GoF Design Patterns in a Smart City System

Wasana Ngaogate*
Ubon Ratchathani University, Warinchamrab, Ubon Ratchathani, Thailand.

* Corresponding author. Tel.: 66-880817270; email: wasana.n@ubu.ac.th
Manuscript submitted January 10, 2014; accepted March 8, 2014.
doi: 10.17706/jsw.14.5.220-226

Abstract: This paper demonstrates how GoF software design patterns can be applied in a smart city system. By giving a case study, observer pattern and composite pattern are used. SoA architecture is applied for showing how a system with heterogeneous devices is structured. Novice software engineers might use this demonstration as a guide for building up a smart city system.

Key words: Smart city, SoA architecture, GoF software design patterns.

1. Introduction

According to an Android application that the author has developed for providing information about temples and their activities that participated in the candle festival at Ubon Ratchathani, Thailand in 2018. It was designed based on Model-View-Controller (MVC) design pattern and implemented Adapter pattern provided by Android SDK. The application works fine by showing temple information such as location, history, gallery, and direction. Tourists can review temples and the festival. Moreover they are notified by real-time messages when they are approaching some interesting activities predefined in the system. A location beacon is installed at the location of each activity of a temple in order to broadcast Bluetooth signal around the area. The mobile application detects the Bluetooth signal when tourists select scan function.

Information stored in the system can be shared with other applications related to Ubon Ratchathani, for example, rating and review of each location, which temples are mostly scanned for locations. On the other hand, the Ubon Candle festival application might need to include information from other applications such as recent announcement about the festival or about the province. Moreover, various types of other applications can use the same beacons for different purposes.

This paper proposes SoA architecture for incorporation between these applications and demonstrates how to apply GoF software design patterns in such architecture. It aims to leverage Ubon Ratchathani to be a smart city. Section II review about smart city and Section III is about Design patterns. A case study of applying GoF design patterns is demonstrated in section IV. Section V is about conclusion and further works.

2. Smart City

Smart city is currently well known word. As Renata Paola Dameri [1] explained that a term of smart city has been defined and named in several contexts such as digital city, knowledge city, and intelligent city. It is important to explicitly define vision of smart city, align initiatives and desired outcomes. The core elements of a smart city are aspects about technology, culture, and environment. Each of them has different role and effect to citizens. José Álvarez-García, *et al.* [2] concluded that smart city has to assure better life quality in the overall by using an appropriate information and communications technologies. Residents should get knowledge on the use of the ICT in order to use more electronic services.

Zhijie Li and Wei Zhang [3] introduced "Four-A-Service" method and designed an evaluation model which is able to estimate the level of intelligent city objectively. The model composes of Anyone-Oriented, Anytime-Oriented, Anywhere-Oriented, and Any device-Oriented. In conclusion, technologies of smart city should be able to identify citizens who use different devices across heterogeneous platforms at any time. Whereas Rama Krishna Reddy Kummithaa and Nathalie Crutzen [4] proposed 3RC framework for clearly explanation of a smart city: Restrictive, Reflective, Rationale, and Critical. Cashless payment is important issue of a smart city. D.Anuradha, *et al*. [5] proposed payment by using Quick Response (QR) code in a smart bus. Liguo Lou, *et al.* [6] also interviewed 247 tourists about QR code while traveling and found that it is pleasurable.

Smart city utilizes various IoT applications which are developed in different domains. Michele Nitti, et.al. [7] presented an IoT architecture for a sustainable tourism application in a smart city. It composes of four layers: Application layer, Service Layer, Virtualization layer, and Real-world layer or physical layer. They claimed that this architecture enhances developers to create templates of objects and services in order to be reused in other systems. Biljana Risteska Stojkoska, Kire Trivodaliev, and Danco Davcev [8] introduced a design of home care system architecture which composes of three levels where data flows between one to another adjacent. Data from sensors located in rooms is called Dew computing level. Fog computing level makes decision. Cloud computing level stores data, discovers knowledge, and makes advance decision.

Jie Lin, *et al.* [9] explained two types of IoT architecture: Three-Layer architecture and SoA-based architecture. Three-Layer architecture composes of three basic layers: perception layer as the sensor layer, network layer as transmission layer, application layer as business layer. Whereas SoA-based architecture composes of four layers: perception layer, network layer, service layer, and application layer. Perception layer collects data from physical devices. Network layer supports data transmission over different networks. Service layer provides proper service for application layer which supports users' requests. Whereas Aditya Tiwary, *et al.* [10] studied various IoT applications including their elements and architectures. They concluded that key elements of IoT system are unique identification for each smart device, sensing devices, communication between devices, data storage and analytics, and Visualization.

Ibrar Yaqoob, *et al*. [11] suggested that quality of service (QoS) of real-time processing in the next IoT architectures should response only the require information. The architecture must enhance communication between heterogeneous devices without interference. They also hinted that practical interoperability can be done by creatively predefined specifications of the components. Victor Basili, *et al.* [12] promoted context-driven research which is driven by concrete needs in particular domains and development projects. Context-driven research focuses on practicality and scalability in realistic conditions. Therefore, collaborations between researchers and industry should be encouraged in order to identify problems, provide solution and evaluation in realistic contexts.

As a software engineer, Brice Morin, Nicolas Harrand, and Franck Fleurey [13] explained that IoT applications consist of large heterogeneous processing nodes. There are tools and techniques for simpler coding in order to hide such heterogeneity. They demonstrated how to apply ThingML with e-Health system. ThingML is model-based approach which abstracts heterogenous platforms and IoT devices.

Andre Dionisio Rocha, *et al.* [14] proposed semantic model for a smart city which allows components to be added or removed as needed. The proposed semantic model composes of classes and subclasses. Each class has a set of data properties and connections between classes. For example, class 'Thing' composes of class 'Event' 'Skill' 'Identity' and 'Parameters'. A class 'Identity' composes of class 'Single Entity' and 'Group

Entity'. Each class communicates to another based on available 'Skills' such as 'SmartHouse' can 'askForGridConnection'. Therefore, software design patterns will be reviewed in the next section.

3. Design Patterns

Software architecture is important discipline. Daniel Feitosa, *et al.* [15] studied five open-source projects and confirmed that software with good architecture always built based on best practices such as design patterns. The code is cleaner with less infraction. Stefan Nadschläger and Josef Küng [16] is good for developing custom knowledge processing systems. It is possible for novice software engineers to recognize and apply design patterns as Jonathan W. Lartigue and Richard Chapman [17] as confirmed. Carmine Gravino and Michele Risi [18] verified that the use of design pattern impacts on the quality of the software.

In terms of Industrial IoT, Gedare Bloom, *et al.* [19] identified common input-output 6 design patterns for control system: Closed-loop, Cloud-in-the-Loop, Open-Loop, Cloud-on-the-Loop, Publisher, and Device-to-Device. Each pattern supports particular purpose of industrial control system. For example, Publisher pattern facilitates data collection from field devices whereas Device-to-Device pattern enhances shared data among peer machines. In terms of Smart Contract in the Ethereum Ecosystem, Maximilian Wohrer and Uwe Zdun [20] categorized widely used 18 design patterns into 5 categories: Action and Control, Authorization, Lifecycle, Maintenance, and Security. For example, authorization consists of Ownership pattern and Access Restriction pattern.

Gang-of-Four design patterns are well-known software patterns. Muhammad Noman Riaz [21] surveyed existing software design patterns and concluded how GoF design patterns impact software quality in terms of maintainability, evolution, performance, and faults. According to the finding, software maintainability is provided by Abstract pattern, Composite pattern and Decorator pattern. Whereas Bruno L. Sousa, *et al.* [22] investigated large class in five Java project developed with design patterns. They concluded that Composite pattern and Factory Method pattern have a low co-occurrence with long method whereas Template method and Observer method have a high co-occurrence with large class and long method. However, Sofia Charalampidou, *et al.* [23] suggested that misuse of Decorator pattern with composite objects can decrease system cohesion and weaken modularity and maintainability.

Apostolos Ampatzoglou, *et al.* [24] studied how GoF design patterns effects stability of classes. They found that Singleton, FaçadeMediator, Observer, Composite, and Decorator patterns are able to confront changes of other classes. Aggregation relationship is less stable than Component. Public super class is more stable than subclass.

Foutse Khomh and Yann-Gaël Guéhéneuc [25] suggested research community in design patterns to focus on four issues: Patterns from devlopers' behaviour, patterns of developers' behaviour, patterns for building systems, and theories of software patterns. Also, Vardhan A. and Chaturvedi A. [26] recommended that ontology driven approach support better implementation than GoF when modifiability, reusability and flexibility is crucial. However, Feitosa D., Avgeriou P., Ampatzoglou A., Nakagawa E.Y. [27] commented pattern grime tends to increase linearly.

4. A Case Study

As Jie Lin, *et al.* [9] explained a service-oriented architecture, SoA-based architecture, composes of four layers. This paper provides examples in each layer. Perception layer collects data from heterogeneous physical devices, for example, tourists' locations detected by beacons or GPS, smart buses' locations detected by GPS or RFID, weather information such as temperature, humidity, rainfall, wind speed, wind direction detected by sensors. Network layer supports data transmission over networks used by those applications. Service layer provides services for application layer which supports users' requests. For example, the Ubon Candle Festival application requests Bluetooth signal broadcasted by beacons, smart bus application requests GPS locations from buses in order to show on a map.

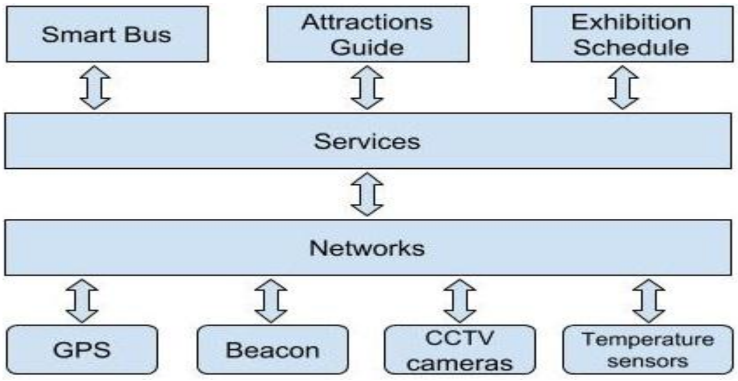

Fig. 1. SoA architecture of a smart city.

Observer pattern is applied as Fig. 2. There are several types of service in a smart city such as SmartBusService, AttractionService, and ExhibitionService. Each service has its own information for sharing. Observer is a class which subscribes to each service and waits for notification from service. For example, SmartBusSubscriber is an observer of SmartBusService.

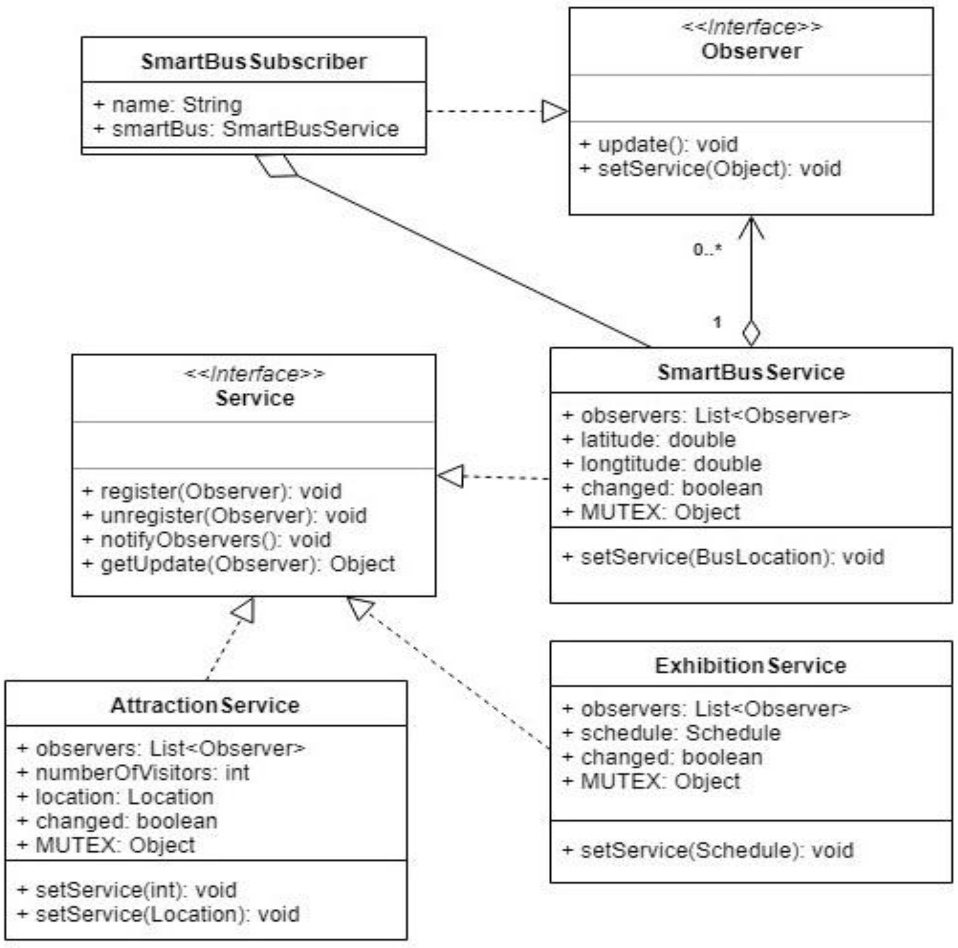

Fig. 2. Observer pattern for an application of a smart city.

Individual systems provide information to services in an application of a smart city. For example, a

SmartBus system has two types of bus: State bus and Private bus. It provides a bus location to public. An Exhibition system provides schedule and streaming video captured from CCTV. Whenever things change in individual systems, information is broadcast to other observers in an application of a smart city.

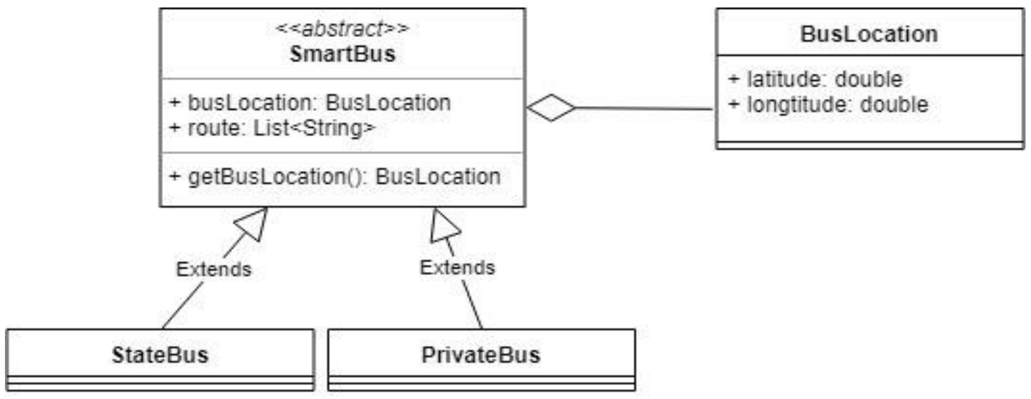

Fig. 3. Some classes in smart bus system.

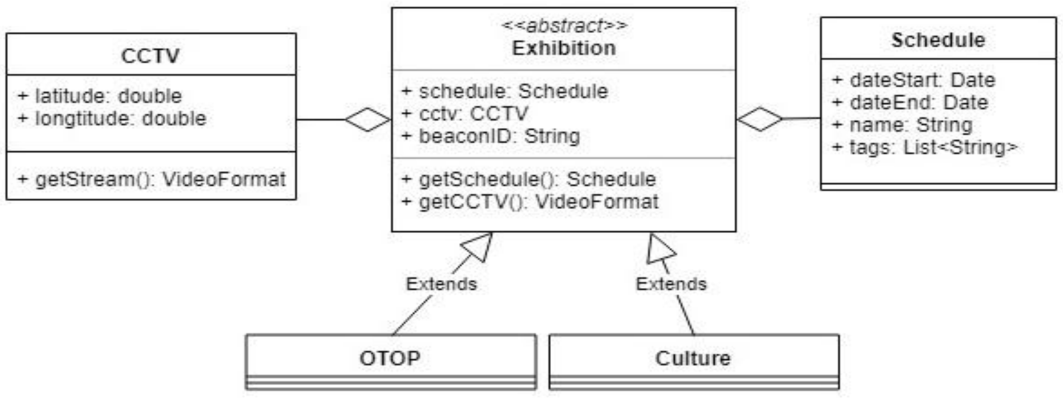

Fig. 4. Some classes in exhibition system.

Composite pattern can be applied by grouping all services as below. An “initiate()” method of “AllServices” class manages all operations needed in an application of a smart city.

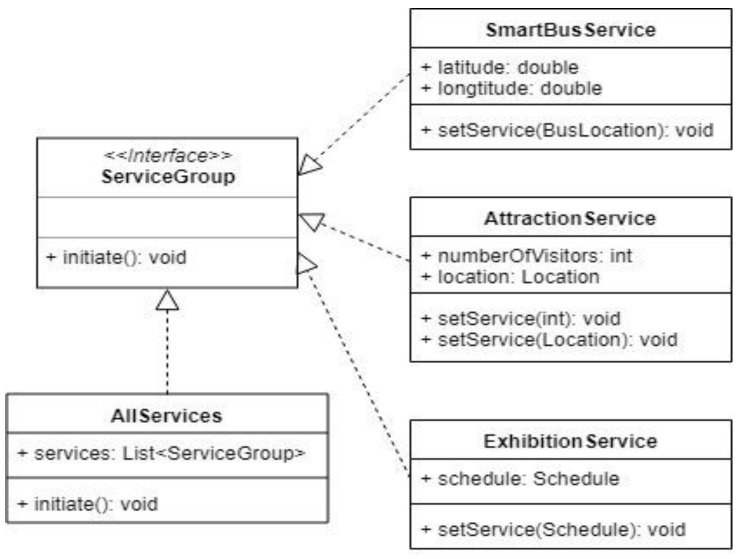

Fig. 4. Some classes in exhibition system.

5. Conclusion and Further Works

As shown in a case study, GoF design patterns can simply applied to a smart city system. Novice software engineers can follow this demonstration as a guide. The most important concern is designing both software architecture and components so that the system can be regularly maintained and scaled up in the future.

In order to provide a full demonstration, the Ubon Candle Festival application should be modified as the proposed suggestion along with some other applications related to a smart city.

References

[1] Renata, P. D. (2017). *Smart City Implementation, Creating Economic and Public Value in Innovative Urban Systems.* Springer International Publishing AG.

[2] José, Á. G. (2016). Smart city and tourism: an analysis of development of caceres (spain) as a smart city. *Sustainable Smart Cities Innovation, Technology, and Knowledge Management.*

[3] Li, Z., & Zhang, W. (2017). Four-a-services oriented evaluated system of intelligent city. *International Symposium for Intelligent Transportation and Smart City (ITASC).*

[4] Rama, K. R. K., & Nathalie, C. (2017). How do we understand smart cities? An evolutionary perspective. Cities. *The International Journal of Urban Policy and Planning*, *67,* 43-52,

[5] Anuradha, D., Durga, D. M. V., Keerthana, K., & Dhanasree, K. (2018). Smart bus ticket system using QR code in Android App*. International Research Journal of Engineering and Technology (IRJET)*, *5(3).*

[6] Lou, L., Tian, Z., & Koh, J. (2017). Tourist satisfaction enhancement using mobile QR code payment: An empirical investigation. *Sustainability, 9(7),* 1186.

[7] Michele, N. (2017). IoT Architecture for a sustainable tourism application in a smart city environment. *Mobile Information Systems, 2017.*

[8] Biljana, R. S., Kire, T., & Danco, D. (2017). Internet of things framework for home care systems. *Wireless Communications and Mobile Computing, 2017.*

[9] Lin, J. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal, 4(5).*

[10] Aditya, T. (2018). Internet of things (IoT): Research, architectures and applications. *International Journal on Future Revolution in Computer Science & Communication Engineering*, *4(3).*

[11] Ibrar, Y. (2018). Internet of things architecture: recent advances, taxonomy, requirements, and open challenges*. IEEE Wireless Communications, 24(3),* 2017.

[12] Victor, B. L. B., Domenico ,B., Shiva, N., Fabrizio, P., & Mehrdad, S. (2018). Software engineering research and industry: A symbiotic relationship to foster impact*. IEEE Software, July.*

[13] Brice, M., Nicolas, H., & Franck, F. (2017). Model-based software engineering to tame the IoT jungle. *IEEE Software, January/February.*

[14] Andre, D. R. (2017). Semantic model to perform pluggability of heterogeneous smart devices into smart city environment. *Service Orientation in Holonic and Multi-Agent Manufacturing, Studies in Computational Intelligence.*

[15] Daniel, F. (2019). What can violations of good practices tell about the relationship between GoF patterns and run-time quality attributes? *Information and Software Technology, 105(2019),* 1-16.

[16] Stefan, N., & Josef, K. (2017). Analysis of GoF design patterns used in knowledge processing systems. *Proceedings of the 22nd European Conference on Pattern Languages of Programs EuroPLoP.*

[17] Jonathan, W. L., & Richard, C. (2018). Comprehension and application of design patterns by novice software engineers. *The Annual ACM Southeast Conference (ACMSE), 24.*

[18] Carmine, G., & Michele, R. (2017). How the use of design patterns affects the quality of software systems: A preliminary investigation. *Proceedings of 43rd Euromicro Conference on Software*

Engineering and Advanced Applications.

[19] Gedare, B. (2018). Design patterns for the industrial internet of things. *Proceedings of 14th IEEE International Workshop on Factory Communication Systems (WFCS).*

[20] Maximilian, W., & Uwe, Z. (2018). Design patterns for smart contracts in the Ethereum ecosystem. *Proceedings of IEEE International Conference on Blockchain 2018.*

[21] Muhammad, N. R. (2018). Impact of software design patterns on the quality of software : a comparative study. *Proceedings of International Conference on Computing, Mathematics and Engineering Technologies – iCoMET.*

[22] Bruno, L. S. (2017). Evaluating co-occurrence of GOF design patterns with god class and long method bad smells. *XIII Brazilian Symposium on Information Systems, Lavras, Minas Gerais.*

[23] Sofia, C. (2017). A theoretical model for capturing the impact of design patterns on quality: The decorator case study. *Proceedings of the Symposium on Applied Computing SAC'17* (pp. 1231-1238).

[24] Apostolos, A. (2015). The effect of GoF design patterns on stability: A case study. *IEEE Transactions on Software Engineering, 41(8).*

[25] Foutse, K., & Yann-Gaël, G. (2018). Design patterns impact on software quality: Where are the theories? *Proceedings of IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER).*

[26] Vardhan, A., & Chaturvedi, A. (2017). Ontology-driven shopping cart and its comparative analysis. *Communications in Computer and Information Science, Springer, Singapore.*

[27] Feitosa, D., Avgeriou, P., Ampatzoglou, A., & Nakagawa, E. Y. (2017). The evolution of design pattern grime: an industrial case study. *Product-Focused Software Process Improvement PROFES 2017. Lecture Notes in Computer Science, 10611.* Springer.

Wasana Ngaogate was born in Ubon Ratchathani, Thailand. After finishing her first degree, she had worked as a programmer at Bangkok Bank for two years. She finished her master degree in computer science from the University of Warwick, England and worked as a lecturer at Ubon Ratchathani University since 1995. She is interested in software engineering particularly in design patterns implemented in various domains of practical applications.