Advanced Science and Technology Letters
Vol.107 (Ubiquitous Science and Engineering 2015), pp.16-19
http://dx.doi.org/10.14257/astl.2015.107.05

Deployable Singleton Pattern for UAV Ground Stations

Yunsick Sung[1,*], Jeonghoon Kwak[2],

[1] Faculty of Computer Engineering, Keimyung University, Daegu 704-701, South Korea,
yunsick@kmu.ac.kr
[2] Dept. of Computer Engineering, Graduate School, Keimyung University,
Daegu 704-701, South Korea,
jeonghoon@kmu.ac.kr

Abstract. As the number of unmanned aerial vehicle (UAV) ground station functions increases, the ground station complexity increases. The ground station structure should be flexible to support the variety of functions. To increase the ground station flexibility, singleton patterns are usually applied to single-instance-based classes in a ground station, such as an event manager class or a UAV manager class. However, given that no class can inherit from a singleton class, the singleton pattern prevents the expandability of a ground station. Therefore, the singleton pattern structure should be improved. This paper proposes a deployable singleton pattern that enables the structure to expand by utilizing overridden functions.

1 Introduction

Recently, as the roles of *Unmanned Aerial Vehicles* (UAVs) have diversified, the role of the ground stations that control UAVs has also expanded. For example, structure research has been conducted on ground stations that control multiple and hetero UAVs [1]. The ground station provides the user interfaces that control multiple UAVs simultaneously. However, the *singleton pattern* [2], one of the *Gang of Four* (*GoF) object-oriented design patterns*, is usually applied repeatedly to manage the variety of events and UAVs. Given that singleton pattern cannot expand a *singleton class* by defining inherited classes it, expanding the structure of a ground station is very difficult. For a more flexible ground station structure, the singleton pattern should be improved.

The research that improves traditional GoF design patterns using the example of a secure entrance system is discussed [3]. To design a secure entrance system, object-oriented programming is applied first. Then, to simplify the design and implementation of the secure entrance system after dividing it into multiple sub-systems, *aspect-oriented programming* (AOP) and design patterns are also applied.

For a crop-harvest survey, a location-based service (LBS)-based mobile system is introduced [4]. By applying design patterns, the reusability and expandability are improved. A real-time operating system increases the usability of multiple-joint-based

*Corresponding Author: Yunsick Sung (yunsick@kmu.ac.kr)

ISSN: 2287-1233 ASTL

small educational robots [5]. When a real-time operating system is applied, the mobile system's performance is improved. However, the system software complexity increases. To solve the complexity problem, the design pattern and refactoring are applied together. In addition, the design information around design patterns is deducted and the reference flows of the design patterns are revealed [6]. By utilizing the design information and the reference flows, the expandable points of the classes are extracted.

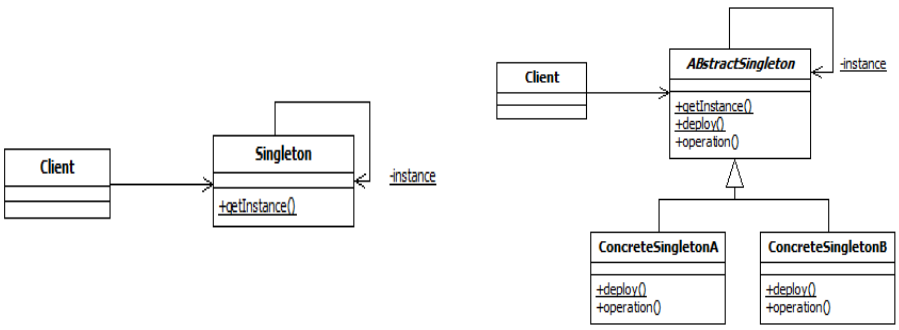
This paper proposes a method that extends the structure of singleton pattern to support the flexible structure of a ground station. Expanding the singleton class gives it the flexibility to add additional functions. This paper is consisted as follows. Section 2 proposes an expanded approach of the singleton pattern. Section 3 concludes our proposal.

2 Deployable Singleton Pattern

Fig. 1(a) shows the singleton pattern of the traditional GoF design pattern. The singleton pattern creates only one instance and prevents the creation of any more. The instance is created the first time the static *getInstance* function is called or the static initialization function is called. The created instance of the singleton class can be accessed through the static getInstance function.

Fig. 1(b) displays the proposed deployable singleton pattern. An *AbstractSingleton* is defined as a parent class of singleton classes and has one instance of one of singleton classes as a member variable. Classes *ConcreteSingletonA* and *ConcreteSingletonB* are inherited from the AbstractSingleton class and defined as its child classes, which is different from the traditional singleton pattern.

In the proposed pattern, only one instance of ConcreteSingletonA or ConcreteSingletonB can be created. If the instance is created more than once, only the last-created instance can be utilized. The class is determined by calling the static *deploy* function.



(a) GoF Singleton Pattern (b) Deployable Singleton Pattern

Fig. 1. Proposed Deployable Singleton Pattern

To utilize the functions of child classes ConcreteSingletonA and ConcreteSingletonB, their *operation* function is redefined (overridden). Then, the operation function of the AbstractSingleton class is called, which invokes the operation function of the child classes.

GoF design patterns have a *factory method pattern*, which is utilized when the creation time of a certain class cannot be estimated. Given that it has not yet been determined which AbstractSingleton child class will be used, the factory method pattern can be applied to select one of the child classes later. In the deployable singleton pattern, a factory method pattern is applied, as shown in Fig. 2. The child classes of *AbstractFactor*, *ConcreteFactoryA* and *ConcreteFactoryB*, are determined by calling the *factoryMethod* function.

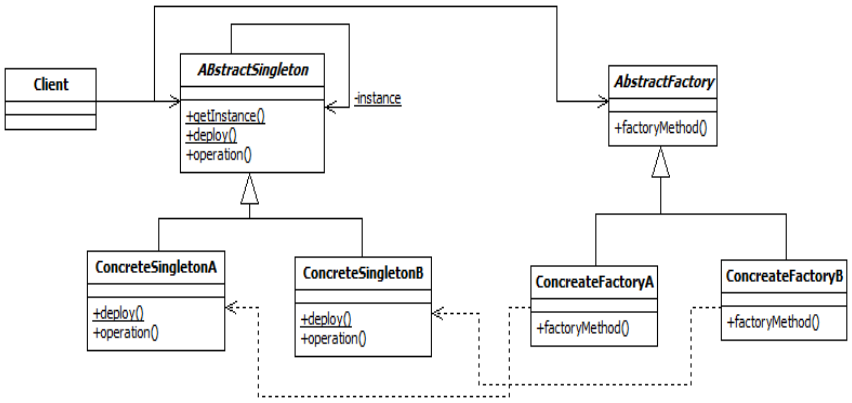


Fig. 2. Deployable Singleton Pattern with Factory Method Pattern

As a result of utilizing the singleton and factory method patterns, the proposed deployable singleton pattern has the following advantages. First, a client class can access a created instance anywhere by calling the getInstance function. Next, generating an instance of the singleton class can be delayed, which enables the extension of AbstractSingleton. Next, a class that will be created can be determined and utilized after defining all the classes.

3 Conclusion

This paper proposed deployable singleton pattern that expands on the traditional singleton pattern. A singleton class in the traditional singleton pattern was expanded by defining inherited child classes; only one of the inherited classes can be selected and utilized by calling the static deploy function and the static getInstance function. The inherited classes define and call their function operation based on the overridden functions. In addition, the inherited classes can be selected and created by applying the factory method pattern.

Acknowledgement. This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A1A1005955)

References

1. Sung, Y., Kwak, J., Yang, D., Park, Y.: Ground Station Design for the control of Multi Heterogeneous UAVs, The Korea Multimedia Society Spring Conference, Andong, Gyeongbuk, Korea, (2015)
2. Erich, G., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software, Pearson, pp. 496 (2007)
3. Kim, T., Cheon, H., Lee., H.: Development of Secure Entrance System using AOP and Design Pattern, The Korea Academia-Industrial Cooperation Society, Vol. 11, No. 3, pp. 943-950 (2010)
4. LEE, H., BAEK, J., MUN, Y.: Implementation of Mobile System based on LBS using Design Patern, Journal of the Korean Association of Geographic Information Studies, Vol. 12, No. 1, pp. 26-35 (2009)
5. Son, H., Kim, W., Ahn, H., Kim, R.: Aplying Design Patern & Refactoring on Implementing RTOS for the Smal Educational Multi-Joint Robot, The Journal of The Institute of Internet, Broadcasting and Communication, Vol. 9, No. 3, pp. 217-224 (2009)
6. Kim, H., Park, C., Kim, T., Yoo, C., Lee, H.: Identifcation of the Extension Points of Design Paterns Based on Refrence Flows, Information processing society journal D, Vol. 19-D, No. 4, pp. 293-298 (2012)

