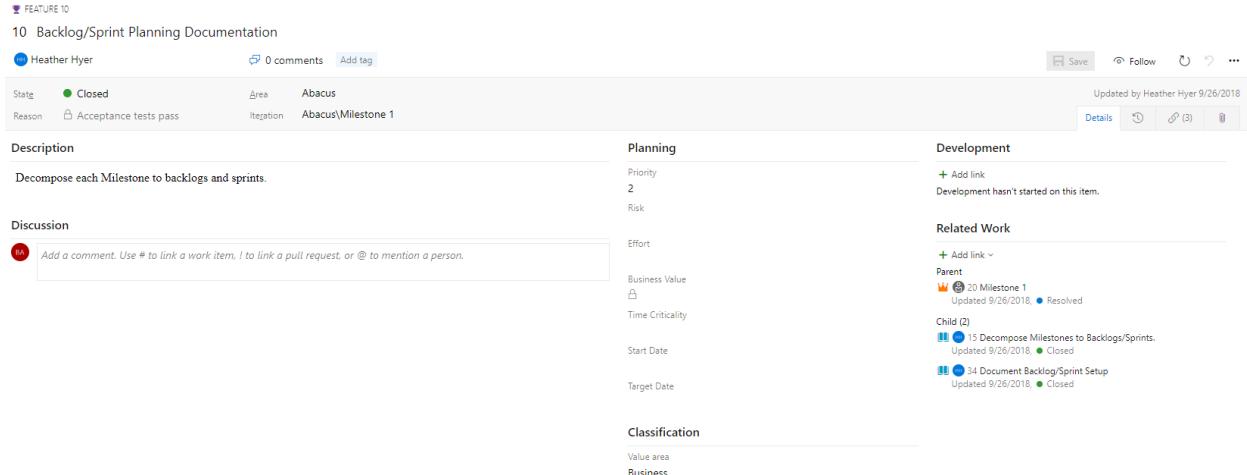
Related Work

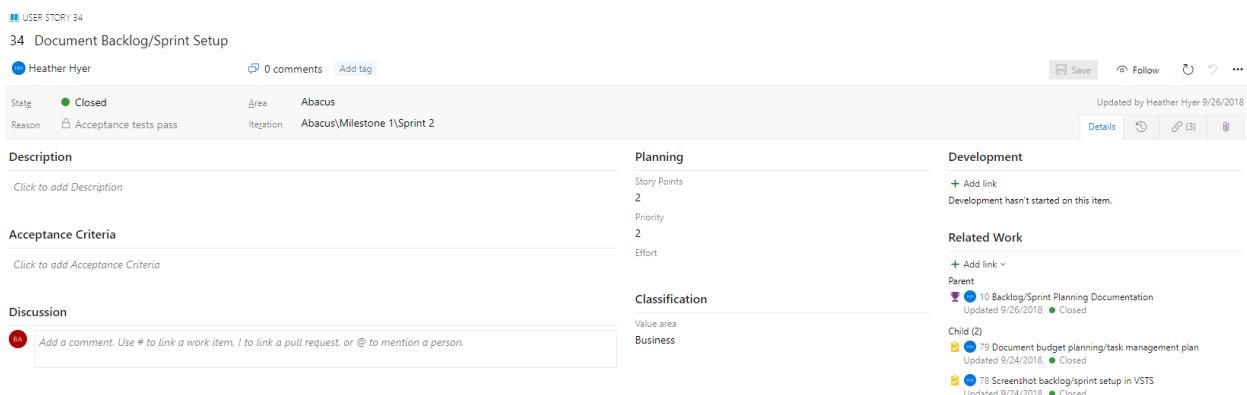
+ Add link

Child (9)

- 1 VSTS Setup Documentation
Updated 9/26/2018, ● Closed
- 10 Chapter Summaries
Updated 9/26/2018, ● Closed
- 8 GUI Wireframes
Updated 9/26/2018, ● Closed
- 5 SCRUM/Kanban lessons documentation
Updated 9/26/2018, ● Closed
- 10 Backlog/Sprint Planning Documentation
Updated 9/26/2018, ● Closed
- 4 Meeting Logs
Updated 9/26/2018, ● Closed
- 7 UML Diagrams
Updated 9/26/2018, ● Closed
- 12 Test Case Examples
Updated 9/26/2018, ● Closed
- 11 Use Case Examples
Updated 9/26/2018, ● Closed

Each feature contained User Stories, which contained tasks and sprints and sometimes other stories





Our User Stories were placed into weekly sprints that were determined by our weekly meetings. The sprints used 6 states. New, Active, Ready for Review, Failed review, Passed review, and Closed.

New: used as the untouched column this way our team lead could determine what wasn't getting done and reach out to the assigned person and see if there were any blocking issues preventing them from commencing.

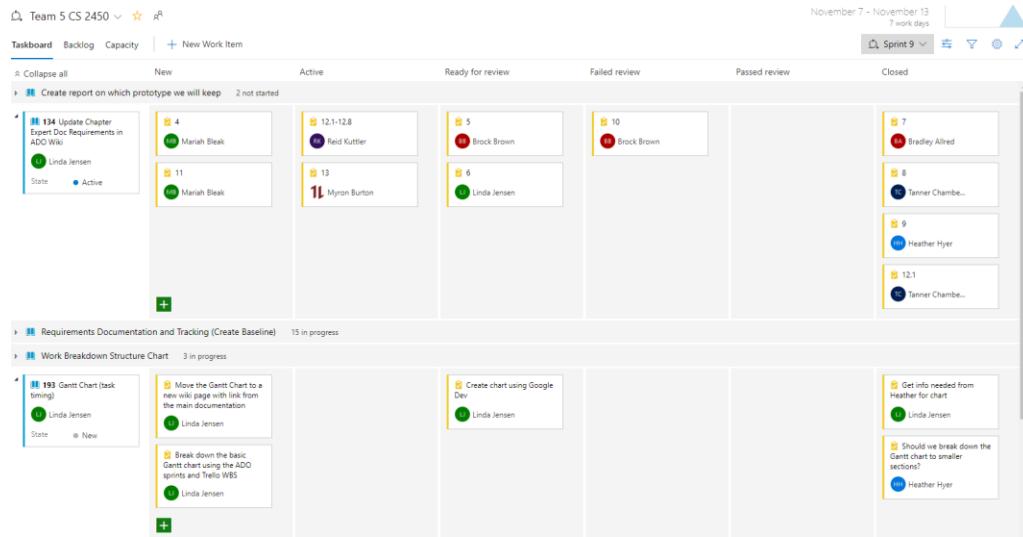
Active: once a task was begun, the assignee would change the status to active.

Ready for Review: let our QA know what tasks were needing to be looked at.

Failed review: if the task was not completed to specification, it was sent to Failed review. The assignee would receive an email and letting them know that more was to be done for the issue

Passed review: once completed to spec, the task was placed here for final approval

Closed: the Team lead would close the issue once his final approval was given.



The Code

Our code repository and version control were implemented through ADO. The Commits can be seen using 3 filters. History type, author, and a date range. Also available for view are the branches, pushes and pull requests. These can all be done in a general context, or in the context of an individual file or folder.

Name	Last change	Commits
abacusEx.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
abacusExWithWords.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
abacusFour.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
abacusOne.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
abacusThree.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
abacusTwo.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
addButton.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
choicePageTitle.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
countButton.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
fiveButton.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
fourButton.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
gradePageTitle.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
greenHead.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
kButton.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
logo.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
logoWithTitle.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
modalCloudLarge.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
modalCloudMedium.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
modalCloudNext.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
modalCloudNextWithWords.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer
modalCloudEx.svg	10/24/2018	74ff3c3b Added 34 files to components Heather Hyer

Backlog and Sprint Decomposition

Sprints

We opted to have week long sprints, just based on the amount of work to be done for this final milestone within the timeframe given.

Originally, we had 5 sprints each planned for milestones 1 and 2. However, we had to respond to changes made around milestone 8 to consolidate both milestones into one.

✓ Milestone 2	9/26/2018	10/30/2018
Sprint 3	9/26/2018	10/2/2018
Sprint 4	10/3/2018	10/9/2018
Sprint 5	10/10/2018	10/16/2018
Sprint 6	10/17/2018	10/23/2018
Sprint 7	10/24/2018	10/30/2018
✓ Milestone 3	10/31/2018	12/4/2018
Sprint 8	10/31/2018	11/6/2018
Sprint 9	11/7/2018	11/13/2018
Sprint 10	11/14/2018	11/20/2018
Sprint 11	11/21/2018	11/27/2018
Sprint 12	11/28/2018	12/4/2018

We were able to track our team's velocity using burndown charts and widgets such as the following:



Milestone/Feature/Story Breakdown

We've divided the work for this final milestone into an Epic, Features for that Epic, Stories for each Feature, and Tasks for each Story. Each backlog has a burndown chart to help keep track of progress in and Epic, Feature, or Story-level scope.

Features in this Epic:

Epic	> Final Milestone
Feature	> Requirements Specification (Baseline)
Feature	> Time/Task Tracking (Management)
Feature	> Architecture/Programming
Feature	> Quality Assurance
Feature	> Documentation

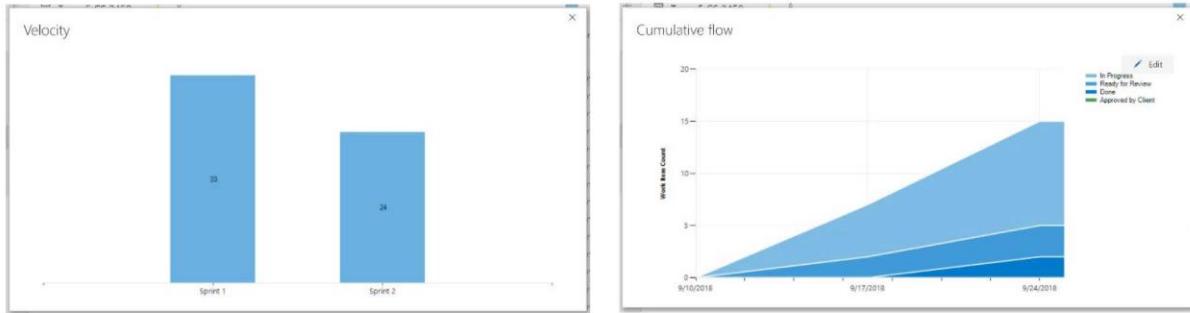
Stories in each Feature (and their respective tasks):

Feature	> Requirements Specification (Baseline)
User Story	> Requirements Documentation and Tracking (Create... ...)
Task	Create tasks for this story ...
Task	introduction
Task	process model
Task	organization of project
Task	standards, guidelines, and procedures
Task	management activities
Task	risks
Task	staffing
Task	methods and techniques
Task	quality assurance
Task	work packages
Task	resources
Task	budget and schedule
Task	changes
Task	delivery
User Story	Functional and Non-Functional Requirements (Delphi ...)
Feature	> Time/Task Tracking (Management)
User Story	> Work Breakdown Structure Chart ...
Task	Add wiki with WBS chart
Task	Break down work
Task	Create tasks for this story

User Story	✓ Gantt Chart (task timing)	
Task	✗ Create chart using Google Dev	
Task	✗ Get info needed from Heather for chart	
Task	✗ Should we break down the Gantt chart to smaller se...	
Task	✗ Move the Gantt Chart to a new wiki page with link f...	
Task	✗ Break down the basic Gantt chart using the ADO sp...	
User Story	✓ Pert Chart (timeboxing)	...
Task	✗ Create tasks for this story	...
Task	✗ Compile elements from WBS and ADO sprints.	
Task	✗ Create a Rough Draft	
Task	✗ Compare draft with Gantt chart and WBS	
Task	✗ Revise and polish	
Task	✗ Submit for team review	
User Story	✓ Task Assignment	
Task	✗ Create Features/Stories in ADO based on WBS	
Task	✗ Assign someone to each Feature/Story	
Task	✗ Create tasks for this story	
User Story	Cost Estimation	
User Story	Configuration Management Plan	...
Feature	✓ Architecture/Programming	
User Story	✓ UML Diagrams	...
Task	✗ Create tasks for this story	
Task	✗ Review code given from programmers to update U...	
User Story	Software Design Approach	
User Story	Software Architecture Assessment	
User Story	Compile Code Snippets	
Feature	✓ Quality Assurance	
User Story	Testing Plan	
User Story	Quality Assurance Control Plan	
User Story	Risk Management	
Feature	✓ Documentation	
User Story	Change Orders	
User Story	Update ADO Documentation	...
User Story	Prototype Justification	
User Story	✓ Compile Documentation into the Binder	...
Task	✗ Compile all Meeting Logs	
Task	✗ Compile Software Development plan	
Task	✗ Go over checklist and make sure all are covered	
Task	✗ Get the Prototype documentation from Myron	

The story-level backlog provides a chart to track the team's velocity. This is useful for budget tracking.

There is also a chart to keep track of the team's cumulative flow, which is helpful for cost estimation.



Task Tracking

Work assigned to Heather Hyer (5)		
	User Story	Task
15	● Resolved	● Decompose Milestones to Backlogs/Sprints.
34	● Active	● Document Backlog/Sprint Setup
79	● Active	● Document budget planning/task management...
78	● Active	● Screenshot backlog/sprint setup in VSTS
10	● Active	● Backlog/Sprint Planning Documentation

There is a widget on the main dashboard that gives each team member an overview of the stories, tasks, and features assigned to them.

Each team member is responsible for creating tasks for each of their assigned stories for a given sprint. They are also responsible for tracking the progress of their tasks and stories and the plan is to do so regularly in order to get an accurate idea of the team's progress.

78 Screenshot backlog/sprint setup in VSTS

Heather Hyer · 0 comments · Add tag

Stat: Active · Area: Abacus · Iteration: Abacus\Timeline 1\Sprint 2

Reason: Work started · Updated by Heather Hyer just now

Description

Screenshot backlog/sprint setup in VSTS and organize them into a document.

Planning

Priority: 2 · Activity:

Effort (Hours)

Original Estimate: 1 · Remaining: 0.5 · Completed: 0.5

Development

+ Add link · Development hasn't started on this item. Create a new branch

Related Work

+ Add link · Parent · 34 Document Backlog/Sprint Setup · Updated an hour ago, Active

Each story is assigned a certain number of story points. The value for these story points follows a modified Fibonacci sequence (1,2, 3, 5, 8, 10) and points are assigned by team consensus according to complexity, difficulty, and estimated time cost. Stories should also include acceptance criteria and a description as needed. Each feature contains a more in-depth description. Stories are also assigned a priority value ranging from 1-4, weighted by what is most vital to the epic as a whole.

The screenshot shows a Microsoft Azure DevOps User Story card for item 24. The card title is "Document 3 reasons for and 3 reasons against SCRUM". Key details include:

- State:** Resolved
- Area:** Abacus
- Reason:** Code complete and unit...
- Iteration:** Abacus\Milestone 1\Sprint 1

The card is updated by Heather Hyer on 9/20/2018. It includes sections for **Description**, **Acceptance Criteria**, **Planning** (Story Points: 1, Priority: 4, Risk: 1), **Development** (status: Development hasn't started on this item, Create a new branch), **Classification** (Value area: Business), and **Related Work** (links to other items like "6 SCRUM/Kanban reasons documentation" and "80 Document 3 reasons for SCRUM").

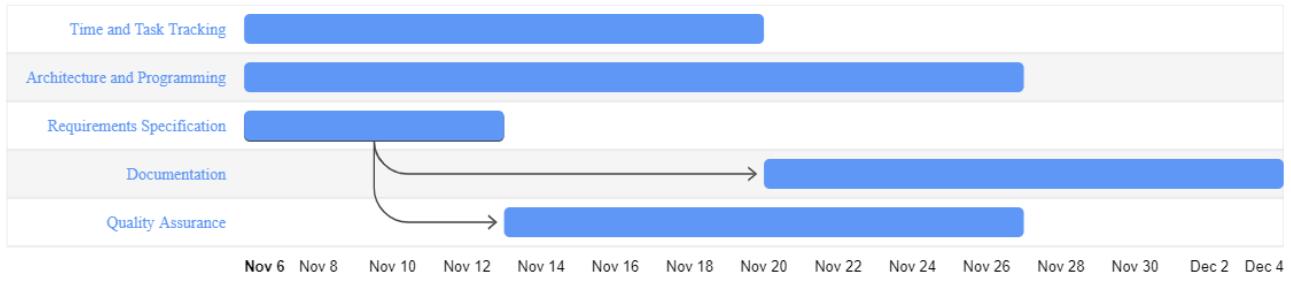
Each team member also keeps track of the tasks they have created for their stories, recording a time estimate (in hours), and reporting the actual time the task took. This is an important tracking mechanism, since this is what is reported in the sprint's burndown chart, which helps us as a team to know how accurate our estimations are and adjust accordingly.

The screenshot shows the Azure DevOps Boards Sprints page for Team S CS 2450. The sprint is defined from September 19 to September 25, with 2 work days remaining. The burndown chart shows the progress of tasks over time. The backlog section lists several tasks, including:

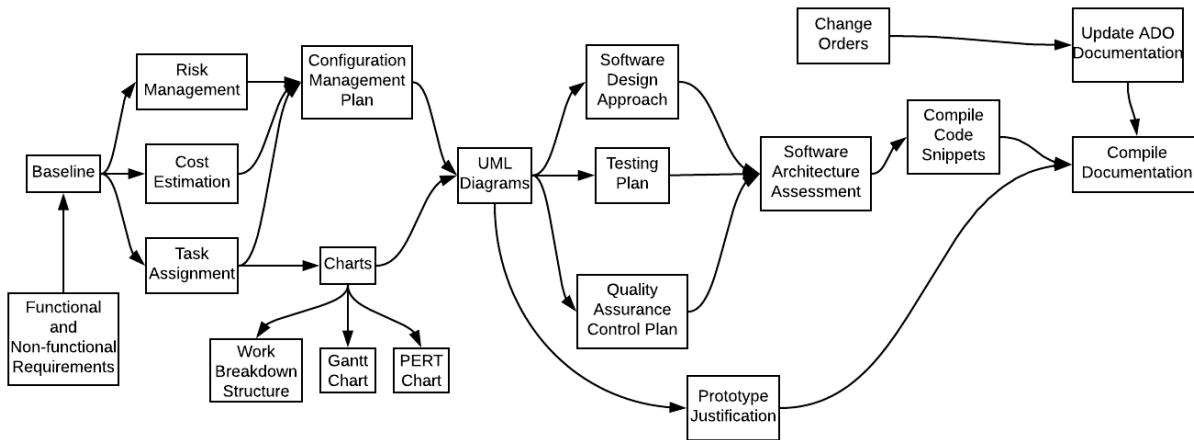
- Document Backlog/sprint setup (Heather Hyer, 2 h, Active)
- Screenshot backlog/sprint setup in VSTS (Heather Hyer, 1 h, Active)
- Document budget planning/task management plan (Heather Hyer, 1 h, Active)
- List and Demonstrate Use case examples (Tanner Chamberlain, 1 h, Active)
- Revise with UML Diagram (Tanner Chamberlain, 1 h, Active)
- Make it look fancy and polished (Tanner Chamberlain, 1 h, Active)
- Create an outline (Tanner Chamberlain, 1 h, Active)

Budget and Schedule

Gantt Chart



Pert Chart



Cost Estimation

Name	Hours Estimate		Current Hours	Est Remaining	
Brock	45		50	10	
Heather	40		25	7	
Linda	35		21	7	
Myron	35		19	8	
Bradley	35		19	8	
Tanner	45		28	6	
Reid	45		23	6	
Mariah	40		30	4	Total Est
	320	320	215	56	271
Cost per hour:	50		Cost per hour:	50	
Total Cost Est	\$16,000		Total Cost:	\$13,550	
	\$13,550.00				
Divide	\$16,000.00				
margin of error		85% initial cost estimation was within 15%			

Chapter Expert Explanation (Chapter 7 – Bradley Allred)

This section (7.3) is generally used in coordination with a cost estimation expert, but since this is a smaller project (Relative to the working world) we can estimate our own manhours cost. It is important to note that adding more people to the project will have diminishing returns. A 20 man-month cost

estimation does not mean you can just throw 80 people at the project and have it done in a week. Also, with this style it is important to tell the customer the uncertainty of the estimation. Research has shown that there is a 75% chance that the actual cost will be within 20% of the estimate. The benefit of this style is that as the project progresses you will continue to get more and more accurate.

Changes

Handling Changes

Proposed changes will be handled with a change order form (see blank form below). Each proposed change will have a cost estimation for the work required to realize the change, and the change request can either be approved or denied. If it is approved the change will be incorporated into the project using Git to track the versioning of the code.

CHANGE ORDER LOG

PROJECT NAME			
LOCATION OF WORK			
CONTRACT NO.		CHANGE ORDER NO.	
REQUESTING PARTY		DATE OF REQUEST	
PROJECT MANAGER		CONTRACTOR	
OWNER		ENGINEER	
CHANGE REQUEST OVERVIEW			
DESCRIPTION OF CHANGE			
REASON FOR CHANGE			
SPECIFICATIONS			
CHANGE IN CONTRACT PRICE		CHANGE IN CONTRACT TIMES	
ORIGINAL PRICE		ORIGINAL TIMES	
NET INCREASE / DECREASE		NET INCREASE / DECREASE	
TOTAL CONTRACT PRICE WITH APPROVED CHANGES		TOTAL CONTRACT TIME WITH APPROVED CHANGES	
COST ESTIMATION		ACCEPTED BY OWNER	
ACCEPTED BY CONTRACTOR			
DATE		DATE	

Proposed Change Order Form

CHANGE ORDER LOG

PROJECT NAME	Abracadabacus		
LOCATION OF WORK	UVU		
CONTRACT NO.	1	CHANGE ORDER NO.	1
REQUESTING PARTY	Partiers-20	DATE OF REQUEST	11/29/2018
PROJECT MANAGER	Brock	CONTRACTOR	Reid Kuttler
OWNER	Professor Sharp	ENGINEER	Reid Kuttler

CHANGE REQUEST OVERVIEW	
DESCRIPTION OF CHANGE	Created a google firebase account for uploading the results of student's abacus attempts. This is in order to keep track of how the students are doing during the lesson. The teacher has an alternate view on the app which shows how the students are doing.
REASON FOR CHANGE	We wanted something to set this project apart and this seemed like an effective tool for teachers to keep track of how their students are doing during the lesson. That way any struggling students can be given extra help.
SPECIFICATIONS	Added sendData class, teacherView class, studentAttempt class, and enterName class. Also modified the Abacus class and Abracadabacus class.

CHANGE IN CONTRACT PRICE		CHANGE IN CONTRACT TIMES	
ORIGINAL PRICE	150 points	ORIGINAL TIMES	215 Hours
NET INCREASE / DECREASE	+10 points	NET INCREASE / DECREASE	+7 hours
TOTAL CONTRACT PRICE WITH APPROVED CHANGES	160 points (10 extra credit)	TOTAL CONTRACT TIME WITH APPROVED CHANGES	222 Hours (7 extra hours)

COST ESTIMATION	215 total hours / 150 total points = 1.43 1.43 * 7 hours = 10.03 extra credit points	ACCEPTED BY OWNER	
ACCEPTED BY CONTRACTOR	Reid Kuttler		
DATE	11/29/2018	DATE	



sharpcraig2450 commented 13 hours ago

Approved for 8 hours per person. This should modify all relevant components of the Milestone document plus the code, interface, use cases, etc.

Delphi Method/Process

Brad	reverse-engineered tutorial database w/ points optional color palette fun music party parrot Easter egg	1 reverse-engineered tutorial 1 database w/ points 2 optional color palette 2 fun music 1 party parrot Easter egg 2 logarithms 1 server 1 Easter egg to read the documentation 1 password to log in 2 tutorial, practice, assignment, test 1 teacher-designed assignments 2 multiplication, division 3 binary, octal, hex, any other base 4 powers	reverse-engineered tutorial database w/ points party parrot Easter egg theme music
Brock	tutorial, practice, assignment, test teacher-designed assignments multiplication, division binary, octal, hex, any other base powers	1 server 1 Easter egg to read the documentation 1 password to log in 2 tutorial, practice, assignment, test 1 teacher-designed assignments 2 multiplication, division	reverse-engineered tutorial database w/ points powers theme music
Linda	logarithms server Easter egg to read the documentation password to log in	3 binary, octal, hex, any other base 1 powers 2 randomized test design 4 screen-shared tutorial for long-distance tutoring	reverse-engineered tutorial database w/ points password to log in theme music
Reid	randomized test design screen-shared tutorial for long-distance tutoring enter name and get grade make it android-compatible accessibility-oriented	2 enter name and get grade 1 make it android-compatible 1 accessibility-oriented 2 make it for blind students 1 give it a fun storyline (pirates attacking, superhero, etc.) 1 pvp abacus competitions 1 user testing 2 square root calculation 1 Boolean algebra 1 theme music	reverse-engineered tutorial database w/ points accessibility-oriented theme music
Myron	make it for blind students give it a fun storyline (pirates attacking, superhero, etc.) pvp abacus competitions user testing square root calculation Boolean algebra		reverse-engineered tutorial database w/ points Boolean algebra theme music
Heather	user testing square root calculation Boolean algebra enter name and get grade make it android-compatible		reverse-engineered tutorial database w/ points make it android-compatible theme music

Mariah	Easter egg to read the documentation password to log in give it a fun storyline (pirates attacking, superhero, etc.) pvp abacus competitions		reverse-engineered tutorial database w/ points pvp abacus competitions theme music
Tanner	server Easter egg to read the documentation password to log in teacher-designed assignments multiplication, division theme music		server Easter egg to read the documentation password to log in teacher-designed assignments multiplication, division theme music

Brad	reverse-engineered tutorial database w/ points party parrot Easter egg theme music
Brock	reverse-engineered tutorial database w/ points powers theme music
Linda	reverse-engineered tutorial database w/ points password to log in theme music
Reid	reverse-engineered tutorial database w/ points accessibility-oriented theme music
Myron	reverse-engineered tutorial database w/ points Boolean algebra theme music
Heather	reverse-engineered tutorial database w/ points make it android-compatible theme music
Mariah	reverse-engineered tutorial database w/ points pvp abacus competitions theme music
Tanner	server Easter egg to read the documentation password to log in teacher-designed assignments multiplication, division theme music

8 reverse-engineered tutorial
 8 database w/ points
 1 pvp abacus competitions
 1 make it android-compatible
 8 theme music
 1 Boolean algebra
 1 accessibility-oriented
 1 password to log in
 1 powers
 1 party parrot Easter egg

Final Three:

Reverse-engineered tutorial
 database w/ points
 theme music

Time:

Brad 8 6 4	Brad 7 6 5	Brad 7 6 6
Brock 6 3 7	Brock 6 4 7	Brock 6 5 7
Linda 4 9 5	Linda 5 8 6	Linda 5 7 6
Reid 5 3 10	Reid 6 4 9	Reid 6 5 9
Mariah 4 8 9	Mariah 5 7 8	Mariah 6 7 8
Heather 9 8 9	Heather 8 7 8	Heather 8 7 8
Tanner 8 8 8	Tanner 7 7 8	Tanner 7 7 8
Myron 7 8 9	Myron 7 7 8	Myron 7 7 8
AVG: 6.5 5.5 7.5	AVG 6 6 7.5	AVG 6.5 7 8

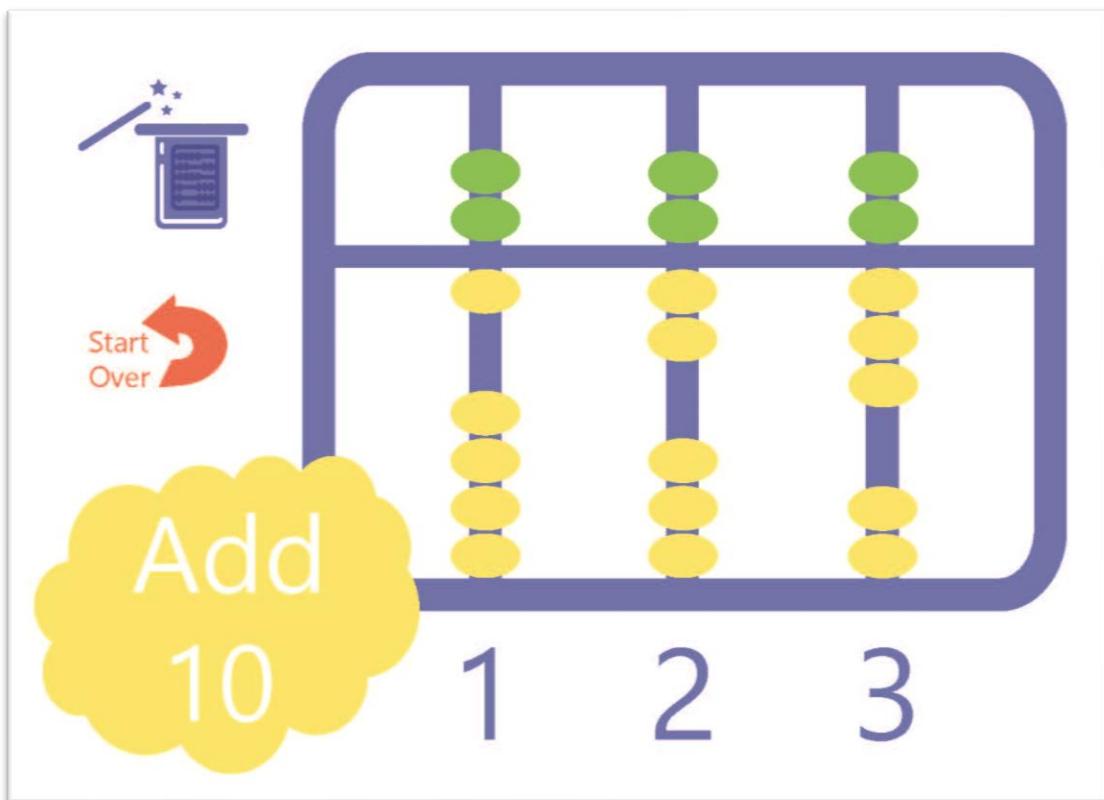
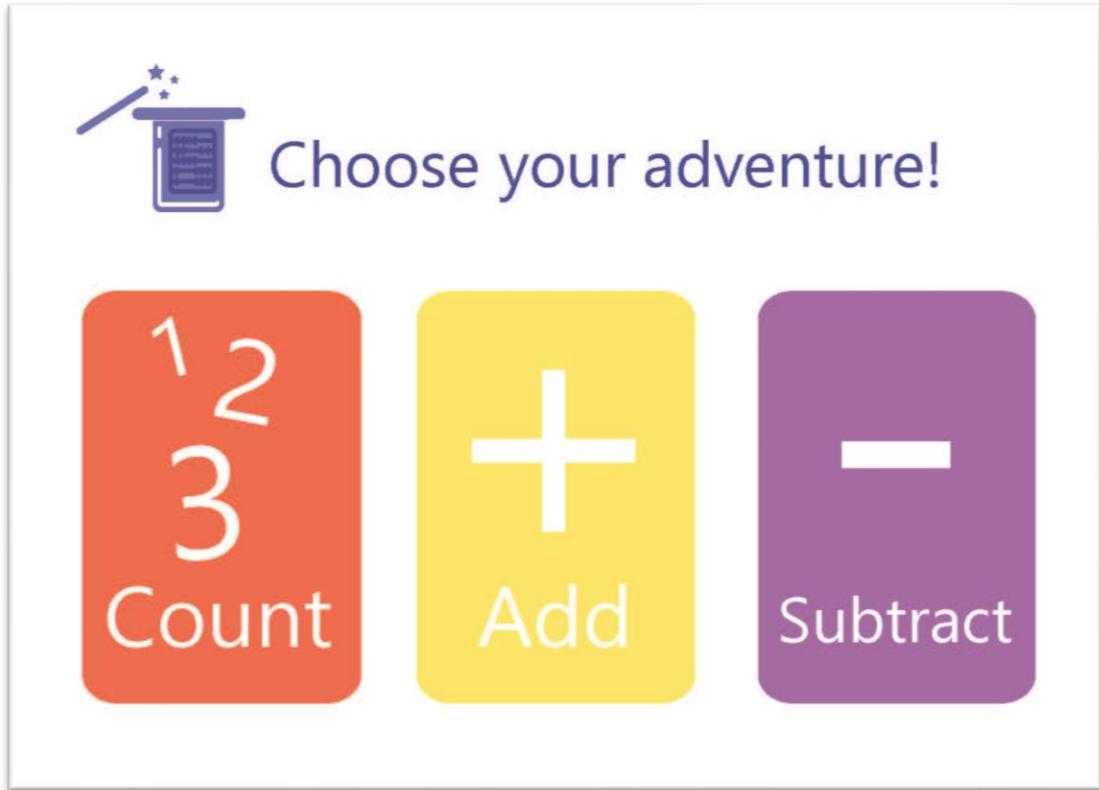
Delivery

It will be distributed via download through a website (myronburton.com/ado)

Base GUI Samples



A welcome screen for the application. At the top, it says "Hello! Welcome to Abracad Abacus". Below this is a question in green text: "What grade are you in?". There are six green rectangular buttons arranged in two rows of three. The top row contains the numbers "K", "1", and "2". The bottom row contains the numbers "3", "4", and "5".



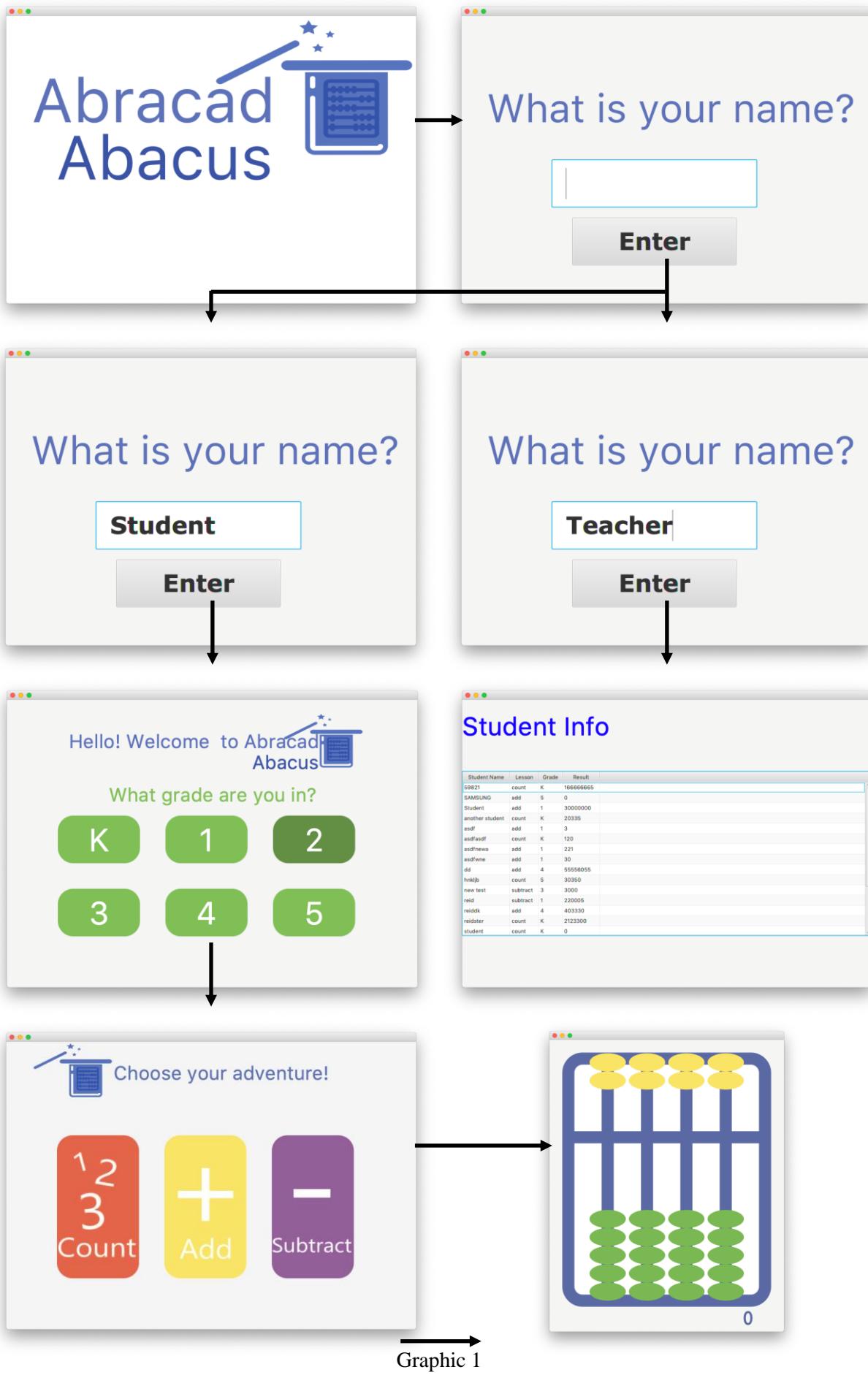
An interactive abacus game. On the left, there's a "Start Over" button with a red arrow. A yellow cloud-shaped button labeled "Add 10" is positioned below the abacus. The abacus itself has three columns of beads. The first column (under the number 1) has two green beads at the top and four yellow beads below. The second column (under the number 2) has two green beads at the top and four yellow beads below. The third column (under the number 3) has two green beads at the top and four yellow beads below.



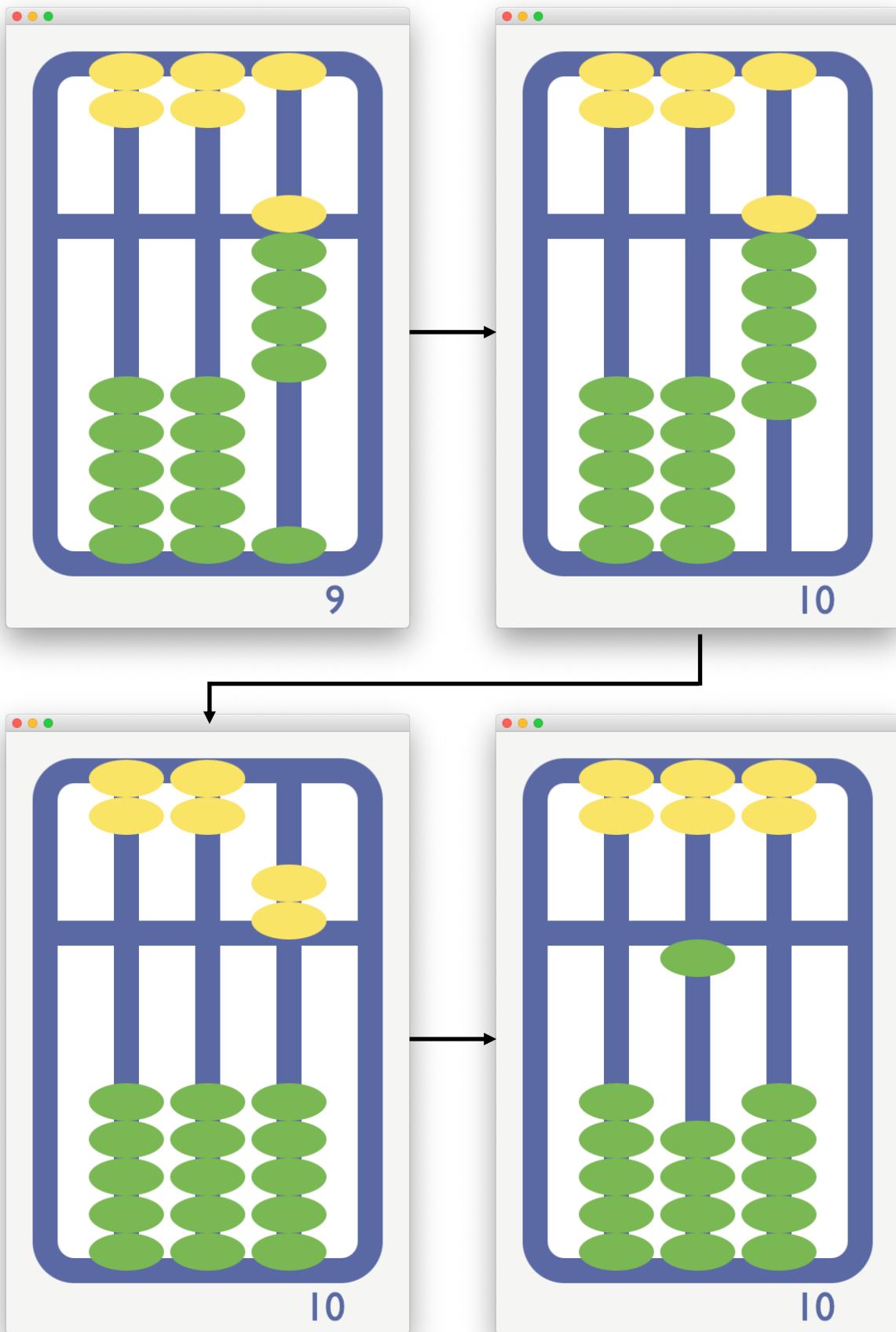
Prototype

Move beads when clicked on:

```
// check to see if we are incrementing or decrementing
// decrement
if (currentCounter == Counter.COUNTED) {
    // move the bead that called
    beads.get(column).get(beadNumber).move();
    // see if there any beads that are below the bead clicked that
    also need to move
    for (int i = 1; beadNumber - i >= 0
        && beads.get(column).get(beadNumber - i).getCounter()
        == Counter.COUNTED; i++) {
        beads.get(column).get(beadNumber - i).move();
    }
}
// increment
else {
    // first move the bead that called this
    beads.get(column).get(beadNumber).move();
```



Graphic 1



```
// see if there any beads that are below the bead clicked that  
Graphic 4
```

```
also need to move
for (int i = 1;
    beads.get(column).get(beadNumber + i).getCounter() ==
    Counter.NOT_COUNTED
    && beadNumber + i < 5; i++) {
    beads.get(column).get(beadNumber + i).move();
}
}
```

Count beads after click:

```
// start fresh
count = 0;

// grabs each list and then each item
// and adds its value to the count.
beads.values().forEach(column -> {
    column.forEach(bead -> {
        count += bead.getValue();
    });
});

// change the textual output of the value
valueOut.setText(String.valueOf(count));
```

AbracadAbacus needs to have a simple design that allows this tool to be transparent in the hands of students, leaving them to feel that they are interacting with an abacus and not as though they were fighting against a program. For this reason a Human Compatible Graphic User Interface (HCGUI) is imperative if this abacus be implemented in the environment. For this reason we have decided to modify greatly how our abacus works.

To make it as simple as possible, all a student has to do is insert their name, select their grade, and choose what type of math is to be performed. This will allocate the number of columns necessary for their project and give tips on how to perform tasks when needed all while being simple and straight forward to the student.

Teachers also have an easy access screen that shows progress of their students. By typing in their name they get a table view of students and what values they have on their abacus. (See graphic 1) Interaction of the abacus comes down to the primary interaction of the user with the beads of the abacus. Students may be informed of the rules of using an abacus such as when moving a fifth bead in a column to reset the column and add a fives bead above the column. (See graphic 3) They must also understand that once the second set of five beads has been made that they must reset the column again, add a fives bead above the column, totaling ten, and then add a bead in the column to the left and resetting the fives bead. This process is outlined in graphic 4.

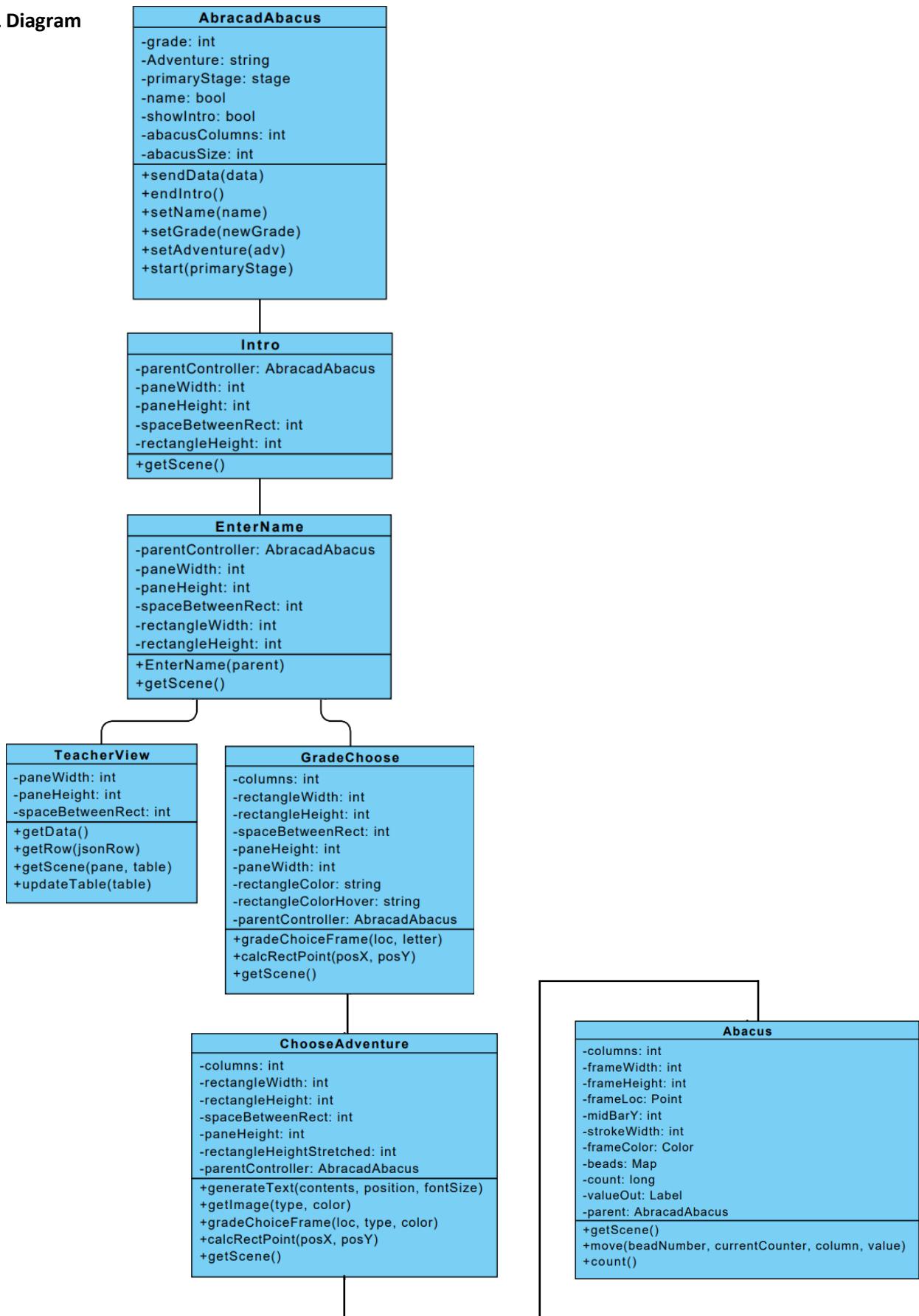
When beads move an update needs to be made to show the sum of the beads. To move the beads we have to check all of the beads in the column that may have to be moved and then move those that should be moved. Once the beads have been moved we will have to tally the value of the beads again to update the decimal output of the display. This is shown in Graphic 5.

While developing the abacus we have implemented a number of test cases to insure that the abacus meet any and all requirements specified. Focusing on the code, these are some of the code based test cases used. Firstly, as code was modified, we used a debugger to make sure that every function had a purpose and was being called correctly following a coverage based testing model. We also included test cases where we would intentionally modify values to insure that the abacus either broke or remained functional as expected. If certain parts could be removed without removing functionality from the abacus and while maintaining specifications then it was done. Completing the modification and removal of pieces of code followed an error based testing model. Another set of testing was done for edge cases. It was discovered that there is a limitation to the number of columns that could be made with the abacus without a major restructure of the code. By limitation of Java's long primitive value, the max value that can be displayed is 9,223,372,036,854,775,807. That means that a

maximum of 18 columns can be used as adding a 19th column introduces the possibility of rolling over the long value which denotes the value of the board. In practice we never display more than 7 columns, but we have found that potential issue and have dealt with it.

Besides code based testing techniques, use based cases have been implemented to insure that the abacus functions as expected. The first, and most basic use case is to introduce someone to the abacus and have them interact with the abacus, moving beads up and down in their respective columns. This was done to insure ease of use as well as proper functionality of an abacus as expected by those who are proficient in the tool. With that testing completed at each update of the master branch, values would be tested after certain math functions performed on the abacus. Among those were multiplication, addition, division, and subtraction. By allowing someone to run these functions on the abacus and checking the output with that displayed by AbracadAbacus, we were able to insure that the application would display appropriate values no matter what value was given. Videos of the use of the abacus as well as photos of the prototype are available online at myronburton.com/ado/

Finally, we have tested the abacus against the requirements specification in a coverage based fashion to ensure that all functionality of the abacus is present. With the change order in effect, that included testing out the networking functionality of the abacus and validating that values would be updated across use sessions of the abacus.

UML Diagram

Chapter Expert Explanation (Chapter 10 – Brock Brown)

We will use a lot of UML Class diagrams

we will use a story board

We will use a use case diagram

Sequence diagrams will be useful.

We will use CRC cards

Prototype Assessment

Between Milestone 1 and the final Milestone, our group decided to throw out the prototype we originally presented –mostly. After discussing it as a team and going over the requirements that were cultivated from our meeting with a teacher, we decided we needed a change. While we scrapped the user-end interface and made it completely different, we did salvage a lot of the backend code from the original prototype.

We kept a lot of the original concept, which isn't saying too much since the basis is a standard abacus, which is what we were tasked to build. We added some other features, such as changing the color of the columns when they are used fully (in groups of five), and we added a number tracker at the bottom of the abacus so students could keep track of what they are calculating.

Another major change that we made was the flow and user pathway for the application, which was one of the more predominant reasons that we had for getting rid of the original prototype. We wanted a product that was more user friendly right from the moment you enter the application. We started with a simple name recognition login, making sure it was simple enough for young children, and the product would tell from the login credentials whether the user was a teacher or a student. From there, it would branch to two different ends of the application: teacher view and a student view. With all these changes on our plate, it was easy for us to decide to dispose of the original prototype and start fresh.

Meeting Logs

Team Name	Meeting Log # 1	Date: 9/27/18	Duration: In class reviews of binders
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	15%	
	Linda Jensen	15%	
	Brad Allred	8.3%	
	Mariah Bleak	15%	
	Myron Burton	15%	
	Reid Kuttler	8.3%	
	Tanner Chamberlain	8.3%	
	Heather Hyer	15%	
Topics Discussed	<p>Team 4</p> <ul style="list-style-type: none"> • We should add how the Wiki will be used • We need more thorough test cases • They didn't have a title on their test cases page <p>Team 3</p> <ul style="list-style-type: none"> • They have nice class diagrams, but they put their code in there as a code snippet, which is wrong • Their report isn't very long compared to the others • We like how big the buttons are for on the GUI but it's pretty rough. • Their chapter summaries aren't really summaries- just saying "This person read this" • The change order form is a little too simple • The use cases look good • They still have it named VSTS in their report instead of Azure <p>Team 2</p> <ul style="list-style-type: none"> • Fancy binder • They didn't do front and back • Spelling mistake: change order forum instead of change order form • Doesn't say who did each chapter • They did a tutorial, which we like and want to use. • The GUI looks really nice • They don't have a class diagram. They have a table of contents but not all the pages are numbered. (they got off on the numbering because the class diagrams aren't there) <p>Team 1</p> <ul style="list-style-type: none"> - Doesn't have a title page - UML diagram is not correct. You're supposed to put your variables at the top (with its classification like int, string), and then functions go below. 		

	<ul style="list-style-type: none">- The GUI looks nice though we're not supposed to be doing multiplication and division yet- Lot of screenshots for Azure Dev Ops, but none of them are explained- They have literal money cost, not in hours. Why does the project manager cost \$100 an hour and the rest \$75?- They don't have test cases. <p>Team 6</p> <ul style="list-style-type: none">- No signatures- Some of their meeting notes are outside of the table- Typo: Scum instead of Scrum- The GUI isn't very graphical... It's just drawn in word instead of showing as how it will look in the app- Lots of code.. Why is their Abacus named Abicus- Screenshots aren't explained. Pages of code and screenshots that aren't explained and are confusing.
--	--

Team 5	Meeting Log # 2	Date: 10/4/18	Duration: Meeting with Client about Baseline
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	13%	
	Linda Jensen	13%	
	Brad Allred	13%	
	Mariah Bleak	13%	
	Myron Burton	13%	
	Reid Kuttler	13%	
	Tanner Chamberlain	9%	
	Heather Hyer	13%	
Topics Discussed		<ul style="list-style-type: none"> - Everyone was involved in this meeting, since this meeting includes the side meeting of just the team. - We discussed backlog planning and sprints in ADO - Figuring out the task planning for the storyboarding, everyone needs to task it out. - Mariah, Heather, and Linda will be meeting with the Educator to get functionality and design input on this coming Tuesday - We need to add stories to Prototype (programming team needs to take the lead on that, since it is more specific) - There's not much we're able to do on the prototype until we meet with the educator to get input- we don't want to go in the wrong direction - Everything is getting tasked out in this meeting and adding all the estimate points. 	
Problems Encountered		<ul style="list-style-type: none"> - There's a lot of backlog planning that needs to happen on our ADO to stay more organized these next few sprints - UML diagram is weak - Keep everything related in the same section - Change the signatures so there's no signatures on the pdf and have it on the printed copy only. - Update the change form to look more professional, and more like the one that the other team copied from the internet. 	

Action Items/ Further Questions	<ul style="list-style-type: none">- Heather will post in Slack the tutorials and examples of online abacus to help Myron figure out implementation- Linda is in charge of doing an example change order form for the documentation.- Check ADO for further assignments each sprint.- Everyone go and add tasks with estimations to all the stories- Add test cases and use cases within the GUI- Do landscape on screen prints to see more- Tanner: make diagrams so that the actors are easier to see where people share things they interact with. Overall it's really good. Add Administrator of teacher in the use cases- Put the three task lists together from the educator, our teacher, and the requirements.
---------------------------------	--

Team 5	Meeting Log # 3	Date: 10/9/18	Duration: Meeting with Educator (Pam Hyer)
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	8.3%	
	Linda Jensen	25%	
	Brad Allred	8.3%	
	Mariah Bleak	8.3%	
	Myron Burton	8.3%	
	Reid Kuttler	8.3%	
	Tanner Chamberlain	8.3%	
	Heather Hyer	25%	
Topics Discussed		<ul style="list-style-type: none"> - More important to have both- can give them a problem but also be able to go through curriculum within the app - We need tutorials for sure on how to use an abacus even as a reminder - The tutorial would show an example problem that would show the abacus actually moving and it would be interactive. <p><u>Learning Requirements for each grade</u></p> <ul style="list-style-type: none"> - K: They need to add and subtract within 10 (any digits within 10 but the sum is no greater than 10). Should be able to count to 100. Big deal is getting 1-1 correspondence, like 1 bead is 1. Helping them understand that this visual means this number. For Kindergarteners should also have just a COUNTING option. Move a bead and see the number change. Questions like: If I have 3 bears and add 2 bears, how many bears do I have. Even if the abacus turned into bears instead of beads. (ADD THIS AS A CHANGE ORDER?) Counting cars, chips, bears, colors, cubes that snap together, so have round, square, and shape. Count to 100 by 1s and by 10s. Be able to add 3 to 7 without starting to count from 1 again. $3+5 = 5+3$ (Number permanence). Add and subtract within 10 with objects and then add and subtract within 5 with numerals. From any number between 1-10 what number do you need to add to make it 10? Called "Make a 10". - 1: Decomposing (taking apart), Putting together (making sets). Word problems are huge, though not as important on an abacus. Base 10 operations. Subtracting within 20. Adding within 100 as long as it's a single digit with a double digit, or a double digit and a multiple of 10. And subtracting in multiples of 10. $90-80=70$. Need to do without counting. We have to move past 10 when adding. 	

	<p>They have to count up to 120. Instead of just pairs of numbers being added, they can have up to 3 numbers being added but it still needs to be less than 20. Be able to count by patterns. Like by 2s or by 5s. Subtract within 20. If you have a 13-4, 13-3 is 10 and then -1. Get to 10 within subtraction. THAT is the hardest one to teach kids because it's more abstract and has more steps. Trying to get them away from counting each object. That one would be really important and teachers would love it.</p> <p>Having the whole column change color when you're supposed to switch to the next column would be helpful. One important thing is $8 + \text{what} = 11$. Fill in the blank problems. Being able to know that a 10 + a 9 is 19. Most first grade teachers stick with add and subtract within 20. Compare two 2 digit numbers.</p> <ul style="list-style-type: none">- 2: Use subtraction with an unknown. $10 - \text{blank} = 8$. All of these requirements are cumulative. Should be able to just know that $12 - 5 = 7$. Add and subtract within 100. Know if a number is odd or even by counting by 2s. Understand that the 3 digits on a 3 digit number mean 100s 10s and 1s. Count within 1000, skip count by 2s, 5s, 10s, and 100s. Give them a number and have them represent it on the abacus. Compare 2 three digit numbers to see which one is bigger. If you wanted to you could do 2 abacuses.- 3: Multiplication starts. It would be too hard to teach the concept of multiplication on an abacus... To help them know that $5 * 7 = 5$ groups of 7. Multiplication on an abacus might just be the actual doing the problem instead of teaching how multiplication works. Basic understanding of division is the reversing of multiplication. Unknown numbers in multiplication and division problems. $25 / \text{what} = 5$. Multiplication and division within 100 but with no remainders. Rounding is huge in 3rd grade. Rounding to the nearest 10 or nearest 100. Add and subtract within 1000. Add and subtract within 1000 and all the times tables 1-10 and 10s to within 100.- 4: Multiply by 2 digit numbers. They don't have to be multiples of 10. 35 is 5 times as many as what? And 7 times as many as what? Wording questions is a weird way. Learning how to represent a verbal statement as a number statement. Find out 2 different ways to multiply to 24. Find all the factor
--	---

	<p>pairs. (12*2, 24*1, etc). Show that factors wouldn't work because you have remainder over here...</p> <p>Multiply 2 digits by 2 digits, or 1,2,3,4 digits by 1 digit. Division is up to 4 digit quotients with a 1 digit divisor. Never over a 1 digit divisor.</p> <ul style="list-style-type: none">- 5: They get into algebra a lot. Combined operations. Add 8 and 7 and then multiply by 2. They learn order of operations in 4th grade. They learn it as an already written statement though, they can't create it from a word problem. Figure out how to do multiple operations on one abacus? HOW does multiplication work on an abacus? They can do powers in 5th grade but only 10^2, 10^3, etc... Multiply and Divide Decimals but not sure if we can do decimals on an abacus. Long Division is a big part of 5th grade also with 2 digit divisors. 5th grade has far fewer standards but a lot more practice needed with these more complex topics. A lot of practice in 5th grade.- Teachers always want a way to track progress of students, or else you would have to look around and check each ipad or computer.- Grade logins: K and 1: if you have a picture symbol that represents them it would be best. They knew that they were a dragon and a red apple. Picture logins are necessary. 2nd grade they can type in a name. Apparently Kindergarteners have to do 7 digit usernames and passwords all the time. It's not easy or good, but the kids can do it.- Assessments make sense as long as the lesson has interactive elements where they can get feedback where they go. Or corrections from the program when they do it wrong or when they submit a problem. It should be able to tell the user what they did wrong. Even if they made them try it again before moving on.- Options can time out, "You've done enough counting today, now choose another activity". Their attention span is less than 10 minutes in Kindergarten. Attention spans are usually the amount of the age in minutes. Max of 10 minutes a day in K and 1st spending in this app. 2 and 3rd graders could go maybe 15. Even giving up any time to a software program is tough for a teacher.- Do a bunch of 1 minute activities for the little ones especially. Never let a session be more than 15 minutes. Little colored bar that is going across the bottom so they can know how long this is and how
--	--

	<p>much more they have to do. Especially because they can't get a huge error ending message. As soon as they finish an activity it gives them a choice again. But gray out an activity after a while of playing the same one. You have to finish the other sections before you can get new questions in the favorite one.</p> <ul style="list-style-type: none">- Being able to see their own progress is huge.- Digital rewards are AWESOME. You got another star in your bank (or something like that). You can earn stars for every activity and then every so often they could use their stars to buy stuff to show up in the game. If the tutorial guy was a robot then you could use the stars to buy different hats for the robot or whatever. Avatar builder as you earn things for it. They loved stocking the rocket ship the most.- http://www.corestandards.org/ for math for the particular grade. It's in graduate level language but it's a great resource.- You could have the buttons on the front page fill up with color as they've done each of the assignments in each.
Action Items/ Further Questions	<ul style="list-style-type: none">- The educator used a website with kids and had them click their grade level and kindergarteners were able to click K and all the grades up could pick too.

Team 5	Meeting Log # 4	Date: 10/25/18	Duration: Meeting with Professor Sharp outside of classroom
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	18.75%	
	Linda Jensen	18.75%	
	Brad Allred	6.25%	
	Mariah Bleak	18.75%	
	Myron Burton	6.25%	
	Reid Kuttler	6.25%	
	Tanner Chamberlain	6.25%	
	Heather Hyer	18.75%	
Topics Discussed		<ul style="list-style-type: none"> - Milestone 2 and 3 will now be combined. - The next major exercise should cover the functional and non functional requirements and the delphi method. - You should have requirements for each of the 5 actors, including ourselves. You would have to do maintenance, docs that explain to the sharegolder what our intentions are, come up with images and animations for how the interface should work. - Sometimes need to do meta stories to figure out what we need for the project. Our non-functioning requirement is based on a certain version of a product so that we don't want to worry about updating mid-project. - WE better see a really good baseline document from the requirements. We create change orders from the delphi list that we create in our team. - Baseline requires a requirements specification - We write the requirements specification and we're supposed to prepare one for each of the 5 actors, and that is inspired by the milestone documents online. We add in things like "We will be creating this on visual studio" or something - It's things we need to do to make the thing to figure it out. - Everything that's required from the customer's list is what we're required to do. Develop the lists of tasks. - If you prepare the requirements specification from the requirements solicitation and you must elicit them from the shareholders (we're shareholders) - We're assuming that it's on each individual device, forget cell phones. - 5 actors: IT, student, teacher, team, admin (our teach) 	

	<ul style="list-style-type: none"> - Notes from educator will go into the baseline document for us. - Prepare all records and documents and assign it to Linda. Ex. A requirement is: having meetings. - Prototype just demonstrates that we can make this work, but when we throw out that you'll do mult and division that involves a major interface change. Which is where change orders come in. - If the teacher doesn't understand it then it was a bad design or it was a bad presentation - Make sure this is going on the big picture.
Action Items/ Further Questions	<ul style="list-style-type: none"> - Get better at being chapter experts. We should really be following the book more - We are not doing the app on the internet! Don't add internet capability, we're going to make it all on individual components. - Linda: make the document as the requirements specification. Track the assigned task to that list of requirements. Change orders are for on top of that. Req specification should have space for 3 or 10 or 2 tasks. In the tasks you should have what requirements it will cover. Summarize it as best you can, because the more detail it is the harder it is to maintain it.

Team 5	Meeting Log # 5	Date: 10/30/18	Duration: Team Meeting
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	13%	
	Linda Jensen	13%	
	Brad Allred	13%	
	Mariah Bleak	9%	
	Myron Burton	13%	
	Reid Kuttler	13%	
	Tanner Chamberlain	13%	
	Heather Hyer	13%	
Topics Discussed		<ul style="list-style-type: none"> - Discussing the mentor discussion outline and preparing for the meeting with our Mentor that will be on Thursday - The chapter expert is the one that needs to add to the requirements - We need to do bullet points in the chapter summaries for our suggestions for the requirements. 	
Action Items/ Further Questions		<ul style="list-style-type: none"> - Put milestone 1 documents into the wiki for reference - Everyone needs to update their chapter summaries to include the suggestions for our requirements specification - Publishing a document in the wiki for all of us to edit our chapter summary suggestions. - Everyone before tomorrow needs to go back through their chapters and add in things to the wiki 	

Team 5	Meeting Log # 6	Date: 11/1/18	Duration: Team Meeting
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed	<ul style="list-style-type: none"> - Making some goals for our final milestone so that we can be prepared. - Looking at the Delphi process that each of us sent in to Brock - Going over non-functional and functional. Non-functional doesn't change the functionality of the project (like adding a color to the GUI) <p>From Teacher</p> <ul style="list-style-type: none"> - Go through milestone 1 and anything you assumed you didn't have to have- you do! Establish and assign how you will do it. Do a WBS. Put people at the bottom of that structure that feel like they can add to that. - Team manager can look at the WBS and add people as necessary to finish deadlines and have it all laid out. Have it defined instead of locked in. Anticipate for the project plan - At a minimum have 1.2, 2.1 completed plus at least a stab at the other stuff. If you're going to use pages 40-41 you need to have an outline done as to where you are. Use the chapter experts to assign each of these sections. Listing the progress we're making in the outline document. Everyone should be adding to this document by having it tasked out to everyone - Simple WBS: group requirements, they develop into work items, code, documentation, cost estimation, some of the major big chunks. Then put below that smaller chunks and then you stick some people on the smaller chunks. And once you've done that it creates the Pert chart. - Break documentation into the smaller pieces. In the timebox it's going to get done by the end of this semester. Turn that into a Pert chart which is the one that says that the weekly documentation is an 		

	<p>hour a week. And the milestone doc is 4 hours per week and you add it up and compare it to the rest of the time. Everyone should be rotating through this and working the same amount</p> <ul style="list-style-type: none"> - The book discusses the processes until chapter 14. A chapter expert should be able to tell everyone what each thing in your chapter means. Be responsible for your chapter so that you can be responsible for part. - This outline should've happened at the start. - What you come in here with in the review is the state of every part of this outline document. He will review it and let us know how we're doing. There should be a part in each part of the review. - Just update the document online and he should be able to see it. We probably should update him when we want him to read through it. - You should have your requirements document done. The 5 actors and their requirements. That is what we FOR SURE need to have by next Thursday. - Delphi is after all this and then that's when the change order comes in. - When you do cost estimation, do it by hours. Do the cost estimation for the change order and that's how many extra credit points we will barter for. That's how we can see how good we've done with estimating cost. - Create a state chart for the milestone "done" "almost done" "need to be started" etc.
Action Items/ Further Questions	<p>Goals for this week:</p> <ul style="list-style-type: none"> - All chapter experts edit their summaries to include suggestions for our project and documentation - Linda: Create a change order example form - Heather: Update feature requirements for this last milestone on ADO - Heather: Non-functional objective: put together color schemes for GUI - Myron, Reid, Tanner: Meet together and figure out what's needed for the prototype coding - Brad: Check-in/Check-out history - Brock: Storyboard first draft <p>Week from today we should be launched to take off into the details of everything</p>

Team 5	Meeting Log # 7	Date: 11/8/18	Duration: Team Progress Meeting with Client
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed		<p>Expecting</p> <ul style="list-style-type: none"> - Seeing the completion of chapter 1 and 2 in documentation. - Starting building stuff from chapter 4 and chapter 8. Remember in Gantt chart to include time boxes. - We should have 4 stories (for 4 weeks) and we should put the tasks in each story based on the Gantt chart. We try to estimate our time because of the Gantt chart - Take all the stuff from milestone 1 that we're done with (like choosing one method over the other) - M1 documentation should be added to the end of the 5th story in this milestone - Add slack time into the Gantt chart - What is our software quality plan? - V&V verification and validation. We did what we're supposed to do and we did it the right way - Not just a list of requirements plus a brief statement about what it does and what it means - We're in a competition to get this project. - Make sure our prototype addresses risk management. What if we had to have disability. - Have we addressed and thought about the risks? - If you finish something don't wait for way down the road for teacher to look at it. Put all of teacher's tasks in a separate story. <p>Additional things</p> <ul style="list-style-type: none"> - As shown by our state diagram... make sure that everything applies to what we're using. Make sure you don't see any risk problems in these last 4 chapters 	

Action Items/ Further Questions	<ul style="list-style-type: none">- A week from today he will have looked at things to check on things. Pay attention to time boxes and make sure we have things finished.- Get ready for change order forms to get extra credit forms. Do a time estimate to achieve it and then somehow show a calculation for how we get to the amount of extra credit. 120 points per how many man hours. That's the ratio you'll do for a change order form. Then we'll negotiate. Just make a story that will be a proposal of an extra feature for teacher to approve or not and THEN we make the file for the change order form.- Your idea for how you'll deal with the last 4 chapters should continue on through the project.- Make a team org chart. A series of org charts for each story and task. These are for figuring out who's in charge of getting things done.- Take the use case, code snippet, GUI, test case to all go together in the documentation.- Pages 40-41. You need to have an organization of final milestone of those items on page 40-41. Are we going to do test cases, GUI, snippets, and use cases all together as one group instead of all separate groups. He wants to see that, the complete project plans, all these charts by next time.
---------------------------------	---

Team 5	Meeting Log # 8	Date: 11/13/18	Duration: Team Meeting
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed		<ul style="list-style-type: none"> - Do a blank org chart and make one for documents, one for code, whatever. Put everyone's names in the boxes and everyone will put their level of availability for involvement for each task. - The Gantt chart should've been made after the Pertt Chart. - Need to have configuration management done before the others. - All diagrams should come from chapter 10, the class diagrams come from that and the code should be moving along - The chapters in gray need to be a big dent during Thanksgiving break - Everyone agree and understand where we're at. - Look at the Gantt chart to be able to see who has the time for each thing 	
Action Items/ Further Questions		<ul style="list-style-type: none"> - Do the org charts with the hours and the task tracking so that everyone can tell who's doing what. - Pert Chart - Add the man hours to the Gantt Chart, and who's on each task so that people can see who's on which assignments - Brock needs to be sending out assignments for us to do during the break while he's in Mexico 	

Team 5	Meeting Log # 9	Date: 11/29/18	Duration: Team Progress Meeting with Client
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed		<ul style="list-style-type: none"> - You can use chapter 11 to organize quite a bit but not everything - Sit down and draw out the formatting of where everything will go for the documentation - Organize so the GUI, code snippets, UML, class diagrams involved, and do that into use case diagrams. You might end up with 3 pages on just student use cases. - Be able to do views relative to what we're talking about - Right now the software development plan should be done and we should be organizing our code and organizing the final documentation - We can't use a test driven development technique because we already have so much code written - We can use data flow coverage for sure 	
Action Items/ Further Questions		<ul style="list-style-type: none"> - Give the reason for Java in part of the description of software architecture - Describe whether you should keep this prototype or not. That goes into chapter 11. - Compile it so that we that someone won't have to go back and forth. - Add to chapter 11 or 12 that Java we use objects because they're easier to manipulate. - V&V should be part of the quality control document. - We should have listed the major requirements of each player. - For software maintenance be able to say how we will maintain the fixes on the program. Will people go out to each school? What do we do? - Make a change order for tracking students scores. - Make a change order for Make a reverse tutorial (you randomly insert a number and then move the beads) 	

MILESTONE 1

Team 5

Team Manager: Brock Brown

Azure Dev Ops: Brad Allred

Lead Programmer: Myron Burton

Programmer: Tanner Chamberlain

Programmer: Reid Kuttler

Lead Architect: Mariah Bleak

Scrum/QA Lead: Heather Hyer

Scribe: Linda Jensen

Software Engineering (CS 2450) - Fall 2018

9/04/18 - 9/25/18

Table of Contents: (page numbers based off the original printed version)

2-10... Meeting Logs

11... SCRUM and Kanban Research

12... UML Diagrams

13-14... GUI Layout

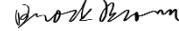
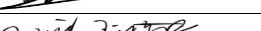
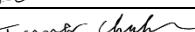
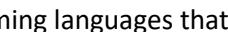
15-17... Azure DevOps (Visual Studio Team Services)

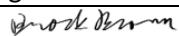
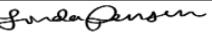
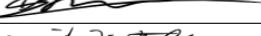
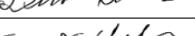
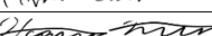
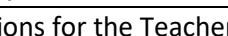
18-22... Backlog and Sprint Decomposition

23... Use Case Examples and Test Case Examples

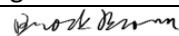
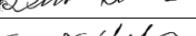
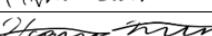
24... Change Order Form

25-31... Chapter Summaries

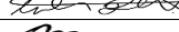
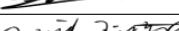
Team Name	Meeting Log # 1	Date: 8/28/18	Duration: 1 hour
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed	<ul style="list-style-type: none"> • Programming languages that team members are comfortable with. • Strengths and Weaknesses of everyone in the group. See "Finished Items" for assignments • Communication (email, slack, phone numbers) 		
Finished Items	<ul style="list-style-type: none"> • Traded Joshua Blackhurst for Bradley Allred. • Gained Heather Hyer from Team 3 for QA Testing and Agile Processes. • The official programming language will be Java because that is what most people in the group are comfortable with, and will have the most support in doing. • Linda will be Lead Scribe but will help with GUI, Myron will lead GUI but will help with Scribe. • Team Manager: Brock Brown • Foundation Server: Brad Allred • Lead Programmer: Myron Burton • Programmer: Tanner Chamberlain • Programmer: Reid Kuttler • Scribe: Linda Jensen • Lead Architect/ Designer: Mariah Bleak • Scrum/Agile/XP Master: Heather Hyer 		
Unfinished Items	<ul style="list-style-type: none"> • Everyone still needs to finish Module 0 and read Chapters 1-3 • Assignments of Chapters of the book 		

Team Name	Meeting Log # 2	Date: 9/11/18	Duration: 1.15 hours (class period)
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed	<p>Developing Questions for the Teacher:</p> <ul style="list-style-type: none"> • Make sure that Java is approved as a programming language? • What exactly are we tracking in Team Foundation Server? Is that for documents or is that for how we plan out tasks/ Can we use something else for task tracking? • Is cost estimation/scheduling/budget tracking on Team Foundation Server? Is that different from task tracking? • Do we need a 100% working abacus by the end of the course, or will it just be a prototype still? • Heather and Mariah compiled more questions for design aspects of the program (need design questions answered before UML can be created) 		
Obstacles Encountered	<ul style="list-style-type: none"> • Only one of us knows how to use an abacus (Myron) • VSTS instead of TFS, the teacher said it's easier because you can create new projects easily. 		
Finished Items	<ul style="list-style-type: none"> • Going to use Git instead of Azure • Going to use Scrum and Kanban • Myron has a server that he can host the abacus at. He has a domain and knows how to build websites and servers. He can also help with the cloning process using Git. • Heather set up a Trello for us to track our tasks before the TFS gets set up. 		

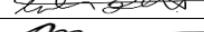
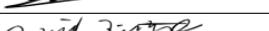
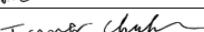
Unfinished Items	<ul style="list-style-type: none"> • Make UML Diagrams • VSTS Setup (Brad) • Task Tracking • Decomposing Each milestone (scheduling) • Use test case examples • Go through milestone 1 as a group
Notes	<ul style="list-style-type: none"> • Heather and Linda will work on the design of the abacus • We don't have a designated tester, but Heather works in QA and will take the lead on that with help from other team members. • Everyone needs to put their github information on slack so that Reid can add us into his repository for the abacus prototype • Every Tuesday from now on is instructor, every Thursday is client • The app is for elementary students (K-5)
Action Items	<ul style="list-style-type: none"> • Mariah take the lead in UML Diagrams • Heather take the lead on test cases • Everyone learn how to use an abacus • Linda and Heather take the lead on design mockups

Team Name	Meeting Log # 3	Date: 9/13/18	Duration: 1.15 hours (class period)
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak (on Slack)	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed		<ul style="list-style-type: none"> Meeting schedules VSTS and how to set it up. Cross collaborated with other teams 	
Obstacles Encountered		<ul style="list-style-type: none"> We didn't know we had to email to schedule a time to meet with the Client, we assumed that it would be every Thursday. So we now don't have a meeting with the Client and are at a disadvantage 	
Finished Items		<ul style="list-style-type: none"> Using Scrum and Kanban Set up an outside team meeting for Monday night at 7 in the library Emailed the client to set up a meeting for thursday 	
Unfinished Items		<ul style="list-style-type: none"> UML Diagrams Documentation Decomposing Milestones into the VSTS Going through the entire Milestone 1 and tasking it out completely 	

Action Items	<ul style="list-style-type: none">• By Thursday everyone needs to have finished reading their assigned chapters and writing a small summary to send to me so that I can put it in the documentation• By Tuesday Myron and Mariah will have a working draft of the UML diagrams
--------------	---

Team Name	Meeting Log # 4	Date: 9/17/18	Duration: 1 hour (meeting in the library)
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed		<ul style="list-style-type: none"> • A VSTS demonstration from Heather. • A UML demonstration and walkthrough from Mariah. • Use Case assigned to Tanner and Test Case assigned to Myron • Tasked out things on VSTS 	
Obstacles Encountered		<ul style="list-style-type: none"> • We need to figure out the story points for each • This milestone will be heavier on the documentation and setup members, and later milestones will be heavier on the programming team. This is fine, just needed to be addressed. 	
Finished Items		<ul style="list-style-type: none"> • Brock finished the 3 reasons for against Kanban and SCRUM. It's been added to the documentation 	
Unfinished Items		<ul style="list-style-type: none"> • Sprint 2 is starting tomorrow and we still need to finish reading our chapters for sprint 1. 	

Notes	<ul style="list-style-type: none">• Do we want something that's already programmed in or should the teacher be making these exercises from scratch (where does the curriculum come from? This is a question for the client on Thursday)
Action Items	<ul style="list-style-type: none">• There has been some debate on programming language. The programmers will figure it out and get back to the group.

Team Name	Meeting Log # 5	Date: 9/20/18	Duration: 1/2 hour (in class meeting with Client)
Team Members (* team leader)	Name	Contribution	Signature
	*Brock Brown	12.5%	
	Linda Jensen	12.5%	
	Brad Allred	12.5%	
	Mariah Bleak	12.5%	
	Myron Burton	12.5%	
	Reid Kuttler	12.5%	
	Tanner Chamberlain	12.5%	
	Heather Hyer	12.5%	
Topics Discussed (Client Answers to Questions)		<ul style="list-style-type: none"> • Grade level is K-5 • Go to the education section of UVU to figure out how to teach primary level kids • Time level of kids on the app needs to be flexible. • All we're worried about is the interface on this prototype • This should be a flexible tool and be able to new curriculum, and to have presets • How do you give a kindergarten kid a tutorial? We need it as a presentable tool • It's a piece of software we're offering to schools for those that want it. • Don't get too broad with this. Our job is to design an interface that will work with K-5 children to use an abacus. How the teacher uses it is not an issue right now. It's not important for Milestone 1 • Adjust the abacus for K-5 levels, Kindergarteners don't know numbers up to 1000 but 5th graders do • Maybe on a tablet, maybe on a cellphone, maybe on a computer, maybe ONLY a computer. We don't care yet, design needs to meet the educational parameters. How they use the tool is up to them. • We don't know what we have to do until we talk to an educational resource • We need cost estimation, planning, etc. • He wants us to have it so if you type in a number, it will show up in the implementation of the abacus • 1s and 5s will be a different color, but make sure a colorblind person will be able to tell the difference still • In a real abacus you drag it, we don't want it to just click • Make it customizable column numbers (put a setting in so you can change the mode) • If they pass a test, they can move up a level (with more columns) 	

	<ul style="list-style-type: none">• Change the name to be easier for kids?? He had trouble reading it.• Don't limit your thinking to someone else's idea. We've gone way too far for these app ideas, we need to dial it back and go a lot more basic.• Do we need passwords? Logging in is too complicated. Just save what level they're at.• We're doing way too much. No divide, no multiply. No assignments. Meet the requirements of milestone 1• Only need functions for add and subtract• Don't go beyond what they're asking for, do what they're asking for very well.• Addition, Subtraction, ability to change the columns, and that's IT
Finished Items	<ul style="list-style-type: none">• Marking story points in Azure Dev Ops

SCRUM and Kanban Research

SCRUM:

Support of SCRUM:

- Set sprint periods facilitate the continuation of the project, easier and faster to catch problems in development.
- The mandatory 15-minute meetings help to keep everyone accountable at all times.
- The product owner dictates the product, which keeps it unified.

Against SCRUM:

- The mandatory 15-minute meetings can negatively affect the amount of informal interaction
- The structure doesn't allow for a larger project that might take longer than the allotted time.
- Due to predefined sprints, process is less flexible.

Kanban:

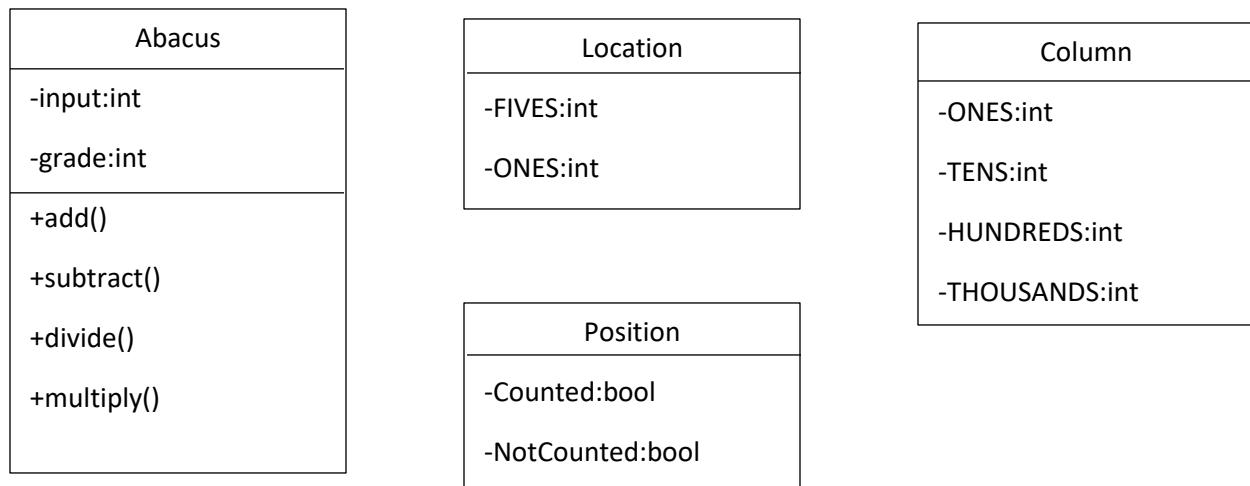
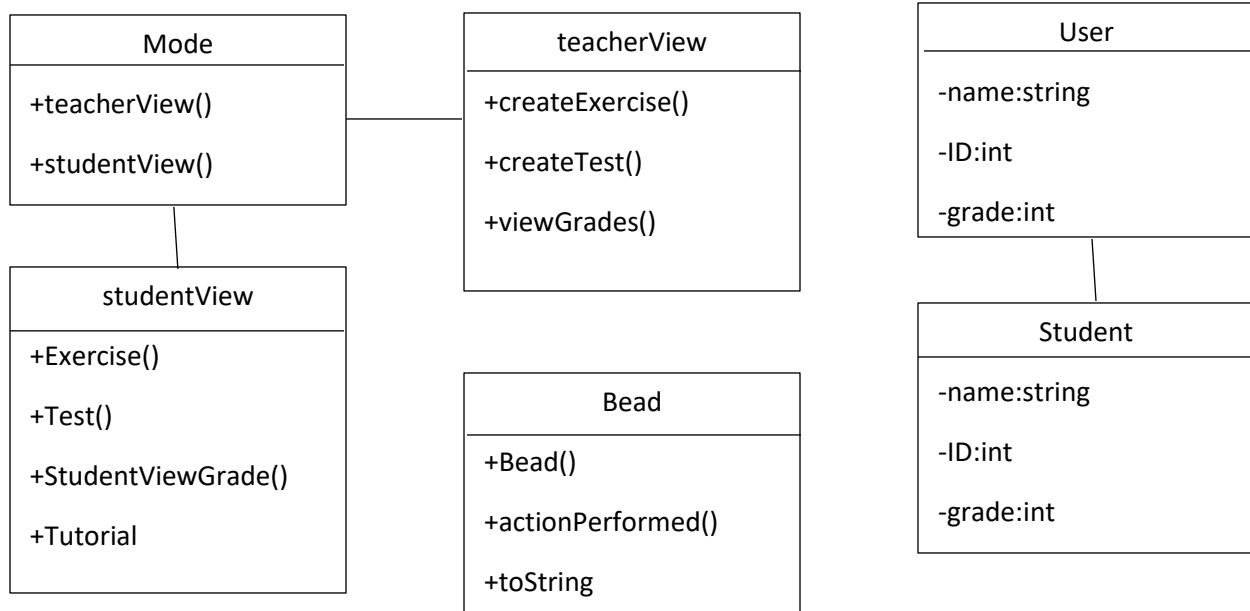
Support of Kanban:

- Limiting Work-In-Progress (WIP) forces small changes to be implemented quickly
- The quick, small changes will give rise to new questions quickly, so we can tackle the next problem
- Encourages each person to take ownership of his/her part in the project

Against Kanban:

- Limited WIP may produce the Convoy effect, where one large task slows everything else down
- Definition of "done" could very easily be confused and project parts could move more quickly than they should.
- Lack of deadlines could lead to conflict over a couple of procrastinators

UML Diagram



GUI Layout

Welcome to AbracadAbacus!

What grade are you in?

K 1
2 3
4 5

[Back](#)

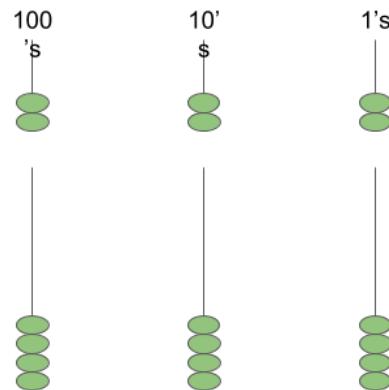
Abacus:

```
// obtain column values and then build the number of beads necessary
int cursor = 0;
Column[] cols = new Column[column];
for (Column c: Column.values()) {
    if (cursor < columns) {
        cols[cursor] = c;
        cursor++;
    } else {
        break;
    }
}

// allocate array with number of beads necessary
beads = new Bead[columns * 7];
// insertion gives bead ID
int insertion = 0;
// build all the beads
for (int i = 0; i < columns; i++) {
    // first two beads are the fives beads.
    beads[i * 7] = new Bead(cols[i],
        Location.FIVES, Position.NOT_COUNTED,
        insertion++, this);
    beads[i * 7 + 1] = new Bead(cols[i],
        Location.FIVES, Position.NOT_COUNTED,
        insertion++, this);
    // loop to add the five ones beads
    for (int j = 2; j < 7; j++) {
        beads[i * 7 + j] = new Bead(cols[i],
            Location.ONES,
            Position.NOT_COUNTED, insertion++, this);
    }
}
```

[Back](#)

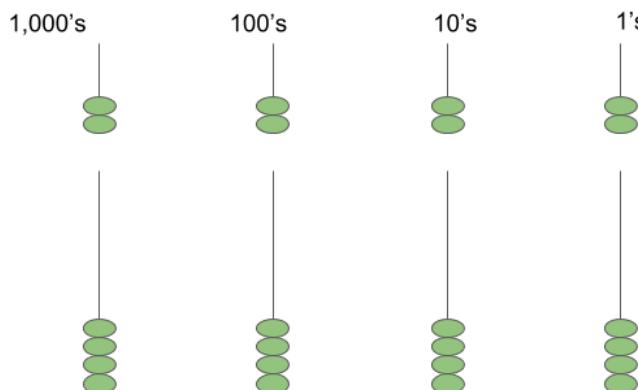
Abacus:



```
// allocate array with number of beads necessary
beads = new Bead[columns * 7];
// insertion gives bead ID
int insertion = 0;
// build all the beads
for (int i = 0; i < columns; i++) {
    // first two beads are the fives beads.
    beads[i * 7] = new Bead(cols[i],
Location.FIVES, Position.NOT_COUNTED,
insertion++, this);
    beads[i * 7 + 1] = new Bead(cols[i],
Location.FIVES, Position.NOT_COUNTED,
insertion++, this);
    // loop to add the five ones beads
    for (int j = 2; j < 7; j++) {
        beads[i * 7 + j] = new
Bead(cols[i], Location.ONES,
Position.NOT_COUNTED, insertion++, this);
    }
}
```

[Back](#)

Abacus:



```
// allocate array with number of beads necessary
beads = new Bead[columns * 7];
// insertion gives bead ID
int insertion = 0;
// build all the beads
for (int i = 0; i < columns; i++) {
    // first two beads are the fives beads.
    beads[i * 7] = new Bead(cols[i],
Location.FIVES, Position.NOT_COUNTED,
insertion++, this);
    beads[i * 7 + 1] = new Bead(cols[i],
Location.FIVES, Position.NOT_COUNTED,
insertion++, this);
    // loop to add the five ones beads
    for (int j = 2; j < 7; j++) {
        beads[i * 7 + j] = new
Bead(cols[i], Location.ONES,
Position.NOT_COUNTED, insertion++, this);
    }
}
```

Azure DevOps setup

We will be using Azure DevOps for our management software. The workaround that we decided to use for Permissions is to have a universal account, which anyone can access. We also have plans to swap permissions for individual team members as necessary as the project progresses.

Display Name	Username Or Scope
Bradley Allred	brad-allred@hotmail.com
Brock Brown	brownbr32@gmail.com
dummyparrot@outlook.com	dummyparrot@outlook.com
Heather Hyer	heatherahyer@yahoo.com
Linda Jensen	linda_jensen@outlook.com
Mariah Bleak	mlbleak@gmail.com
Myron Burton	pichu766@gmail.com
Reid Kuttler	reidkuttler@gmail.com
sharpraig2450	sharpraig2450@outlook.com
Tanner Chamberlain	tannerchamb@gmail.com

Technologies we plan to implement are as follows:

1. Boards (including Work Items, Backlogs, and Sprints)
2. Repos (source control)
3. Pipelines (release management)
4. Test Plans

Since we found we could use Azure as a one-stop shop for most of everything we need for development, we thought that if we committed to learning these processes we could really streamline our progress.

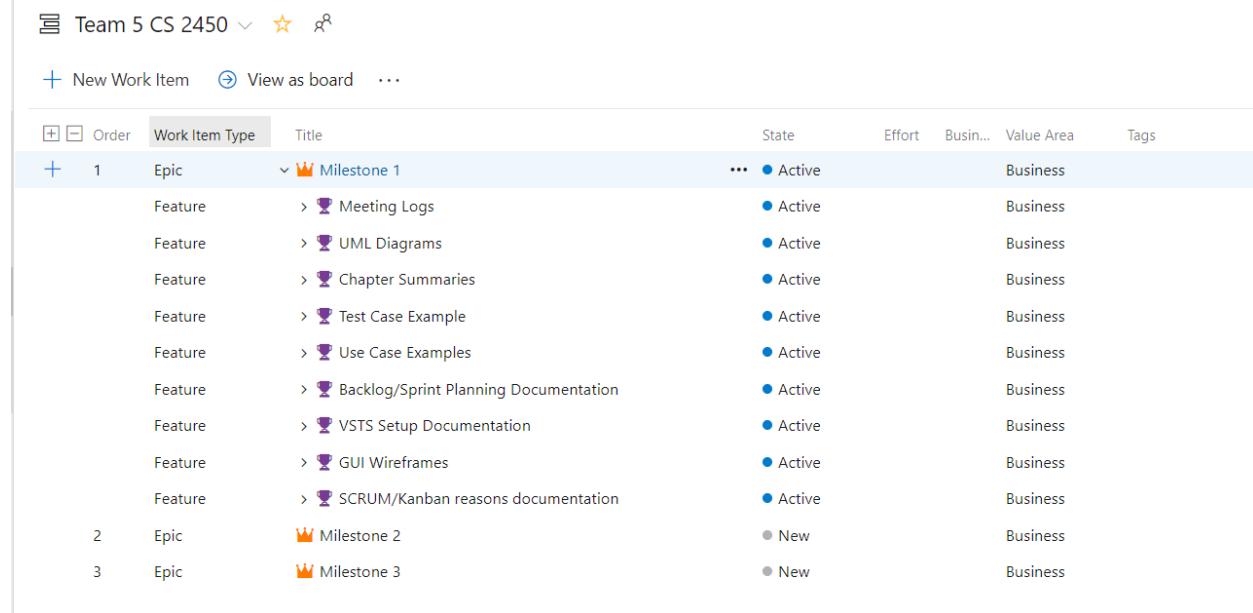
In the boards section, there are 4 parts we plan to use.

Work items is the team members home page in which they can go to see their own tasks. The page is also customizable to the individual team member's preference.

Work Items

Assigned to me	New Work Item	Open in Queries	Column Options	Recycle Bin
<input type="text"/> Filter by keyword				
ID	Title	State	Area Path	Comments
33	Document VSTS setup	Active	Abacus	9/20/2018 9:28 PM
62	Chapter 7	New	Abacus	9/14/2018 10:15 PM
45	Chapter 7	New	Abacus	9/13/2018 11:33 PM

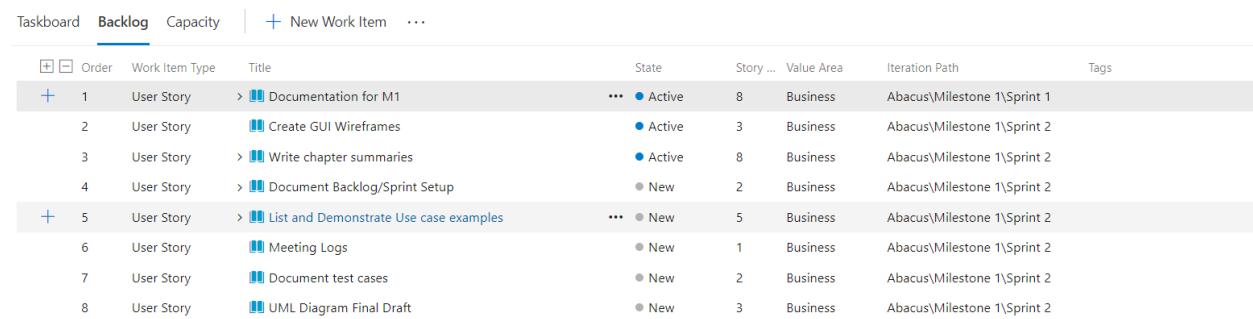
Boards and Backlogs will display the milestones and their progress including the task inside each milestone. Again, there is a level of customization that we will take greater advantage of as the project progresses.



The screenshot shows a project board titled "Team 5 CS 2450". At the top, there are buttons for "New Work Item", "View as board", and three dots. Below the header, there is a table with columns: Order, Work Item Type, Title, State, Effort, Business, Value Area, and Tags. The table data is as follows:

Order	Work Item Type	Title	State	Effort	Business	Value Area	Tags
1	Epic	💡 Milestone 1	... ● Active		Business		
	Feature	> 📋 Meeting Logs	● Active		Business		
	Feature	> 📋 UML Diagrams	● Active		Business		
	Feature	> 📋 Chapter Summaries	● Active		Business		
	Feature	> 📋 Test Case Example	● Active		Business		
	Feature	> 📋 Use Case Examples	● Active		Business		
	Feature	> 📋 Backlog/Sprint Planning Documentation	● Active		Business		
	Feature	> 📋 VSTS Setup Documentation	● Active		Business		
	Feature	> 📋 GUI Wireframes	● Active		Business		
	Feature	> 📋 SCRUM/Kanban reasons documentation	● Active		Business		
2	Epic	💡 Milestone 2	● New		Business		
3	Epic	💡 Milestone 3	● New		Business		

Sprints will be used to display the more urgent tasks. These will be anything with an upcoming due date including milestones and tasks.



The screenshot shows a backlog view with tabs for "Taskboard", "Backlog", and "Capacity". The "Backlog" tab is selected. At the top, there are buttons for "New Work Item" and three dots. Below the header, there is a table with columns: Order, Work Item Type, Title, State, Story..., Value Area, Iteration Path, and Tags. The table data is as follows:

Order	Work Item Type	Title	State	Story...	Value Area	Iteration Path	Tags
1	User Story	> 📋 Documentation for M1	... ● Active	8	Business	Abacus\Milestone 1\Sprint 1	
2	User Story	> 📋 Create GUI Wireframes	● Active	3	Business	Abacus\Milestone 1\Sprint 2	
3	User Story	> 📋 Write chapter summaries	● Active	8	Business	Abacus\Milestone 1\Sprint 2	
4	User Story	> 📋 Document Backlog/Sprint Setup	● New	2	Business	Abacus\Milestone 1\Sprint 2	
5	User Story	> 📋 List and Demonstrate Use case examples	... ● New	5	Business	Abacus\Milestone 1\Sprint 2	
6	User Story	> 📋 Meeting Logs	● New	1	Business	Abacus\Milestone 1\Sprint 2	
7	User Story	> 📋 Document test cases	● New	2	Business	Abacus\Milestone 1\Sprint 2	
8	User Story	> 📋 UML Diagram Final Draft	● New	3	Business	Abacus\Milestone 1\Sprint 2	

Repos has many subgroups and we will be using all of them. There is a file explorer for the source, a log of commits, pushes, and branches. ADO also has pull requests, which are useful for code reviews before merging changes.

The screenshot shows the Source Explorer and Commit Log interface in ADO. On the left, the Source Explorer displays a tree view of the repository structure under the 'Abacus' group. The 'public' folder contains 'favicon.ico', 'index.html', and 'manifest.json'. The 'src' folder contains 'components/abacus-column', which is expanded to show 'abacus-bead' (containing 'abacus-bead.css' and 'abacus-bead.js') and 'abacus-column.js'. It also contains 'containers', 'abacus.js', and 'App.css'. On the right, the Commit Log shows a detailed history of commits. The commits are listed in a table with columns for Author, Authored Date, and Message. The commit history starts with 'Set up CI with Azure Pipelines' by Heather Hyer on 9/14/2018 at 3:21 AM, followed by several merge pull requests from reiddk, and various changes made by Reid, Brock, Heather-Hyer, and Reid Kuttler.

Author	Authored Date	Message
Heather Hyer	9/14/2018 3:21 AM	Set up CI with Azure Pipelines
reiddk	9/14/2018 2:49 AM	Merge pull request #5 from reiddk/to-chinese-abacus
Reid	9/13/2018 11:29 PM	no longer allow drag beyond black bar
Reid	9/13/2018 11:06 PM	made the beads be 3 dimensional and animated
reiddk	9/11/2018 10:33 PM	Merge pull request #4 from reiddk/to-chinese-abacus
Reid	9/11/2018 10:17 PM	fix multipliers
reiddk	9/11/2018 10:10 PM	Merge pull request #3 from reiddk/to-chinese-abacus
Reid	9/11/2018 9:45 PM	to-chinese-abacus
reiddk	9/11/2018 9:46 PM	Merge pull request #2 from reiddk/to-chinese-abacus
Reid	9/11/2018 5:28 AM	first draft
reiddk	9/6/2018 10:08 PM	Merge pull request #1 from reiddk/blocksBranch
Brock	9/6/2018 10:05 PM	hogus balogus
Heather-Hyer	9/6/2018 10:01 PM	Merge branch 'master' of https://github.com/reiddk/abacus
Brock	9/6/2018 9:55 PM	fixed problem
Heather-Hyer	9/6/2018 9:57 PM	changed the box-shadow on css
Reid Kuttler	9/6/2018 7:58 PM	next draft
Reid	9/5/2018 12:42 AM	draft1

(commit log)

The screenshot shows the File Explorer interface in ADO. The left pane displays the file structure under the 'Abacus' group. The 'public' folder contains 'favicon.ico', 'index.html', and 'manifest.json'. The 'src' folder contains 'components/abacus-column', which is expanded to show 'abacus-bead' (containing 'abacus-bead.css' and 'abacus-bead.js') and 'abacus-column.js'. It also contains 'containers', 'abacus.js', and 'App.css'. The right pane shows the contents of the 'abacus-bead' folder, listing 'abacus-bead.css' and 'abacus-bead.js'. The 'abacus-bead.css' file was last changed on 9/13/2018 by Reid, with the message 'made the beads be 3 dimensional and animated'. The 'abacus-bead.js' file was last changed on 9/13/2018 by Reid, with the message 'no longer allow drag beyond black bar'.

(file explorer)

The Pipelines section will be used for release management. We are still early in the development process, so we haven't had much to look into for this or for Test plans. But we will be able to track build version and plan releases around testing as needed.

Backlog and Sprint Decomposition

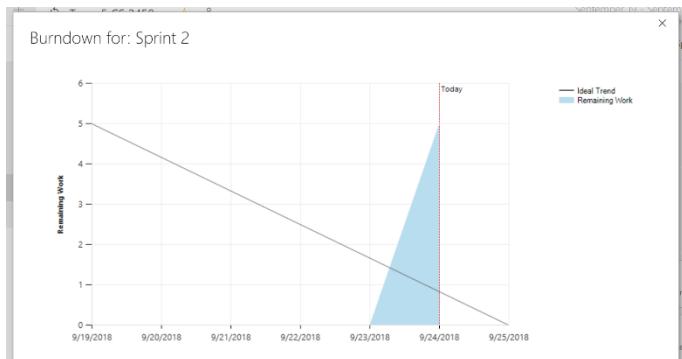
Sprints

We opted to have week long sprints, just based on the amount of work to be done for each milestone within the timeframe given.

In total we will have 12 sprints this semester, distributed among the corresponding milestones.

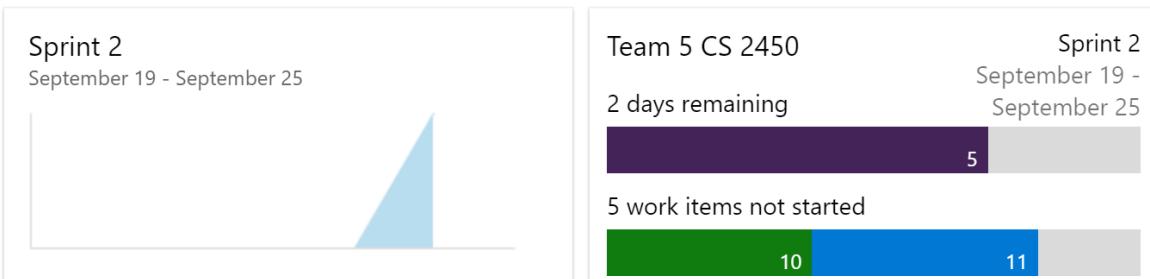
If we strictly follow this schedule, we should have no problem completing the work for each milestone in a timely manner.

Iterations	Start Date	End Date
Abacus		
Milestone 1	... 9/12/2018	9/25/2018
Sprint 1	9/12/2018	9/18/2018
Sprint 2	9/19/2018	9/25/2018
Milestone 2	9/26/2018	10/30/2018
Sprint 3	9/26/2018	10/2/2018
Sprint 4	10/3/2018	10/9/2018
Sprint 5	10/10/2018	10/16/2018
Sprint 6	10/17/2018	10/23/2018
Sprint 7	10/24/2018	10/30/2018
Milestone 3	10/31/2018	12/4/2018
Sprint 8	10/31/2018	11/6/2018
Sprint 9	11/7/2018	11/13/2018
Sprint 10	11/14/2018	11/20/2018
Sprint 11	11/21/2018	11/27/2018
Sprint 12	11/28/2018	12/4/2018



On the Sprint's taskboard, there is a burndown chart that provides a visual representation of the team's planned progress versus our actual progress.

There are also widgets on the main dashboard showing the burndown/overview for the current sprint.



Milestone/Feature/Story Breakdown

We've divided the work for this semester into Epics (one for each milestone), Features for each Epic, and Stories for each Feature. Each backlog has a burndown chart to help keep track of progress in and Epic, Feature, or Story-level scope.

The screenshot shows the Azure DevOps Backlogs page for the team "Team 5 CS 2450". The left sidebar is visible with options like Overview, Boards, Work Items, Boards, Backlogs, Sprints, Queries, and Repos. The main area displays a table of work items under the "Epics" section. The table has columns for Order, Work Item Type, Title, State, Effort, Business Area, and Tags. The data is as follows:

Order	Work Item Type	Title	State	Effort	Business Area	Tags
1	Epic	> 🏁 Milestone 1	Active		Business	
	Feature	> 📄 Meeting Logs	Active		Business	
	Feature	> 📄 UML Diagrams	Active		Business	
	User Story	> 📄 UML Diagram Rough Draft	Active		Business	
	Task	>Create UML diagram rough draft	Active		Business	
	User Story	UML Diagram Final Draft	Active		Business	
	Feature	> 📄 Chapter Summaries	Active		Business	
	Feature	> 📄 Test Case Example	Active		Business	

Breakdown of Epics:

The screenshot shows the Azure DevOps Backlogs page for the team "Team 5 CS 2450". The left sidebar is visible with options like Overview, Boards, Work Items, Boards, Backlogs, Sprints, Queries, and Repos. The main area displays a table of work items under the "Epics" section. The table has columns for Order, Work Item Type, Title, State, Effort, Business Area, and Tags. The data is as follows:

Order	Work Item Type	Title	State	Effort	Business Area	Tags
1	Epic	> 🏁 Milestone 1	Active		Business	
2	Epic	> 🏁 Milestone 2	New		Business	
3	Epic	> 🏁 Milestone 3	New		Business	

Breakdown of Features for the first Epic:

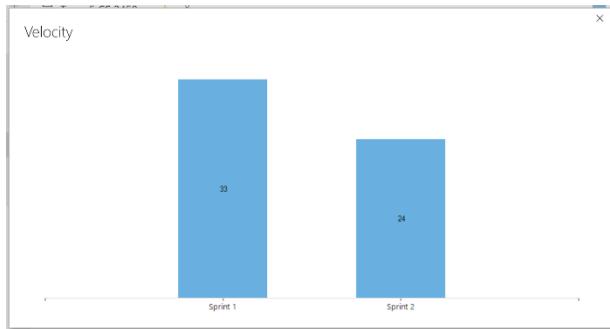
The screenshot shows the Azure DevOps Backlogs page for the team "Team 5 CS 2450". The left sidebar is visible with options like Overview, Boards, Work Items, Boards, Backlogs, Sprints, Queries, and Repos. The main area displays a table of work items under the "Features" section. The table has columns for Order, Work Item Type, Title, State, Effort, Business Area, and Tags. The data is as follows:

Order	Work Item Type	Title	State	Effort	Business Area	Tags
1	Feature	> 📄 Meeting Logs	Active		Business	
2	Feature	> 📄 UML Diagrams	Active		Business	
3	Feature	> 📄 Chapter Summaries	Active		Business	
4	Feature	> 📄 Test Case Example	Active		Business	
5	Feature	> 📄 Use Case Examples	Active		Business	
6	Feature	> 📄 Backlog/Sprint Planning Documentation	Active		Business	
7	Feature	> 📄 VSTS Setup Documentation	Active		Business	
8	Feature	> 📄 GUI Wireframes	Active		Business	
9	Feature	> 📄 SCRUM/Kanban reasons documentation	Active		Business	

Breakdown of Stories for the first Epic:

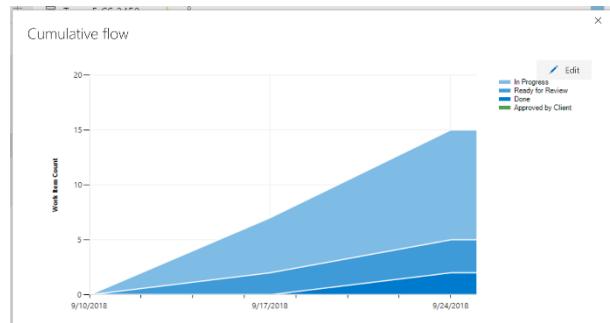
Team 5 CS 2450

Order	Work Item Type	Title	State	Story ...	Value Area	Iteration Path	Tags
1	User Story	> Documentation for M1	... ● Active	8	Business	Abacus\Milestone 1\Sprint 1	
2	User Story	> Create GUI Wireframes	● Active	3	Business	Abacus\Milestone 1\Sprint 2	
3	User Story	> Meeting Logs	● Resolved	2	Business	Abacus\Milestone 1\Sprint 1	
4	User Story	> UML Diagram Rough Draft	● Active	3	Business	Abacus\Milestone 1\Sprint 1	
5	User Story	> Read chapters	● Active	10	Business	Abacus\Milestone 1\Sprint 1	
6	User Story	> Write chapter summaries	● Active	8	Business	Abacus\Milestone 1\Sprint 2	
7	User Story	> Document 3 reasons for and 3 reasons against SCRUM	● Resolved	1	Business	Abacus\Milestone 1\Sprint 1	
8	User Story	> Document 3 reasons for and 3 reasons against Kanban	● Resolved	1	Business	Abacus\Milestone 1\Sprint 1	
9	User Story	> Document VSTS setup	● Resolved	3	Business	Abacus\Milestone 1\Sprint 1	
10	User Story	> Decompose Milestones to Backlogs/Sprints.	● Resolved	5	Business	Abacus\Milestone 1\Sprint 1	
11	User Story	> Document Backlog/Sprint Setup	● Active	2	Business	Abacus\Milestone 1\Sprint 2	
12	User Story	> List and Demonstrate Use case examples	● Active	5	Business	Abacus\Milestone 1\Sprint 2	
13	User Story	> Meeting Logs	● Active	1	Business	Abacus\Milestone 1\Sprint 2	
14	User Story	> Document test cases	● Active	2	Business	Abacus\Milestone 1\Sprint 2	
15	User Story	> UML Diagram Final Draft	● Active	3	Business	Abacus\Milestone 1\Sprint 2	



The story-level backlog provides a chart to track the team's velocity. This is useful for budget tracking.

There is also a chart to keep track of the team's cumulative flow, which is helpful for cost estimation.



Task Tracking

Work assigned to Heather Hyer (5)

ID	State	Title
15	● Resolved	Decompose Milestones to Backlogs/Sprints.
34	● Active	Document Backlog/Sprint Setup
79	● Active	Document budget planning/task management...
78	● Active	Screenshot backlog/sprint setup in VSTS
10	● Active	Backlog/Sprint Planning Documentation

There is a widget on the main dashboard that gives each team member an overview of the stories, tasks, and features assigned to them.

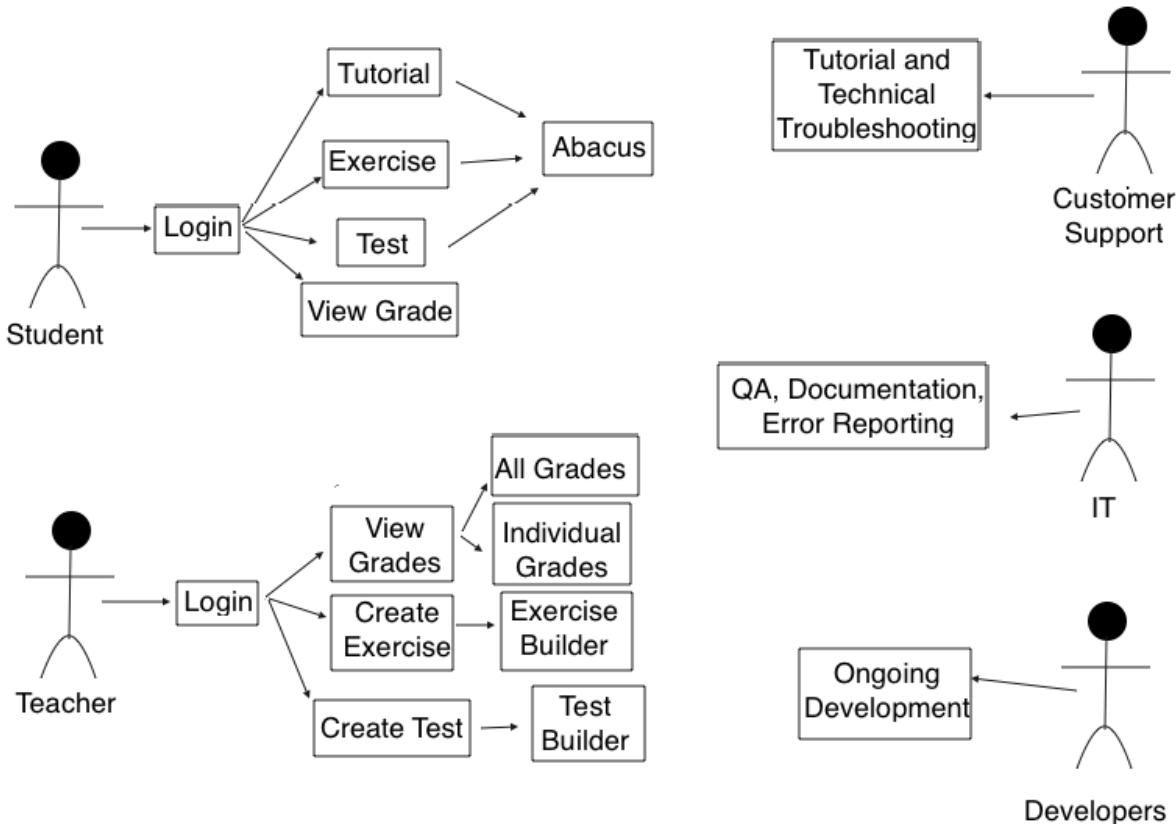
Each team member is responsible for creating tasks for each of their assigned stories for a given sprint. They are also responsible for tracking the progress of their tasks and stories and the plan is to do so regularly in order to get an accurate idea of the team's progress.

Column	Work Item	Owner	State
New	Document Backlog/Sprint Setup	Heather Hyer	2 h
Active	Screenshot backlog/sprint setup in VSTS	Heather Hyer	1
Active	Document budget planning/task management plan	Heather Hyer	1
Closed	Chapter 13	Myron Burton	
New	List and Demonstrate Use case examples	Tanner Chamberlain	2 h
Active	Revise with UML Diagram	Tanner Chamberlain	
Active	Make it look fancy and polished	Tanner Chamberlain	
Closed	Create an outline	Tanner Chamberlain	

Each story is assigned a certain number of story points. The value for these story points follows a modified Fibonacci sequence (1,2, 3, 5, 8, 10) and points are assigned by team consensus according to complexity, difficulty, and estimated time cost. Stories should also include acceptance criteria and a description as needed. Each feature contains a more in-depth description. Stories are also assigned a priority value ranging from 1-4, weighted by what is most vital to the epic as a whole.

Each team member also keeps track of the tasks they have created for their stories, recording a time estimate (in hours), and reporting the actual time the task took. This is an important tracking mechanism, since this is what is reported in the sprint's burndown chart, which helps us as a team to know how accurate our estimations are and adjust accordingly.

Use Case Examples



Test Case Examples

Using a coverage based text we will be able to create a bead and use a `toString` method to get expected values from the bead. This can test that bead creation works. Using the created bead we will run another test within the same case to make sure that moving of the bead behaves as expected, again using the `toString` to compare values.

The main abacus class will have a test case that affirms that the number of columns shown matches the number of columns provided. It will also affirm that the value of the abacus matches the expected value after a set number of moves. This will test edge cases for the largest column and for moving the max number of beads. It will also perform these actions on the largest, smallest and a select few other board sizes. This will ensure that when moving beads around the board that they will update the value appropriately.

Change Order Form – Team 5

Description of Proposed Change:

Date Requested: _____

Proposed Change Order Value: _____

Estimated Extra Time Required: _____

Approved Denied

Final Negotiated Price: _____

Signature of Customer

Date:

Signature of Team Manager

X _____ Date: _____

Chapter Summaries

Chapter 4: Mariah Bleak

Configuration Management

Configuration management comprises of careful procedures that are necessary to manage every element over the lifespan of a large software system or a distributed development project. Throughout the lifetime of a system or project, there may at any given time exist different versions of the software, depending on what updates have been given to the client and if any programmers are working on older versions.

Baseline –official version of the complete set of documents related to the project.

Configuration items are the items contained in the baseline.

Ex:

- Source code components
- The requirements specifications
- The design documentation
- The test plan
- Test cases
- Test results
- The user manual

Configuration management takes care of controlling the release and change of CRs throughout the software life cycle.

CCB (Configuration control board) ensures that any change to the baseline is properly authorized and executed. When a change request is submitted to the CCB, they need info on how the change will affect the product and the development process. If approved, the change request becomes a work package which will need to be scheduled. All changes and status of changes need to be recorded, this ensures that if needed to go back to a previous version or to monitor what changes have been made there is a clear record.

Chapter 5: Brock Brown

It's really important to decide what the project goals are, i.e. fastest, least memory use, etc. so everyone knows and can comply with said goals. Along with these goals must come the explicit expectation of clear notes/pseudocode in the file. As we progress we will further clarify the standards for both in-code documentation and out-of-code documentation. I think that the ideal communication is an informal and open dialogue, which I believe supports my goal of an integration-style leadership and team.

Chapter 6: Linda Jensen

Chapter 6 is about managing software quality. It's important that quality be a key factor in designing software- it can't be added as an afterthought. You can improve the quality by improving the product or process. This section can also be where we calculate the complexity of the project. If the code is mostly brute force, it's not high in quality. You need to figure out your scale of determining complexity however, so that you can compare them with other projects. Quality attributes include: correctness, reliability, efficiency, integrity, usability, maintainability, testability, flexibility, portability, reusability,

and interoperability. These qualities fall into 3 categories: product operation, product revision, and product transition. There are many different perspectives on what gives a program quality. Some may say it depends on how well it conforms to the users' needs, some say it's how much money it made vs time spent on it. Overall, there needs to be a test for quality agreed on by the shareholders because quality can be hard to measure quantitatively.

Chapter 7: Bradley Allred

The models outlined in 7.1 are cost estimation functions where the variables are based of previous similar projects, they are useful when there is a lot of data to sort through to generate an estimate. The problem is for this project we do not have a lot of data to base our estimation on. Section 7.2 is cost estimation for experts which does not apply to our situation. However, it does include some techniques that will be useful for us. Section 7.4 is called Agile cost estimation and provides the least accurate cost estimation because it places emphasis on the ability to change rather than having a set plan.

The best fit for cost estimation for our situation is found in section 7.3 Distribution of manpower over time. this section is generally used in coordination with a cost estimation expert, but since this is a smaller project (Relative to the working world) we can estimate our own manhours cost. It is important to note that adding more people to the project will have diminishing returns. A 20 man-month cost estimation does not mean you can just throw 80 people at the project and have it done in a week. Also, with this style it is important to tell the customer the uncertainty of the estimation. Research has shown that there is a 75% chance that the actual cost will be within 20% of the estimate. The benefit of this style is that as the project progresses you will continue to get more and more accurate.

Chapter 8: Tanner Chamberlain

Chapter 8 can be applied by using project planning and control in our project. Planning and controlling the project is vital to making sure the project does what it's supposed to and works correctly. In order to plan and control the project, variables such as staff experience, the goals of the project, and the tools used in the project need to be considered. Knowledge of certain categories is needed prior to planning. During our project, we need to ascertain the client's needs for the product, familiarize ourselves with the tools, and determine our resources to create the project. Obtaining these, we'll be able to determine what we need to focus on and what problems we'll be facing. This is a form of risk management, where we can see what areas we're weak in and plan on mitigating those weaknesses. Finally, setting goals and keeping track of progress are focused around the milestone reports created during development.

Chapter 9: Heather Hyer

Requirements engineering is a cyclical process involving four types of activity: elicitation (understanding the problem), specification (describing the problem), validation (agreeing upon the problem), and negotiation (fitting the problem to the situation at hand). Documentation and Management should be incorporated into each of these activities. Requirements engineering takes into account both social and cognitive issues. Requirements engineering is taking a problem to be solved and returning the requirements specifications that will be used to solve the problem. Prioritizing requirements includes breaking things up into must haves, should haves, could haves, and won't haves. Requirements engineering activities include: asking, task analysis, scenario-based analysis (think-aloud), ethnography, form analysis, natural language descriptions, derivation from an existing system, business process redesign (BPR), and prototyping.

I think right now our team needs to focus on asking the client what the product must have and leaving our own ideas about what it could have or even what we think it should have off the side for now. We'll also want to ensure we're specifying requirements for our tasks as we move forward into the next milestone. I think we should also look into doing scenario-based analysis, both on our own, and through interviewing the client and potential users (teachers and students).

Chapter 10: Brock Brown

There are 4 modeling perspectives that are analyzed in UML diagrams: entity-relationship modeling, finite state machines, data flow diagrams, and CRC cards. I think that the class diagrams, the sequence diagrams, and the use case diagrams will be the most important.

Chapter 11: Mariah Bleak

Software architecture concerns the large-scale structure of software systems. This large-scale structure reflects the early, essential design decisions. This decision process involves negotiating and balancing of functional and quality requirements on one hand, and possible solutions on the other hand. Software architecture is not a phase strictly following requirements engineering, but the two are intertwined. In this chapter, we discuss how to design, document and evaluate software architectures.

Three purposes:

- Vehicle for communication among stakeholders
- Captures early design decisions
- Transferable abstraction of a system

Forces that influence architecture:

- Developmental organization
- Background and expertise of the architect
- Technical and organizational environment

In the software architecture, the global structure of the system has been decided upon. This global structure captures the early, major design decisions. Whether a design decision is major or not really can only be ascertained with hindsight, when we try to change the system.

Stakeholders speak with architect back and forth over requirements and quality, when they reach an agreement that feeds into the development.

Attribute Driven Design (ADD) is described as a top down decomposition process. In each iteration, one or a few components are selected for further decomposition. In the first iteration, there is only one component, 'the system'. From the set of quality attribute scenarios, an important quality attribute is selected that will be handled in the current refinement step. For instance, in our library system, we may have decided on a first decomposition of the system into three layers: a presentation layer, a business logic layer, and a data layer. In a next ADD step, we may decide to decompose the presentation layer, and select usability as the quality attribute that drives this decomposition. A pattern is then selected that satisfies the quality attribute. For instance, a data validation pattern (Folmer et al., 2003) may be applied to verify whether data items have been entered correctly. Finally, the set of quality attribute scenarios is verified and refined, to prepare for the next iteration.

Backlog is a list of issues that currently need to be tackled in the development/design.

There are different types of undocumented design decisions:

- The design decision is implicit: the architect is unaware of the decision, or it concerns 'of course' knowledge.
- The design decision is explicit but undocumented: the architect takes a decision for a very specific reason. The reasoning is not documented, and thus is likely to vaporize over time.
- The design decision is explicit, and explicitly undocumented: the reasoning is hidden.

Design Decisions Process:

Element	Description
Issues	Design issues being addressed by this decision
Decision	The decision taken
Status	The status of the decision (pending, approved)
Assumptions	The underlying assumptions about the environment in which the decision is taken
Alternatives	Alternatives considered for this decision
Rationale	An explanation of why the decision was chosen
Implications	Implications of the decision
Notes	Any additional info

View: representation of a whole system from the perspective of a related set of concerns.

Viewpoint: establishes the purposes and audience for a view and the techniques or methods employed in constructing a view.

Module viewpoints give a static view of the system. They are usually depicted in the form of box and line diagrams where the boxes denote system components and the lines denote some relation between those components.

Component and connector viewpoints give a dynamic view of the system, i.e. they describe the system in execution. Again, they are usually depicted as box and line diagrams.

Allocation viewpoints give a relation the system and its environment, such as who is responsible for which part of the system.

Decomposition: In a decomposition viewpoint, elements are related by the 'is a submodule of' relation. Larger elements are composed of smaller ones (top down).

Uses: In a uses viewpoint, the relation between elements is 'uses' (A calls B, A passes information to B, etc.).

Layered: The layered viewpoint is a special case of the uses viewpoint. It is useful if we want to view the system as a series of layers, where elements from layer n can only use elements from layers <n. Layers can often be interpreted as virtual machines.

Class: The class viewpoint describes how certain elements are a generalization of other elements. The relation between elements is ‘inherits from’. It is obviously most applicable for object-oriented systems.

Conceptual/Logical viewpoint: describes the system in terms of major design elements and their interactions.

Implementation viewpoint: gives a view of the system in terms of modules or packages and layers.

Process: The process viewpoint describes the system as a series of processes, connected by communication or synchronization links.

Concurrency: To determine opportunities for parallelism, a sequence of computations that can be allocated to a separate physical thread later in the design process is collected in a ‘logical thread’.

Shared data: This viewpoint shows how persistent data is produced, stored and consumed. It is particularly useful if the system centers around the manipulation of large amounts of data.

Client-server: To describe a system that consists of cooperating clients and servers. The connectors are the protocols and messages that clients and servers exchange.

Deployment: This viewpoint shows how software is assigned to hardware elements, and which communication paths are used.

Implementation: This viewpoint indicates how software is mapped onto file structures. It is used in the management of development activities and for build processes.

Work assignment: Shows who is doing what. This viewpoint is used to determine which knowledge is needed where.

Component Types:

Type	Description
Computational	The component performs a computation of some sort
Memory	Maintains a collection of persistent, structured data to be shared by a number of other components
Manager	Contains a state and a number of associated operations
Controller	Governs the time sequence of other events.

Architecture Tradeoff Analysis Method(ATAM):

1. Present method to stakeholders

2. Present business drivers (by project manager)
3. Present architecture (by lead architect)
4. Identify architectural approaches
5. Generate quality attribute tree
6. Analyze architectural approaches
7. Brainstorm and prioritize scenarios
8. Analyze architectural approaches
9. Present results

Chapter 12.1: Tanner Chamberlain

Chapter 12.1 can be applied by designing our software efficiently with the use of modules. A module is a portion of code that does a specific task, for example a class in Java or struct in C. Using modules effectively gives many benefits, such as readability and adaptability. Effective module usage employs information hiding, which means each module gives and takes information on a need-to-know basis. This process means that unnecessary information is abstracted from each module, so data is contained and clean. Individual modules also need to contain only relevant data and processes. Modules whose elements all function for one purpose are the most effective, while those that have unnecessary functions or try to do multiple processes are messier and less effective. Perfect module implementation is very difficult, but doing it well will lead to less errors, more readable code, and an adaptive program.

Chapter 12.1-12.8: Reid Kuttler

The essence of the design process is that the system is decomposed into parts that each have less complexity than the whole. Some form of abstraction is always used in this process.

- The constituents of a module should belong together and be friends.
- The interfaces between modules should be as thin as possible.
- Each module should hide one secret. Information hiding is a powerful design principle where each module is characterized by a secret which it hides from its environment.
- The structure of a system, should have a simple and regular shape when depicted as a graph.

They discussed four design methods in this chapter

1. Functional decomposition
2. Data flow design
3. Data structure design
4. Object-oriented design

The first three methods have been around the longest. Object-oriented analysis and design came later, but is now the most widely used approach.

However simply using object oriented design does not guarantee a good design. Good practices still have to be followed.

A design pattern is a recurring structure of communicating components that solves a general design problem within a particular context. These design patterns describe best practices and represent the collective experience of some of the most experienced and successful software designers. In a similar way antipatterns describe bad experiences

Finally the design itself must also be documented. IEEE may serve as a guideline for this documentation as it lists a number of attributes for each component of the design.

Chapter 13: Myron Burton

Testing is one of the most important parts of Software Engineering. Just like we wouldn't sell cars that haven't been crash tested, or build bridges that hadn't been tested for strength and durability, we shouldn't build programs without proper testing. To test there are three basic models and two ideologies that can be used. The methods include coverage based, fault based, and error based testing. Coverage makes sure that all of the lines of code are actually worth their weight and that they can be run. Fault based testing introduces possible errors and makes sure that the program is able to handle these issues properly. Error based testing checks key "stress" points of a program, areas that are commonly issues among programs. This can all be accomplished using either black box or white box testing, where black box testing doesn't consider implementation while white box knows the implementation of the program. There are two different ways to build test cases which are top down and bottom up methods. Top down considers the system as a whole and goes into specifics using stubs which simulate functions of components not yet implemented. Bottom up uses drivers that generate testing environments for the code.

Testing comes in different stages and can be modified based on the model used and the client specifications. Unit testing is the most basic stage which is testing specific parts of the code. This helps test each individual component in an integral method. System testing compares a program with its specifications and the user documentation given. There may also be Acceptance testing which is similar to system testing but is done by supervision of the client. Finally there is installation testing which makes sure that the program works in various environments.

Testing is documented in what IEEE calls a verification and validation plan. This report covers the design specification, the procedure specification, and an item transmittal report showing how each feature will be tested, what sequence testing takes and what items will be tested respectively. After testing is done a test log is composed, an incident report with incidents that need further investigation is written, and a summary report shows the overall evaluation of the testing and any findings that should be reported