The Application of Software Engineering to Moving Goods Mobile App

Katherine Snyder and Kevin Daimi
Department of Mathematics, Computer Science and Software Engineering
University of Detroit Mercy,
4001 McNichols Road, Detroit, MI 48221
{snyderke, daimikj}@udmercy.edu

Abstract—GoodTurn - the moving goods mobile system, is designed and implemented by the University of Detroit Mercy with a grant from Ford Motor Company. The app is intended to facilitate and manage Ford employees' donation of their time and vehicles to serve the community by moving goods and resources. This paper introduces the requirements, analysis, and design of the GoodTurn system for iPhone environment. The software tools needed for its development will be highlighted.

Index Terms—GoodTurn, Requirements, Specification, Design, System Models, Architecture, Interface

I. INTRODUCTION

The goal of the Goods Moving Mobile System, GoodTurn, is to allow non-profit requesters to select drivers who volunteer their time and vehicles to move goods and materials from donors to a location specified by requesting organizations. Requesters are either Non-Profit Organization (NPO) or Non-Government Organizations (NGO). A team of software engineers from the University of Detroit Mercy designed and implemented this system following software engineering processes. Currently, GoodTurn runs on iPhone only. The next phase of development will allow for the app to be used on Android-based phones. The Ford Motor Company offered a grant to build this system after three Ford employees proposed the idea for the app. Xcode [1] was used to develop the GoodTurn application using the Swift programming language [2]. Furthermore, Firebase 3.0 was utilized for the underlying database and to capture analytics of app use [3].

GoodTurn was developed using a thorough requirement engineering process and was implemented using an agile software development approach. Unlike the classical software development techniques, agile methods involve extensive collaboration and face-to-face communication with the customer. Agile requirements engineering defines the way of planning, executing and reasoning about requirements engineering activities without having to wait for the requirements to be completed before analysis and design starts [4]. Once a subset of requirements is available, analysis, design, and programming begin. It is broadly established in software engineering (SE) research that requirements engineering (RE) is one of the most decisive sub-process of software development. There is a wide unanimity that understanding software requirements is crucial for designing the right software system [5], making software requirements extremely important. Recent studies revealed that 56% of system defects are the result of poor requirements and requirements errors cost 10 times more than coding errors [6]. There are many methodologies to develop software requirements through interactions with the clients. Substantial software defects can be detected during the requirement analysis phase. This is considered a sensitive task in which mistakes or incorrect perceptions may result in a major catastrophe for the software product [7]. Agile Software Development (ASD) is frequently adopted to handle the growing complexity in system development. Hybrid development models with the integration of User-Centered Design (UCD) can be utilized with the aim of delivering competitive products with a suitable user experience [8].

In the development of the GoodTurn app, software modeling notations were also used to facilitate the analysis of some of the requirements. UML notation, use-cases, and data-flow diagrams were employed where useful. Model notations provide efficient graphical views for sharing knowledge between the professionals responsible for documenting information and those who need to understand it and put it into practice [9]. Model notations are also useful for communicating hardware issues. Vogel-Heuser, Braun, Kormann, and Friedrich [10] indicated that object-oriented model based design can be constructively utilized in industry and that the code automatically derived from the UML model can be implemented on industrial Programmable Logic Controllers (PLCs) without supplementary work. Modeling notations also facilitate converting an analysis model to a software design seamlessly and efficiently implementing it into a programming language [11].

Software architecture is normally the first design artifact that focuses on quality issues, such as performance, reliability, and security. In addition, it is a
reference point for other development activities including coding and maintenance [12]. Conventionally, software architecture is considered the result of the software architecture design process usually symbolized by a set of components and connectors. Currently, the set of design decisions crafted by the software architect complements the solution-oriented definition of software architecture [13]. Visualization of the software architecture supports software engineers in reasoning about the functionality and features of a software system without the need to get involved in coding and implementation details [14]. Alenezi [15] stressed that software systems are becoming more complex, larger, more integrated, and are implemented using various technologies, which need to be managed and organized to ensure quality product. Quality attributes of the product require an architecture to enable designers to assess and analyze them. The GoodTurn system relied on the three-tiered client-server architecture style that includes iPhone software, application server, and database server.

User interface (UI) is the locus of interaction between the user and computer software. The success and failure of a software application is influenced by User Interface Design (UID). The ease of using a software and the time needed to learn the software are impacted by UID [16]. In the past, application functionality dominated most of the attention, effort and development time. User interaction was looked upon as the least significant aspect in developing a software application [17]. When the services of a software system need to be described, two main elements must be considered: User Interface (UI) which is presented to the user, and the User Actions on this UI. The system reacts to a user action through on-screen feedback with the possibility of asking for further information and providing error or help messages [18]. Designing User Interfaces (UIs) is a creative and human-intensive activity, which prevents the adoption of computer-aided tools to explore alternative solutions [19]. The GoodTurn system followed the standards and polices of Apple’s current user interface conventions.

This paper presents the design and implementation of the GoodTurn software system. Section II presents the Functional Requirements together with the specification. Section III provides the Design Constraints. Nonfunctional Requirements are discussed in Section IV. The System Models and System Architecture are organized in Sections V and VI respectively. Section VII represents the User Interface. Finally, conclusions are presented in Section VIII.

II. FUNCTIONAL REQUIREMENTS

Functional requirements embody the foundation for any system construction. They dictate what the system should do. For the GoodTurn app, functional requirements were collected from Ford employees, non-profit organizations (NPOs), non-government organizations (NGOs), and the public. Samples of these requirements are depicted below. Note that the term “requesters” refer to NPOs and NGOs.

A. Login Screen

Requirement: The login screen must contain an option to save the user’s email.

Specification:
1. The login screen must advance the user to the requestor or driver screen if the user has an active login session.
2. The login screen must have an input for the user to input their username/email.
3. The login screen must have an input for the user to input their password.
4. The application should allow the user to register if they do not already have an account.
5. The application must authenticate the user’s email and password combination.
6. The application must contain an option for the user to initiate the recovery of password process.

B. Account Management

Requirement-1: The system must allow deactivation of a user’s account.

Specification-1:
1. Set account status field to ‘deactivate’ in firebase.
2. The user will receive a notification when logging in if their account is deactivated.
3. The requestor will not be able to create new jobs if their account is deactivated.
4. The driver will not be able to view any jobs if their account is deactivated.

Requirement-2:
The system must allow the requestor to reject a specific driver in the future.

Specification-2:
1. Add the driver to the blacklisted list for the requester’s account in firebase.
2. The application must allow for the requestor to remove a driver from the blacklist.
3. Remove the driver from the blacklisted list for the user’s account in firebase.

Requirements-3: The system must allow the driver to reject a specific requestor/organization in the future.
Specification-3:
1. Add the requestor to the blacklist for the user's account in firebase.
2. Allow the driver to remove a requestor from the blacklist.
3. Remove the requestor from the blacklist for the user's account in firebase.

Requirement-4: The system must be able to put a user under review.

Specification-4:
1. If a user requests reactivation, then put their account under review.
2. Set account status field to 'review' in firebase.

C. Registration Screen

Requirement: The system should allow the user to register if they do not already have an account.

Specification:
1. A user should be able to specify whether they are a driver, requestor, or both.
2. The registration screen must have an input text field for the user to enter their full name.
3. The registration screen must have an input text field for the user to enter their address.
4. The registration screen must have an input text field for the user to enter their company name.
5. The registration screen must have an input text field for the user to enter their phone number.
6. The registration screen must have an input text field for the user to enter their email address.
7. The registration screen must have an input text field for the user to input their new password.
8. The registration screen must have an input text field for the user to input their new password again to verify they entered the same password.
9. The registration screen component must verify that the password conforms to a set of text restrictions: at least 8 characters, one uppercase letter, one lowercase letter, and one special character.
10. The registration screen component must verify that the two password fields match.
11. The registration screen must have an input text field for drivers to input the vehicle make they will be using to deliver goods.
12. The registration screen must have an input text field for drivers to input the vehicle model they will be using to deliver goods.
13. The registration screen must have an input text field for drivers to input the vehicle year they will be using to deliver goods.
14. The registration screen must have an input text field for the drivers to input the license plate of the vehicle they will be using to deliver the goods.
15. The registration screen must have a button for users to register their entered data.
16. The registration component will verify the password and check if all the fields are correctly populated.
17. If successful, hash the password and create a new user on firebase with all the information populated by the user. If not successful, then inform the user what went wrong.

D. Job Completion

Requirement-1: The system must allow rating of users that were involved in a job.

Specification-1:
1. Users will be rated on a number system, 1 to 5, where 5 being the highest quality.
2. Drivers provide their rating of the requestor.
3. Requestors provide their rating of the driver.
4. Average ratings will be associated with the perspective driver or requestor.

Requirement-2: The system must allow users that were involved in a job to provide feedback.

Specification-2:
1. The feedback given by users should be a text input for the user.
2. The application must allow the driver to provide any feedback they have for the requestor.
3. The requestor screen must have an input for the requestor to provide any feedback they have for the driver.

E. Driver Screen/Dashboard

Requirement-1: The system must provide a list of available jobs.

Specification-1:
1. A job must contain the items to be transported.
2. A job must contain the address of the pick-up and drop-off.
3. A job must specify the miles from the pick-up location to the drop-off location.
4. A job must include the estimated time for completion.
5. The job list should be updated to reflect jobs that are accepted, rejected, or no longer available.
6. A job must include what type of vehicle is required.
7. A job must include the time of pick-up.

Requirement-2: The system should allow the driver to accept a job.

Specification-2:
1. The requestor must be notified that their job has been accepted.
2. A reminder should be issued for the driver regarding the accepted job.
3. The job should be set as no longer available for other drivers.

Requirement-3: The system must allow the driver to cancel an accepted job.

Specification-3:
1. If a job is cancelled by the driver, the requestor must be informed that their job was cancelled.
2. If a job is cancelled by the driver, the job must be reposted for other drivers.

F. New Job

Requirement: The system must allow the requester to start a new job.

Specification:
1. The new job screen must have an optional input field for the user to specify a new item to the job.
2. The new job screen must have an input field for the user to specify the number of items to be added to the new job.
3. The new job screen must allow the user to add to the list of items for the new job.
4. The new job screen must allow the user to remove an item from the list of items for a new job.
5. The new job screen must allow the user to move to the next screen to specify pickup and drop-off locations.
6. The new job screen must allow the user to specify the size of vehicle needed, such as pickup truck, van, or sedan.
7. The new job screen must allow the user to specify if any heavy items are in the job.
8. The new job screen must contain a search address field for the user to enter their pickup location for the job.
9. The new job screen must contain a search address field for the user to enter their drop-off location for the job.
10. The new job screen component must calculate the miles from the pickup location to the drop-off location.
11. The new job screen must allow the user to save the job. The requestor should then be brought back to the Requestor Dashboard.
12. The new job screen must have an option for cancelling the current job being created.

G. Requester Screen/Dashboard

Requirement-1: The system must allow canceling a job that is in the queue.

Specification-1:
1. The job must be removed from the available jobs.
2. The driver should be alerted that the job has been cancelled if job is accepted.

Requirement-2: The system must allow the requestor to modify a job.

Specification-2:
1. The requestor should not be allowed to modify a job within 24 hours of the accepted job's scheduled deliver date.
2. The driver must be notified if the requestor makes any changes to an accepted job.
3. Driver should re-accept the changes to the job.
4. If a driver rejects a new job, job gets reposted to available drivers.
5. The requestor is notified of the driver's decision to accept or reject the modified job.

H. Support/Feedback

Requirement: The system must allow users to provide feedback, submit problems, or seek help.

Specification:
1. The support screen should have a way to specify what type of problem they are inquiring about or whether they are providing feedback.
2. The support screen must have an option for users to inquire about a lost item.
3. The support screen must have an option for users to file a complaint.
4. The file-complaint must provide users with an option to file complaints against another user.
5. The support screen must have an option to provide feedback on the vehicle used to transport items.
6. A password should be allowed to reset.
7. The support screen must have an option to report a broken link.
8. The support screen must have an option to delete the user's own account.

9. The support screen must have an option to allow the user to request reactivation of their account.

I. Profile

Requirement: The system must have a profile screen for the users to review or change their information.

Specification:
1. The profile screen must contain the user's name that can be changed.
2. The profile screen must contain the user's address that can be changed.
3. The requester’s profile screen must contain the user's company name that can be changed.
4. The requester’s profile screen must contain an option for how users wish to be notified.
5. The driver's profile screen must have an option for adding a vehicle.
6. The driver's profile screen must have an option to edit the details of their vehicle.
7. The driver's profile screen must have an option for removing a vehicle.
8. The profile screen must contain a button for the user to save their changes.

III. DESIGN CONSTRAINTS

GoodTurn was developed under the following constraints:

- The system must be developed using the client-server methodology.
- The clients will be accessing the app using iPhones 5 or more recent models.
- Information about drivers and requesters should be stored using Firebase database.
- Swift should be the programming language used.
- The system must be developed under the Xcode development environment.

IV. NONFUNCTIONAL REQUIREMENTS

Nonfunctional requirements represent constraints on the functional requirements. They express some quality characteristics that the software system should possess. In this section, performance, usability, security, privacy, reliability, and maintainability features of the GoodTurn system will be illustrated.

A. Performance
- The system should allow drivers and requesters to sign in within 5 seconds.
- Drivers and requesters registration should not take more than 5 seconds.
- Deactivation and reactivation of accounts should take 5 seconds.
- Displaying blacklisted drivers for a specific requester should take no more than 5 seconds.
- The system should display the job history details within 5 seconds.
- Completed jobs for a requester should be listed within 3 seconds.
- The system should display the list of available jobs within 5 seconds.

B. Usability
- Drivers and requesters should be able to use the system without any training.
- System administrator should be provided with online training.
- The system should provide messages to guide the users when invalid information is entered.

C. Security
- Drivers, requesters, and system administrators should be authenticated.
- Messages exchanged between all parties (drivers, requesters, system administrators) should be confidential.
- No party can deny sending a message to another party.
- No part can deny receiving a message from another party.
- No authorized party should be denied service.
- An authorized party will be denied service.

D. Privacy
- The system should not disclose requester information to non-drivers.
- The system should not disclose a driver information to non-requesters.
- The system should not disclose driver’s information to another driver.
- The list of rejected drivers for a requester should not be available to other requesters.
- The list of rejected requesters for a driver should not be available to other drivers.
E. Reliability and Availability

- The system must detect, isolate, and report faults.
- The mean time between failures should be six months.
- The system should be up and running within an hour after a failure.
- The system should be backed up weekly.
- Backup copies must be stored at a different location specified by the NPO/NGO (Requester or mover).

F. Maintainability

- Errors should be easily corrected using effective documentation.
- Additional features should be added without considerable changes to the design.
- The system should be easily ported to iPad.
- The system should be easily ported to Android.

V. SYSTEM MODELS

System models help the requirement engineer to understand the functionality of the system. In addition, models help the analyst when communicating with stakeholders. Software engineering relies on the use of abstract models to portray and infer the properties of software systems. A system model symbolizes attributes of a system and its development environment. Normally, systems have many details. However, system models concentrate on few vital pieces of the system to be fully understood and facilitate reasoning.

A use-case tells a formalized story about how an end-user interacts with a system under a specific set of situations. The story may be narrative text, an outline of tasks or interactions, a template-based description, or a diagrammatic representation [20]. Regardless of its form, a use-case represents the system from the end-user’s point of view. Examples of use-cases for the GoodTurn system are given in Figures 1 and 2 below.

VI. SYSTEM ARCHITECTURE

A software system architecture represents the structure of the system comprised of the elements of the system and the relationships between them. It depicts the high-level structures of a software system. In addition to the relationships between the elements, the architecture emphasizes the characteristics of both the elements and relationships. Architecture is normally thought of as a blueprint for the software system, in other words, the big picture. The architecture is the main artifact for further designs of components, interfaces, data, and code.
By examining the architecture, one can conclude how multiple software elements collaborate to fulfill their tasks. Figure 3 illustrates the architectural style used by the system. GoodTurn follows the three-tiered client-server architectural style. In this style, the client presentation includes the iPhones with the GoodTurn application installed. The business logic is represented by the GoodTurn application server. Finally, the Firebase database server represents the information system.

Figure 3. GoodTurn Architectural Style

Figure 4. Top-Level Decomposition

Figure 5. Second/Third Levels Decomposition
The GoodTurn system architecture is functionally decomposed into various functional components. Decomposition assist designers to recognize and identify the key problems that the system needs to tackle. Figure 4 demonstrates the top-level and first-level decomposition. The component, MVS Startup, is further decomposed into second and third levels in Figure 5.

VII. USER INTERFACE

A user interface is the means by which users interact with, manipulate, control, and use the system. Current software systems have a graphical user interface (GUI). A graphical user interface relies on several components, such as menus, toolbars, windows, and buttons.

The first step in interface design is to identify the humans who will interact with the system. For the GoodTurn system, the humans included drivers, requesters, and system administrators. Then, scenarios for each way the user can interact with the system were developed. The classes needed to implement these scenarios were designed and integrated with other classes of the system. The user interface of GoodTurn User Interface is illustrated below. Figures 6 and 7 provide samples of the User Interface.

A. Splash Interface

This interface will be the first screen the user encounters when starting the application. It will only show for a brief time (2 seconds) while the application is loading the first screen.

B. Login Interface

The Login screen will be seen immediately after the splash screen. If the user is already logged in, then the appropriate dashboard screen will be loaded next. This screen contains fields for the user to enter their email/password combination. It also contains a register/create new account button so that the user can navigate to the Registration screen.

C. Registration Interface

When a new user/account is being created, this interface will be displayed. The first screen will be seen regardless of user type. The second screen will only be available if a user registers as a driver. After the user fills out all the information they can confirm by pressing a button. Then, terms of service will be displayed to the user, which they must agree to before the account is created. Once the account is created, the appropriate dashboard will be loaded for the user.

For both drivers and requesters (movers), all general details of the account will be encompassed on this screen including name, address, company, phone number, email, password, and confirm password. Drivers will be able to enter Vehicle Make, Vehicle Model, Vehicle Year, Vehicle Color, Vehicle License Plate.

D. Requester Dashboard

A requester, who has successfully logged into their account, will arrive at this screen next. This screen will display all jobs the requester has scheduled with options to edit jobs, cancel jobs, or create new jobs. Jobs should also have a status to let requesters know if the job has been accepted, on route, completed, or still pending (searching for driver). There are options to navigate to these screens: Profile, Support/Feedback, Driver Dashboard (if registered as driver as well).

E. Driver Dashboard

A driver who has successfully logged into their account will arrive at this screen. This screen will display a job list that will be populated with any jobs made by requesters. The driver will be able to sort jobs by distance to pick-up, distance between pick-up and drop-off, pick-up time, or vehicle type. The driver can accept a job and the requester will be notified that their job was accepted. The driver will also be able to reject a job. This will just dismiss the job from their list. There are options to navigate to the screens: Profile, Support/Feedback, Requestor Dashboard (if registered as a requestor as well).

F. Profile Interface

This screen is available to all users of the application. The screen gives information about the user and the ability to edit this information. The Profile screen for any user includes: name, address, company, and notification settings. If the user is a driver, it will also provide vehicle information with the ability to add, edit, or remove a vehicle from their profile. An option to save changes is also displayed.

G. New Job Interface

When requesters want to create a new job for drivers, this screen becomes available. Requestors can enter information about the job they wish to create, such as what they expect to be transported, how many items need to be transported, the size of the vehicle they think they will require, whether there will be heavy objects to be lifted, where the items will be picked up, and where the items will be dropped off. This screen has options for submitting the job details and creating a new job as well as cancelling the created job, and removing all job details.
H. Support/Feedback Interface

This screen should be available to all users of the application at any time. If an account is locked, this will be the screen used to request unlocking the account. This screen has options for stating lost items, filing a complaint, filing a complaint against a specific user, giving feedback on a driver, resetting password, reporting a broken link, deleting an account, reactivation of an account, and “other” to cover any other possible help required. These are represented as drop down menus to select a topic and then appropriate text fields based on the selected topic will be shown to report accordingly.

VIII. CONCLUSION

The application of software engineering to the creation of the GoodTurn system was described. Specifically, the sub-processes; requirements engineering, modeling, software architecture, and user interface were highlighted. It was concluded that the best development approach would be the agile process. The user interface was constrained by the standards and polices of Apple for iPhone applications. Since iPhones, application server, and a database server were needed, the GoodTurn system architecture followed the three-tiered client-server architecture. Future work will expand the scope to include Android-based phones.

REFERENCES


