5-10-2019

Inventory Management for Sports Equipment: Agile Project Management

Nicholas Barnard
The College at Brockport, nbarnard@twcny.rr.com

Inventory Management for Sports Equipment: Agile Project Management

A Senior Honors Thesis

Submitted in Partial Fulfillment of the Requirements
for Graduation in the Honors College

By
Nicholas Barnard
Computer Science Major

The College at Brockport
May 10, 2019

Thesis Director: Dr. Sandeep Mitra, Professor, Computer Science

Table of Contents

Abstract

Introduction

Software Development Methodologies
- Waterfall development
  - Description
  - Positives & negatives
- Spiral development
  - Description
  - Positive & negatives
- Agile development
  - Description
  - Positive & negatives
  - Implementation
- Comparison

Project Management
- Roles of a project manager
- Agile project management

Design Process
- Initial requirements capture
- Use cases
- Sequence diagrams
- GUI mockups & State Diagram

Results/Discussion

References

Acknowledgments

I. Abstract

The goal of this Agile software development project is to create an application to manage the inventory and the rental process of an on-campus organization. At the start of the project, I met with the client to understand their current business processes, which were largely paper-based. I then undertook a requirement capture process to better understand the features needed in the envisaged software application, making sure to keep close contact with the customer. A design phase was started using an Agile Modeling approach to create a minimal model that primarily outlines the behavior of the software application, relying mainly on UML sequence diagrams and GUI mockups/state diagrams. After each check-in with the client the design was modified accordingly. The main objective is to practice the Agile methodology to its fullest, both in the overall project management and, especially, in the development and testing phases.

II. Introduction

The College at Brockport has a multitude of services offered to students throughout their campus. The academic department for Kinesiology, Sport Studies, and Physical Education, KSSPE for short, provides a lot of opportunities to its students. Courses within this department range from scuba diving, skill levelled sports, coaching, and lecturing. In order to offer these experiential courses, the department houses equipment that both professors and students can reserve to use. All of this equipment is

housed in the aptly named KSSPE Equipment Room within the athletic complex on the campus. Currently, if a professor or student desires to reserve and take out equipment they notify either a student worker in the room, or the staff member that oversees the room. The equipment is then set aside for them to pick up when they need it, and after it is returned, shelved. To keep track of reservations, and currently checked out equipment, a weekly paper and clipboard system is utilized to track the day to day operations. This thesis project undertook the development of software for inventory and reservation management. This system will keep track of the users and equipment involved with the system allowing for the addition, modification, and removal of users and equipment. The system will also be able to track any equipment currently reserved, returned, and the generation of reports for data tracking. The goal is that this software will reduce the amount of paper use, provide an ease of equipment tracking, and prevent equipment loss. This software development project, undertaken by a team of four students, and a faculty advisor, was broken into two distinct sections, that provide summaries of the two major parts of the development, project design and management, and project development and deployment. This paper will focus on the former.

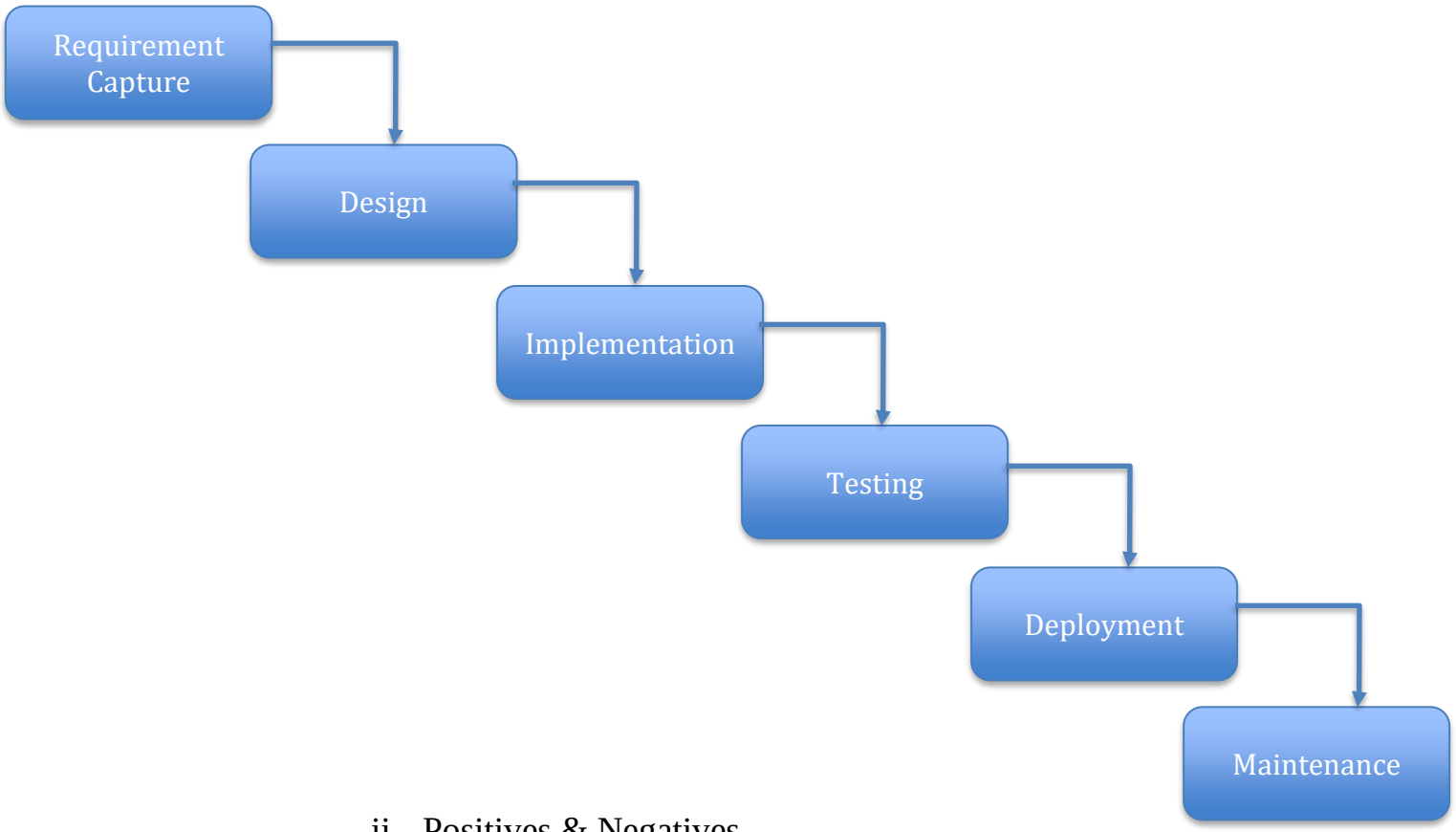
III. Software Development Methodologies

a. Waterfall Development

i. Description

This method, also known as the linear-sequential life cycle model, was the first process model used in software development. The method is simplistic by nature, in short, the workflow for this model is that each phase of development must be completed before the next phase is initiated. After a phase is moved on from, going backwards between phases does not occur. This linear sequential movement removes the possibility

of phase overlapping. This method is typically broken up into six separate phases: requirement capture and analysis, design, implementation, testing, deployment, maintenance. The first phase is where all requirements that the desired system must meet are figured out and documented. In the design phase, the requirements are studied, and models are made to clarify the software architecture. In implementation, the software code is written in units that are then later tested for proper functionality in the testing phase. After all tests are passed, the software is deployed to the client and maintained for any bugs or issues that might occur post deployment. The figure below depicts the flow of phases within the waterfall method.



ii. Positives & Negatives

The main positives of the Waterfall method come from its strict workflow. Since it is a simple phase to phase transition it is easy to understand and manage the workload.

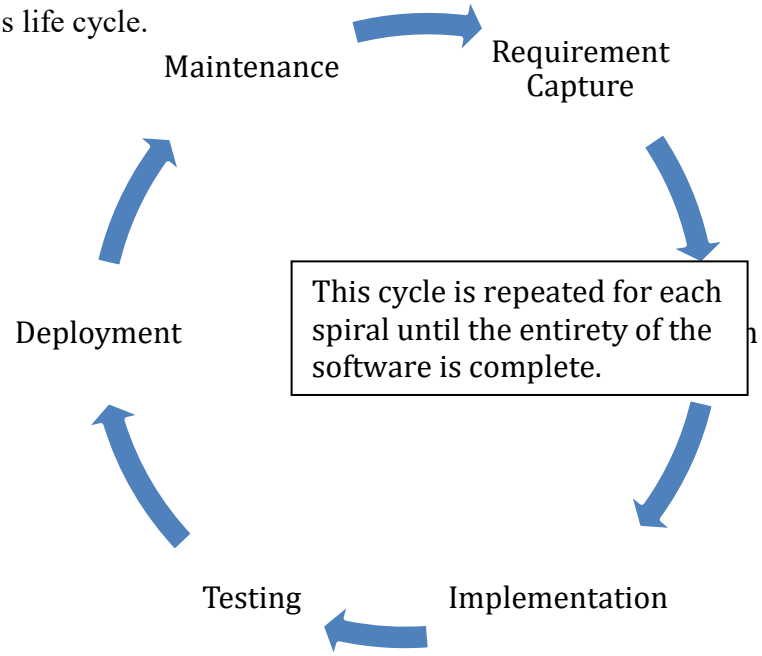
Task completion is easy to follow and review as it is done on a phase by phase basis. Lastly, since the phases do not overlap, each phase's start and end point are easy to pinpoint. The main negatives of the Waterfall method are in the lack of revision. Due to the lack of backtracking to prior phases, changing requirements can not be accommodated which can upset clients. This model also does not have truly visible results until late in the life cycle which means there can be a long wait time before a part of the product is released for client review.

b. Spiral Development

i. Description

The Spiral method follows a less strict phase structure than the Waterfall method. As its name states, the phases of the software development life cycle are organized now in a spiral formation. The spiral method contains the same core six phases that the Waterfall method had, requirements capture, design, implementation, testing, deployment, and maintenance. This method starts with an initial requirement capture to gain an overall understanding of the required software, but with a focus on a smaller portion of the entire project. This focus will be the section of the software completed in the first spiral. Each spiral consists of the six phases, and at the end of each spiral, a new portion of the software is deployed to the client. When a new spiral starts, any client feedback from the previous spiral is addressed in the new requirement capture, as well as

any initial requirement revisions or additions. Below is a figure of one spiral within the spiral method's life cycle.



ii. Positives & Negatives

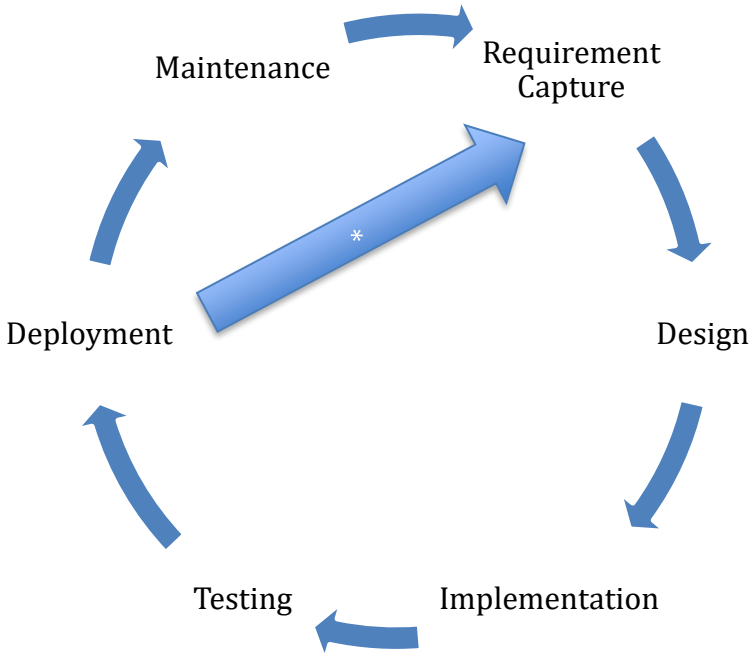
The positives of the Spiral method are almost in direct contrast with that of the Waterfall method. Deployment of the product to the client is faster as it is broken up into partial deployments that the client can get their hands on throughout development. Requirement changes are also able to be handled with ease due to the requirement capture phases at the start of each spiral. Increased requirement capture, however, can also become a negative. This can lead to repetitive cycles if the client continually changes their mind. The end of the project can also be hard to predict due to the ever-changing requirements and difficulty to predict the duration of each spiral's workload.

c. Agile Development

i. Description

The Agile method has similarities to the spiral method, but with clear end points for the workload contained within each run through of the six phases, primarily referred to as sprints. The six phases are then broken up into smaller subphases to increase manageability. Each sprint in the Agile method contains a very specific section of the

overall project. Until that section meets the clients needs, the sprint is not complete. Once complete, a new section of the project is initiated in a new sprint. The primary focus of the Agile method is adaptability and client satisfaction. This is the method chosen for the development of the KSSPE Inventory Management System software. Below is an example of a sprint within the Agile method.



*At the deployment phase, if the client is content, the sprint's product is deployed, maintained, and the next sprint is initiated. Otherwise, the client feedback is taken back to the team, and the sprint is restarted

ii. Positives & Negatives

Going off of its primary focus, the biggest positives with the Agile method are increased adaptability and client satisfaction. That being said, this method also makes it easy for fixes, increased teamwork, and efficient product deployment. With increased adaptability comes the negative of feature creep, the never-ending addition of requirements by the client. This also makes it hard to predict an end to the project. As this

method is also heavily dependent on client interaction, if the client is vague with some details, the developing team can have difficulties meeting these requirements.

iii. Implementation

As previously mentioned, the Agile method was chosen for the KSSP Inventory Management software development. The Agile method was the frame we followed for the software development lifecycle with our own alterations. In the early stages the first few sprints were to produce the use cases, sequence diagrams, GUI mockups and state diagram. The later sprints were centered around completing the code. In these sprints, related features were tackled in each sprint to produce usable pieces of the software. This software development life cycle provided a unique take on project management and revealed key features that will be discussed later.

d. Comparison

Looking at the different development methodologies presented here, it can be seen that one of the factors in development is picking a strategy that will best suit the developer and client needs. The Waterfall method is best used in small project situations where the requirements are well known up front, and will not change in order to ensure the client is satisfied with the result in a timely manner. The Spiral method is best used in larger projects with complex requirements that can easily be altered by either the client or developer. Lastly, the Agile method is best in large project environments with high client collaboration and with a team that can easily respond to changing requirements. Going into this project, it was known that the project would be of large scale which ruled out an effective use of the Waterfall method. Agile was decided on based off of client accessibility, the existence of vague initial requirements that would need alterations later

on in development, the given time frame for project completion, and the formation of a well-practiced team able to adapt to the impending changes.

IV. Project Management

a. Roles of a Project Manager

Project managers are the people that oversee a project from inception to completion. They manage all parts of the project and are in charge of their planning and execution. Specifically, they are in charge of planning the project scope, assembling and leading a team, time management, progress monitoring, and they are the main point of contact with the client. Planning project scope involved gathering the information required to start the project, and doing the initial requirements capture. Assembling the team was centralized around finding reliable, knowledgeable, and hard-working peers. Overseeing time management was tied into progress monitoring, as loose deadlines were created to ensure sprints were completed, or close to completion on a weekly to biweekly basis. This allowed for the project as a whole to be completed on time. The role of project manager allowed for a new insight on software development. On past software development projects, being a member of the team only allowed for limited access to project details. As project manager, all details go through you, and you are in charge of keeping open lines of communication within the team, and with the client to ensure client satisfaction and proper project completion.

b. Agile Project Management

Using the Agile method requires that the project manager make sure that the core principles of Agile are followed. For software development to be Agile, contact with the

client must remain constant, sprints should be designed and completed in a timely manner, and client feedback needs to be taken into account before completing each sprint. This meant that as project manager, the designated sprints had to be manageable and completable within a decent time frame, and that after sprint completion, the client was brought in to review the sprint and provide feedback that would steer the sprint, and future sprints, in their desired direction. Meeting with the client was key to project completion as it led to the discovery of components missed in the initial requirement capture that were vital to the overall design of the database and project features.

V. Design Process

a. Initial Requirement Capture

For this project, the initial requirement capture started with an in-person meeting. At this meeting, the client was questioned on the current operations of the equipment room. This meeting followed by a trip to the equipment room to observe these operations in action. The notes taken at this initial meeting were then taken to the first development team meeting where a list of features that the software would need to complete was created. This initial list was then taken back to the client in order to ensure that all needed features were accounted for. It was at this point that the client made it known that they did not want the checking out of equipment to be called a rental, but instead to be called a reservation.

b. Use Cases

A use case is a list of actions that describe how users interact with a system to complete a task. The use cases for this project were determined from the list of features created after the initial requirement capture. A total of nineteen different use cases were created for the project in order to properly define the required software features. On the following pages are an example of two use cases from the project. It was during the use case sprint that another key design feature was uncovered. If a person as already registered in the system, as either a worker or borrower, then their information would be in the system already. Therefore, if a worker became and borrower, or vice versa, then this registration process could be expedited. After checking with the client to ensure that workers could also be borrowers, the registration process was altered into two separate use cases to take care of either possibility.

Use Case Name:	5. Register a Borrower Without Existing Banner ID

Description:

A new Borrower is registered without an existing Banner Id

Preconditions:

Workflow:

1. Receptionist requests Worker credentials
2. Worker provides Receptionist with credentials
3. Receptionist retrieves the Worker's Worker record from the Worker folder using the information provided in step 2.
4. Worker informs Receptionist they would like to register a borrower.
5. Receptionist requests the Worker for the borrower's BannerID, first name, last name, email address, and phone number.
6. Worker provides the Receptionist with the information requested in step 5.
7. Receptionist validates the data provided in step 6.
8. Receptionist fails to find a Person record with the BannerID provided in step 6.
9. Receptionist creates a new Person record with the Banner ID, first name, last name, email address and phone number provided in step 6.
10. Receptionist files the Person record into the Person folder.
11. Receptionist creates a new Borrower record with the Banner ID provided in step 6, sets the status to 'Active' and sets the 'date added' and 'date last updated' fields to the current date.
12. Receptionist files the Borrower record into the Borrower folder.
13. Receptionist informs the Worker that the Borrower was successfully registered.

Results:

New Borrower is added to the system

Alternates:

A Person record is found

Entities Involved:

Person, Receptionist, "Person" Folder

Use Case Name:	6. Register a Borrower with Existing Banner ID

Description:

A new Borrower is registered with an existing Banner Id

Preconditions:

Workflow:

1. Receptionist requests Worker credentials
2. Worker provides Receptionist with credentials
3. Receptionist retrieves the Worker's Worker record from the Worker folder using the information provided in step 2.
4. Worker informs Receptionist they would like to register a Borrower.
5. Receptionist requests the Worker for the borrower's BannerID, first name, last name, email address, and phone number.
6. Worker provides the Receptionist with the information requested in step 5.
7. Receptionist validates the data provided in step 6.
8. Receptionist finds a Person record with the BannerID provided in step 6.
9. Receptionist creates a new Borrower record with the Banner ID provided in step 6, sets the status to 'Active' and sets the 'date added' and 'date last updated' fields to the current date.
10. Receptionist files the Borrower record into the Borrower folder.
11. Receptionist informs the Worker that the Borrower was successfully registered.

Results:

New Borrower is added to the system

Alternates:

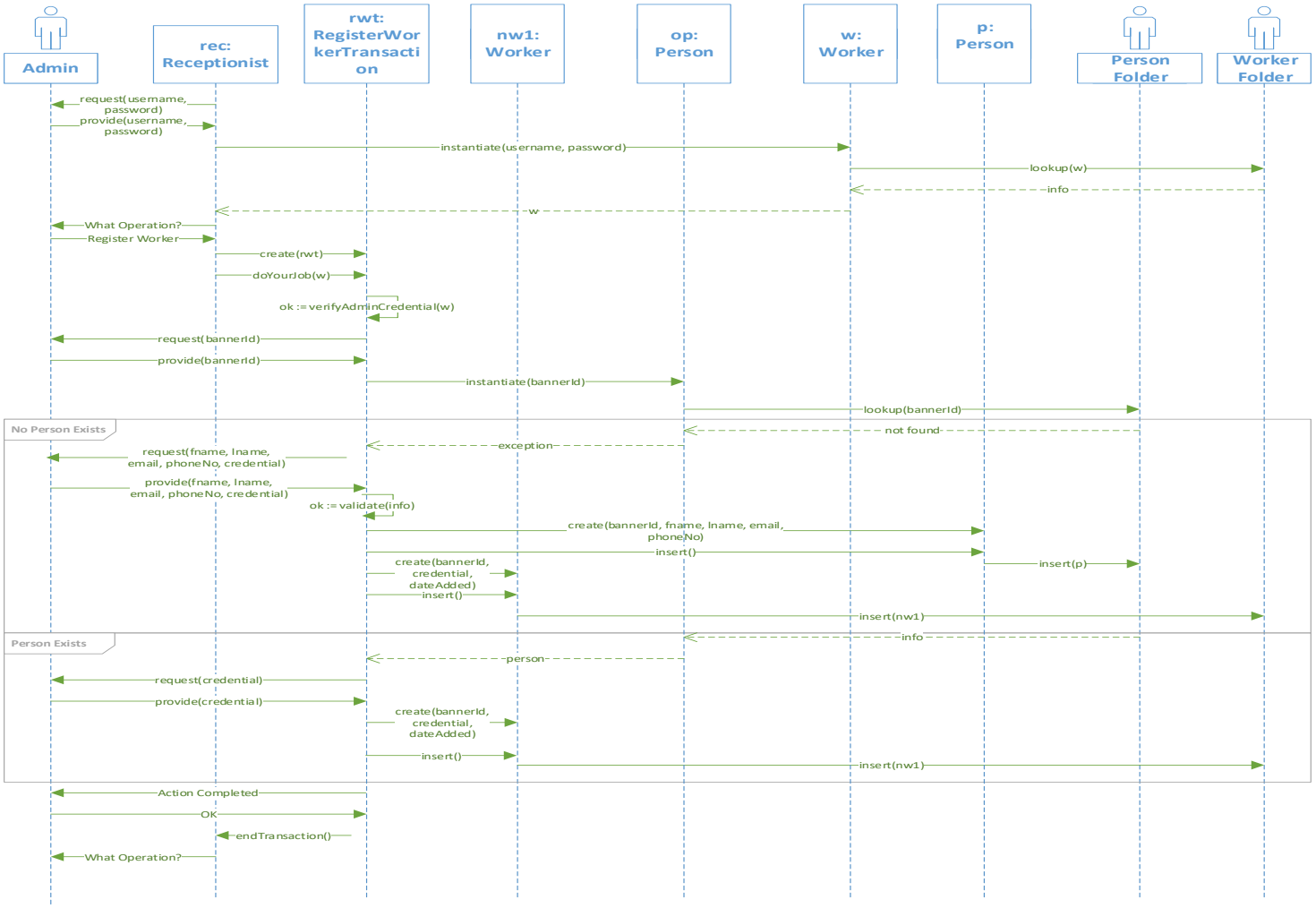
No Person records match the information provided.

Entities Involved:

Person, Receptionist, "Person" Folder

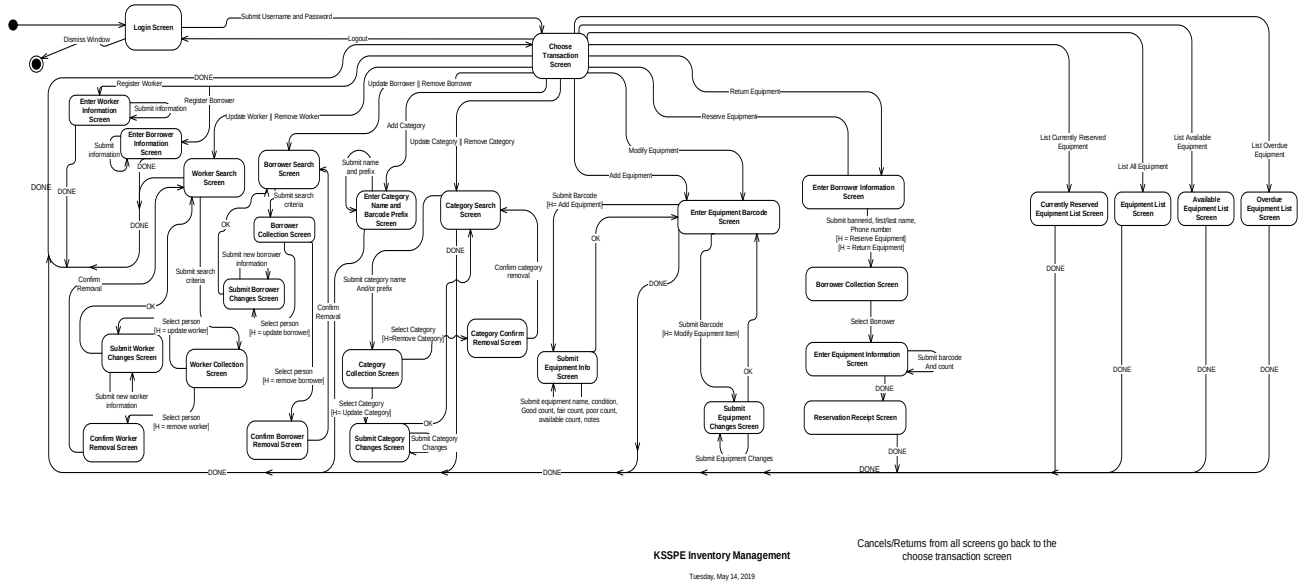
c. Sequence Diagrams

Sequence diagrams depict interactions between the user and the system, and how the classes within the system interact with one another and the database to complete a given feature. This visual flow of information in a timeline fashion is a great way for the developer to fully understand how the classes will need to share information in order to complete the task at hand. The below sequence diagram depicts the process for registering a worker into the system. The admin on the left-hand side is the user and the boxes to the right are the classes, objects, and database tables required to complete the action, and the lines show the transfer of information.



d. GUI Mockups & State Diagram

The state diagram is a large diagram that shows the different screens the user will need to interact with in order to use the system for any given operation. These screens are called states and each state becomes a different view in the Guided User Interface, GUI. This diagram also acted as the basis for the created GUI mockups.



The GUI mockups are a very basic illustration of what the different views for the system will look like. These mockups allow for the client to get a basic idea of system functionality before code development begins. This crucial meeting walks the client through the system and allows them to make suggestions on the GUI layout and how it affects system functionality. After this meeting, the client informed us they wanted the reservation feature to be the most prominent on the main page as it would be the most used by the workers.

VI. Results/Discussion

The project as a whole was fairly successful. Undergoing the software development process as a project manager was a surreal experience. It was accomplishing to seek out a client, set up meetings to understand their needs, and coordinate the development of a product to meet these needs. Forming the team was simple enough because of previous coursework that put me in a team environment. The team met every week at the same time. This meeting was crucial to ensuring work was consistently being completed and we remained on track to complete the project on time. Outside of these meetings, the team kept open communication through emails, group chats, and use of the GitHub version controller.

Following the Agile method was easy enough because the client was always available via email if not in person, so any questions the team had, I could easily address with the client, and return to the team with prompt answers. This methodology was ideal for the project as our team had an open line of communication, and as evident by the design phases there were a decent amount of alterations to the original requirements that would have been difficult discover, confirm, and make work without an Agile environment.

It may take some time for the deployed software to be eased into use, as the current operations have been used for some time, and it may be a difficult habit to break. If the system is being used by the end of next year, I would consider this a success. As this would mean it met expectations and requirements, is being used, and a resistance to change did not result in software failure.

VII. References

Tutorialspoint.com. (n.d.). SDLC Tutorial. Retrieved from https://www.tutorialspoint.com/sdlc/index.htm

What is Agile Project Management? - Definition from WhatIs.com. (n.d.). Retrieved from https://searchcio.techtarget.com/definition/Agile-project-management

VIII. Acknowledgements

I would like to thank my team of fellow undergraduate computer science students who were a driving force for this project.

Liam Allport – College at Brockport, Class of 2019
*Matthew Fritschi – College at Brockport, Class of 2019
Lucas Wing – College at Brockport, Class of 2019

I would also like to thank our faculty advisor Dr. Sandeep Mitra who helped me immensely in the project manager role and performed a plethora of code reviews to ensure we produced quality extensible and maintainable code.

*A fellow honors college computer science student, documented parts of this project in his own thesis focusing on the code development and use of the Observer framework.