“Clue-Less” Project Plan

Team Undecided

1. Scope

1.1. Project Objectives

1.1.1. Participate in the engineering of a deliverable software product.

1.1.2. Exercise the skills and knowledge gained in the Intro to Software Engineering class.

1.1.3. Practice and apply sound software engineering approaches and show evidence of how this was accomplished.

1.2. Project Life Cycle Description.

1.2.1. Staged Delivery Model. The project deliverables follow a progression closely resembling a Staged Delivery Life Cycle Model that has specific requirements articulated and planned incremental releases (skeletal, minimal, target, and dream). As such, this is the most appropriate life cycle choice.

1.2.2. Releases under the Staged Delivery Model

1.2.2.1. Skeletal, due October 22, 2019

1.2.2.2. Minimal, due November 12, 2019

1.2.2.3. Target, due December 10, 2019

1.2.2.4. Dream features that can be accomplished before the Target due date will be delivered concurrently.

1.3. Deliverables (these are also milestones)

1.3.1. Vision Document, due October 08, 2019

1.3.2. Software Requirements Specification, due October 15, 2019

1.3.3. Skeletal System Build and Demo (see 1.2.2)

1.3.4. Software Design Document, due November 05, 2019

1.3.5. Minimal System Build and Demo (see 1.2.2)

1.3.6. Target System Build and Demo (see 1.2.2)

1.4. Features

1.4.1. Skeletal Build (architecture)

1.4.1.1. Game “Board” consisting of 21 spaces (9 room type and 12 hallways type) and their adjacencies

1.4.1.2. Game entities data (characters, weapons, etc.)

1.4.1.3. Answer Generation and tracking (combination of weapon, character, room)

1.4.1.4. Guessing ability

1.4.2. Minimal Build (most important functionality)

1.4.2.1. Terminal / Text-based interface

1.4.2.2. Game state

1.4.2.2.1. Includes where each piece is on the grid

1.4.2.2.2. Must also include whether a player was moved to a room by a different player (because then they can guess).
1.4.2.2.3. Must include opening state since players must move to the hallway on their first move.
1.4.2.2.4. Must include ordered turns.
1.4.2.3. Player Hand
1.4.2.3.1. Must be “dealt” by the game
1.4.2.3.2. Must not be broadcast
1.4.2.4. Game Synchronization and Notifications
1.4.2.4.1. Game must be multiplayer through some network infrastructure
1.4.2.4.2. All “table commands” (things that would be noticeable by other players in a physical game) must be broadcast to all player terminals
1.4.2.5. Movement functionality – must take game state and user choice as input and output a different game state
1.4.2.5.1. Includes limits on destinations
1.4.2.5.2. Differentiates between move to hallway and move to room (move to room prompts a guess command)
1.4.2.5.3. Must include rejection (you cannot move to a hallway where someone else is)
1.4.2.6. Deny command – when another player guesses
1.4.2.6.1. Must include options to deny based on person, deny based on weapon, deny based on location, or cannot deny
1.4.2.7. Accuse command – if denied by a different player, must eliminate that player
1.4.2.8. Guess command
1.4.2.8.1. Must move the guessed player

1.4.3. Target (target)
1.4.3.1. GUI input rather than text
1.4.3.2. Display options for each move rather than force user to know the command to use
1.4.3.3. Prompt for user to "randomly" select secret answer rather than game choosing
1.4.3.4. Verify deny commands
1.4.3.5. Allow for "skipping" turns if there are not 6 players

1.4.4. Dream (what you could do if you had enough time)
1.4.4.1. Option for user to load their own character name
1.4.4.2. Option for user to load their own character image
1.4.4.3. Sound effects for different guessing results

2. Organizational Structure

2.1. Management/Team Structure

2.1.1. Project Manager: Sean Walsh. Main point of contact for team communication, status checks, and performs the role as leader for conflict resolution. Ensures all deadlines are met by driving the

2.1.2. Lead Architect – Joel Huddleston. Responsible for the high level organization and interactions between modules in the product. Researches and recommends best practices and technologies.

2.1.3. Lead Programmer – Andrew Johnson. Directs and develops the features required for the delivery. Assigns and coordinates with developers to alleviate roadblocks when needed.

2.1.4. Lead Tester – Kira Ullman. Coordinates testing and develops scenarios/ edge cases for testing. Reports errors or problems found to other team members.

2.1.5. Lead Software Quality Assurance Engineer – Trey Hoffman. Ensures that activities contributing toward software development are performed to standards and implemented accordingly. Includes version control, code review, testing, documentation, and releases. Establishes and enforces productions standards.

2.1.6. Lead Configuration Management Editor – Sean Walsh. Ensures that the software is consistent in its performance and functionality from one version the next.

2.1.7. Deliverables Editor – Andrew Johnson. Finalizes and proofs all documents submitted on behalf of the team. Ensures consistent formatting and flow of the document.

2.2. Work Breakdown Structure

3. Monitoring and Control Procedures

3.1. Introduction

3.1.1. This Software Quality Plan establishes the standards and methods that will be used to perform the quality assurance functions of the Undecided Team "Clue-less" project.

3.1.2. Responsibility

3.1.2.1. A single member of the team will be identified as the software quality manager. In this role, that member will approve all software work products to be presented to the customer.

3.1.3. Scope

3.1.3.1. The Software Quality Plan establishes the following:

3.1.3.1.1. Identifies the role of quality assurance in the activities of software development

3.1.3.1.2. Establishes the activities and work products performed in the quality assurance role

3.1.3.1.3. Identifies the work products anticipated in the performance of quality assurance activities

3.1.4. References

3.1.4.1. The following external documents will be used as standards for the quality assurance function:

3.1.4.1.1. Johns Hopkins Java coding style

3.2. Communication and Review

3.2.1. The team will communicate using the Blackboard group discussion tool while developing products. For each work product, a discussion will be created to announce preliminary work product documents, discuss options for revision and determine the final form of the document.

3.2.2. Team members will use email to notify the group of status of their products such as a product being ready for review.

3.2.3. During code development, a minimum of two team members will be required to accept software modules into the project.

3.2.4. Documents will be considered accepted for submission when the quality manager determines that consensus is reached on the final form of the document.

3.3. Configuration Management

3.3.1. General

3.3.1.1. The configuration management function ensures that software modules meet requirements for compilation using the project established source language and version.

3.3.2. Source Language

3.3.2.1. The project will establish a single language standard for all source modules. The vendor and exact version of this language will remain constant throughout the development life-cycle.

3.3.3. Development Environment

3.3.3.1. Individual software module contributors will choose their own development environments, including any preferred editor, compiler, debugger and run-time.

3.3.4. Compiler Requirements

3.3.4.1. All software modules must be submitted to a common repository to perform the integration testing function. Modules will consist of text source in the assigned language.

3.3.5. Code Repository

3.3.5.1. The team will use a single software repository for all source files included in project deliveries. All team members will have equal access to source files in order to facilitate the unit testing function.

3.3.6. Build System

3.3.6.1. A software build team will consist of 1 or 2 members. It will be the responsibility of the software build team to build all deliverable program modules in their final form. At every stage of the project, exactly one build server will be used. No software changes will be allowed on the build system in the development tool chain for the duration of the project.

3.4. Software Testing

3.4.1. General.

3.4.1.1. All software will be tested according to criteria to be developed during design.

3.4.2. Unit Tests

3.4.2.1. The purpose of unit testing is to verify that each unit of a software system performs as designed.

3.4.2.2. Each code module will be tested against the requirements document before acceptance. Such tests may be performed outside of the build system.

3.4.3. Integration Tests

3.4.3.1. The purpose of this level of testing is to expose faults in the interaction between integrated units (multiple modules) within a system.

3.4.3.2. All code must pass integration test acceptance after compilation by the build team using the official build system.

3.4.4. System Tests

3.4.4.1. The software system will be tested "end-to-end" at each build phase (skeletal, minimal, target) to ensure compliance with specified functional and non-function requirements.

3.4.5. Acceptance Tests

3.4.5.1. The software product will be tested by the grader to ensure that the product is an acceptable deliverable at the due dates of the particular phases.

4. Project Schedule Estimate

4.1. See included document: project_schedule.pdf

5. Project Risks and Contingencies

5.1. See included document: risk_management_plan.pdf