Jaybirds

Project Plan

Foundations of Software Engineering

Fall 2023

Akinwale | Brothers | Liu | Olson

Photo Credit: [The Johns Hopkins Hub]

1. Scope

1.1 Project Objectives

This project will result in a computerized, simplified version of the popular board game Clue®. The game will use a graphic layout of nine rooms, six weapons, and six characters.

1.2 Life Cycle Description: Design-to-Schedule

This project will use a variation of the staged delivery model called Design-to-Schedule. The project team will deliver working products in increments that are tied to instructor-required due dates. In the first phase of the life cycle, the team will focus on project planning and definition and then complete a requirements analysis. Once phase one is complete, the project team will deliver at least three and as many as four product releases that follow a cycle of detailed design, code, test, and deliver.[1] The project will deliver an architecture in in the first increment, followed by the most essential components of the game in the second increment, and then build toward higher grades with additional functionality in subsequent increments.

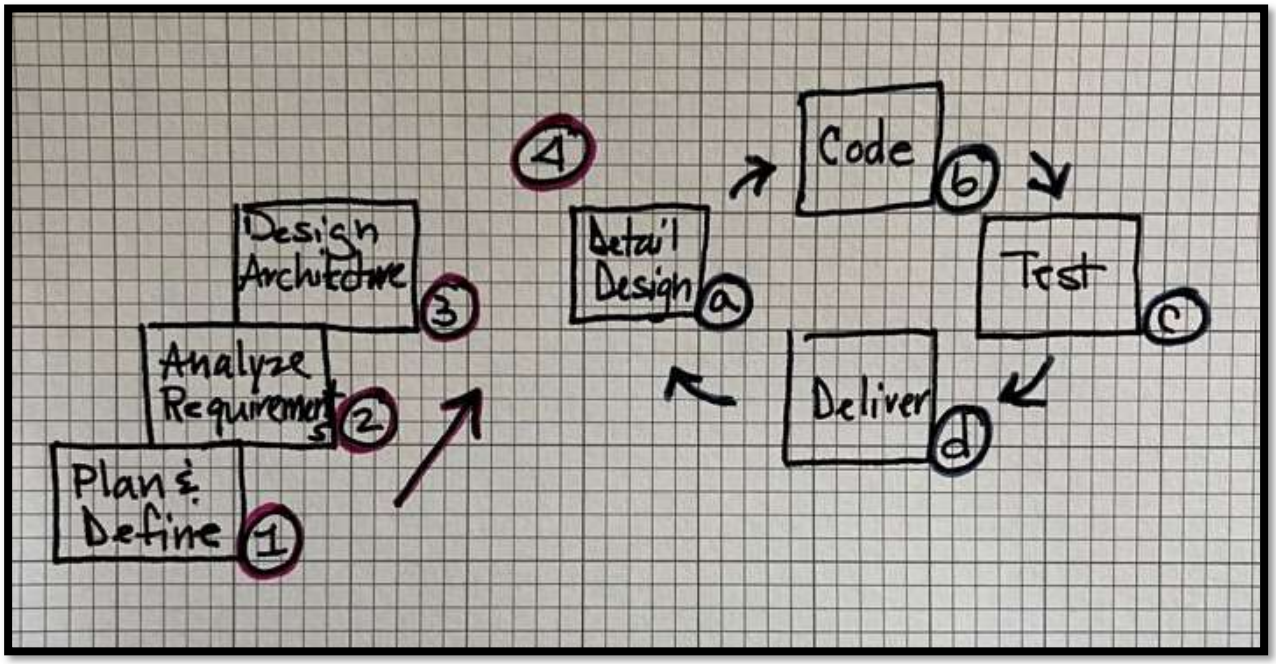

Figure 1: Design-to-Schedule Project Lifecycle

[1] Foundations of Software Engineering Course Materials, Module 3, Lecture C.

1.3 Deliverables

Product Type	Deliverable	Format
Code	Increment 1: Architecture	Python Script
Code	Increment 2: Minimal Text Based Client	Python Script
Code	Increment 3: Target with GUI	Python Script
Code	*Increment 4: Dream | Optional*	*Python Script*
Document	Project Plan	PDF
Document	Project Vision	PDF
Document	Requirements	PDF
Document	Design	PDF
Video + Credit List	Demo Increment 1	.mp4/.mov
Video + Credit List	Demo Increment 2	.mp4/.mov
Video + Credit List	Demo Final Increment	.mp4/.mov

2. Phases and Schedule

2.1 Milestones with Work Breakdown Structure (WBS)

The project will pass through four critical milestones underpinned by a series of work tasks.

★ **Milestone 1: Planning and Design Phase Complete**

Team Charter | Project Plan | Vision Document | Requirements Specification

During this phase, the project team will complete key planning documents including the team charter, project plan, and vision document. The project team will apply the use case technique to specify requirements that will guide the architecture decisions and detailed code design during the next phase as well as set up efficient and effective unit testing.

★ **Milestone 2: Stable Architecture Complete**

Program Diagram | Main Method | Object-Oriented Classes | Increment | Video Demo

During this phase, the project team will build and test the foundations of the program including a main method with the key movements defined and the object classes for weapons, characters, and rooms.

★ **Milestone 3: Stable Minimal Increment**

Instances | Movement and Game Logic | Increment | Video Demo

During this phase, the project team will build out the instances of each class. Once those classes are implemented, the team can refine the logic in the main function for taking turns, monitoring movement on the board, and the "winning" condition.

★ **Milestone 4: Target System Increment Complete**

Web Server | Interactive GUI | Increment | Video Demo

During this phase, the project team will build a server to host the game logic implemented in the previous increments. The server will be exposed to the broader internet via ngrok and allow for multiple users to connect to the same game lobby. The game logic from previous increments will be tied to an interactive frontend GUI.

★ **Dream System**

Memorable Images | Animation | Music | Game Chat

If time allows, the project team plans to add some extra features to the game.

2.2 Schedule

Date	Task and SubTasks	Deliverable	Dependencies
~~**Mon 11 Sep**~~	~~**1.00 Team Charter Complete**~~	~~**Team Charter**~~	~~**N.A**~~
~~Sun 24 Sep~~	~~1.01: Draft and edit project plan~~		~~1.00~~
Mon 25 Sep	1.02: Finalize changes to project plan		1.01
Mon 25 Sep	**1.03: Project Plan Complete**	**Project Plan**	**1.02**
Tue 26 Sep	2.00: Draft and share use case list		1.03
Wed 27 Sep	2.01: Jaybirds select use cases to draft		2.00

Sun 1 Oct	2.02: Share use cases for team review and edit.		2.00
Mon 2 Oct	3.00: Draft, share diagram for program		1.03
Mon 2 Oct	3.01: Format Main Class		3.00
Sun 8 Oct	2.06: Draft and edit vision document		1.03
Mon 9 Oct	2.07: Finalize changes to vision document		2.06
Mon 9 Oct	**2.08: Vision Document Complete**	**Project Vision**	**2.00-2.07**
Tue 10 Oct	2.09: Merge use cases into reqs draft		2.08
Wed 11 Oct	3.02: Write object class: character.		3.00
Thur 12 Oct	3.03: Write basic character move methods		3.02
Fri 12 Oct	3.04: Write object classes: weapon, room.		3.00
Sat 13 Oct	3.05: Unit Testing		3.00-3.04
Sun 15 Oct	3.06: Functional Testing		3.05
Sun 15 Oct	2.10: Finalize changes to requirements		2.08
Mon 16 Oct	**2.11: Requirements Document Complete**	**Requirements Document**	**2.10**
Tue 17 Oct	3.07: Fix remaining errors		3.06
Fri 20 Oct	3.08: Integrated Testing		3.07
Sat 21 Oct	3.09: Film Demo Video		3.08
Mon 23 Oct	**3.10: Stable Architecture**	**Increment 1 + Demo Video**	**3.09**
Tue 24 Oct	4.00: Write turn taking logic		3.10
Wed 25 Oct	4.01: Write movement logic		3.10
Thur 26 Oct	4.02: Write winning condition logic		3.10
Sun 5 Nov	4.03: Draft and edit design document		3.10
Mon 6 Nov	**4.04: Design Document Complete**	**Design Document**	**4.03**

Tue 7 Nov	4.05: Unit Testing		4.00-4.04
Wed 8 Nov	4.06: Functional Testing		4.05
Thur 9 Nov	4.07: Fix remaining errors		4.06
Fri 10 Nov	4.08: Integrated Testing		4.07
Sat 11 Nov	4.09: Film Demo Video		4.08
Mon 13 Nov	**4.10: Minimal System Increment**	**Increment 2 + Demo Video**	**4.00-4.09**
Tues 14 Nov	5.00: Write basic webserver framework		4.10
Wed 15 Nov	5.01: Design GUI layout		5.00
Thurs 16 Nov	5.02: Add logic for peer to peer interaction		5.00
Sun 19 Nov	5.03: Integrate minimal system with new web framework.		5.02
Tue 5 Dec	5.04: Unit Testing		5.03
Wed 6 Dec	5.05: Functional Testing		5.04
Thur 7 Dec	5.06: Fix remaining errors		5.05
Fri 8 Dec	5.07: Integrated Testing		5.06
Sat 9 Dec	5.08: Film Demo Video		5.07
Mon 11 Dec	**5.09: Target System Increment Complete**	**Increment 3 + Demo Video**	**5.08**

3. Quality Plan

The project will define quality as "fitness for use"[2] and design quality control activities–quality assurance, testing, and configuration management practices–to verify the product is free from faults or defects that could prevent a user from playing the game.[3]

[2] Frank Tsui, Orlando Karam, Barbar Bernal, *Essentials of Software Engineering,* 4th Edition (2018), Quoting Juran and DeFeo, p211..
[3] Frank Tsui, Orlando Karam, Barbar Bernal, *Essentials of Software Engineering,* 4th Edition (2018), p212.

3.1 Quality Assurance

The project takes a broad view of quality assurance and includes team preparation and process planning.[4] The project team will double as the quality assurance team. While this is not a best practice in industry,[5] it is the most realistic option for this exercise. Some initial quality assurance steps include:

- The project team members will write detailed use case requirements that enable object-oriented, modular development as well as sound testing.
- In addition to use cases, the project team will document edge cases and test them before the delivery of each increment to ensure consistent user experience.
- The project team has begun preparing for the GUI-increment with research into sound implementation strategies and techniques.

3.2 Testing

The test plan will enable the project team to identify defects and fix problems before an increment is delivered.[6] The testing process includes:

- **Unit testing**: Programmers will draw from use cases to design unit tests that stress a single procedure, method or class. The programmers will aim to try and "break" a function in order to build confidence that the code will not produce bugs later.
- **Functional testing:** Paired programmers will work together to merge new code and perform functional testing that verifies newly combined units function without defect.
- **Integrated testing:** All programmers will test the integrated system as part of a team inspection in advance of a demo and increment delivery. The team members will test the system from the user (a game player) perspective and record all errors for action.[7]

3.3 Configuration Management

The objective for the project configuration management plan is to put safeguards around change that will allow the team to efficiently build and deliver working code in increments. The project team will manage change through a set of team processes and tools in GitHub that should enable reproducibility, traceability, and reporting.[8]

- The `main` branch of the repository will contain the current production implementation of the game. It will be forked and frozen for each milestone completed. These frozen

[4] Frank Tsui, Orlando Karam, Barbar Bernal, *Essentials of Software Engineering,* 4th Edition (2018), p210.
[5] Frank Tsui, Orlando Karam, Barbar Bernal, *Essentials of Software Engineering,* 4th Edition (2018), p210.
[6] Frank Tsui, Orlando Karam, Barbar Bernal, *Essentials of Software Engineering,* 4th Edition (2018), p210, 213.
[7] 712-713
[8] Braude, M.J. & Bernstein, M.E. (2016). *Software Engineering: Modern Approaches (2nd ed.).* John Wiley & Sons.

branches will be versioned as v1.0, v2.0, etc. The first increment will be a baseline architecture of the code and later versions will include additional functionality.

- The team will list explicit tasks grouped under milestones on a Kanban type board in the team GitHub repository on the `Issues` page. Individual tasks will be assigned to developers according to developer abilities and project requirements.
- A merge/pull request will be tied to each task's development branch. This request will describe the task and how it has been implemented. It will also act as a reference to all commits for a given task.
- Programmers will complete each task on development branches and make regular commits to the main–each time a feature is successfully implemented and tested. In addition to the commit, the programmer will draft a concise commit message describing what was changed or added.
- The project team will implement a code review process that includes unit tests for the code prior to merge and a functional test with paired programmers after a merge. The goal of this step is to ensure no known broken code merges to the main branch.

4. Risk Assessment Plan

The project risk plan lists known risks and prioritizes the risks with an assessed high level of impact severity or likelihood for mitigation.[9] The project team acknowledges two key kinds of risk for this project:[10]

- Technical: Bugs, software incompatibility.
- Organizational: Staff Programming Language Knowledge, Schedule

Risk: Bugs

Likelihood: High

Impact: Potentially High

Mitigation: Apply unit, functional, and integrating testing process to ensure increments deliver working code.

Monitoring: Programmers can meet in pairs or as a group to address known bugs and perform testing.

Risk: Software Knowledge The project decided to use Python as our primary programming language, because it is good common ground and most of us are familiar with it. However, it is not a primary language for anyone on the team.

[9] Roger Pressman and Bruce Maxim, *Software Engineering: A Practitioner's Approach,* 9th Edition, (2020) p538.
[10] Ian Sommerville, *Software Engineering,* 10th Edition, (2017), p648.

Likelihood: High

Impact: Medium

High Mitigation Strategy: Programmers will help each other with syntax and functions that might be unique to the programming language. The programmers may be able to apply AI tools to “translate” a favored programming language into Python.

Monitoring: The project team expects this risk will diminish over time as programmers gain experience with Python.

Risk: Software Incompatibility

Likelihood: Low

Impact: Potentially High

Mitigation: Regularly test the software on different platforms and environments and address compatibility issues promptly.

Monitoring: Programmers will stay abreast of updates and changes to dependent software and test for compatibility regularly.

Risk: Schedule/Location The schedule for producing a working game is aggressive and the programmers have numerous additional responsibilities throughout the schedule. The programmers work from different locations and have different schedules with only limited overlap, which will make in-person or sametime virtual collaboration challenging.

Likelihood: Medium

Impact: Medium

Mitigation: The team will use a detailed schedule to ensure the work is broken down into manageable chunks that show measurable progress each week and help us resolve collaboration issues early in the project. The team has identified tools to simplify remote development and collaboration.

5. Conclusion The project team will conduct periodic check-ins and an ‘after action’ assessment following the delivery of the final increment to ensure continuous learning and improvement.