University of Bamberg

Internship Report
Mobile Software Frameworks and Object Oriented Programming at Favendo GmbH

April 2018 - July 2018

by
Chandan Sarkar
MSc. International Software Systems Science
University of Bamberg

Favendo GmbH
An Der Spinnerei 15
Bamberg, Germany 96047

Supervisor and Organization Contact:
Christian Motz
Director Legal-HR
christian.motz@favendo.com

Bamberg, August 8, 2018

Contents

List of Figures

List of Tables

1 Acknowledgment

I would like to thank Mr. Christian Motz, the Director Legal/HR at Favendo GmbH, for giving me the opportunity to do an internship within the organization.

2 Introduction

I have participated in an internship program offered by Favendo GmbH [1] in Bamberg for four months starting from April 2018 and concluding on July 2018. Favendo is a software company established on 2014 with development sites mainly in Bamberg and Jena in Germany, provides cutting edge Location Based Services to its internationally spread customer base. Favendo is also part of Fewclicks corporate group [3] which closely deals with Web Application Services and Internet of Things based solutions. Favendo has provided location based services and solutions to customers from various demographics. Some of the prominent examples are Audi AG, SAP and Mediterranean Shipping Company cruise ships. This internship is part of my ongoing masters program at University of Bamberg in Germany.

Core services offered by Favendo can be divided into several categories as referred from the Favendo Official references [1]:

- **WayFinding**

  Way finding and indoor navigation is one of the key services that is offered with the help of hardware components such as beacons and the Commander software framework, in conjunction with a highly available backend server component. This service helps the users in finding their navigation path from a given source to destination at the indoor venues such as airport, shopping malls or large cruise ships with hand-held smart devices. A high degree of accuracy helps in precisely locating a point of interest in large complex venues. Normally, a combination of several technologies such as low energy consuming crypto-beacons along with core development framework of popular software ecosystems are used to bring the solution to the customer.

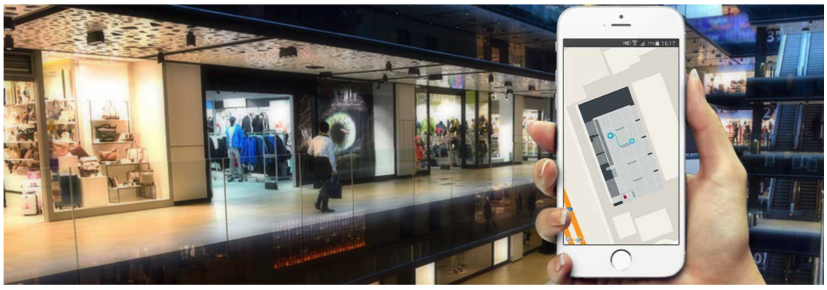

Figure 1: Way Finding sample application from Favendo Official [1]

WayFinding carries value both for external visitors as well as the authority using the Favendo provided solution. On one hand visitors can find their way to desired destination in a large building or complex. On the other hand, respective authorities can use visitors data through analytics and use the same for proximity marketing and many other thoughtful purposes in order to boost the business.

- **Asset Tracking**

  Asset Tracking was always been a challenge worth considering for large industries especially in sectors like construction, hospital management or large recreational venues like cruise ships. It is crucial to search items of interest in the hour of need and ensure security for them. Asset Tracking system developed by Favendo can be helpful in tracking the positions/potential utilizations of equipments or in certain cases even human beings. It is far more than just keeping track of matters. It could also be helpful in routing and smart arrival/availability predictions.

Figure 2: Asset Tracking concept from Favendo Official [1]

Favendo offers hardware/asset tags based on crypto-beacons and RFID

technology as well as software framework that goes hand in hand in order to create an Internet of Things based real time implementation of Asset Tracking. Use of bluloc [4] crypto-beacons makes a durable installation of signal radiators which are intercepted by beacon controllers and analyzed by the software systems to generate useful knowledge.

- **Proximity Marketing**

  Proximity Marketing has introduced a new paradigm of customer engagement for the businesses in order to improve sales. The key idea of the proximity marketing often relies upon the accurate positioning systems in place and it exposes unprecedented opportunities to improve business as a whole. Proximity marketing can be utilized to reach customers with making venue-specific products or offers information available.

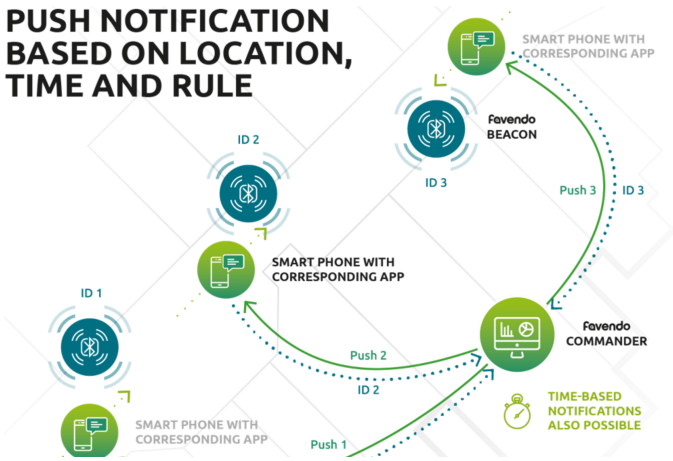

Figure 3: Proximity Marketing concept from Favendo Official [1]

We can assume a very simple use case of making the content or commercial offers available in mobile device of potential customers when they enter a specific zone or in close vicinity with the point of interest. The solution offered by Favendo is implemented with beacons making it independent of the network connectivity.

- **Analytics**

  Analytics utility is built in within the web based dashboard application that Favendo offers to customers. This datasets covers a wide range such as the purchasing trends, navigation patterns or interactions with tracked objects or assets.

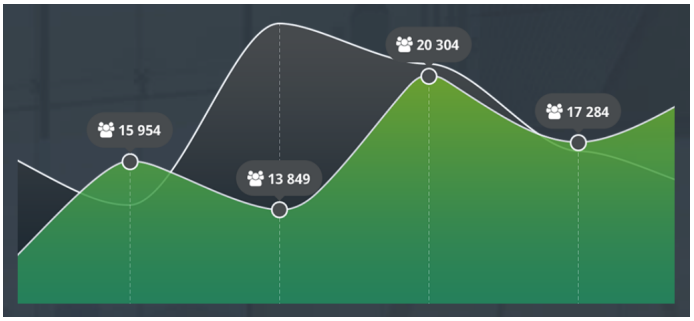

Figure 4: Analytics concept from Favendo Official [1]

This invaluable dataset could help customers with determining the possible optimization scopes in their product lines or services offered as well taking rational decisions improving the customer engagement. The image 4 shows a concept study made for an indoor shopping center to have an overall idea of visitors movement over a course of time.

2.1 Description

Efficient framework system plays a key role in the services that Favendo offers to its customers and works as the enabler and customization point for consuming location based service. It consists of several related subsystems and goes by the project name Commander [5]. Commander is executed with a highly available backend server side module that often securely hosts the classified information and a highly customizable mobile application SDK that is used to integrate location based services offered by Favendo to the mobile application developed for popular mobile software ecosystems such as iOS an Android. In order to provide the indoor navigation/positioning utility this framework comes with a map implementation customizable as per specific customer needs.

During my internship I was engaged with various enhancements and bugfixes for this framework especially applicable for the iOS mobile platform. I have participated in the research and development efforts targeted towards the next major release of the commander mobile framework. I have learned a great deal about the iOS ecosystem in general and application programming constructs and relevant design patterns both with Swift and Objective-C as programming languages. I have participated in development efforts of reusable frameworks in the mobile application development. In the next

sections I have provide a vivid summary of the subject matters that I have learned or closely dealt with and my reflection of the same.

2.2 Key Concepts and Activities

In this section I illustrate the subject matters that formed the core of my internship and the concepts I have dealt with.

2.2.1 Object Oriented Programming and Design Patterns

M. D. Smith et al. [6] have described the object oriented design paradigm as the specification of task to be performed in terms of the associated objects and properties/behaviors of the objects. They have further illustrated an object as an instantiated entity having some operations it is capable to respond. It has state which could be impacted by the associated operations. Objects can invoke operations of other objects by passing messages. Smith et al. [6] has also described a class as an interface specifying the properties and behaviors of an object. Properties defined inside the class determines the state of the object when instantiated. OOP paradigm provides abstraction and encapsulation contributing to security and modularity the real power of re-usability comes in the form of inheritance. OOP has contributed in developing maintainable code for many large and sophisticated software applications.

Despite the capabilities that OOP has offered, modern software development can not be realized without adherence of appropriate architecture and design paradigm which needs us to adhere to certain disciplines while writing software application code. Erich Gamma et al.(The Gang of Four) [2] have mentioned that design patterns are the mechanism to identify, name and abstract away common themes in object oriented designs. They are generally based on the intent behind the design and they identify the collaboration, rolls and responsibilities of different objects in building a software application or solution. A very common frequently used example in mobile applications demographic is Model View Controller(MVC) design pattern which is often referred from the development of Smalltalk-80 programming environment as illustrated in the work of Krasner et al. [7]. The key focus of the MVC design patterns is to separate the functional units of an application for modularity and easier maintenance. It separates the class instances encapsulating data

and operations related to the application domain as `Model`, the presentation and display of the application state as `View` and user interaction and response with the model and the view as `Controller`. MVC is one of the popular design patterns often realized while developing standalone mobile applications. While there are many popular design patterns, in my internship I have worked with three popular object oriented design patterns as briefly illustrated below.

- **Mediator**

  The Gang of Four [2] describes the Mediator design pattern with an object that encapsulates the cooperation and interactions between several other objects by preventing them refer to each other explicitly. Object orientation suggests to reasonably distribute the application behavior among several objects. But since often one object needs to interact with other objects in the object oriented design it may result in many interconnection among the objects. While this distribution of responsibilities makes the system modular but uncontrolled interconnection and interdependencies makes the system appears as monolithic.

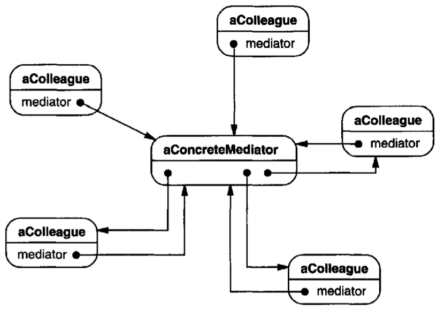

Figure 5: Generic Mediator diagram as explained by The Gang of Four [2]

  We can assign such collective behavior to a mediator object and let it coordinate among a group of objects and serve as an intermediary. The individual objects only know the mediator, resulting in reduced interconnection. In the diagram 5 we have `aConcreteMediator` type acting as an intermediary among the several objects often referred as `aColleague` as per The Gang of Four [2]. Favendo Commander framework has several individually shippable modules responsible for various tasks aiding location based services. We have used Mediator design pattern to coordinate among these modules in the next release of the framework in order to create an complete service provided by the framework.

- **Strategy**

  The Gang of Four [2] defines the Strategy design pattern as the capability of encapsulating a family of algorithms and making them interchangeable. In essence for different clients that are going to use the algorithm we can easily swap one strategy for other as per the clients needs and the underlying algorithm would also differ accordingly.

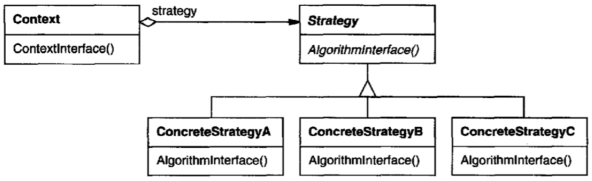

Figure 6: Generic Strategy diagram as explained by The Gang of Four [2]

Strategy often also called Policy design pattern, is applicable when several related objects having commonalities, only differs in certain behavior at runtime. In the basic structure 6 provided by The Gang of Four [2] we see that `Context` refers to the object that exposes a common interface which has a reference to a `Strategy` which could be an abstract base type or a protocol in certain languages. And we have several child types e.g. `ConcreteStrategyA B` and so on which derive from base type `Strategy`. From the perspective of `Context` it only has a strategy to carryout some task. Depending on the requirement, we can swap one child type for other in order to provide different outcomes for different strategies. Often in our framework in concern, we needed to record the specifications received from beacons hidden in a given zone in order to measure the outcome from our positioning algorithms. We could use different serialization strategies for these recording and we have found using strategy design pattern in this regard makes the implementation cleaner.

- **Delegate**

  Delegation as the Swift documentation [8] illustrates, is a design pattern that features an object to hand over certain responsibilities to another object. So the later becomes so called delegate of the former. Wikipedia reference [9] on delegation describes it in a more general sense such that delegation is defined by the evaluation of some properties by one object(delegate) in the context of another (sender). Delegation is a useful design pattern in OOP and its implementation

can be observed throughout the software ecosystems designed by Apple e.g iOS, watchOS, tvOS, macOS and even its predecessor NextStep as mentioned in [9].

Delegation is relatively simple design pattern and often contributes a great deal to achieve a clean code in software applications. Particularly in Apple software ecosystems, specifically in the languages like Objective-C and Swift, delegate pattern is implemented using a language construct called `protocol` [9] which is analogous to interfaces in other languages like Java. In our framework we have made use of delegate pattern extensively in order to achieve a cleaner object to object interactions as it is prescribed by Apple.

2.2.2 Test Driven Development

TDD

2.2.3 Beacon Ranging and Overview of Indoor Positioning

Beacon Ranging and Positioning.

2.2.4 Cross-platform Framework Development

Cross platform framework development

3 Reflection

This section will describe the reflection on the internship.

4 Conclusion

This section will have the conclusion.

5 References

References

[1] Favendo gmbh official. [Online]. Available: https://www.favendo.com

[2] E. Gamma, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[3] Fewclicks official. [Online]. Available: https://fewclicks.io

[4] Favendo bluloc specification. [Online]. Available: https://www.favendo.com/beacons/

[5] Favendo commander. [Online]. Available: https://www.favendo.com/commander/

[6] B. Smith, "Object-oriented programming," in *AdvancED ActionScript 3.0: Design Patterns*. Springer, 2011, pp. 1–25.

[7] G. E. Krasner, S. T. Pope *et al.*, "A description of the model-view-controller user interface paradigm in the smalltalk-80 system," *Journal of object oriented programming*, vol. 1, no. 3, pp. 26–49, 1988.

[8] Swift delegation. [Online]. Available: https://docs.swift.org/swift-book/LanguageGuide/Protocols.html

[9] Delegation wikipedia reference. [Online]. Available: https://en.wikipedia.org/wiki/Delegation_(object-oriented_programming)