The Use of Design Patterns in a Location-Based GPS Application

David Gillibrand and Khawar Hameed

Staffordshire University,
The Octagon, Beaconside, Stafford, ST18 0AD

Abstract

The development of location-based systems and applications presents a number of challenges – including those of designing and developing for a range of heterogeneous mobile device types, the associated spectrum of programming languages, and the incorporation of spatial concepts into applied software solutions. This paper addresses these challenges by presenting a harmonised approach to the construction of GPS location-based applications that is based on Design Patterns. The context of location-based systems is presented, followed by several design patterns - including the Observer and Bridge Design Patterns, which are described and applied to the application. Finally the benefits of using Design Patterns in this framework-oriented approach are discussed and future related work in the area of systems design for mobile applications is outlined.

Keywords: *Reusable software, Object Orientation, Design Patterns, Mobile Applications, Location-Based Systems.*

1. Introduction

The idea of design patterns came from a building architect Christopher Alexander who wrote a book "A Pattern Language: Towns, Buildings, Construction" in 1977 [1]. The idea of what a pattern is, is summed up in the following quote by Alexander: "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." Design Patterns have been used in object-oriented design since the widely acclaimed book Design Patterns: Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson and Vlissides [2] commonly known as the Gang of Four. It contains a description of the concepts of patterns, plus a catalog of 23 design patterns with their full documentation. They took Alexanders' idea of a pattern and applied it to software Engineering. Essentially a software design pattern is a piece of literature that describes a well tried out solution to a given problem in a given context. As part of that description the essential elements are the pattern name, problem description, problem solution and any consequences of using the pattern. The benefit of using patterns is to make the software more reusable - by building in quality attributes of software such as extensibility, maintainability and readability. The principle of design patterns therefore provides a basis for the development of specific schema based on systemic rigour at a low level of abstraction in systems modeling essentially providing a framework to encapsulate what are potentially complex application domains and the associated interactions between system components. These affordances provide the motivation for our paper that of conceptualising, capturing and modelling location-based systems and dynamic spatial data associated with the design and development of location-based applications -such as those used in navigation systems, using the relatively formal construct of design patterns.

2. Location-Based Systems and GPS

Having introduced the subject of design patterns and the motivation, the application domain of location-based systems is presented. This provides the basis for the subsequent articulation of design patterns and aims to demonstrate the transformation of a somewhat conceptual and abstract notion of space into an applied schema constructed using design patterns. These provide the foundation for programmers to construct code that underpins location-based applications.

Location-based systems are those which exploit and leverage the concept of mobility in context of local or remote environmental conditions and factors, and are founded on the core principle of *anyplace* as the driving rationale. Essentially, location-based systems deliver information that is relevant to users in context of their location at any particular point in time and where the focus or contextualisation of information and services is governed by location [3][4]. Furthermore, this information and the associated services can be defined as triggered or user-requested [4]. Developments in location-based systems have been driven by regulatory requirements such as international legislation and a growing awareness of the commercial opportunities facilitated by exploiting the technical ability to provide

value-added information and enhanced experience to mobile consumers, and through emerging demand levels, typical services and associated business models [5].
Location-based systems are strongly coupled to the concept of context within mobile computing systems and form a special class of context-aware systems [6]. Location is a determinant in that it contributes significantly to the universe of discourse created - essentially, all activities are hosted within a particular environmental location and context. A key characteristic of location-based systems is the changing physical location of the mobile user, which may be continual - such as when in a moving vehicle or walking, or periodic - where there are periods of short-term or transient residency of the user in a location.

Kakihara and Sorensen [7] discuss the view of spatial mobility as one dimension of mobility that is, 'the most immediate aspect of mobility' in that the physical locational space provides the immersive context for objects within that space. This discussion further articulates three composite aspects – that of the mobility of objects, the mobility of symbols, and the mobility of space itself. The dimensional aspects of location lead to a potentially more complex universe of discourse comprising the determination of object positioning within space (for example using co-ordinate geometry) where location identification is not only based on triangulation of co-ordinates - where each co-ordinate represents a particular dimension, but also based on time – where objects move through space and time. In this case, objects can be deemed to possess an orthogonal property where *different locations in time* exist for those objects.

The transformation of these spatial concepts into real-world deployments of location-based systems is also evident. Within the public sector there is acknowledgement of the unique features and advantages of mobile technologies to enhance engagement between governmental institutions and the citizens they serve through the development of innovative location-based services and new methods of interaction [8]. Specific examples of mobile location-based applications include those concerned with supporting front-line emergency services for public security and safety [6] with a range of associated improvement and efficiency gains being reported.

Private sector interest in mobile location-based systems is underpinned by new commercial and revenue-generating opportunities evidenced primarily by numerous consumer-oriented applications in a number of categories including navigation and travel, social networking, leisure and entertainment. For example, in September 2009 Apple, through its mobile application distribution channel - the 'App Store', had twenty different categories of mobile applications with 268 applications within the 'travel' category, and a large number of applications across all categories that exploited the user's location profile as part of the application configuration.

Supporting technology for location-based systems typically includes mobile network platforms for determining location including wide area systems such Cell Identification in mobile radio networks, Global Positioning Systems (GPS), and Broadband Satellites [5] and more localised sensor technologies such as WiFi (802.11), Bluetooth, and Radio Frequency Identification (RFID) [9]. Spatial databases provide the core repository infrastructure to host multi-dimensional data, with associated data models and query capabilities that enable location-based queries to be satisfied. The end-to-end delivery of mobile location-based systems includes a number of stakeholders, each of which is critical to the operation of the complete system. These include mobile network operators, content providers and aggregators, technology infrastructure providers, application service providers, and device manufacturers. As the potential scope and opportunities offered by mobile location-based systems increase, there is a risk of increasing complexity leading to evaluation of suitable business models and frameworks and components that address the overall aggregation of services [10][11]. Whilst appreciating this increasing complexity at higher levels of abstraction in location-based systems, we posit that an equal focus and effort on the use of design patterns to formalize and structure the lower-level construction of such systems is of merit.

The remainder of this section introduces the example of a GPS application as a component of a location-based system. This illustration is subsequently developed and serves a vehicle for the articulation of the associated design patterns.

The GPS application consists of reading data from a GPS receiver which constantly sends a stream of $GPRMC sentences to a GPS class. An example of a sentence is: $GPRMC, 140036,A, 5226.5059, N, 00207.6806, W,2.0, 064.64, 120710,001.0,E*34 where 194322 is the time of fix (14:00:36 UTC), A is a navigation receiver warning (A = OK, V = warning), 5226.5059,N is Latitude (52 deg. 26.5059 min North), 00207.6806,W is Longitude (002 deg. 07.6806 min West), 2.0 is Speed over ground (Knots), 064.64 is Course Made Good(degrees), 120710 is Date of fix (12 July 2010), 001.0,E is the Magnetic variation (1.0 deg East), *34 is the mandatory checksum.

The application then continually reads, parses and stores the sentences as records in a buffer. Those records are then available for the application to read. The application displays three views, a text view which as well as displaying the basic information in the GPS sentence also displays the distance travelled and average speed, a compass which uses the course made good part of the sentence (the user needs to be travelling at about 3knots for this to display a meaningful value) and a breadcrumb trail which shows the trail as well as minimum height, maximum height and the ascent (the difference between them) see Figure 1.

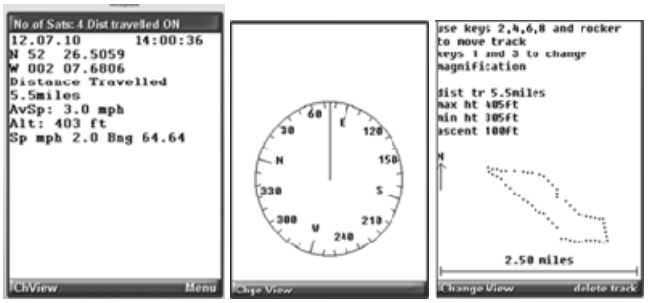

Fig. 1. Different views from a GPS application

3. Design Patterns Used in a GPS Application

The principle design patterns used are the Observer, Strategy and Bridge design pattern. What follows is a description of those patterns followed by an explanation of how those patterns can be applied to the GPS application.

3.1 Observer Design Pattern

The intent of the pattern as described by Gamma et al. is to define a one-to-many dependency between objects so that when one object state changes all the other objects are notified and change state accordingly. The observer pattern can be applied in any of the following situations: When an abstraction has two aspects, one dependent on the others, encapsulating these aspects in separate objects lets you vary and reuse them independently; when a change to one object requires changing others, and you don't know how many objects need to be changed; When an object should be able to notify others without making assumptions about who these objects are [2]. Figure 2 shows a class diagram that represents the general case of the observer design pattern.

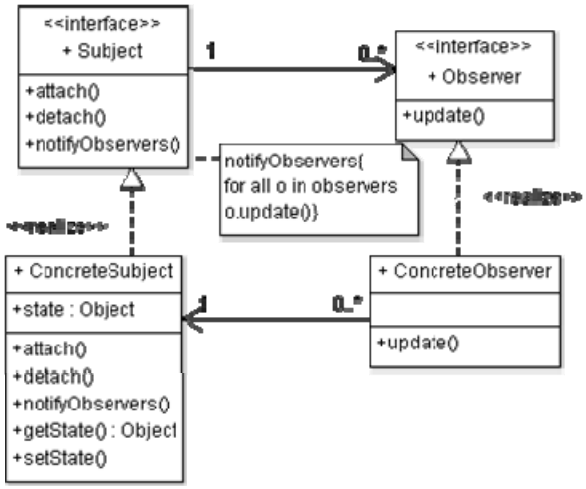

Fig. 2. The Observer Design Pattern

The ConcreteSubject class contains the data (state) with associated get and set methods and a list of observers. There are methods to add or delete an observer, (attach and detach). The notifyObservers method iterates through the observers list in the ConcreteSubject class invoking the update method in each observer object. The update method in the Observer object then gets the state (or data) of the Subject object. The Observer pattern can be varied with respect to the update protocol. The Pull model protocol (which we've just described) can be implemented in java by sending a changeEvent object to the observers (views) every time the data or state is changed in the Subject Object. On receiving this object the views obtain the latest data from the subject by invoking the update method. Alternatively there is the Push model protocol - when data is changed, the Subject sends a message to the Observers saying that the data has changed and also sends an additional argument that describes the state change. This comes in as an argument to the update method, so the observer has already got the latest data without having to invoke the getState method of the Subject.

3.2 Strategy Design Pattern

The Strategy Pattern "defines a family of algorithms, encapsulates each one, and makes them interchangeable. The Strategy pattern lets the algorithm vary independently from clients that use it" [12]. Instead of using case statements to differentiate between different algorithms, a more flexible design is to encapsulate each transformation algorithm as a separate class. The class diagram for Strategy is shown in Figure 3.

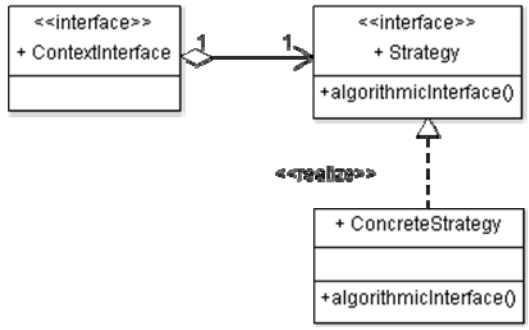

Fig. 3. The Strategy Design Pattern

3.3 Bridge Design Pattern

A more flexible alternative to the strategy design pattern is the Bridge design pattern which allows you to vary the abstractions as well as the implementation by placing them into two different class hierarchies - see Figure 4.

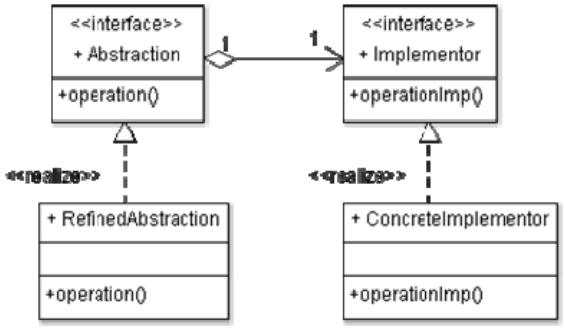

Fig 4. The Bridge Design Pattern

4. Applying Patterns to a GPS Application

4.1 Application of the Observer Design Pattern

The GPS application lends itself to the observer design pattern. The Subject or the data of the application is encapsulated by a Record class. The Record class has attributes that reflect the information contained in the GPS sentence. The three views (Figure 1) need to be updated every time the Record object changes its values. Figure 5 shows how the observer design pattern has been incorporated into the GPS application.

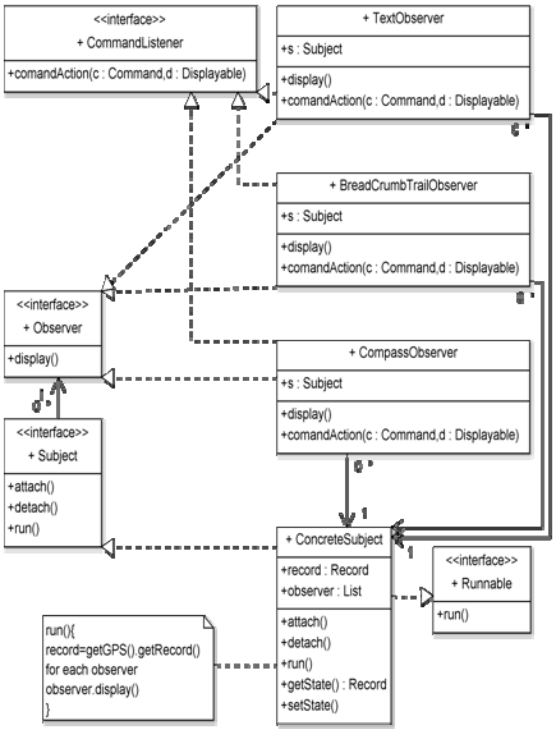

Fig. 5. The Observer Design Pattern Applied to a GPS Application

The ConcreteSubject has a List of the three observers (the views as shown in Figure 1). It also contains the state (record). This record is constantly updated from the GPSreceiver. The views or observers need to be constantly refreshed with the current value of the record. The ConcreteSubject invokes its own thread of execution (it implements the Runnable interface) with the run method. The run method takes the place of the notifyObservers method in the general case (Figure 2). The run method iterates through the three observers and invokes their display method. The observers have an object reference to the Subject passed to them in their constructor method and are then able to invoke the getState method of the Subject. This is essentially using the Pull model protocol. The use of the Observer design pattern allows you to easily extend the application with further views if required and very little modification to the existing code making it a modular and robust design. It decouples the data from the views.

The Observer design pattern is the principle one used in MVC (model view controller) architecture. The model is the data of the application, the views are different views of the data and are responsible for drawing that data on the screen and the control is responsible for handling the user input and then updating the model or the view. In our

application the function of the control is split between the Subject which gets the record from a GPS class (record=getGPS().getRecord()) and the Observers which implement the CommandListener interface which can then receive user input commands .
The application also has the ability to change the position format. It can be changed to the OSGrid reference system which is the British national grid system, instead of lat and long, in which case the position format is displayed in terms of northings and eastings as found on an
Ordnance Survey map - see figure 6.

Fig. 6 A view showing the British National Grid reference system

4.2 Application of the Bridge Design Pattern

The calculation that transforms from a lat/long coordinate system to the OSgrid coordinate system is quite complicated and, in the future, when extending the application it might be deemed appropriate to include other geographical grid systems for transformation. One way of implementing such a functional request might be to use a case statement and passing in a value which reflects the grid system to be chosen. However, this design is inflexible and would involve a rewrite of the class that contained the case statement for each change made. A better way would be to use the Bridge design pattern. In the GPS application, this is applied by creating a new class for each transformation algorithm - see Figure 7 making that part of the application easily extendable.

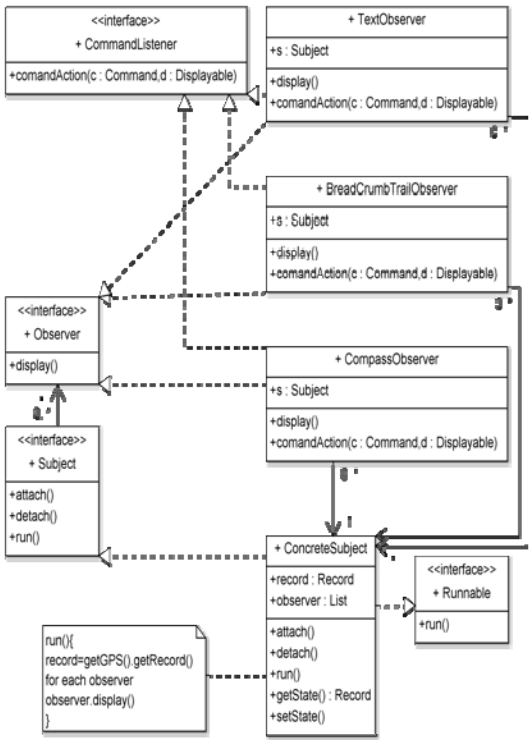

Fig. 7. The Bridge Design Pattern applied to a GPS application

5. Conclusions

The use of design patterns brings many benefits in the software development process, especially in terms of code reuse and maintainability. Much of the software that results offers the desirable properties of high cohesion and low coupling. Many of the patterns are well documented and categorised according to different criteria - for example the Gang of Four categorises its patterns by scope (Object or Class) and their purpose (creational, structural or behavioural). The key to successfully using design patterns is to learn and understand the pattern and to classify it in a meaningful way. Once understood, using patterns become second nature and if well documented allows a design level of abstraction to be visible with the added benefits of improved communication between developers.

The use of design patterns to underpin the development of application-specific components of location-based system warrants particular attention. The increasing focus on the development and adoption of location-based systems provided fertile ground for developing schematic and reusable constructs that provide a vehicle for capturing conceptual aspects of location and the translation of these into applied and usable notation for sytem developers. Our focus on design patterns for location-based systems

has also been incorporated into a Masters level course in Mobile Applications and Systems where students are taught the conceptual basis of mobility and location-based systems, and the associated development of software underpinned by design patterns. The approach to design patterns in this paper can be adopted by both scholars and practitioners in industry. Our future work in this area is concerned with development of schematic constructs to model the multi-dimensional aspects of mobility – those of time, space, and context and to position these at different levels of abstraction within the system development process - such as design patterns at lower levels of abstraction, and enterprise architecture-based constructs at higher levels of abstraction. In doing so, we aim to focus on a systemic and structured approach to the development of mobile applications and systems.

References

[1] C. Alexander Pattern Language: Towns, Buildings, Construction. 1977

[2] Gamma , Helm, Johnson, Vlissides, Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley 1994

[3] Duri, S., Cole, A., Munson, J. & Christensen, J., 2001, WMC '01: Proceedings of the 1st international workshop on Mobile commerce, An approach to providing a seamless end- user experience for location-aware applications. ACM, pp. 20-5.

[4] D'Roza, T. & Bilchev, G., 2003, An overview of location-based services, BT Technology Journal, 21(1), pp. 20-7.

[5] Rao, B. & Minakakis, L., 2003, Evolution of mobile location-based services, Commun. ACM, 46(12), pp. 61-5.

[6] Streefkerk, J.W., van Esch-Bussemakers, M.P. & Neerincx, M.A., 2008, MobileHCI '08: Proceedings of the 10th international conference on Human computer interaction with mobile devices and services, Field evaluation of a mobile location-based notification system for police officers. ACM, pp. 101-8.

[7] Kakihara, M. & Sørensen, C., 2001, Expanding the 'mobility' concept, SIGGROUP Bull., 22(3), pp. 33-7.Trimi, & Sheng 2008

[8] Trimi, S. & Sheng, H., 2008, Emerging trends in M-government, Commun. ACM, 51(5), pp. 53-8.

[9] Johnson, S., 2007, A framework for mobile context-aware applications, BT Technology Journal, 25(2), pp. 106-11.

[10] Aphrodite & Evaggelia, 2001, Business models and transactions in mobile electronic commerce: requirements and properties, Computer Networks, 37(2), pp. 221-36.

[11] de Reuver, M. & Haaker, T., 2009, Designing viable business models for context-aware mobile services, Telematics and Informatics, 26(3), pp. 240-8.

[12] Freeman, Freeman, Sierra, Bates Head First Design Patterns O'Reilly 2004

David Gillibrand is a Senior Lecturer in the Faculty of Computing, Engineering & Technology at Staffordshire University. His research is in the area of Object-Oriented technologies, Design Patterns, Enterprise Applications, Mobile programming, Databases, and System methods. He has had publications in object-oriented journals and delivered courses in system design to industry.

Khawar Hameed is a Principal Lecturer in the Faculty of Computing, Engineering & Technology at Staffordshire University. His research is in the area of mobile and remote working, enterprise mobility, and mobile learning. He has been a key driver in the adoption of mobile computing and technology within the Faculty's portfolio and has helped drive the development of undergraduate and post- graduate degrees in this technology area. He has contributed extensively to the development and delivery of externally funded projects and academic-industrial collaborations in mobile/wireless technology that aim to develop and enhance the collective intellectual capital that supports the growth of mobile and wireless systems as a discipline both within academia and in industry.