Deployment Pattern

Youngsu Son[1],JiwonKim[2], DongukKim[3], Jinho Jang[4]

Samsung Electronics [1,2,3], Hanyang University[4]

alroad.son[1], jiwon.ss.kim[2], dude.kim[3] @samsung.net, undersense3538@gmail.com[4]

Abstract

Software that is deployed in the market needs to be continuously evolved. In order to be long-running, it is necessary to accept client's requirements. Software's Deployment has become essential for software's evolution. Therefore many vendors provide the deployment application [1],[2].

Keywords

Deployment, Update, Upgrade, Push, Polling, Half-Push/Half-Polling

1. Introduction

Software must reflect the changes of the real world. If not, software will decay over time. No matter how well-made, software must always be revised based on client's requirements changes and the needs for new services. Due to this, many applications currently support deployment services in order to improve the customer satisfaction. Through this deployment service, the clients can use the latest services without installing a new program on every time. In this paper, we discuss an important pattern for deployment services which is applicable to various environments.

2. Background & Example

Background

In the Client-Server Model, Push or Polling are general methods to notify the updates at the server to the clients. Push refers to actively updating clients by the server. The server requests access to the clients and pushes data to them. The advantage of this approach is that you can fully utilize the server resources within the limit of network bandwidth. However, this method assumes that the client will always be alive. And it also requires the server to keep information about clients.

The figure 2 shows how Push method works. When some event (ex; upgrade) occurs and the server needs to connect to the clients, the server can control the workload autonomously in consideration of its capacity and the bandwidth. Numbers in the picture represent a sequence of operations.

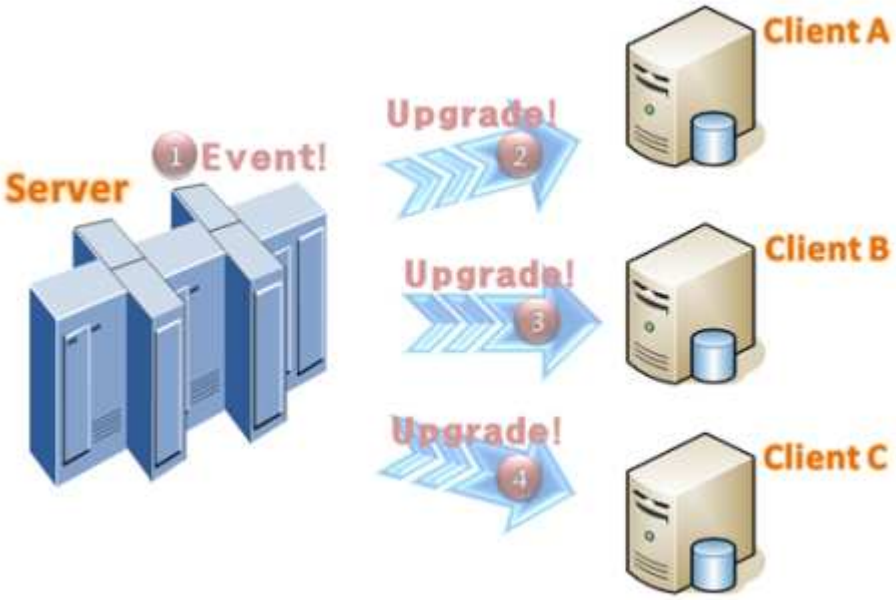

Figure 1. Push Upgrade Method

On the other hand, when using polling method, a client has the initiative. The client requests data transmission from the server. Because the server is more stable and using this method guarantees the server is alive, the chance of the problem is very low.

However, In Polling method, as opposed to Push, the client checks repeatedly whether any event has occurred or not. And all of clients would attempt to access to the server at the same time in the worst case because there is no fixed order. The figure below shows such case.

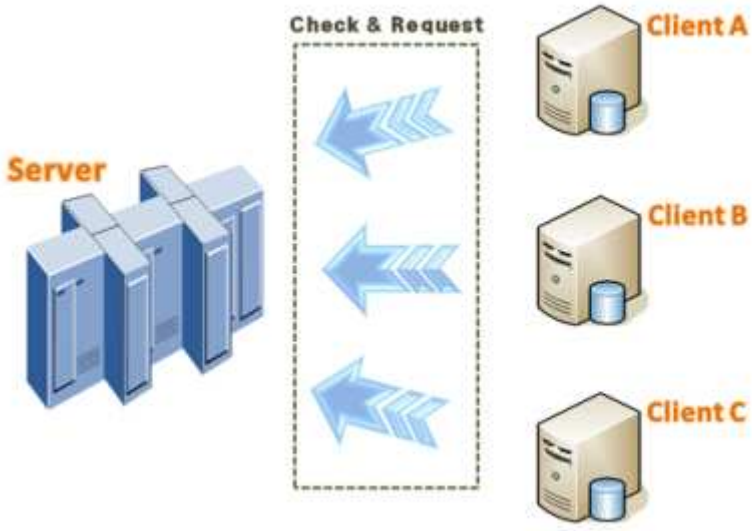

Figure 2. Poll Upgrade Method

Example

For example, let's assume that we are developing an office automation system for buildings located closely together in a downtown. There are various types of devices in the system and they are connected to a wired or wireless network. In addition, requirement is to keep the software in each device up-to-date. The server will provide the latest software via client-server model.

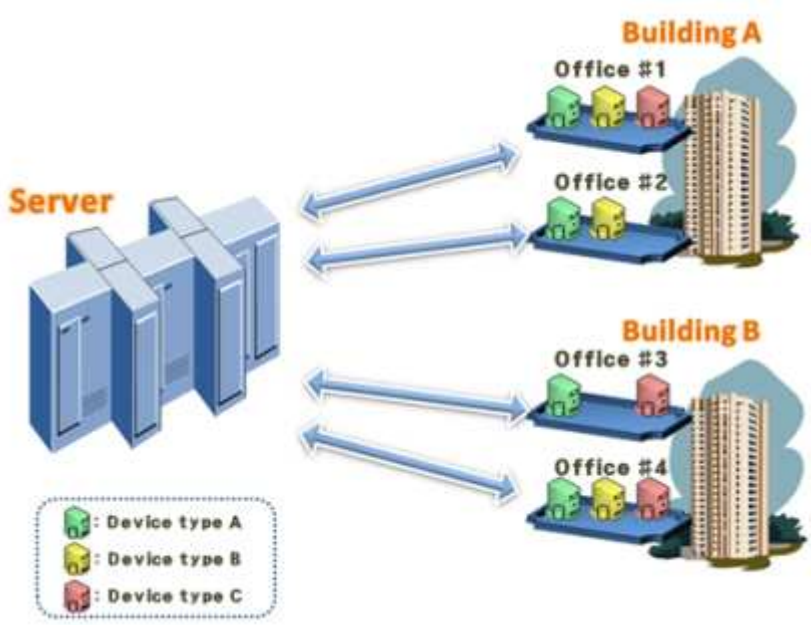

Figure 3. Building system which consists of various types of devices

Some devices should be updated emergently. For example, when Security system's server or module is updated, emergent update is quite needed. However, printer driver update in office is not urgent.

3. Context

When software in use is distributed, there are many considerations.

1. A case that each client group uses different program version.
2. A case that each client group needs to be upgraded in different timing.
3. A case that each client group needs to use different distribution strategies.
   (Ex. Every 3 hour, every week)

Let's think of management system for businesses. They can have different service version and distribution strategies according to each client group.

Furthermore, some of companies might want to upgrade in not busy time, and not want to

download the whole program every time new version released but only download changed module to avoid server's overload. Clients should pay much money for download not impossibly.

4. Problem

- Are different versions of upgrade for each client group possible?
- Are different strategies of upgrade for each client group possible?
- Is the upgrade in time clients want possible?

5. Forces

- Distribution strategies which satisfies client's needs.
- Upgrade strategies of least change for new version.

6. Solution.

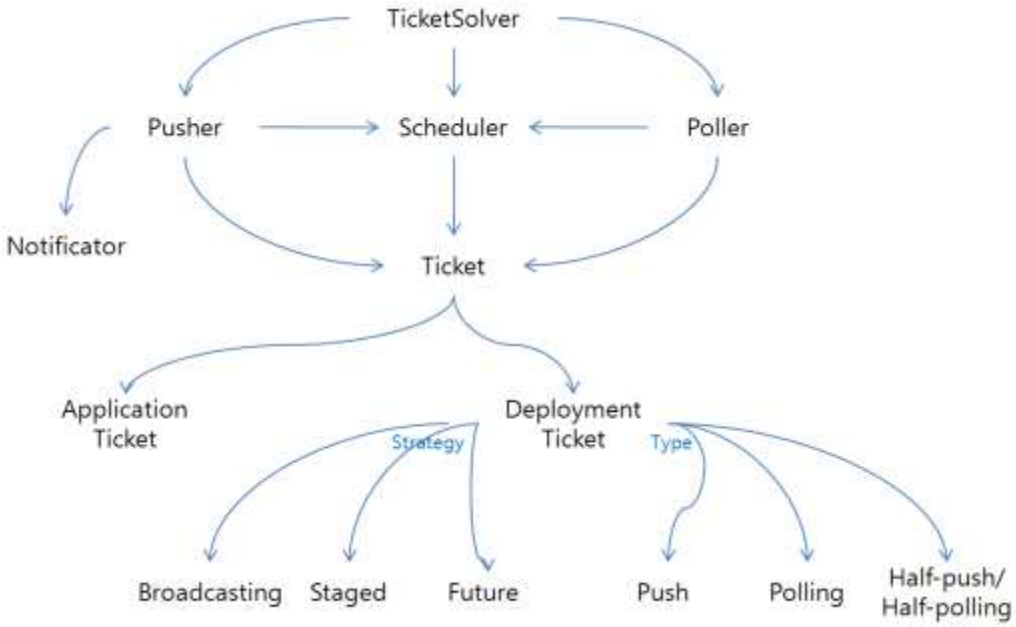

Figure 4. Deployment Pattern Map

This pattern is divided into Ticket and Deployment Component (Pusher, Poller, and Scheduler). Ticket describes information of file/module/component to deploy and deployment strategy and type and so on. Deployment Component using ticket information supports various deployment ways by dynamic channel composition

7. Structure

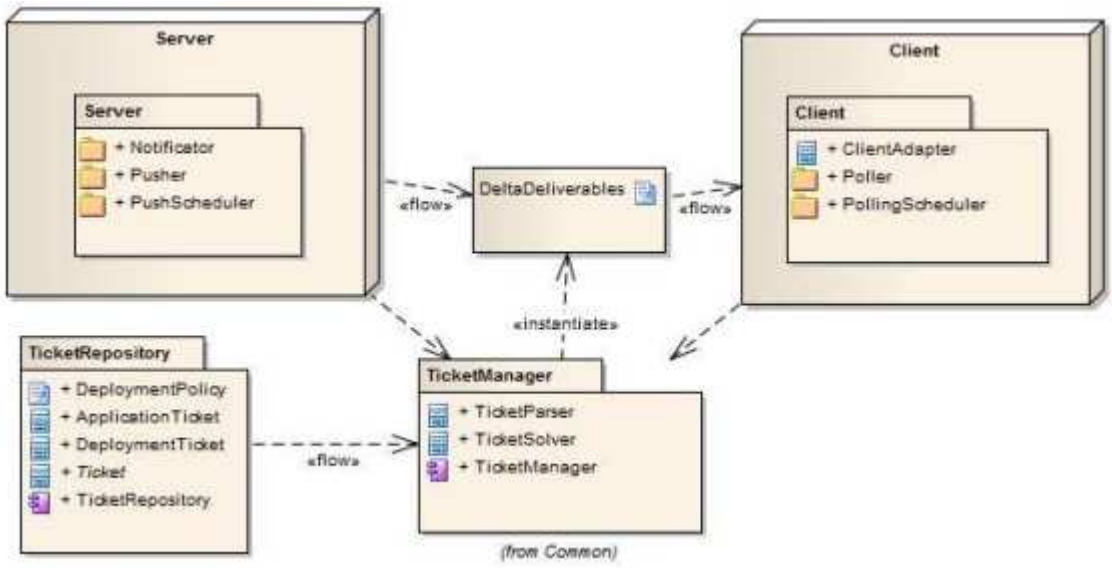

Figure 5. Overview Structure of Deployment Pattern

7.1 Ticket

Upgrade Ticket can be used for software distribution with various distribution strategies according to each client group. It prevents server's upgrade overload by managing files and user's setting information and provides optimized and the reduced upgrade based on system environment. Upgrade Ticket largely consists of Application Ticket and Deployment Ticket.

- **Application Ticket**

  Application Ticket includes Software information which will be distributed such as dependency of application and assembly, file, and permission. The ticket manages file information which upgrades and download from distribution server not the whole module, but only changed module.

- **Deployment Ticket**

  Deployment Ticket includes a distribution strategy (Broadcasting, Staged, Future) and a type of update (Push - Publisher/Subscriber [7], Polling, Half-Push/Half-Polling [11], [13]).

7.2 Scheduler

There are two schedulers, Push scheduler and Polling scheduler in Upgrade system using Ticket. Each scheduler might use 3 types of ways for upgrade.

- Broadcasting: Deploys it to all the clients at the same time.
- Staged: Deploys it to each client group.
- Future: Makes a reservation and deploys it at the reserved time.

7.3 Ticket Solver

According to Deployment strategy such as Push, Polling, and Half-Push/Half-Polling, it dynamically deploys and redeploys correspondent components runtime. It is based on Composite Message [4] and Pipe&Filter [7].

7.4 Pusher

It is used for simultaneous update of all clients. (It is used in Half-Push/Half-Polling afterwards referred and just forwards scheduling information) Clients who want to get updated register their own version information and access information to Pusher. The connection is 1:1 and all the clients are registered in a list (Publisher-Subscriber) [7] and Pusher distributes modules to

Clients as per their Policies.

7.5 Poller

All clients that needs to simultaneously ensure the consistency and systems that don't need urgent upgrade use Polling Clients that clients spontaneously download modules. Poller takes a role of a receptionist and PollingScheduler decides when modules are downloaded

7.6 Notificator

It is used when Push Update used. In case of Push Upgrade, it passes a Channel made by Ticket Solver. The Channel is used for comparison of version information and extraction of modules which will be updated. In other words, DeltaDelieverables are derived. These modules are used to notify to Pollers.

8. Dynamics

8.1 Dynamic Channel Composition by Ticket information

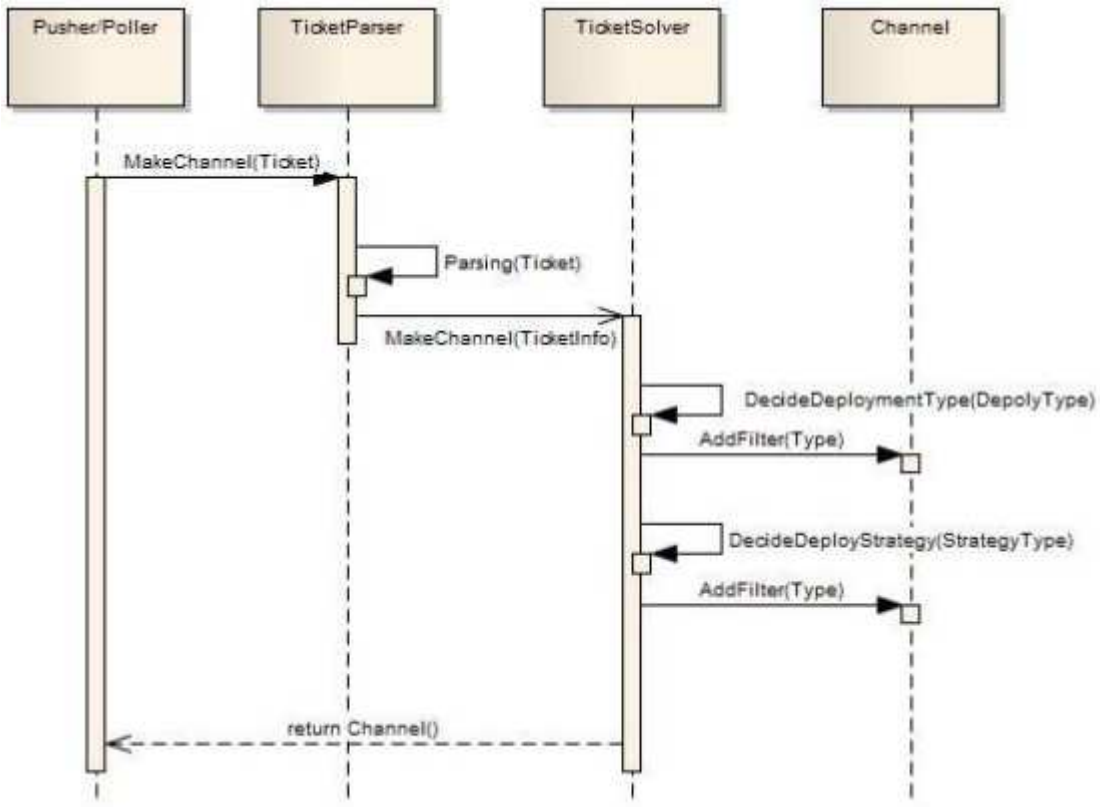

Figure 6. Dynamic Channel Composition

1. When a Ticket added to TicketRepository, ticket information is delivered to a correspondent object based on Deployment type.

A. In Push, Pusher notifies Ticket information added. In this case, Pusher dynamically makes a transport channel.

B. In Polling, Poller gets notified information that Ticket information is added. In this case, only Poller dynamically makes a transport channel.

C. In Half-Push/Half-Polling, both Pusher and Poller get update information. In this case, both Poller and Pusher dynamically make a transport channel.

2. TicketParser analyses Ticket information about how to distribute (push, polling, half-push/half-polling), and what strategy to use (broadcasting, stage, future).

3. This information is delivered to Solver (Distributer) and then Filter is made according to

Distribution method and Strategy. As per a strategy decided such as Push and Polling, a channel is created by dynamically composed filters.

A. By checking Application Ticket, difference between a previously distributed program and latest one is confirmed and a list of files for upgrade are created.

B. Components are distributed as per deployment (Broadcasting, Stage, Future) policy.

8.2 Push Deployment

1. Clients to be updated register themselves to Pusher.

2. A connection is managed as 1:1 and client list is registered in server.

3. Each client' job sequence is controlled by Pusher's Scheduler, and Pusher is in charge of distribution of one client.

4. A module's information for distribution is described in Application Ticket and Push for one client is done as per this information.

A. In case of Broadcasting, the application modules are distributed to all clients already registered.
B. In case of Staged, the application modules are distributed to correspondent staged clients.
C. In case of Future, the application modules are distributed to clients when the reserved time comes.

8.3 Polling Deployment

1. Sends Application Ticket information in upgrade request program or gets a ticket from Ticket Manager in remote.

2. Dynamically makes a channel as per policy after parsing Ticket information by Ticket Parser.

3. Downloads modules from Server.

A. In case of Broadcasting, instantly downloads modules from server.

B. In case of Staged, downloads only correspondent staged modules.

C. In case of Future, downloads modules when the reserved time comes.

8.4 Half-Push/Half-Polling Deployment

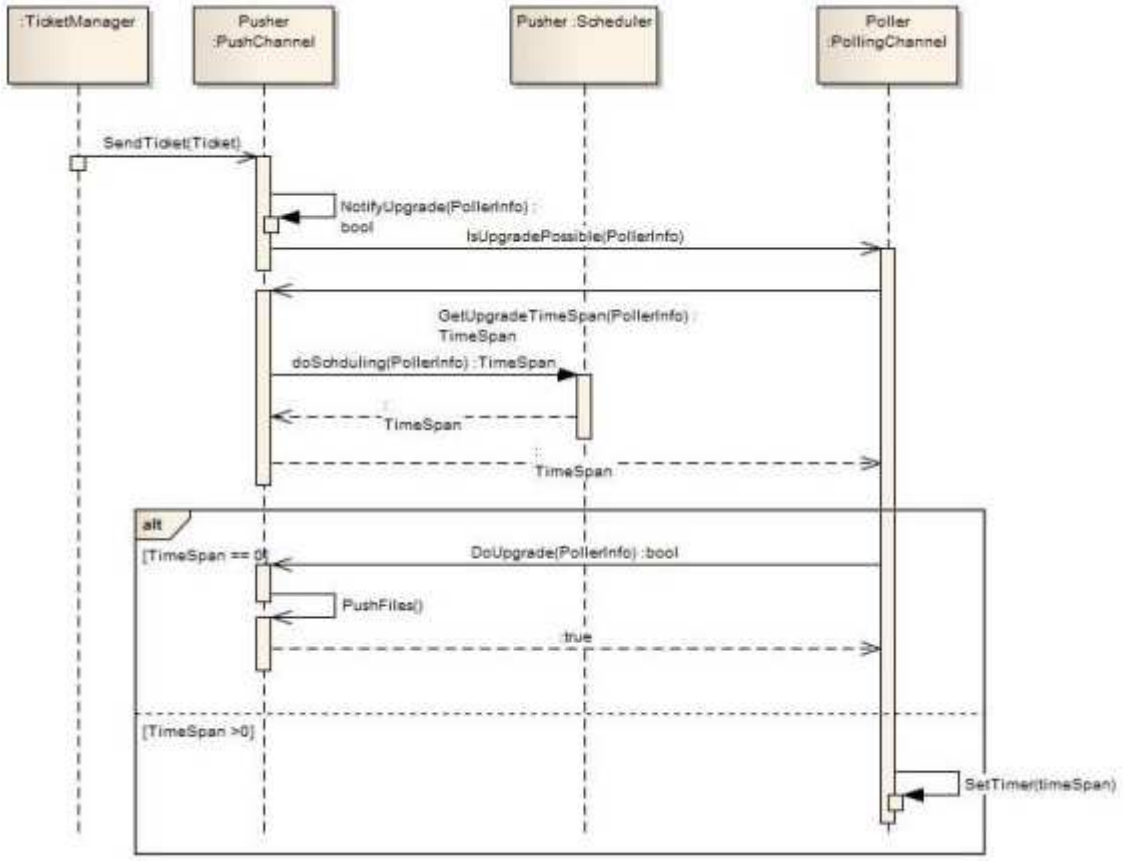

Figure 7. Half-Push/Half-Polling

1. When Ticket is delivered from Ticketmanager to TicketSolver, a channel is dynamically created as per Section 6.2.1
2. PusherChannel has information of Pollers to upgrade and asks poller if currently upgradable.
3. PushChannel extracts a list of Pollers to be urgently upgraded based on Deployment policy. For example, broadcasting should update all pollers and future should only delegate clients which reached reserved time to scheduler.
4. Scheduler sends Timespan about when to upgrade to current Pollers.
5. Pollers request the upgrade to the server when the reserved time comes and download files.

9. Known Use and Variants

Known Uses

- ZeroInstall [15]

ZeroInstall is a decentralised cross-distribution software installation system. Other features include full support for shared libraries, sharing between users, and integration with native platform package managers.

ZeroInstall provide Solver. Solver creates architecture in runtime as per Deployment Policy. Also it offers Application and Deployment Ticket as Feed and Policy objects mentioned in this paper.

- Clickonce [14]

The core principle of ClickOnce is to bring the ease of deployment of web applications to the Windows user. In addition, ClickOnce aims to solve three other problems with conventional deployment models: the difficulty in updating a deployed application, the impact of an application to the user's computer, and the need for administrator permissions to install applications.

A ClickOnce deployment is controlled through the use of two XMLmanifestfiles: a deployment manifest and an application manifest. The manifests are in the same XML format as the Side-by-SideAssembly implementation. The deployment manifest (*.applicationfile) describes the deployment model: the current version, update behavior, publisher identity along with digital signature; this manifest is intended to be authored by administrators who handle deployment.

The application manifest(*.exe.manifestfile) describes the application assemblies, dependent libraries and lists permissions required by the application. This file is intended to beauthored by the application developer .In order to launch a ClickOnce application, a user clicks on its deployment manifest file.

- FF Clickonce [16]

This, a Clickonce .NET's variant, is created to distribute the latest version of Firefox extension by Firefox Community

Variant.

-OMG CORBA Event Service [1]

The Event service of RealTime CORBA it is not for upgrade, but the Event Channel method that replaces push method with pull (polling) method as data transmission method. In order for various message deliver, this paper suggest not only Push, Pull(Polling) but a variety of supplier and consumer such as passive pull supplier - active pull consumer.

10. Consequence

The advantages of this pattern include:

- Dynamic deployment components and supports various upgrade methods.
- Being able to upgrade specific selected clients by grouping.

Possible disadvantages are:

- necessity of maintenance due to complexity than simple update method.
- necessity of many distribution components (Client and Ticket Manager as well as Server)
- It is difficult to apply to a system such as P2P, where server and client information changes frequently.

11. Related Patterns

Publisher-Subscriber [7]
Also called the Observer pattern, it is used to synchronize the information between two components-Publisher and Subscriber. Copying the database from Publisher to Subscriber can be a typical example. It is used when the pattern encounters a non-stop talker object, in other words, when overload occurs because of non-stop polling.

Composite Message [4]
This pattern is used for marshaling/un-marshaling data, extending and adding messages you want to transfer while passing through layers. It is also used to create a transmission protocol for each device (Poller) in environments heterogeneous to the Half-Push/Half-Polling pattern. It is used in various distributed middleware.

Pipe & Filter [7]
This pattern is used when adding or filtering messages you want to transmit flexibly according to the circumstance, used internally in the aforementioned Composite Message. It is also used to filter unwanted data when building an Event Channel.

Half-Push/Half-Polling [13]
Push causes much less load because push can upgrade specific client but there is cumbersome monitoring to keep stopped clients on latest version. The Half-Push/Half-Polling pattern mixes these two different ways, keeping their advantages, eliminating their disadvantages.

12. References

[1] Timothy H.Harrison, david L. Levine, and Douglas C. Schmidt, "The Design and Performance of a Real-time CORBA Event Service," Proceedings of OOPSLA '97, Atlanta, Georgia, October, 1997

[2] Robert S. Hanmer, "WatchDog", Patterns for Fault Tolerant Software, WILEY, 2007

[3] James C. Hu, Douglas Schmidt, "JAWS: A Framework for High-Performance Web Servers", Domain-Specific Application Frameworks: Frameworks Experience By Industry, John Wiley & Sons, October, 1999.

[4] Aamond Sane, Roy Campbell, "Composite Messages: A Structural Pattern for Communication between Components", OOPSLA' 95 Workshop on Design Patterns for Concurrent, Distributed, and Parallel Object-Oriented Systems, 1995.

[5] Martin Fowler, Kent Beck, John Brant, William Opdyke, Don Roberts, "Refactoring : Improving the Design of Existing Code", Addison-Wesley Professional , 1999

[6] Douglas C. Schmidt, Michael Stal, Hans Rohert, and Frank Buschmann, "Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects", WILEY, 2000.

[7] Frank Buschmann , Regine Meunier , Hans Rohnert , Peter Sommerlad , Michael Stal, "Pattern-Oriented Software Architecture Volume 1: A System of Patterns", WILEY, 1996

[9] F. Cristian, "Probabilistic Clock Synchronization, Distributed Computing, vol. 3.

[10] R. Gusella, S. Zatti: "The Accuracy of the Clock Synchronization Achieved by TEMPO in Berkeley UNIX 4.3 BSD", IEEE Transactions on Software Engineering, Vol. 15, July 1989

[11] Jamadagni, Satish., Umesh M.N**.,"**A PUSH download architecture for software defined radios**",** 2000 IEEE international symposium on personal wireless communication

[12]SameerAjmani,BarbaraLiskov,LiubaShrira,"SchedulingandSimulation:HowtoUpgradeDistributedSystems"

[13] Youngsu Son, Sangwon Ko, Jinho Jang, Hyukjoon Lee, Jemon Jeon, Jungsun Kim, "Half-Push/Half-Polling", 2009, Proceedings of the 16th Conference on Pattern Languages of Programs

[14] "ClickOnce - Microsoft Deployment Application" , http://en.wikipedia.org/wiki/ClickOnce

[15] "ZeroInstall – Python Deployment Application" http://en.wikipedia.org/wiki/Zero_Install

[16] "FFClickOnce – A Firefox extension for running .NET ClickOnce applications" , http://www.softwarepunk.com/ffclickonce/

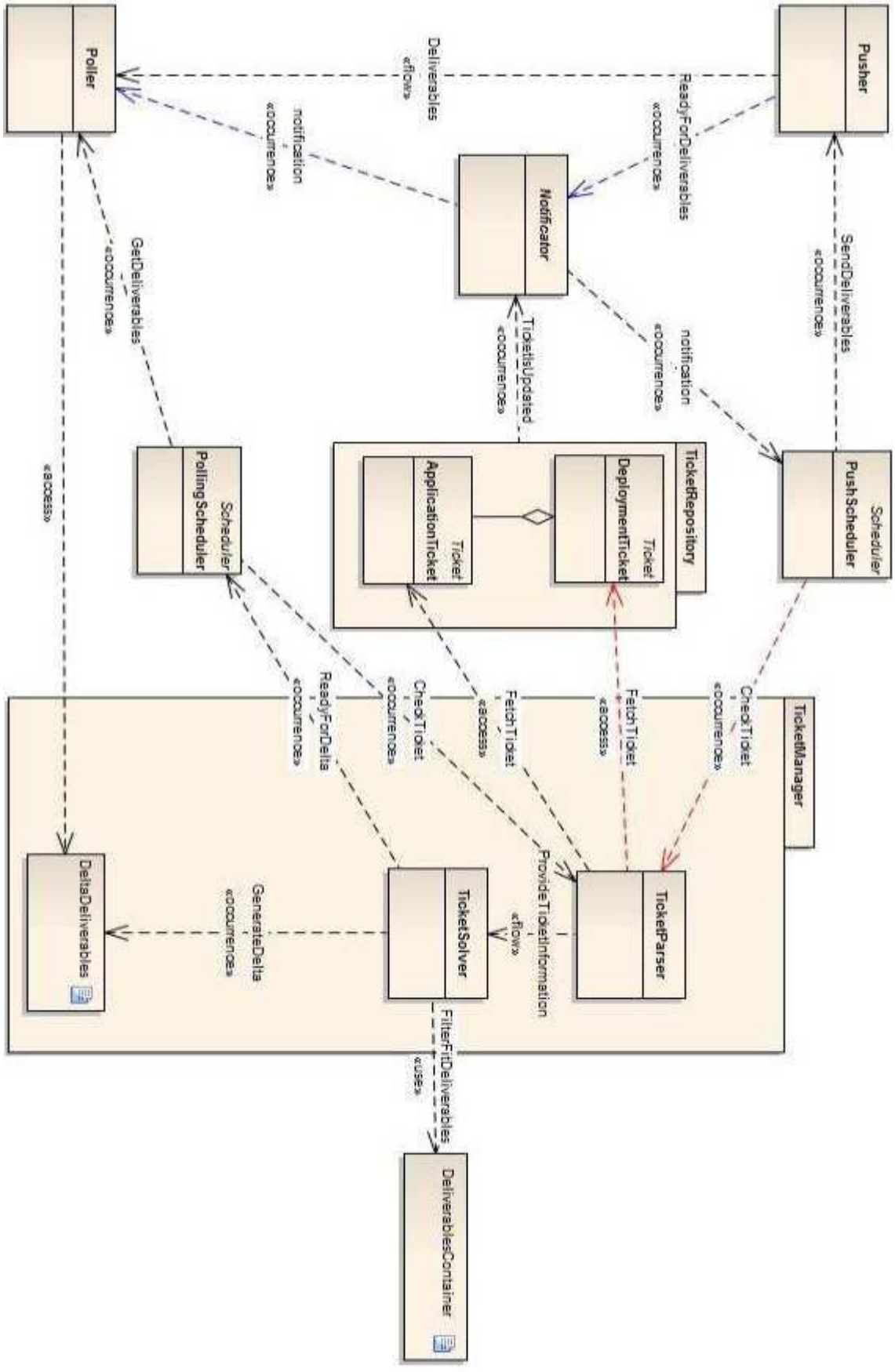

Figure 8. Component Diagram of Deployment Pattern