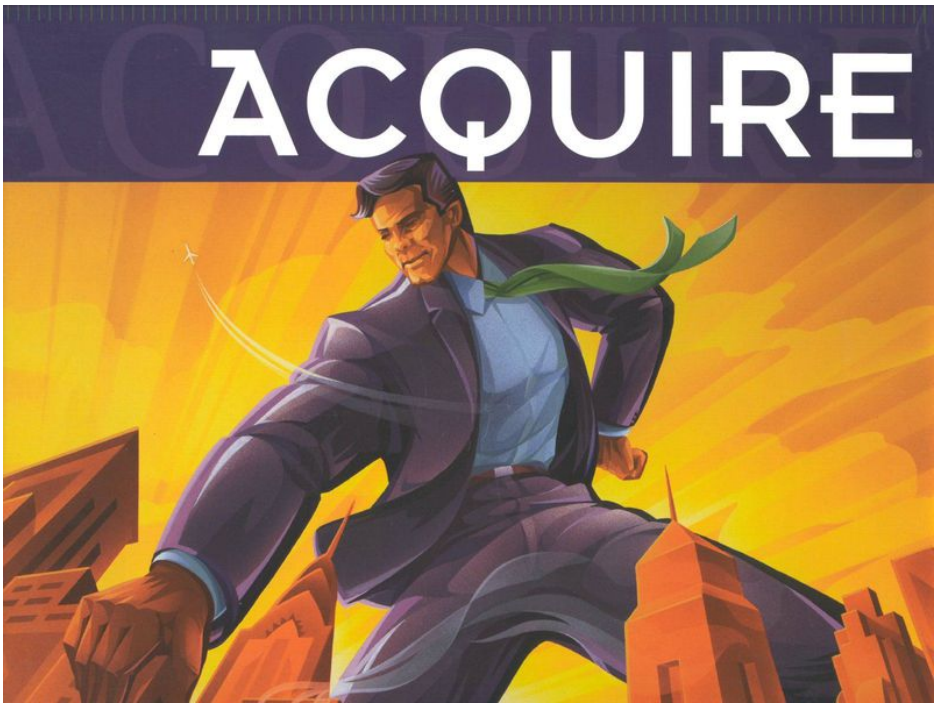
Work-Package 1: Project Guidelines

CS 3321/INFO 3307 Project

Isaac Griffith

August 2019



CS 3321/INFO 3307 Project

Isaac Griffith

Informatics and Computer Science
Idaho State University

Project Guideline

Prepared for CS 3321/INFO 3307

Table of Contents

1 Project Description

Your team has been tasked with the development of a digital version of the Acquire board game. From boardgamegeek.com: *"In Acquire, each player strategically invests in businesses, trying to retain a majority of stock. As the businesses grow with tile placements, they also start merging, giving the majority stockholders of the acquired business sizable bonuses, which can then be used to reinvest into other chains. All of the investors in the acquired company can then cash in their stocks for current value or trade them 2-for-1 for shares of the newer, larger business. The game is a race to acquire the greatest wealth."*

You can find the official rules for the game at the following url: `http://www.wizards.com/avalonhill/rules/acquire.pdf`

I suggest studying these rules as they are the basis for your requirements.

2 Project Objectives

Team Dynamics

- Create and follow an agenda for a team meeting.
- Identify and justify necessary roles in a software development team.
- Understand the sources, hazards, and potential benefits of team conflict.
- Apply a conflict resolution strategy in a team meeting.
- Demonstrate through involvement in a team project the central elements of team building and team management.
- Create a team by identifying appropriate roles and assigning roles to team members.
- Assess and provide feedback to teams and individuals on their performance in a team setting.

Project Management

- Use an *ad hoc* method to estimate software development effort (e.g., time) and compare to actual effort required.
- Use a project management tool to assist in the assignment and tracking of tasks in a software development project.
- Explain how risk affects decisions in the software development process.
- Demonstrate a systematic approach to the task of identifying hazards and risks in a particular situation.
- Conduct a cost/benefit analysis for a risk mitigation approach.
- Identify and analyze some of the risks for an entire system that arise from aspects other than the software.
- Use a defect tracking tool to manage software defects in a small software project.

Process

- Assess a development effort and recommend potential changes by participating in process improvement or engaging in a project retrospective.
- Define a user-centered design process that explicitly takes account of the fact that the user is not like the developer of their acquaintances.

Measurement

- Use project metrics to describe the current state of a project.
- Track the progress of some stage in a project using appropriate project metrics.
- Compare simple software size and cost estimation techniques.
- Estimate the impact of a change request to an existing product of medium size.

Analysis

- Identify both functional and non-functional requirements in a given requirements specification for a software system.
- Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system.
- Use a common, non-formal method to model and specify the requirements for a medium-size software system.
- Explain the relationships between the requirements for a software product and its design, using appropriate models.
- Investigate the impact of software architectures selection on the design of a simple system.
- For an identified user group, undertake and document an analysis of their needs.
- Conduct a quantitative evaluation and discuss/report the results.

Design

- Use a design paradigm to design a simple software system, and explain how system design principles have been applied in this design.
- Construct models of the design of a simple software system that are appropriate for the paradigm used to design it.
- For a simple system suitable for a given scenario, discuss and select an appropriate design paradigm.
- Create appropriate model for the structure and behavior of software products from their requirements specifications.
- Design a contract for a typical small software component for use in a given system.

Development

- Create a prototype of a software system to mitigate risk in requirements.
- Select suitable components for use in the design of a software product.
- Use refactoring in the process of modifying a software component.
- Develop and use a conceptual vocabulary for analyzing human interaction with software: affordance, conceptual model, feedback, and so forth.
- Create a simple application, together with help and documentation, that supports a graphical user interface.

3 The Process and Tools

For the duration of this project your team will follow a specific software engineering process. The process your team follows will be a decision left to the team. Each team may choose from one of the following three process models: * Scrum * ScrumBan * Lean Software Development

It is up the team and its members to study the selected methodology and identify how the team will follow the processes methods to achieve their goals.

To Help follow this process and to make the collaborative effort more effective each team will be using several tools to make the process most effective.

- Programming Language: Each team will select from one of the following programming languages
  - Java
  - C#
  - C++
  - Python
- Integrated Development Environment: I would suggest that each team select an appropriate development environment such as Visual Studio or those created by JetBrains (I would recommend the latter over the former).
- Modeling: To collaboratively create UML Models, each team is required to utilzie LucidCharts.
- Configuration Management and Version Control: Each team is required to utilize GitHub to manage their project. Each team will have a project created for them, they are to use this and associated components for the duration of the project. An invitation to the project will be delivered during project kickoff.
- Project Management and Planning: Each team will be required to identify and utilize a tool for project management and planning.
- Documentation: The team will utilize the project's GitHub wiki to document their project as they progress. Two main documents will be need to be created. The first is the project proposal and the second is the final report.
- Automated Build: Each team is required to connect their project to Travis CI in order to ensure that their project may be built automatically and outside their individual team's machines.

4 Project Deliverables

4.1 D01 – Project Proposal (Due: Monday 10/14/2019 @ 2300)

You will be assigned teams on the Monday of the 7th Week of class. During this week it is your job, as a team, to:

- Review the Project Guidelines
- Each team member should familiarize themselves with LucidCharts and its integration with Google Docs.
- Each team will need to select, learn, and follow one of the following software development lifecycles for use throughout their project: Scrum, ScrumBan, Lean Software Development.
- Get to know your team members and identify your roles on the team. **Roles should be assigned based on the SDLC selected**
- Each team member will be sent a link to a GitHub repository for their team. Each team member will need to join that repository and ensure that they have access to it. **I suggest designating one individual to maintain the team's gitflow.**
- Each team will need to plan the initial iteration of their project.

During this first iteration, you will need to compose and submit the initial System Proposal for your project. This should be documented within your repository wiki, and must include the following sections:

- **Project Vision**: A summary of the essential information describing the project and most importantly your concept of the projects Minimum Viable Product breakdown for both implementation iterations.
- **Process and Tools**: This section describes the Software Development Process you will be using, why you selected this process, and how your team will adhere to and follow this process. You will also need to describe the roles that each team member will be fulfilling from the process, your planning process, and how this process will extend across the three iterations of the project. This section should also indicate the frameworks, tools, and languages you intend to use and the rationale justifying the selection of each. You will also need to identify how you are going to follow and maintain gitflow.
- **Requirements Definition**: A listing of the User Stories and their task lists identified during the Analysis phase of the project. In the first phase (planning) this needs to be the first development iteration's requirements.
- **Structural Models**: This section contains the class diagram models for the system. These include any domain models, and other structural models which fully describe the system.
- **Behavioral Models**: A set of sequence diagrams and behavioral-state machines that describe the internal behavior of the system. In addition any major design decisions made should be documented as well.

- **Functional Models**: A set of activity diagrams that illustrate the basic processes or external functionality that the systems needs to support.
- **User Interface**: A series of sketches documenting the design and layout of the User Interface.

Note: Each diagram that you include in your report must be accompanied by a description and explanation within the text of the enclosing section

4.2 D02 – Iteration Phase 1 (Due: 11/04/2019)

At the close of the initial iteration the following must be complete:

- Each team must have completed a working version of the software representing their initial Minimum Viable Product. And be capable of demonstrating this either in class or during a meeting, if requested.
- Each team will need to maintain and update their system proposal in the Git.
- Each will need to update their plan for the second development iteration.
- Each team will need to identify the features for the next iteration of development, and update the minimum viable production section of their proposal.
- Setup and complete a Code Review with Professor Griffith prior end of iteration.

4.3 D03 – Final Documentation and Project Presentation (Due: 12/05/2019)

You will need to compose and submit the initial System Proposal for your project. This should be documented within your repository wiki, and must include the following sections:

- **Project Vision**: A summary of the essential information describing the project and most importantly your concept of the projects Minimum Viable Product breakdown for both implementation iterations.
- **Process and Tools**: This section describes the Software Development Process you will be using, why you selected this process, and how your team will adhere to and follow this process. You will also need to describe the roles that each team member will be fulfilling from the process, your planning process, and how this process will extend across the three iterations of the project. This section should also indicate the frameworks, tools, and languages you intend to use and the rationale justifying the selection of each. You will also need to identify how you are going to follow and maintain gitflow.
- **Requirements Definition**: A listing of the User Stories and their task lists identified during the Analysis phase of the project. In the first phase (planning) this needs to be the first development iteration's requirements.
- Structural Models: This section contains the class diagram models for the system. These include any domain models, and other structural models which fully describe the system.

- **Behavioral Models**: A set of sequence diagrams and behavioral-state machines that describe the internal behavior of the system. In addition any major design decisions made should be documented as well.
- **Functional Models**: A set of activity diagrams that illustrate the basic processes or external functionality that the systems needs to support.
- User Interface: A series of sketches documenting the design and layout of the User Interface.
- **Lessons Learnt**: A listing of issues that occurred during execution of the project. This does not include personnel issues but should focus on issues in framework selection, language selection, operating the selected software process, etc.

Note: Each diagram that you include in your report must be accompanied by a description and explanation within the text of the enclosing section

Additionally, each team will need to present their final project as part of the project demonstrations during the last day of class. **(Due: 12/05/2019)**

- Each team will be given 20 minutes to demonstrate the project. This demo should consist of a description of their concept of the MVP, the tools and languages used, and a working version of the project. The presentation will be graded by the other teams as well as Professor Griffith.
- Additionally each team member will need to complete a review of their other team members. These reviews will be held in the strictest of confidence and remain anonymous. There is a form that will be used, one per teammate, and will be submitted via moodle.
- Each team will need to schedule a Iteration Retrospective near the end of the second development iteration, with Professor Griffith. **(Due: 12/02/2019)**
- Each team member must complete Peer Reviews **(Due: 12/11/2019)**

5 Grading

This chapter describes the grading criteria for each deliverable for both the team and the individual members of a team. This chapter also describes the single guaranteed path to failure on this project. We will begin with the latter.

5.1 Single Guaranteed Path to Failure

This project can easily be failed as a team in the following way:

- As a team disregard this document, that is do not adhere to the guidelines of the project.
- As a team choose not to follow the prescribed software engineering process for this project.
- As a team do not work towards forming a jelled team working towards the best possible outcome and product for this course.

This project can easily be failed as an individual in the following way:

- As an individual team member not accepting and completing (to the mutual satisfaction of your team members) an equal share of the work (including design, documentation, and most importantly coding).
- Not writing any code on the project. This is an engineering course and a fair share of it is design, but if you don't contribute to the code, you made no contribution to the final project.

5.2 Team Component

Component	Sub-Component	Percentage
Project Proposal		30%
	Planning	40%
	Report	60%
Final Project Report		50%
	Iteration 1 Development	30%
	Iteration 2 Development	30%
	Final Report	60%
Project Presentation		20%

5.3 Individual Component

The projects though graded on a team basis are not actually team assignments, but rather individual assignments. Regardless of how well you as an individual perform in the team, your team score acts as the maximum bound on your individual score. Thus, the team can fail the project or pass the project, and an individual can fail or pass the project based on their performance. Furthermore, it is every team members responsibility to ensure that the team succeeds. That being stated, you individual score is composed of the team score modified by an individual project multiplier (IPM) value.

The IPM value is derived using the following formula:

$$IPM = 0.25 * PEER + 0.5 * CONTRIB\_INDEX + 0.15 * MEETING + 0.1 * PRESENT$$

Where:

- PEER is the aggregated value of your peer reviews (peer reviews about you), which is a value between 0.0 and 1.0.
- CONTRIB_INDEX is the evaluation based on git commits and assigned authorship in the repository concerning code you authored, the apparent equal sharing of backlog items (weighted by story points) assigned and completed by you, and the apparent amount of contribution you put forward in the documentation (design and writing components) of the project. This is a value between 0.0 and 1.0
- MEETING is a factor representing your appearance at scheduled code reviews and team retrospectives. Note that if your team does not conduct these with Professor Griffith, the value is 0. This is a value between 0.0 and 1.0.
- PRESENT is a factor representing your participation during your team's final project presentation. This is a value between 0.0 and 1.0.