INTERIM REPORT:

MOBILE JOB LIFECYCLE AND PARTS MANAGEMENT APPLICATION

Author: Daniel J Workman
Supervisor: Dr. Andrew Jones
Moderator: Dr. Xianfang Sun

Table of Contents

Introduction 3

Proposed Solution 3

Background 4

Barcode Format 5

Inventory Control System (ICS) 5

Approach 5

Feasibility Study 6

Scanning Prototype 1 6

Scanning Prototype 2 7

Web Services Prototype 8

Conclusions 8

Glossary and Abbreviations 11

References 12

Introduction

This project was initially proposed to me by a client I had done work for previously but I declined it at the time as I was in my first year of university and felt the project scope too great to take on in my spare time. However as I reached my final year the client still required a solution and I felt the project would make a suitable final year project.

The client, ATM Solutions (Europe) Limited, provides repairs and servicing to the ATM industry. This service is provided both in-house (faulty parts are sent to their office) and also in the field (technicians visit sites). ATM Solutions has an inventory control system (named ICS) that tracks all parts by barcode from reception to dispatch. Unfortunately the field service technicians are not provided with any system that allows received or returning parts to be scanned and reported back to the inventory control system. The lack of a system for use in the field by the technicians causes significant errors to be introduced. The current procedural controls used by the field technicians are inefficient and prone to human error.

In addition to tracking parts to and from the technicians there is also a requirement to support job tasking and reporting.

In support of job tasking, the following information must be communicated to the field technicians throughout their working day:

- Job number and priority.
- Details for each job including company name, address and contact details.
- The type of each job whether it be an installation, repair, service or uplift.

The engineers must then communicate the following job information back to the main office:

- Job acceptance.
- Parts they have used during a visit.
- The status of each job, whether they have completed it or aborted it (and why).

Their current method of achieving this is via the use of email and phone calls with an operator at the head office.

As previously stated, this current method is inefficient for the following reasons:

- Numerous phone calls are both time consuming and costly.
- Relaying data such as part numbers during a voice call is prone to human error.

Proposed Solution

The proposed solution is to provide each engineer with an application which will run on their existing mobile smart phone as provided by the company. All engineers are provided with an Apple iPhone to use as their business phone. The client would therefore like a custom built iPhone application to address the issues they have with their current procedures. The application should be built such that it focuses on the full lifecycle of a job.

The client requires the following as a minimum:

- Provide a method of downloading a list of jobs for a specific engineer on that day.
- Allow the user to select a job and view its details.
- Allow the user to accept a job and to set a job as active (all other jobs for the day will then become inaccessible).
- Allow the user to add the parts they have used to the job by scanning barcodes, reversion to manually entering part numbers is required when a barcode is damaged and cannot be read.
- Allow the user to complete or abort a job (a valid reason is needed if a job is aborted).

The client has also requested the following features they would like but are not a requirement of the final deliverable:

- Provide extra information with each job such as:
    - Parking details
    - A 'street view' map showing the location of the site (most likely making use of the built in mapping features provided with the operating system)
        - This could also make use of GPS data as provided by the device.
    - An area where specific documents for the site can be displayed, these would most likely be in a PDF format.
- View a list of all the completed jobs for that day so that the engineer can go back at a later time to confirm any details they may have forgotten.

By using an interactive application the engineers will cut most if not all manual interaction with those at the head office. This saves time and money as not only do engineers spend less of their time talking to people but fewer people are required to operate phones at the office.

After meeting with the client they have provided me with a system flow diagram which I have included in Appendix A. This accurately shows the job centric nature of the application and includes both the required and extra features that have been discussed.

Background

Scanning barcodes using an iPhone is not a new concept. Not only are there libraries which provide this functionality via the devices camera but physical docking devices are manufactured which include a traditional laser scanner that interfaces with software on the iPhone.

These scanners are not only costly (starting costs of approx. £200 per unit) (1) but make the device much bulkier to use and store. Considering that the client expects the device to be a mobile phone first and a scanning tool second it is an unacceptable compromise for each engineer to be carrying such a bulky device. Couple this with the high hardware costs and this becomes an option which the client feels is just not viable. As a result the client wishes only to use the devices built-in camera to scan the barcodes.

Barcode Format

The client uses a standardised barcode format called **Code 128** (2). The format allows for the encoding of all 128 standard ASCII characters and also provides error checking via the use of a check character.

Barcodes are printed onto stickers by the inventory control system using a label printer so that they may be easily attached to parts. They use two differing sizes of barcode for different sized parts. Larger parts or parts that may come in a bag will have a larger sized sticker and barcode. Smaller parts use a smaller sticker and barcode which can be placed on items that are too small to use the larger size of label. Examples of each are included in Appendix B. The code is stored using the following format:

The first two characters of the barcode are always "**!A**" which identifies this as a part number. This is then followed by the 8 digit part number itself. The following is an example of an actual part number used by the client:

```
!A00002234
```

These part numbers correspond directly to those stored within the clients' database. The application will need to store a list of these codes so that they can be sent back to the database to update any necessary stock levels or trigger any reorders.

Inventory Control System (ICS)

The inventory control system currently in use by the client is a desktop C# application which provides complete tracking for all parts at all stages within their business model. To provide access to the data a solution must be reached whereby an interface between the database and the mobile application can be established.

The system currently uses a Microsoft SQL Server database to store its data. After discussions with the client however there should be no need for any direct interactions with the database. The client already has a published interface specification which can be followed to allow for reliable communication with the database. This exists because they already export parts data to other systems which require this knowledge.

The interface specification calls for all data to be passed using XML and has well defined and logical structuring. Using a method such as a web service is should be relatively straightforward to format valid XML from within the application whilst being sure to follow the specification correctly. I have not included a copy of the interface specification at this stage but it will certainly be covered in the final report.

Approach

Before the project had even been accepted as my own final project for my degree I liaised with the client and discussed what it would be that they expected as a deliverable solution. They conveyed

what would be the minimum they would expect for a working application and also other features they would like to see if technologically possible and were feasible within the given timescale considering that I would be undertaking this work alongside the rest of my degree. It was very important that I conveyed the fact that I would not be working on the project fulltime as they would normally expect from one of their own software developers but would be fitting the development schedule in with the rest of my degree programme.

Feasibility Study

Shortly after my discussion with the client a small feasibility study was performed to ascertain whether the project was technologically possible given the constraints they had provided.

Considering the client had stipulated that the solution must run on Apple iPhone devices background research was performed into what existing libraries or services were available for scanning barcodes which made use of the devices built in camera.

Searching the internet the following solutions were found:

- **RedLaser** (3)
  - A commercial solution which has an impressive list of clients who use their product including *Nasa*, *Groupon* and *Buy.com*.
  - For 'Enterprise' users (applications which are internal to the company and not distributed via the App Store) there is an up-front fee of $500 and a charge of $5 for each device running an instance of the application.
- **Shopsavvy SDK** (4)
  - A similar commercial solution which required a large up-front fee to allow use of their SDK ($7,200 per annum) and limits the number of scans available.
  - A downside of this solution is the need for an active internet connection as the image processing is performed on their servers and not on the device itself.
- **ZBar** (5)
  - An open-source solution.
  - Completely free to use and deploy.
  - The reliability of scanning would need to be ascertained as it appears to be a small scale solution with no known users unlike the other two solutions.

Scanning Prototype 1

After submitting the feasibility study to the client they provided me with some useful feedback as how best to proceed. If a solution could provide them with reliable scanning they were happy to pay for the required licensing. The pricing scheme set out by RedLaser for Enterprise users was deemed to be an acceptable price if it provided the reliability they advertised so this solution was chosen to begin creating an early prototype.

The RedLaser SDK provided a very simple example project which could be modified slightly to provide the extra functionality required. This allowed a working application to be up and running quickly. Although the application ran easily within XCode's iPhone simulator there were problems when trying to execute the application on a physical device.

The SDK was extremely stringent about checking for valid licence certificates and would not run until it was provided with a valid one. Development licenses were downloadable from the Developers section of the RedLaser website and were generated using the devices unique ID. Even after this had been done the SDK would still often report seemingly irrelevant or cryptic error messages as to why it could not be used. After spending hours of tweaking and searching through forums it conspired that a new version of the SDK had been uploaded during the time the application had been developed and the generated license file was of an incorrect version to the SDK currently in use. A more sensible error message would have saved much time and as a result some faith in the SDK was lost. Considering the likelihood that similar problems could be encountered in the future the solution no longer appeared to be the best one available.

For the short periods of time that the prototype was working the results of the scanning were very good. Setting the device to scan only Code 128 barcodes and then providing it with some example codes the application was providing fast and reliable scans (never any errors in the returned code) in a range of light conditions and also at differing lengths from the camera lens. The application allowed the user to operate the devices built-in flash as a torch so that barcodes could be detected even when there were no other light sources available.

As a result of the unreliable nature of the SDK when trying to validate licences it was decided that the prototype would be placed on hold and to try an alternative scanning solution to see if the same level of scanning accuracy could be achieved but without the problems associated with trying to keep a set of up to date and valid licence certificates. The fact that it was enough trouble providing a working license for a single device one could only assume that the problem would most likely scale when trying to deploy the application to numerous devices.

Example screenshots of the prototype have been included in Appendix C (Figures 1 & 2) which show the main screen of the application and an example error message which illustrates a common problem which was faced.

Scanning Prototype 2

Considering the ZBar SDK was an open-source project it seemed logical to investigate this solution next. If this could provide a similar level of reliable scanning then it would be the obvious choice to replace the more commercial yet troublesome RedLaser SDK.

A basic iOS application was created and the ZBar SDK added to the project. This process was completed with relative ease and the next stage was to go about creating a basic user interface which would allow the scanning of barcodes and then adding the returned code to an onscreen list. Using the online documentation for the SDK it was possible to disable, in code, all barcodes other than Code 128 from being detected.

The only problems encountered during the development of the prototype were generally related to a lack of knowledge of Objective-C (the language used to write iOS applications) rather than issues relating to the SDK. As there were no license issues to worry about the SDK ran first time and never reported any errors.

As soon as possible the reliability of scanning was tested using some example Code 128 barcodes. The results were also very good and there were no discernable difference between the acquisition times of the RedLaser prototype and this one. The application never reported any errors or any failed scans and as with the other prototype the devices built in flash was utilised to illuminate the barcodes when in low light conditions.

Considering the success of this prototype the RedLaser prototype was officially set aside and the ZBar SDK was selected for use in the final product.

Figures 3-6 in Appendix C show an example usage of the application. Included in Appendix D is the main scanning code from the prototype project, although the project contains more code than this it is not relevant to the actual scanning SDK and has been omitted for the sake of brevity.

Web Services Prototype

After the completion of the scanning prototype the focus moved to researching what was necessary to get an up and running web service. From discussions with the client it was ascertained that all of their in-house software was written in C#. Due to the fact that the service would end up running on their own servers and potentially need to be updated in the future by their own developer's C# seemed the logical language to use to implement the service. Although familiar with both C++ and Java I had not used C# in the past which presented yet another learning curve as I would need to get some experience in the language before I could consider starting the web service implementation.

The client had however already discovered a potential need for a web service for an unrelated project in the past and had created a lightweight web service prototype at that time. They were happy for this to be used as a starting point for this prototype as it effectively illustrated how to use C# and the methods and tools needed to communicate between the web and their database.

Although my own knowledge of C# was improving I found that without a test database to connect to there would be no way of checking that any of the code would work. Due to a combination of time delays from the scanning prototype and a lack of C# knowledge a working prototype was not delivered to the client within the planned timescale.

As a result this must now become the primary concern and work effort when returning to the development in January.

Conclusions

The initial plan was to deliver two working prototypes to the client prior to the writing of this report. The first prototype, the scanning app, was completed adequately and feedback from the client was very good. They were happy with how the application acquired codes and also with the speed and accuracy in which it managed to do this. There were even instances whereby the application was able to acquire a positive scan on poorly printed barcodes which their own physical laser scanners were unable to provide a positive identification.

As a result the client agreed that the rest of the project should continue as planned and that the ZBar scanning library be used in the final version of the application.

The web services prototype was unfortunately less successful. The lack of a dummy database for testing purposes coupled with a lack of intimate C# knowledge meant that a working solution could not be delivered within the agreed timescale. Although work has been done toward this goal it is not yet at a standard whereby it can be used to demonstrate a working link between a Microsoft SQL Server database and a web interface.

As a result I have planned the following so as to resolve this issue as soon as work continues in January:

- Create a basic SQL Server database with some dummy data and find a suitable server to act as a host.
- Continue familiarising myself with C# and the tools it provides for implementing a web service.
- Use what I can learn from the client prototype to help with my own understanding and to therefore speed up the process.

Once both the client and I are happy that using a web service is a suitable solution then implementation of the main application will begin.

The task will be approached as such:

- The system flow diagrams provide an excellent visualisation of how the application should look and feel during execution. These will be used to storyboard a suitable GUI for the application. XCode provides excellent storyboarding tools which allow the user to place UI elements on views and in turn link views to other views.
- Once happy with a suitable user interface implementation of the job lifecycle framework will begin. This will allow jobs to be selected from a list, accepted and completed (or aborted). At this stage they will not be sourced from live database data, a list of dummy jobs will be created to test the framework.
- Once a basic framework is complete the addition of job specific details such as names, addresses and contact details will take place. The extra features the client has requested will not be added at this time and will only be added once all the requirements have been fulfilled and there is time left to implement them.
- At this stage the scanning feature of the application will be implemented. The parts which have been selected for use will be saved back to the data stored with that specific job.
- The final step will be to replace the dummy data used for testing with live data from the external database using the web service. It must be seen that not only can data be downloaded but uploaded to the database also.
- Once all of these steps are completed and if any time remains then other requested features will be looked at in the following priority order:
    - The ability to view any completed jobs from that day.
    - The addition of extra data for each job such as:
        - Parking data.
        - A ‘street view’ map of the site.

- A general purpose area for PDF documents relating to the job.

In light of the delay caused by the web services prototype the project timetable has been updated. The updated version is available in Appendix E.

In summary I believe that the timescales available and the work completed so far still provide enough time and resource to deliver a solution by the final deadline. Having chosen to create an initial prototype using the ZBar SDK instead of the RedLaser SDK would most likely have saved enough time to have completed both prototypes within the original timescales proposed but this was only a minor setback. Even the time spent developing the original prototype provided a good method of learning about the development environment and its tools.

If the project is going to successfully meet the stated deliverables then the work must be undertaken as closely to the timeline as possible. It is also very important to gain as much valuable feedback from the client as possible to ensure that the product is following their vision and also so that any as yet unknown modifications or features can be incorporated. If the implementation is left too late then although it may be considered complete it may not be fit for purpose once delivered to the client.

Glossary and Abbreviations

ATM (Automated Teller Machine): A computerized device providing automated financial transaction services to customers without the need to interact with a human cashier.

IDE (Integrated Development Environment)**:** An application which provides the user with all the necessary tools for developing software for a specific environment.

SDK (Software Development Kit)**:** A suite of software development tools which allow the user to develop software for a particular platform or operating system.

iOS (previously iPhone OS)**:** The operating system deployed on Apple iPhone and Apple iPad devices.

XCode: An IDE designed specifically for the development of iOS and OS X applications.

App Store: A service operated by Apple to distribute applications for iOS devices.

References

1. **Clevenger, Nathan.** iPhoneLife Magazine Barcode Scanning Hardware. *iPhoneLife Magazine.* [Online] 1 1 2012. [Cited: 13 12 2012.] http://www.iphonelife.com/issues/2012January-February/BarcodeScanning.

2. Code 128. *Code 128 Wikipedia entry.* [Online] [Cited: 8 9 2012.] http://en.wikipedia.org/wiki/Code_128.

3. **RedLaser.** RedLaser Developers. *Redlaser.com.* [Online] [Cited: 1 12 2010.] http://redlaser.com/developers/.

4. **ShopSavvy.** ShopSavvy SDK. *ShopSavvy.mobi.* [Online] [Cited: 1 12 2010.] http://shopsavvy.mobi/developers/.

5. **ZBar.** ZBar Bar Code Reader. *ZBar Project Sourceforge page.* [Online] [Cited: 1 12 2010.] http://zbar.sourceforge.net/.