# Pattern Services for Service-Oriented Systems

Eltaher El-Shanta, Dr. Weichang Du

Faculty of Computer Science, University of New Brunswick

UNB EST. 1785 UNIVERSITY OF NEW BRUNSWICK
Computer Science
Fredericton

## Introduction

The goal of patterns in the software field is to create a body of literature to help software developers resolve recurring problems encountered while designing and developing software systems. While patterns enable reuse of abstract design and architecture knowledge, abstractions documented as patterns do not directly yield reusable code (Schmidt, 1997).

In the effort to try to find a solution to the problem of the lack of readily implemented reusable software patterns that can be used to build different software systems, our research aims to achieve the following objectives:

1. Propose Methodologies for implementing Patterns as Services.
2. Convert and apply non-SOA based patterns to Service-Oriented Systems.
3. Design and implement a Pattern as a Service (PaaS) software system. This system functions as a platform for integrating pattern services with service-oriented applications, configuring and deploying those pattern-based service-oriented applications.

## Application Example

The main objective of this research is to put together a general methodology for converting software patterns from different fields and of different granularity levels into pattern services. We adopt a step by step approach of showing how a software design pattern is gradually converted into a pattern service. We will use a stock market data inquiry application as an application example. Figure 1 shows a simplified overview of the example stock data inquiry application.

In the example in Figure 1, the stock data manager component pulls data from Google stock ticker service in a periodic way. Stock data is then dispatched to two stock data displayers. The first one displays the data in a form of chart, while the second displays it as a table.

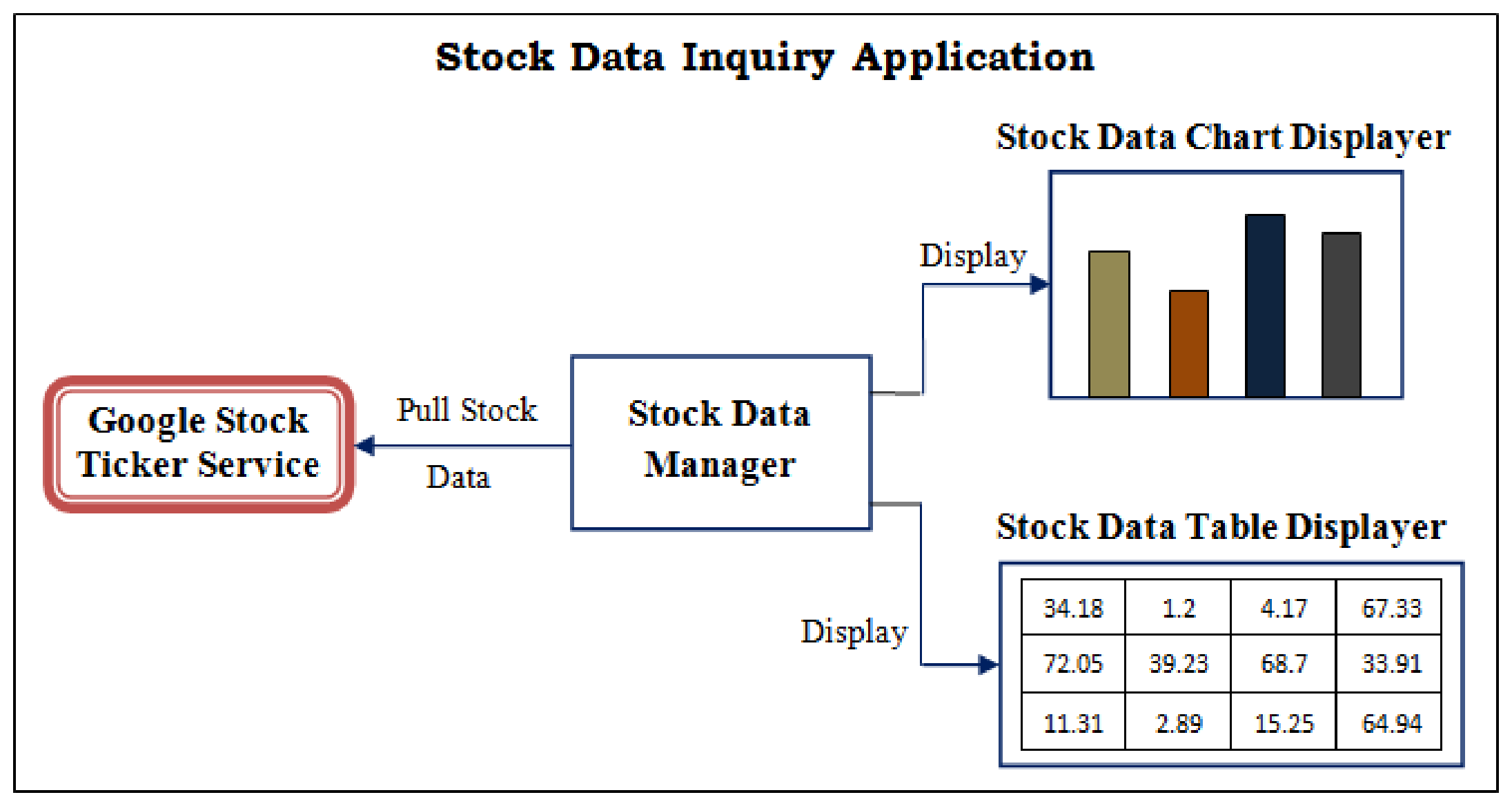

Figure 1: A simplified general view of a stock data inquiry application

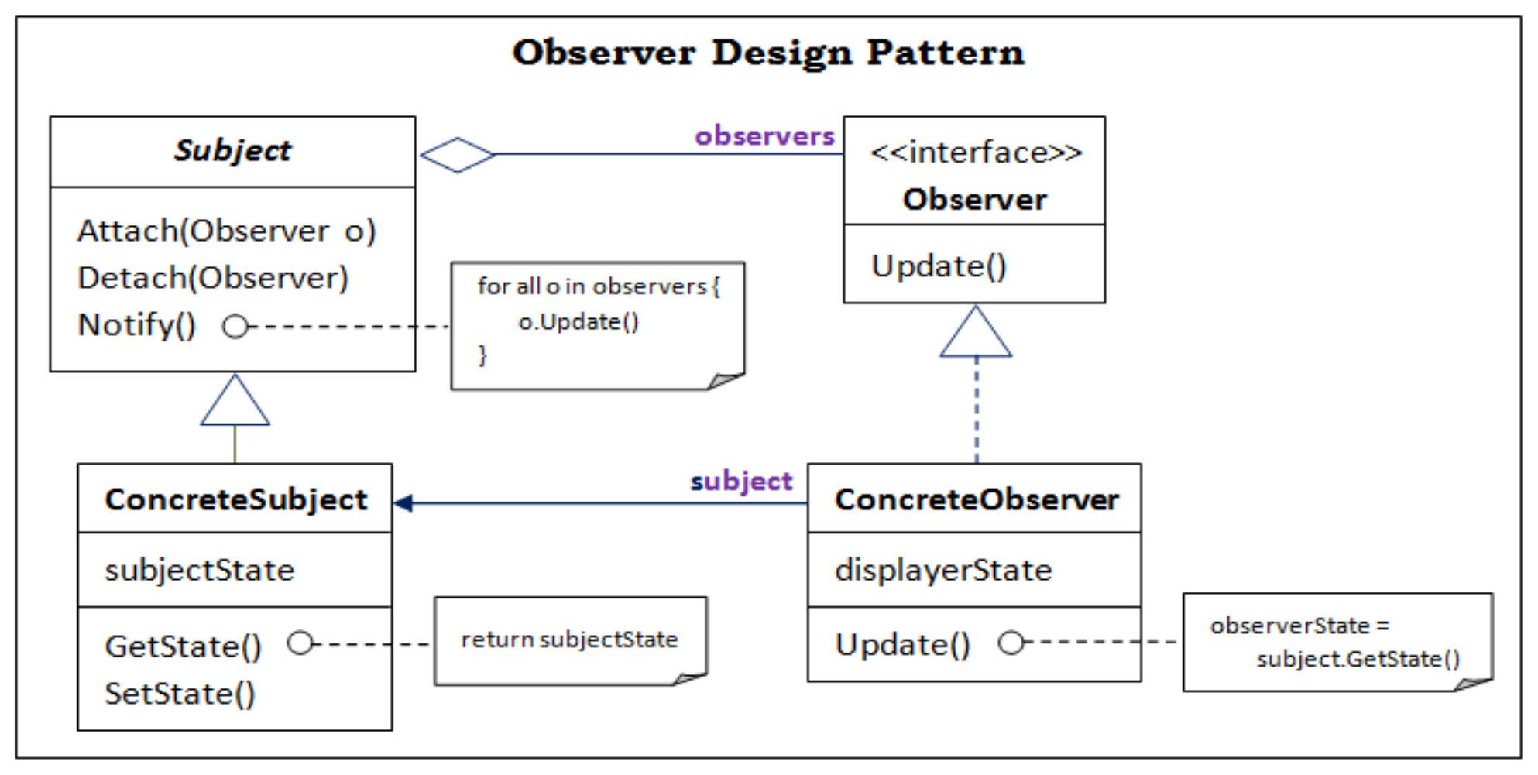

Figure 2: Observer design pattern
(Gamma, Helm, Johnson, & Vlissides, 1995)

The design and implementation of the stock data manager and the stock data displayers are good fit for the observer design pattern. The observer design pattern – Figure 2 - is usually used whenever the state of one or more objects depend on others'.

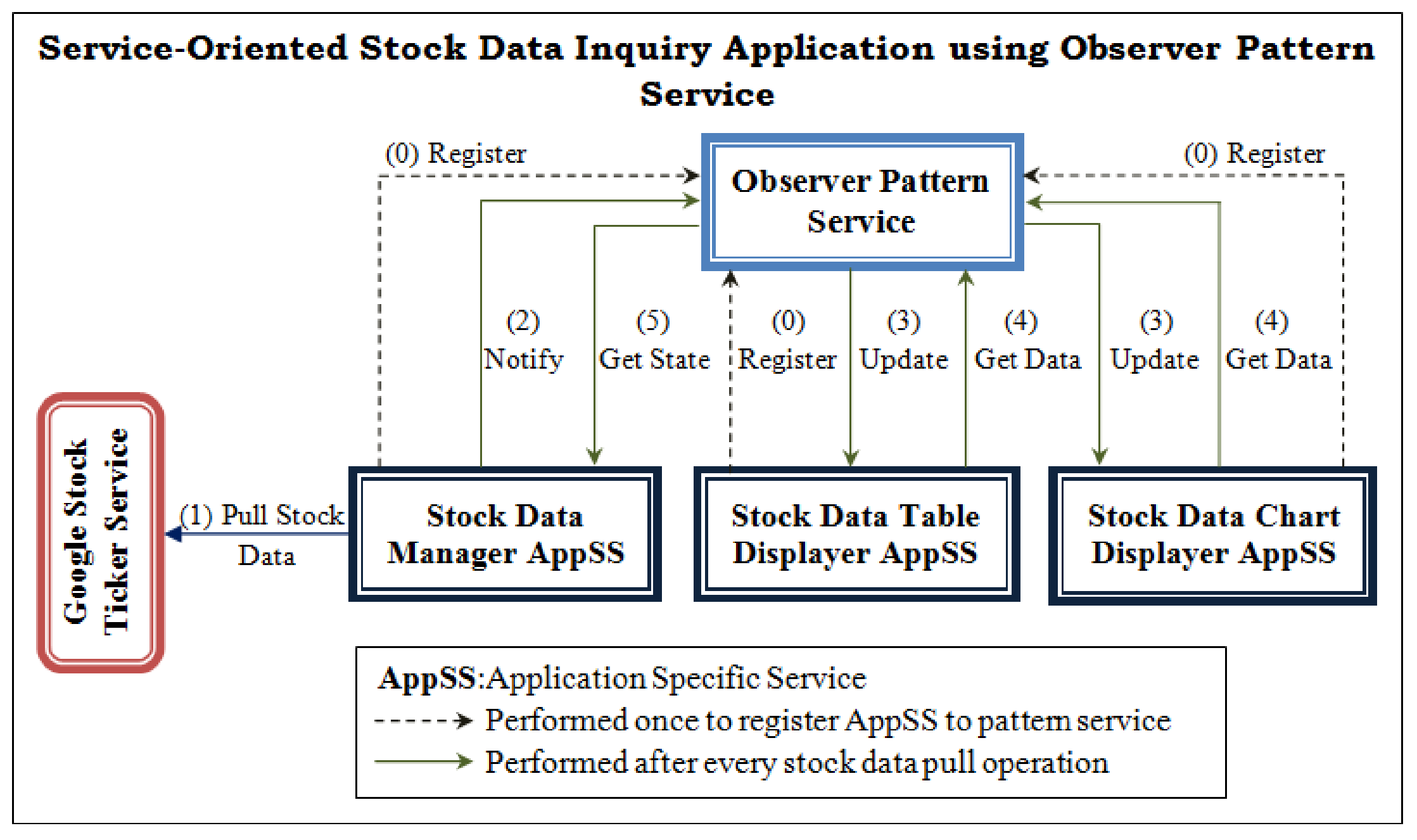
Service-Oriented Stock Data Inquiry Application using Observer Pattern Service
(0) Register
Observer Pattern Service
(0) Register
(2) Notify
(5) Get State
(0) Register
(3) Update
(4) Get Data
(3) Update
(4) Get Data
Google Stock Ticker Service
(1) Pull Stock Data
Stock Data Manager AppSS
Stock Data Table Displayer AppSS
Stock Data Chart Displayer AppSS
AppSS:Application Specific Service
Performed once to register AppSS to pattern service
Performed after every stock data pull operation

Figure 3: An overview of a service-oriented stock data inquiry application using observer pattern service

Figure 3 shows the example service-oriented system implemented using the observer pattern service. The first step in converting the observer design pattern into a pattern service is to implement the generic classes in the abstract layer and their relationships and interaction rules as the major part of a generic pattern service. This part will enforce all pattern rules and will coordinate all the interactions between its classes and objects. Since the pattern's concrete classes are application specific, we leave their implementation to the application developer, who will be using the observer pattern service in his application. We call the services implemented by the application developer - Application Specific Services (AppSS). AppSS are then registered with the generic pattern service to configure it for use with that specific application.

As can be seen from the diagram in Figure 4, the generic observer pattern service contains the implemented generic logic of the observer design pattern. In addition to that, two registration lists are added to the it. One of these lists will store service references to the subject component of the pattern service, which is – in our case- the stock data manager. The second registration list will hold service references to the observer components of the pattern service, which – in our example - are the stock data displayers. Note that the generic observer pattern service provides the interface and implementation for AppSS registration.

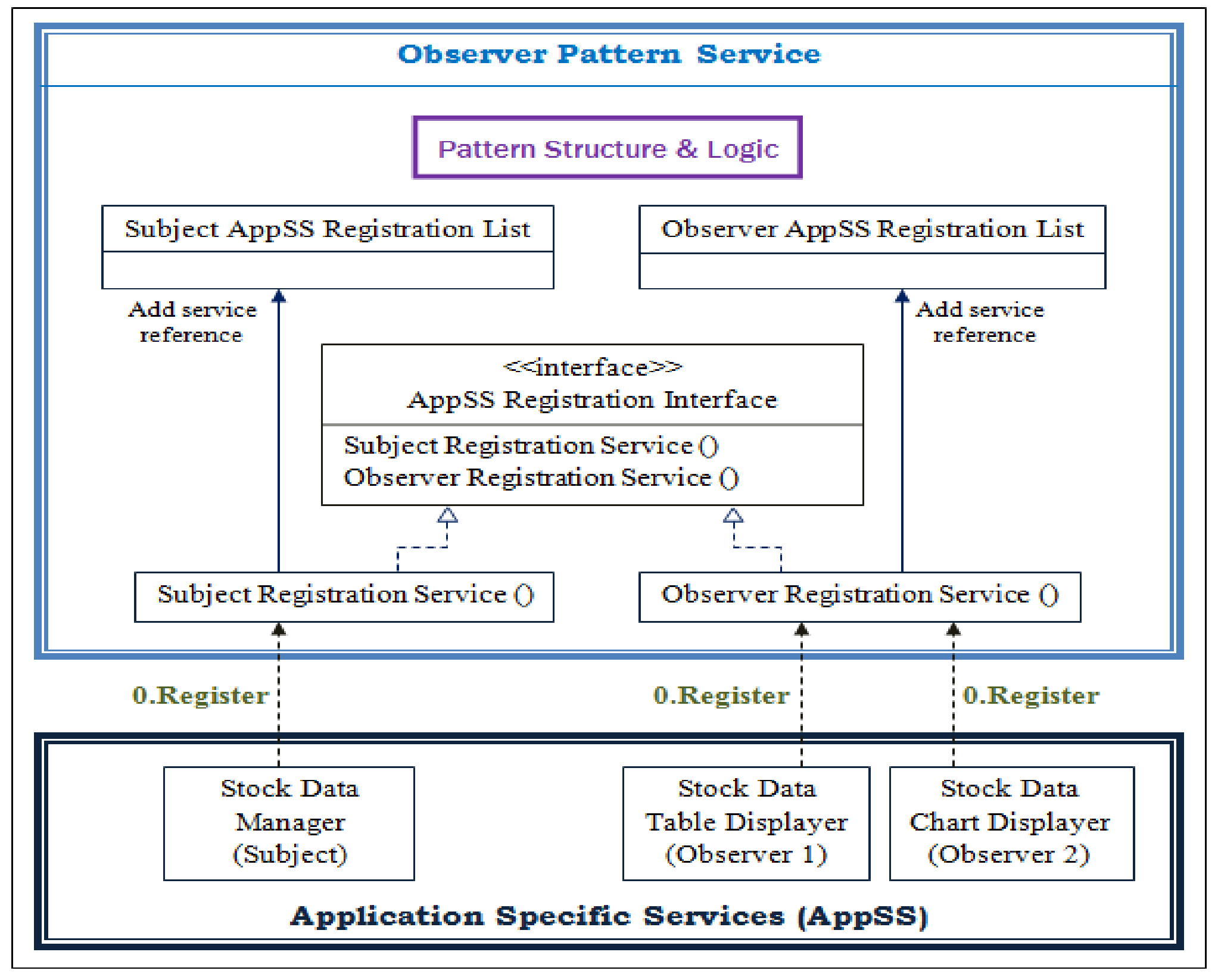
Observer Pattern Service
Pattern Structure & Logic
Subject AppSS Registration List
Observer AppSS Registration List
Add service reference
Add service reference
<<interface>>
AppSS Registration Interface
Subject Registration Service ()
Observer Registration Service ()
Subject Registration Service ()
Observer Registration Service ()
0.Register
0.Register
0.Register
Stock Data Manager (Subject)
Stock Data Table Displayer (Observer 1)
Stock Data Chart Displayer (Observer 2)
Application Specific Services (AppSS)

Figure 4: AppSS to observer pattern service registration components and process

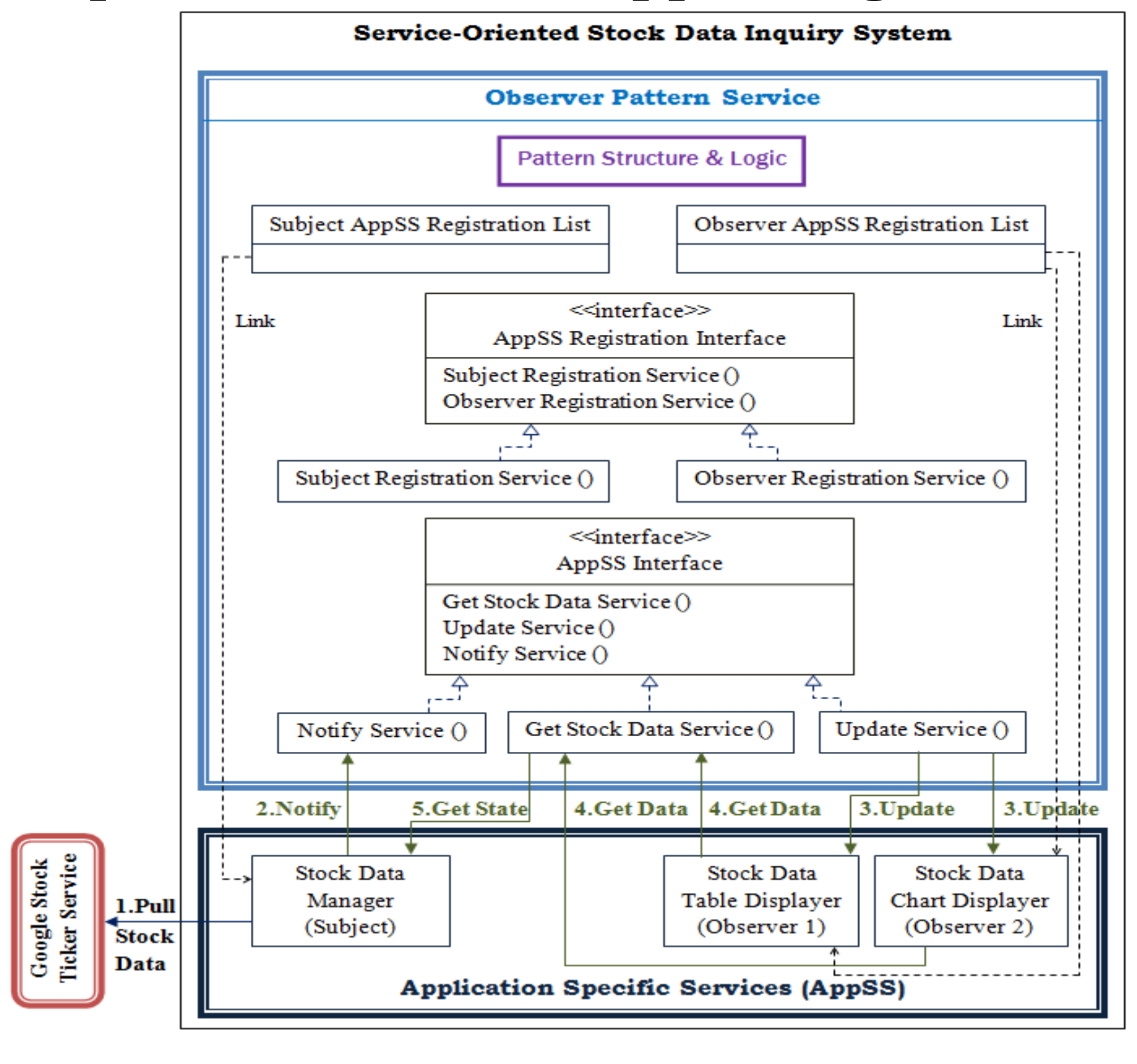
Service-Oriented Stock Data Inquiry System
Observer Pattern Service
Pattern Structure & Logic
Subject AppSS Registration List
Observer AppSS Registration List
Link
Link
<<interface>>
AppSS Registration Interface
Subject Registration Service ()
Observer Registration Service ()
Subject Registration Service ()
Observer Registration Service ()
<<interface>>
AppSS Interface
Get Stock Data Service ()
Update Service ()
Notify Service ()
Notify Service ()
Get Stock Data Service ()
Update Service ()
2.Notify
5.Get State
4.Get Data
4.Get Data
3.Update
3.Update
Google Stock Ticker Service
1.Pull Stock Data
Stock Data Manager (Subject)
Stock Data Table Displayer (Observer 1)
Stock Data Chart Displayer (Observer 2)
Application Specific Services (AppSS)

Figure 5: Service-Oriented stock data inquiry system using the observer pattern service

To enable interaction with the registered AppSS, a set of interfaces and delegation code needs to be created. This can be achieved by running a configuration service that adds the required configuration and delegation code. Figure 5 shows the components and interactions of a service-oriented stock data inquiry system using the observer pattern service. Note the presence of the necessary interface and delegation code that enables the communication between the generic pattern service and its complementing AppSS.

## Pattern as a Service (PaaS) System

The PaaS system is a platform that stores, integrates, deploys and manages pattern services and pattern-based service-oriented systems. As can be see in Figure 6, the Pattern Service Creator implements Pattern Services (PS) and stores them in the Pattern Service Repository. Then the Pattern-based Application Creator selects a PS, studies its documentation, implements any needed Application Specific Services, registers them with the PS and stores the resulting application in the Pattern-based Software Applications Repository. The Pattern-based Application Instance Creator selects the application of interest, configures it, and finally deploys it to the end users of the application.

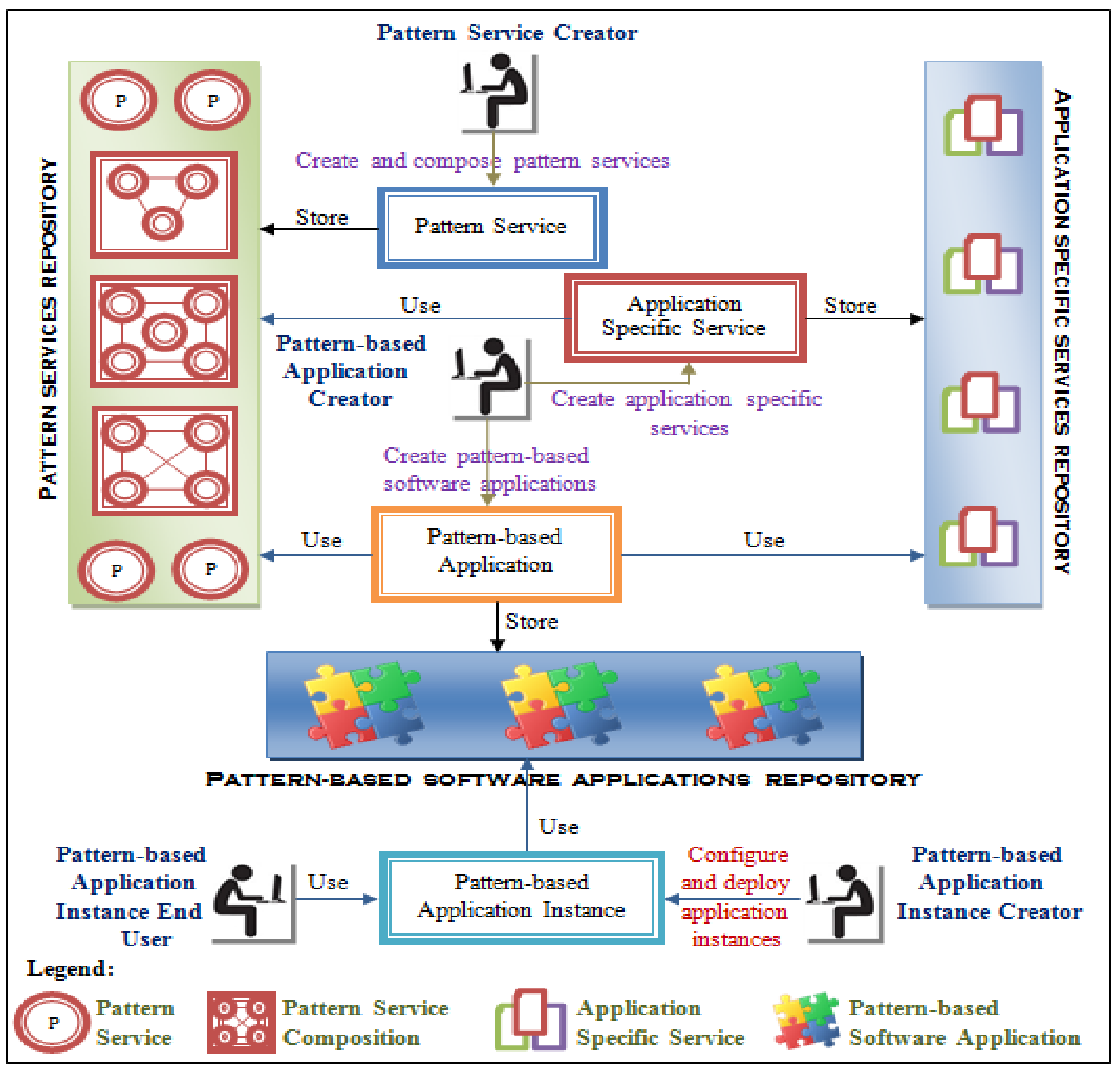
Pattern Service Creator
Create and compose pattern services
PATTERN SERVICES REPOSITORY
Store
Pattern Service
Use
Application Specific Service
Store
Pattern-based Application Creator
Create application specific services
Create pattern-based software applications
APPLICATION SPECIFIC SERVICES REPOSITORY
Use
Pattern-based Application
Use
Store
PATTERN-BASED SOFTWARE APPLICATIONS REPOSITORY
Use
Pattern-based Application Instance End User
Use
Pattern-based Application Instance
Configure and deploy application instances
Pattern-based Application Instance Creator
Legend:
Pattern Service
Pattern Service Composition
Application Specific Service
Pattern-based Software Application

Figure 6: General architecture of the PaaS system.

## References

Schmidt, D. C. (1997). Applying Patterns and Frameworks to Develop Object-Oriented Communication Software. In e. b. Salus, *Handbook of Programming Languages, Volume I.* MacMillan Computer Publishing.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software.* USA: Adison-Wesley.