The Shutdown-Alert Pattern for Online Service Replacement

Shin-Jie Lee

Computer and Network Center, National Cheng Kung University, Taiwan

jielee@mail.ncku.edu.tw

Abstract. *As a software system evolves, new versions of services may be developed to replace the old ones. In contrast to a normal object class locally run on each user's device, a service is usually deployed as a single running instance and shared among multiple users. The conventional solution is that the service provider informs the service consumers of the newly deployed service and asks them to migrate to using the new service. However, the conventional approach would be challenged regarding an unknown number of autonomous service consumers. This paper introduces a pattern, referred to as Shutdown-Alert, for dealing with the problems that may occur in the context of online service replacement. The Shutdown-Alert pattern enables on-the-fly service replacement and is capable of discovering implicit usage dependencies. Its liability and a variant for mitigating the impact are also discussed*.

Categories and Subject Descriptors

•Applied computing → Service-oriented architectures

Keywords

Service replacement, service delivery, service usage dependency, SOA

1. Introduction

A service delivery process usually involves a service provider, multiple service consumers, and a service to be delivered [1]. In a software system, a service can be a remote object [2], a web service [3], a database, or the software itself [4]. As the software system evolves, new versions of services may be developed to replace the old ones. In contrast to a normal object class locally run on each user's device, a service is usually deployed as a single running instance and shared among multiple users. Replacing a service by a new version may need to be conducted on the fly rather than through the way used with a normal objects class, i.e., distributing copies of the new version to the users and installing the copies off-line.

This paper introduces a pattern, referred to as Shutdown-Alert pattern, for dealing with the problems that may occur in the context of online service replacement. The conventional solution to the online service replacement is that the service provider informs the service consumers of the newly deployed service and asks them to migrate to using the new service. However, the conventional approach would be challenged regarding an unknown number of autonomous service consumers. The problem involves two types of service consumers: careless and uninformed users. Careless users are service consumers who reply that they have finished the migration but they did not. An uninformed user is a service consumer who is not informed of the replacement due to some reasons, such as broken connections, missing messages, and non-managed service consumers list. The Shutdown-Alert pattern enables on-the-fly service replacement and is capable of discovering implicit usage dependencies.

This paper is intended to provide a canonical form of the Shutdown-Alert pattern and could be referenced by cloud and web service designers who usually replace their services by better versions on the fly with an unknown number of autonomous service consumers. Section 2 fully describes the formulation of the pattern with discussions of its liability and a variant for mitigating the impact. The concluding remark is presented in Section 3.

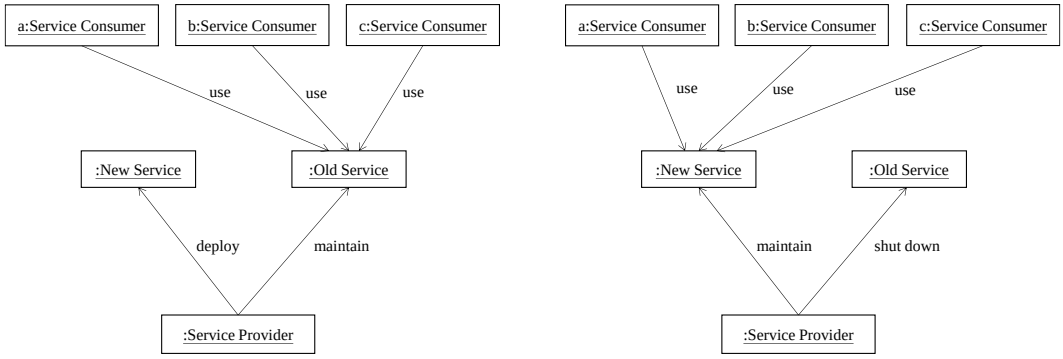
2. The Shutdown-Alert Pattern

2.1 Intent

Discover the implicit usage dependencies from the careless and uninformed service consumers to the old service, and force the service consumers to migrate to using the new service. Before permanently shutting down the old service, the old and new services co-exist and hence the service consumers have time for conducting the migrations on the fly.

2.2 Context

A system consists of a service provider and a number of service consumers (see Figure 1). A service provider or consumer may be a person, a software agent, or a system. The service provider provides and maintains a service. The service is used by the service consumers. For some reason, the service provider intends to replace the service by a new service. The service consumers have to modify their behavior of using the service due to interface changes, which may include changes of service names, input types and output types, invocation sequence of service operations, service locations, and definitions of the semantics of the input and output data. Ideally, the replacement should be conducted on the fly and the service provider has to let the service consumers have time to migrate to using the new service. After the replacement, the old service is shut down and all of the service consumers use the new service maintained by the service provider.



(a) Before replacement (b) After replacement

Figure 1. The system states before and after performing an online service replacement.

2.3 Problem

A conventional solution is that the service provider informs the service consumers of the newly deployed service and asks them to migrate to using the new service. The old service will be shut down once the service provider receives the successful-migration confirmations from all of the informed service consumers. However, the conventional approach would be challenged regarding an unknown number of autonomous service consumers (see Figure 2). The problem involves two types of service consumers: careless and uninformed users. Careless users are service consumers who reply that they has finished the migration but they did not. An uninformed user is a service consumer who is not informed of the replacement due to some reasons, such as broken connections, missing messages, and non-managed service consumers list. As a result, service consumers of these two types will still refer to the old service and fail to continue usages of the old service while the service provider shuts down the service.

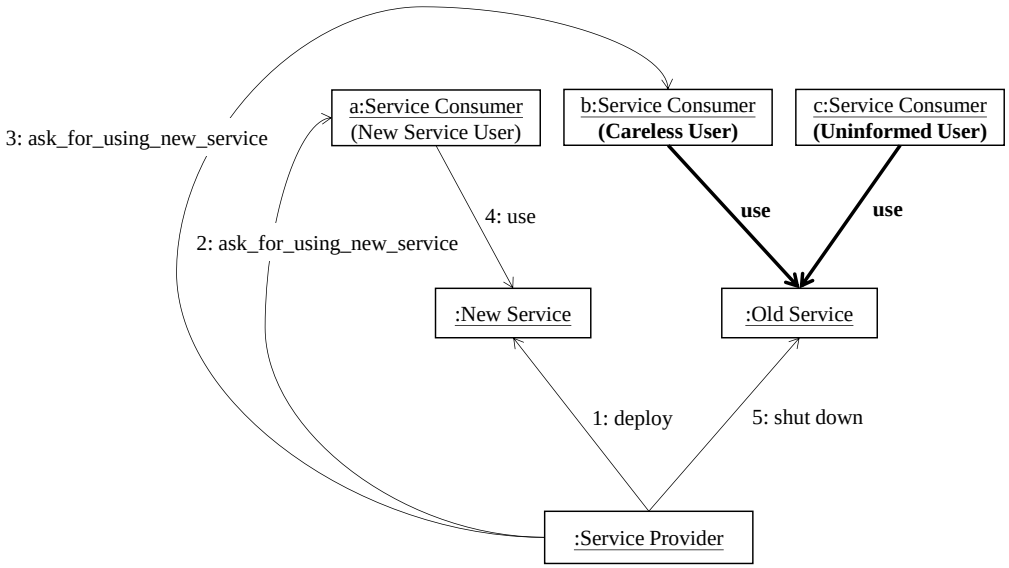


Figure 2. A conventional solution and its problem.

2.4 Solution

Figure 3 presents the UML sequence diagram for the pattern. The iterations of shutdown and restart phases are represented as a loop. In each iteration, the service provider receives an alert from a service consumer, restarts the old service and asks the service consumer for using the new service. The service consumer may then use the new service instead or still use the old

service. After that, the service provider shuts down the old service again. The loop is terminated while there is no more alert sent from a service consumer.

Figure 4 shows the operational concept of the Shutdown-Alert pattern. The pattern involves two repeatedly conducted phases: Shutdown phase and Restart phase. At the first run, the service provider shuts down the old service and waits for alerts sent by careless and uniformed users when they found the old service is unavailable. Once the service provider receives any alerts from users, the service provider at once restarts the old service and requests the users for the service usage migration. Because the users may require some time for the migration, the old service is restarted to continue the service. After the completion of the migrations, the users send successful-migration confirmations to the service provider.

Unfortunately, the number of uninformed users may be unknown and the users who reply they have completed the migrations may still be carelessly conducting the migrations. The service provider conducts a second run of the two phases. The old service is shut down again and then the service provider waits for alerts. Once the service provider receives one or more alerts from some users, the old service is restarted again and the users are asked for the migration in the same way. The two phases are conducted repeatedly until the service provider does not receive any alerts from service consumers by a specified deadline.

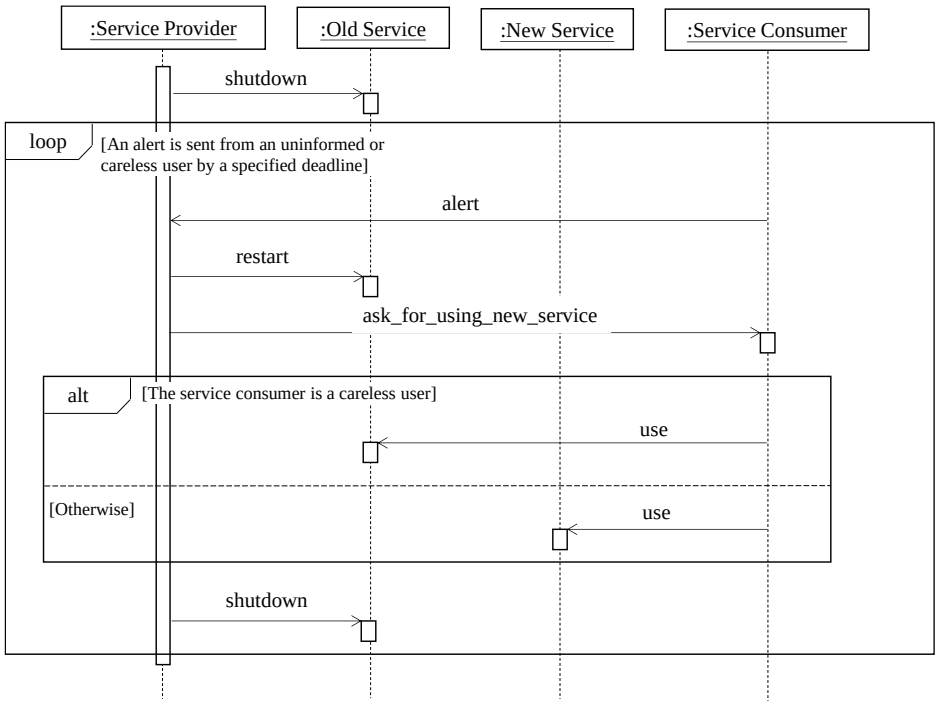
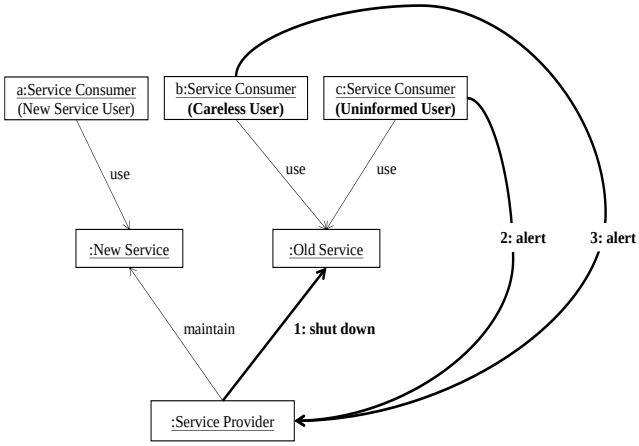
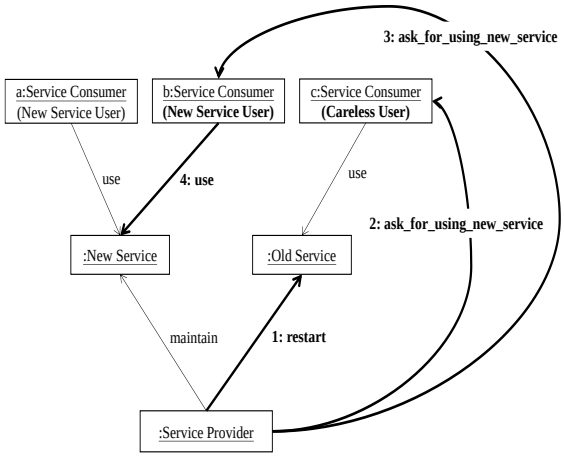


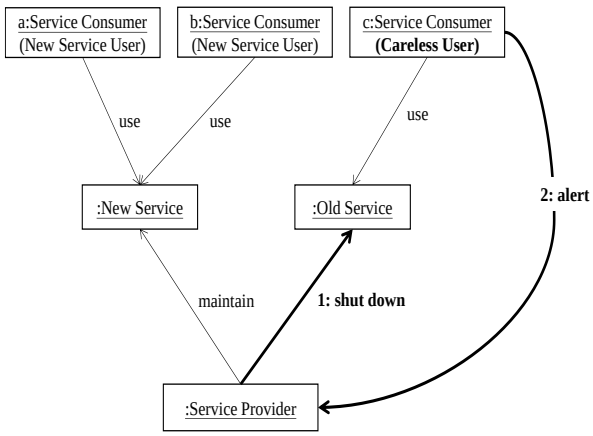
Figure 3. Sequence diagram of the Shutdown-Alert pattern.



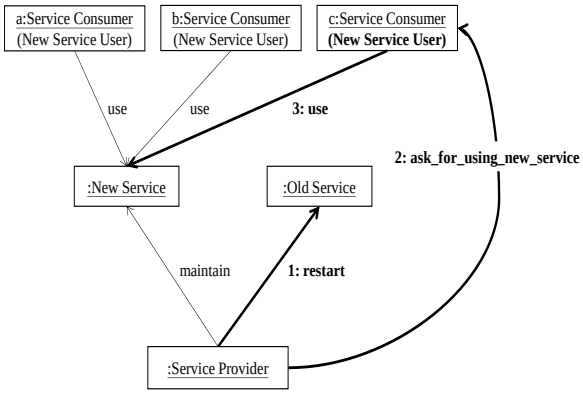
(a) Shutdown phase: 1st run



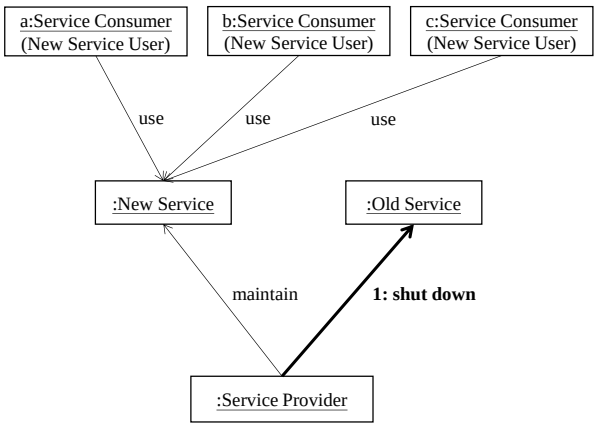
(b) Restart phase: 1st run



(c) Shutdown phase: 2nd run



(d) Restart phase: 2nd run



(e) Shutdown phase: 3rd run

Figure 4. Operational concept of the Shutdown-Alert pattern.

2.5 Consequences

The Shutdown-Alert pattern offers several benefits:

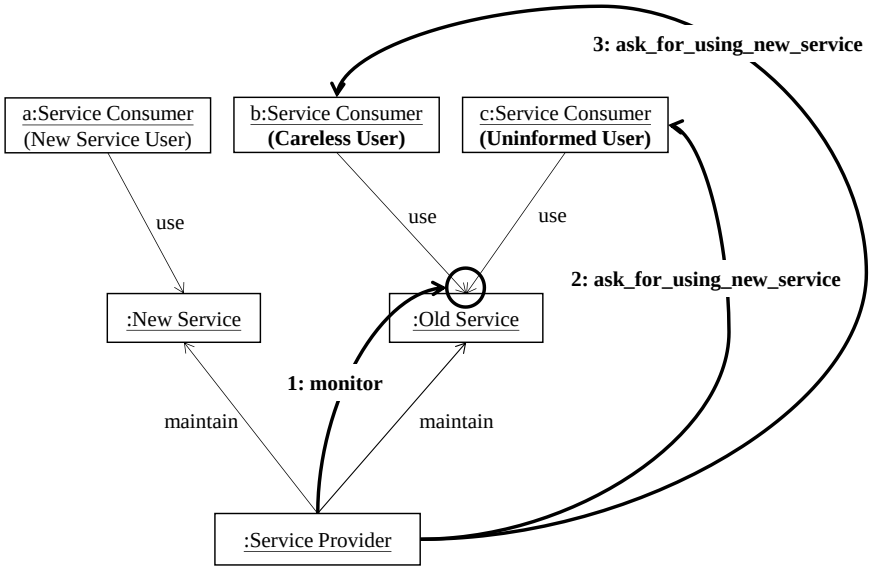
On-the-fly service replacement. The pattern forces all service consumers to migrate to using the new service and the service replacement takes place on the fly. Before permanently shutting down the old service, the old and new services co-exist and hence the service consumers have time for conducting their own migrations.

Implicit usage dependencies discovery. The pattern can discover the implicit usage dependencies from the service consumers to the old service. The dependencies are the service usages of the careless and uninformed service consumers.

The Shutdown-Alert pattern suffers from the following liability: The temporary service shutdowns may impact the functions of the careless and uninformed service consumers.

2.6 Variants

In order to mitigate the impacts caused by the temporary service shutdowns. Another phase, Monitoring, is introduced prior to the Shutdown phase in each run (see Figure 5). In this phase, the service provider monitors the usages of the old service and tries to find out the service consumers based on the tracking information. The service consumers will then be asked for the service-usage migrations. The implementations of the monitoring mechanism could vary for different applications. For example, IP addresses information could help find out the service consumers in a network-connected system.



Monitoring phase

Figure 5. Introducing the Monitoring phase prior to Shutdown phase

2.7 Know uses

The Shutdown-Alert Pattern is currently selected to be applied in our development team to replace an old table field by a new one in a database. In this application, a table field serves as a service used by over 100 software systems. The database administrator serves the role of the service provider and the software applications serve the role of service consumers. Because the software applications should always be available to the users, the table field replacement should take place on the fly. Once all relevant software systems are thoroughly inspected and the relevant dependencies are manually modified, we are going to apply the pattern to find out implicit dependencies.

In 2013, Google used this pattern to replace their Picasa Web Album cloud service with Google+ Photos [6]. They redirected the URL of Picasa Web Album (picasaweb.google.com) to the albums section of Google+ Photos, which means users were unable to access Picasa Web Album directly. From a user's perspective, Picasa Web Album was shut down. Nevertheless, on the redirected web page, a message "Click here to go back to Picasa Web Albums" with a URL (https://picasaweb.google.com/lh/myphotos?noredirect=1) was provided to let users be able to go back to the old Picasa Web interface. From the user's perspective, the old service was restarted. At that time, users would know Google+ Photos is an upgraded version of Picasa Web Albums, which would be discontinued in the future.

This pattern is sometimes used in real world scenarios. For example, a shop, say a gym, has moved to a new place. The shopkeeper starts to inform their members of the new address. However, some members are not informed due to their contact information changes. Moreover, some informed members may forget this matter and still go to the old place. The shopkeeper decides to terminate the service in the old place but to leave a shop assistant to wait for the careless or uninformed members. Once a member comes to the old place, the assistant immediately redirects her/him to the new place. In this situation, because the members are immediately redirected to the new place to enjoy the service in the new place, the service in the old place is not restarted. Nevertheless, the service provider (shopkeeper) is still able to discover the implicit dependencies and makes the replacement on the fly. The shop in the old place will permanently be closed once no more members come to the old place by a specified deadline, say one month later.

2.8 See also

The service consumers can apply Adapter pattern [5] to better manage the old and new service usages if the interfaces of the two versions are incompatible.

3. Conclusion

This paper presents a pattern, Shutdown-Alert, for solving the problems that may occur in the context of online service replacement. The Shutdown-Alert pattern offers the following benefits: (1) the pattern can force all service consumers migrate to using the new service and the service replacement takes place on the fly. (2) The pattern can discover the implicit usage dependencies between the careless and uninformed service consumers and the old service. However, the temporary service shutdowns may impact the functions of the careless and uninformed service consumers. In order to mitigate the impacts caused by the temporary service shutdowns, we introduce another phase, Monitoring, prior to the Shutdown phase for finding out the service consumers of some implicit dependencies through monitoring the usages of the old service. The service consumers will then be asked for the service-usage migrations before entering the Shutdown phase.

Acknowledgements

This research is sponsored by Ministry of Science and Technology under the grant 103-2221-E-006-218 in Taiwan.

References

[1] Lee, J., Ma, S.-P., and Liu, A. 2011. "Service Life Cycle Tools and Technologies: Methods, Trends and Advances," IGI Global.

[2] Java Remote Method Invocation (RMI). https://docs.oracle.com/javase/tutorial/rmi/.

[3] Erl, T. 2004. “Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services,” Prentice Hall.

[4] Papazoglou, M. P. 2003, “Service-Oriented Computing: Concepts, Characteristics and Directions,” Proceedings of the Fourth International Conference on Web Information Systems Engineering.

[5] Gamma, E., Helm, R., Johnson, R., and Vlissides, J. 1994. “Design Patterns: Elements of Reusable Object-Oriented Software,” Addison-Wesley Professional.

[6] http://googlesystem.blogspot.tw/2013/03/picasa-web-albums-redirects-to-google.html