

PROBLEM 2.2.7

The Development Process

INTRODUCTION

Your CollegeApp can be used as the basis of a new app. With modifications, you can create a new app to monitor and collect information for another system or process. For example,

- an ecosystem monitoring app - collect pictures of an ecosystem, the location of the sample, and important information about the samples
- a restaurant app - include a picture of your favorite restaurants (or favorite meal), where they are located, and other details such as the restaurant rating, the ambiance (casual or formal), etc.
- a hikers guide app - collect information about hiking trails, include locations, difficulty rating, and pictures from the hike.

So think of something you would like to collect or monitor and make an app!

Materials

- Computer with Android™ Studio
- Android™ tablet and USB cable, or device emulator
- Tools with which to create visual representations of your code
- Tools for creating and using a sprint task list and product backlog
- Presentation software

RESOURCES



Problem 2.2.7 Requirements
Resources available online



Problem 2.2.7 Rubric
Resources available online

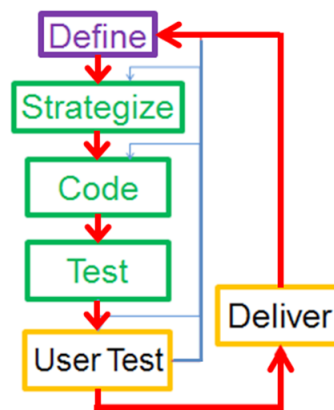
Procedure

Part I: Planning

- 1 While solving this problem, you will learn a software development process called the Agile methodology. It encourages small development steps, frequent testing, and iterative design. As you practice the Agile methodology, make sure to save all artifacts of your process. They will be your resources when you present your solution to the class.
- 2 Review the slideshow.

Software Development Process

1. Define
2. Strategize
3. Code
4. Test
5. User Test
6. Deliver



Computer Science A

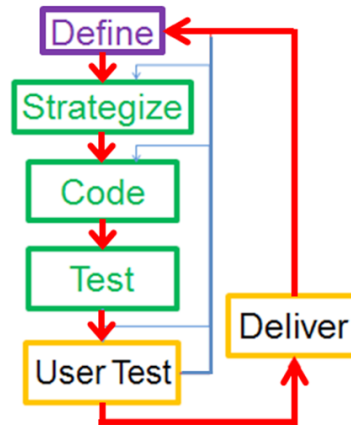
© 2016 Project Lead The Way, Inc.

Most software development processes involve similar steps:

1. Define the problem to be solved. This includes asking questions, interviewing people, and understanding the requirements.
2. Strategize includes exploring options, brainstorming ideas, and designing potential solutions.
3. Code the solution.
4. Test the solution and make sure it solves the problem identified in the first step.
5. Have users test the solution and make sure it works as expected. This is also known as Quality Assurance (QA) testing.
6. Deliver the solution.

Agile Methodology

- Frequent deliverable iteration
- One cycle called a **sprint**



Computer Science A

© 2016 Project Lead The Way, Inc.

In Waterfall software development, the steps are linear. One step has to be completed before the next one begins. This method usually takes longer to deliver a solution and does not involve the client once the problem is defined in step 1.

In Agile software development, the process is not linear. Instead it is a frequent deliverable iteration. Once a problem is defined, programmers will go through frequent cycles developing the solution in increments and presenting it to the client. This way the client can help clarify the specifications of the software as it is being built. This also allows for faster delivery. Agile is very commonly used in industry today.

Each cycle in Agile is called a sprint (or an iteration) and may last 1-4 weeks for a professional team. Programmers code and test multiple times, and even may have to re-strategize, then code and test. The cycle repeats until they are ready to share with the client for feedback. Then they start another sprint, in which they work on another set of requirements or refine something from their previous sprint.

Organizing Agile Work

Backlog

- ☐ User Story #1
 - ☐ User Story #2
 - ☐ User Story #3
 - ☐ User Story #4
 - ☐ Etc.
- Prioritized
 - Top of backlog well defined

Computer Science A

© 2016 Project Lead The Way, Inc.

Programmers usually organize their work using two lists: a backlog and a task list.

The backlog is the list of requirements identified by the client and prioritized from highest to lowest priority. The items in this list are usually written in an informal fashion and are referred to as “user stories”. User stories should always describe the way the user interacts with the software application. For example, “the user should register to use the app”. During each sprint, programmers work on one or two user stories, based on the time estimated to develop (see next slides).

Backlog Example

Backlog

- ☐ Students should register with a school to use the app.
- ☐ Students can enroll in multiple classes.
- ☐ Teachers should be able to post assignments for each class.
- ☐ Students should be able to submit their assignments back to the teacher.
- ☐ Students should be able to view their grades for each class.

Computer Science A

© 2016 Project Lead The Way, Inc.

The example shows a backlog for a project to develop an app to support a student learning management system. The list consists of user stories that describe the student and teacher experiences.

Organizing Agile Work (cont'd)

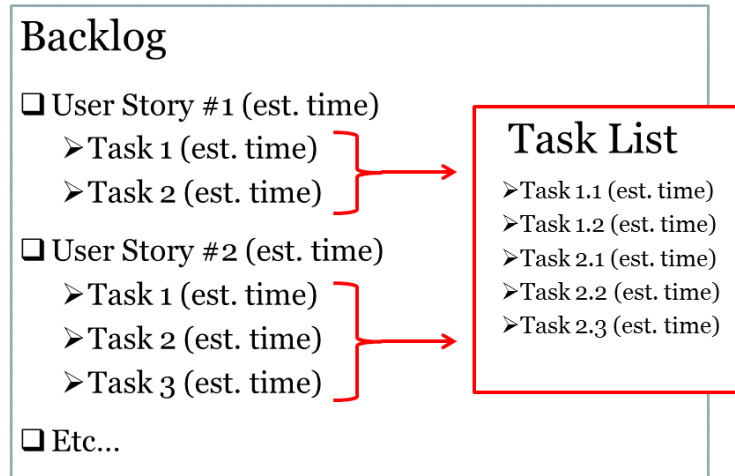
Each sprint or iteration includes:

- Iteration Planning
- Iteration Execution
- Iteration Retrospective

Computer Science A

© 2016 Project Lead The Way, Inc.

Sprint step 1: Iteration Planning



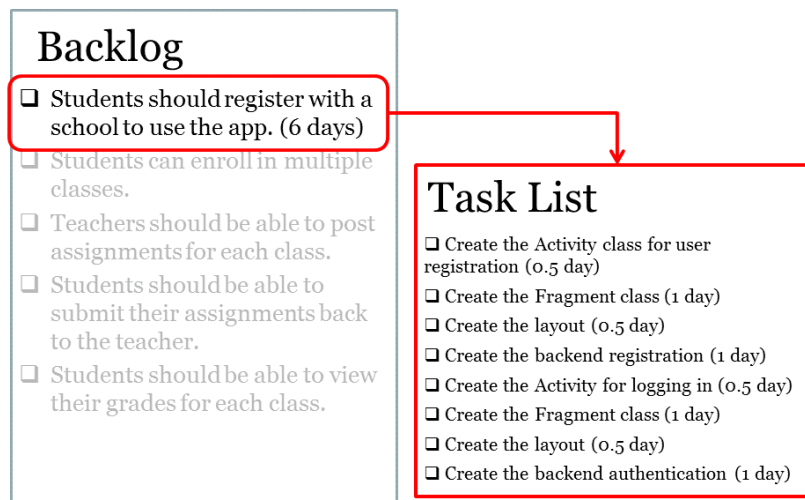
Computer Science A

© 2016 Project Lead The Way, Inc.

The first step of a sprint is the iteration planning step. Programmers begin to plan their work for the first sprint as follows:

1. They write the estimated time to complete each user story. This will help identify how many user stories can be completed in an iteration.
2. Then they choose the user story (or stories) that they would like to complete in the first iteration (based on the estimated time).
3. Each of the chosen user stories is then translated into a list of more specific tasks to be worked on. Tasks can be more technical and describe the nuts and bolts of what must be coded to fulfill the user story.
4. Then they write the estimated time to complete each task. Any task that is still too large should be broken down into smaller tasks and re-estimated. Tasks should be small enough to complete in a day or two.
5. In the case when a user story has a very large number of tasks, the tasks can be split up so that the user story is worked on over two iterations.
6. The final list of tasks to be worked on during a sprint is called a task list.

Task List Example



Computer Science A

© 2016 Project Lead The Way, Inc.

The task list above is an example of what tasks can be generated from the first user story. Note the time estimates on both the backlog and the task list.

Sprint step 2: Iteration Execution

- Divide task list among programmers.
- Start developing:
 - Code
 - Test
 - Do code review
 - Create Javadoc
- Present to client for feedback at the end of the sprint.

Computer Science A

© 2016 Project Lead The Way, Inc.

The retrospective is probably the most important part of the whole sprint. The retrospective reflects on two main questions:

1. What did we do well? Identify successes that can be repeated in the future.
2. What did we NOT do well and how should we address them? Identify failures and lessons learned.

The retrospective enforces the core benefit of iterative development. Bringing what is identified during a retrospective to the next iteration planning is the “magic” that makes agile so effective.

Sprint step 3: Iteration Retrospective

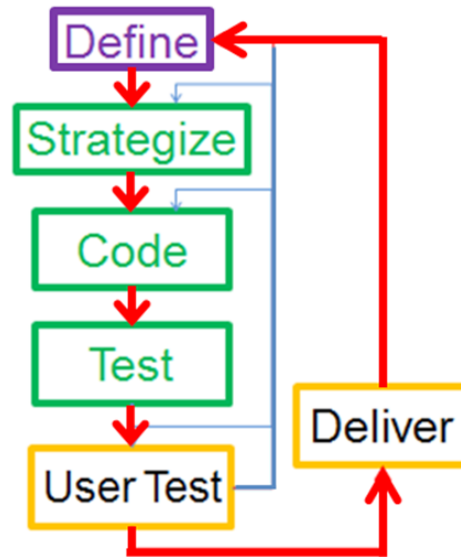
- What did we do well?
- What did we NOT do well and how should we address that in the future?

Computer Science A

© 2016 Project Lead The Way, Inc.

Organizing Agile Work (cont'd)

- Groom the backlog.
- Generate new task list.
- Repeat sprints until product is complete.





Computer Science A

© 2016 Project Lead The Way, Inc.

Then the backlog is groomed (revisited and updated), and a new task list is generated to clearly define the tasks to work on in the next sprint. The cycle of sprints continues until the complete product is delivered.

- 3 As directed by your instructor, form teams and select a problem to work on.
- 4 Identify and interview a variety of stakeholders to get initial ideas for features for your app. Interview classmates, friends, and teachers. Be sure to record their input. Try to get a wide variety of answers, for example, talk to people who share your interest and to people who may be learning something new using your app.
- 5 As directed by your instructor, brainstorm additional ideas for features for your app.
- 6 Select features from your brainstorming and interviews to populate your product backlog.
- 7 Select user stories from the backlog for your team to work on during this sprint and divide up the workload.

Part II: Implementing

- 8 Review the documents described below before you begin.
 - Refer to  **Problem 2.2.7 Requirements** for a list of requirements for the product, the written reflection/interview, and the team presentation.
 - Refer to  **Problem 2.2.7 Rubric** for details on how your solution to this problem will be graded.

- 9 Work as a team to develop your product using the Agile software development process. Refer to the slideshow in Part 1.
- 10 Perform usability testing throughout development. Be sure to test often, going back to test previously working features as you develop new ones.

Part III: Delivery

- 11 Present your product to the class.

Problem 2.2.7

Requirements

Product Requirements List

1. Original Problem Statement
2. Brainstorm List
3. Notes from stakeholder interviews
4. Product Backlog
5. Sprint Task List
6. Source code for the product including at least:
 - a. An original abstract class or interface
 - b. An original concrete class
 - c. An array or `ArrayList` over which the program iterates
 - d. Different `Fragments` to display different functionality
 - e. Navigation between `Fragments`
7. Documentation of features

Written Reflection or Interview Requirements List

Your instructor will ask you either to provide a written reflection or notes from an interview based on the following prompts. All examples should come from original code that was written by either you or someone on your team.

1. Describe a compile time error that you encountered and how it was corrected.
2. Describe a run-time error that you encountered and how it was corrected.
3. Pick one abstract class or interface in the product and explain the role that the class plays in the app, how it could be used in one of the problems that your team did not select, and why the role it fills is more appropriate for either an abstract class or interface.
4. Pick one class and explain what each of the following do and how they fit in with the rest of the code:
 - Constructors
 - Service methods
 - Helper methods
 - Encapsulation
5. Explain an example of iteration over either an array or an `ArrayList`.

Team Presentation Requirements List

1. The problem statement
2. A demonstration of the features of the app that the team created
3. An example of a particularly challenging debugging situation accompanied by an explanation of the process involved to fix the bug
4. A description of the version control system that the team chose, as well as rationale for choosing it
5. A description of the current state of the product with respect to Agile Methodology, specifically, problem statement, product backlog, and sprint task list
6. A description of next steps for development

Problem 2.2.7 Rubric

RESOURCES



Problem 2.2.7 Rubric

Resources available online

Product Requirements

Criteria	4	3	2	1
1. Problem Statement	Thoroughly and effectively communicates why this problem is particularly relevant and important to each team member	Mostly communicates why this problem is particularly relevant and important to each team member	Somewhat communicates why this problem is particularly relevant and important to each team member	Ineffectively communicates why this problem is particularly relevant and important to each team member
2. Brainstorm List	Thoroughly and thoughtfully explores possible solutions to the problem statement	Thoughtfully explores possible solutions to the problem statement	Briefly explores possible solutions to the problem statement	Incompletely explores possible solutions to the problem statement
3. Stakeholder Interview Notes	Detailed and varied interview notes related to the problem statement	Detailed interview notes related to the problem statement	Somewhat detailed and varied interview notes related to the problem statement	Incomplete or ineffective interview notes
4. Product Backlog	Completely represents the backlog in an organized manner	Represents most or the backlog in a moderately organized manner	Represents some of the backlog in a somewhat organized manner	Incomplete or unorganized representation of the backlog

Criteria	4	3	2	1
5. Sprint Task List	Completely represents all tasks in an organized manner	Represents most or all tasks in a moderately organized manner	Represents some tasks in a somewhat organized manner	Incomplete or unorganized representation of task list
6. Source Code: a. Original abstract class or interface b. Original concrete class c. Array or ArrayList iteration d. Fragment functionality e. Fragment Navigation	Addresses all required elements using a logical and organized implementation	Addresses most of the required elements using a logical and organized implementation	Addresses most of the required elements in but may be unorganized or illogically implemented	Missing a majority of the required elements
7. Documentation	All new features are thoroughly explained in terms of purpose and use	Most new features are explained in terms of purpose and use	Some of the new features are explained in terms of purpose and use	New features are inadequately explained in terms of purpose and use

Written Reflection or Interview Requirements

Criteria	4	3	2	1
1. Compile-time error	Description of a error, why it occurred, and how it was corrected	Description of a error, missing either why it occurred or how it was corrected	Description of error without reason or correction	Incomplete or incorrect description of error
2. Run-time or logic error	Description of a error, why it occurred, and how it was corrected	Description of a error, missing either why it occurred or how it was corrected	Description of error without reason or correction	Incomplete or incorrect description of error

Criteria	4	3	2	1
3. Abstract class / Interface	Thorough description of: <ul style="list-style-type: none"> • purpose • why an abstract class or interface was chosen • example of re-use outside of app 	A mostly thorough description of: <ul style="list-style-type: none"> • purpose • why an abstract class or interface was chosen • example of re-use outside of app 	A moderate description of: <ul style="list-style-type: none"> • purpose • why an abstract class or interface was chosen • example of re-use outside of app 	An incomplete description of: <ul style="list-style-type: none"> • purpose • why an abstract class or interface was chosen • example of re-use outside of app
4. Concrete Class	Thoroughly describes all <ul style="list-style-type: none"> • Constructors • Service methods • Helper methods • Abstraction of the class as they relate to other classes in the app 	Mostly describes <ul style="list-style-type: none"> • Constructors • Service methods • Helper methods • Abstraction of the class as they relate to other classes in the app 	Somewhat describes <ul style="list-style-type: none"> • Constructors • Service methods • Helper methods • Abstraction of the class as they relate to other classes in the app 	Inaccurately or incompletely describes all <ul style="list-style-type: none"> • Constructors • Service methods • Helper methods • Abstraction of the class as they relate to other classes in the app
5. Array or ArrayList Iteration	Explanation of a sample iteration is accurate and thorough	Explanation of a sample iteration is mostly accurate and thorough	Explanation of a sample iteration is somewhat accurate and thorough	Explanation of a sample iteration is inaccurate or incomplete

Presentation Requirements

Criteria	4	3	2	1
1. Problem Statement	Thoroughly describes the problem statement	Mostly describes the problem statement	Briefly describes the problem statement	Inadequately describes the problem statement
2. App Features	Fully demonstrates all features with clarity and thoroughness	Demonstrates most or all features, however clarity or thoroughness may be absent	Demonstrates most or all features, however clarity and thoroughness are both absent	Demonstrates very few of the app features, the demonstration is neither clear nor thorough
3. Debugging Situation	Thoroughly discusses the problem and the process involved to fix it	Mostly discusses the problem and the process involved to fix it	Briefly discusses the problem, or omits the process involved to fix it	Minimally discusses the problem and omits the process involved to fix it
4. Version Control	Discusses the method and rationale		Discusses the method <i>or</i> rationale	
5. Agile Methodology	Reviews the current state of product, thoroughly describing backlogs and sprint tasks	Reviews the current state of product, describing most backlogs and sprint tasks tml>	Reviews the current state of product including some of the backlogs and sprint tasks	Reviews the current state of product omitting backlog or sprint tasks
6. Next Steps	The next steps are logical and insightful, and the team thoroughly discusses them	The next steps are mostly logical and insightful, and the team discusses them	The next steps are somewhat logical and the team mentions a few next steps	The next steps are minimal and the team inadequately discusses them