# CSCE 361 Course Project, Increment 1

Due: February 4, 2019

## 1 Assignment

Review the project description for your project. Communicate with your TA to clarify the system's requirements. You can do this during your TA's general office hours, during your mid-increment feedback session, and/or through Piazza. There is a folder for each project on the course's Piazza site; questions and answers about the system's requirements that are placed there would be visible to the other teams working the same project (this is not a bad thing). Each team will still need to complete its own Requirements Analysis Document (RAD).

I expect all team members to participate in requirements elicitation and the drafting of your RAD.

While you won't be writing any code during this increment, you should start thinking forward to implementation. By the end of Increment 2, you will need to have decided on the target platform and the implementation language. *You may use any appropriate implementation language or other technology, provided everyone on your team knows that languages or technology before the implementation phase of the waterfall. Similarly, your target platform may be any reasonable target platform, provided everyone on your team can write code for that platform before the implementation phase of the waterfall.*

### 1.1 Requirements Analysis Document

A template for the Requirements Analysis Document is on the next page. Prepare your own document using the sections described in the template. You may leave Section 2 (Current System) blank. The sub-subsections for the Subsection 3.3 (Nonfunctional Requirements) are examples; use those that are appropriate for your system. Begin preparing your use cases; you do not yet need complete set of use cases. For your structural model) and behavioral model indicate *TBD* for now. For your user interface), provide a wireframe of the main user interface. You can sketch it by hand or using a tool such as `draw.io` or `gomockingbird.com`.

As you prepare section 3, be sure that each functional requirement, each nonfunctional requirement, and each use case has a distinct paragraph number. This will help as you trace design decisions and test cases back to requirements. Further, be sure that each functional and nonfunctional requirement is testable (falsifiable): if you cannot answer the question "How will I objectively show that the system does not satisfy this requirement?" then the requirement is not testable.

## 1.2 Configuration Management

You may want to use a collaborative-editing tool such as Google Docs to work on your RAD. In your `git.unl.edu` repository, create a directory called Requirements. Save your RAD as a pdf file and place it in the Requirements directory. The file that is in your Git repository at 11:59pm on February 4 will be the document we review during grading. *Be sure that your Git repository is private and accessible only to the members of your team, your TA, and the professor.*

# 2 Requirements Analysis Document Template[1]

1. Introduction
    - 1.1. Purpose of the system
    - 1.2. Scope of the system
    - 1.3. Objectives and success criteria
    - 1.4. Definitions, acronyms, and abbreviations
    - 1.5. References
    - 1.6. Overview
2. Current System
3. Proposed System
    - 3.1. Overview
    - 3.2. Functional Requirements
    - 3.3. Nonfunctional Requirements
        - 3.3.1. Usability
        - 3.3.2. Reliability
        - 3.3.3. Performance
        - 3.3.4. Supportability
        - 3.3.5. Implementation
        - 3.3.6. Interface
        - 3.3.7. Packaging
        - 3.3.8. Legal
    - 3.4. Models
        - 3.4.1. Use cases
        - 3.4.2. Structural model
        - 3.4.3. Behavioral model
        - 3.4.4. User Interface
4. Glossary

*The first section of the RAD is an Introduction. Its purpose is to provide a brief overview of the function of the system and the reasons for its development, its scope, and references to the development context (e.g., reference to the problem statement written by the client, references to existing systems, feasibility studies). The introduction also includes the objectives and success criteria of the project.*

*The second section, Current system, describes the current state of affairs. If the new system will replace an existing system, this section describes the functionality and the problems of the current system. Otherwise, this section describes how the tasks supported by the new system are accomplished now.*

*The third section documents the requirements elicitation and the analysis model of the new system. The overview presents a functional overview of the system. Functional requirements describe the high-level functionality of the system. Nonfunctional requirements describe user-level requirements that are not directly related to functionality. This may include usability, reliability, performance, supportability, implementation, interface, operational, packaging, legal requirements, and possibly others. System models*

[1]Modified from template provided by Bruegge & Dutoit

*describes the scenarios, use cases, structural model, and behavioral models for the system. This section contains the complete functional specification, including mock-ups illustrating the user interface of the system and navigational paths representing the sequence of screens. The subsections Structural model and Behavioral model are written during the Analysis activity.*

*At the end is the glossary of important terms, to ensure consistency in the specification and to ensure that we use the client's terms.*

# 3 Rubric

_____ 10 points for following directions

4: repository on git.unl.edu is private and shared with the TA and the professor

3: RAD is a pdf file

3: RAD is in the Requirements directory in the repository

_____ 20 points for completing Section 1

_____ 10 points for completing Subsection 3.1

_____ 20 points for Functional Requirements

10: a reasonably complete set of requirements. Do not deduct for minor oversights. For significant oversights, deduct based on your judgment of what fraction of the system's requirements are missing.

10: requirements are testable. Deduct based on the fraction of the listed requirements that are not testable.

comment-only: listed requirement is a non-functional requirement

comment-only: overly-compounded requirements

_____ 15 points for Nonfunctional Requirements

10: at least five non-functional requirements.

5: requirements are testable.

comment-only: listed requirement is a functional requirement

comment-only: overly-compounded requirements

_____ 10 points for starting the use case model

_____ 10 points for preparing a wireframe of the main user interface

_____ 5 points for starting Glossary

# 4 Package Tracker

Bohn's Drones is a courier service using small unmanned aerial systems (SUAS) to deliver packages within the Lincoln and Omaha, Nebraska, areas. When a customer needs a package to be delivered, a SUAS is dispatched from a nearby depot to pick up the package.

Because the SUAS has a limited range, there are depots every ten miles along I-80 between Seward and the Missouri River. There are also depots in Lincoln at the intersection of O Street and 27th Street, at the intersection of O Street and 84th Street, and at the intersection of 84th Street and Nebraska Highway 2. When a SUAS with a package arrives at a depot, the package is handed off to another SUAS which will carry the package to the next depot or the destination (if the destination is within range).

A customer (both the sender and the receiver) should be able to observe the status of a delivery, to include the point of origin and the destination, when the SUAS was dispatched, when the SUAS picked up the package, when the package was handed off to another SUAS at each depot visited, and when the package was delivered. A customer should also be able to generate a delivery request, which will cause a SUAS to be dispatched automatically to pick up the package.

The Bohn's Drones staff should be able to observe where each SUAS is, whether at a depot, between depots, en route to/from a customer, or at a customer's location. The staff should be able to observe which package is aboard which SUAS. Just as the customers can, the staff should be able to observe a package's status. While dispatching a SUAS to pick up a package is automatic, the staff should be able to dispatch an empty SUAS from one depot to another.

# 5 Boggle Game

Procrastination Pastimes wants you to develop a computer game of Boggle. The program should implement the rules of the pencil-and-paper version of Boggle as faithfully as is possible, with one exception: there will only be one player. The rules for the pencil-and-paper version of Boggle are available at

`https://www.hasbro.com/common/instruct/boggle.pdf`

The sixteen dice have the following letters:

| | | | |
|---|---|---|---|
| R I F O B X | I F E H E Y | D E N O W S | U T O K N D |
| H M S R A O | L U P E T S | A C I T O A | Y L G K U E |
| Qu B M J O A | E H I S P N | V E T I G N | B A L I Y T |
| E Z A V N D | R A L E S C | U W I L R G | P A C E M D |

When the player indicates that they wish to start a new game, the dice should briefly be animated (please do not create a 3D physics model). The dice can settle into any permutation of the sixteen die positions, and each die can have any of its six faces showing. When the dice settle, the timer should begin. The timer should be displayed so the player has a clear

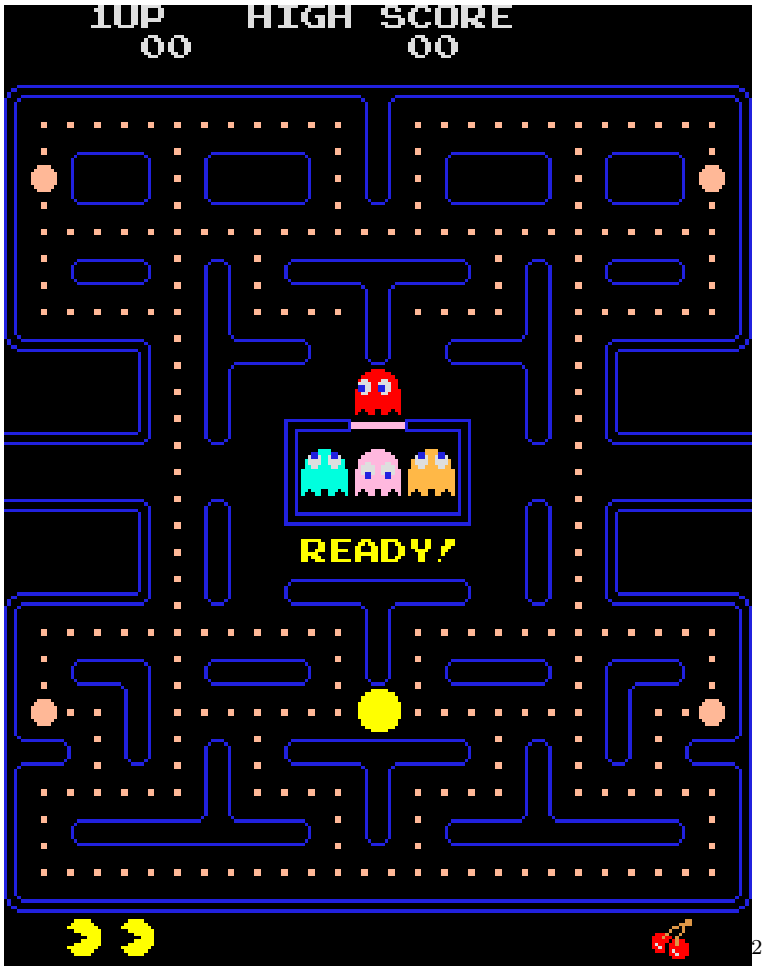
visual indication of time progressing; you may choose whether this is a representation of an egg timer, a dial timer, or some other visualization.

The player should then be able to enter as many words as they can until the timer runs out. The method for entering the words should be appropriate for the target platform. If the user enters a word that is too short or is a duplicate of a previously-entered word, the program should silently allow the error to occur, so as not to distract the player.

After the timer has run out, the program should remove duplicates words, words that are too short, and words that do not appear in the corpus. The program should then calculate and display the player's score.

# 6 City Traffic Simulator

Mayor Pat Mann wants to be able to find a good arrangement of traffic lights and stop signs for the city of Pacopolis. The Pacopolis city map looks like this:




[2]

[2]Pac-Man maze obtained from http://pacman.wikia.com/wiki/Maze

The ghosts, the large yellow circle that's Pac-Man with a closed mouth, and the word "READY!" are there because this is a screenshot of a game, and they have no significance in the application you'll write.

The space between two dots is $\frac{1}{8}$ mile. All roads are two-way roads, and the speed limit on all roads is 30mph. An intersection is any location where a car has more more than one option for its next direction. The intersection itself is at the dot in the center of the intersections on the map (the size of the intersections on the map are not to scale). Each intersection must have either a traffic light or stop signs.

A traffic light alternates between allowing north-south traffic and allowing east-west traffic. Traffic lights stay cycle between traffic directions every two minutes. When changing from green to red, a light is yellow exactly long enough for a car to transit the intersection. You can assume that an intersection is exactly large enough to hold one car traveling in each direction, and therefore you can simplify the traffic light as one that has no yellow light provided that cars enter the intersection only if they have the green light and there is no cross-traffic in the intersection.

An intersection with all-way stop signs admits cars into the intersection based on their order of arrival (if multiple cars arrive at the exact same time, use random selection to break the tie). If the intersection has stop signs for only north-south traffic or for only east-west traffic, then cars in the directions without stop signs may immediately enter the intersection, and cars in the directions with stop signs must stop and may enter the intersection only if there is not cross-traffic about to enter the intersection.

Some simplifying assumptions about cars' speeds: A car that is stopped remains stationary for $\frac{1}{2}$ second after the reason for its stoppage has ended (*i.e.*, the light has turned green, or the intersection has cleared, or the stopped car ahead of it has moved forward. When a stopped car begins moving, it will travel 15mph for the first $\frac{1}{8}$ mile and 30mph thereafter until it stops. When a car must stop, it stops immediately (there is no need to slow down). All cars honor these speeds exactly.

The roads that proceed off of the map to the east and west are ramps to/from the local highway.

All cars have a starting point on the map and a destination. The start & end points should be able to be established by the user, but not necessarily within the application our students will write. You might consider using comma-separated values, which the end user could create in a spreadsheet. Mayor Mann will use your program to find the combination of traffic lights and stop signs that gets most cars to their destinations the fastest. You do not need to find that combination; you need to create an application that will allow the mayor to try different combinations.