Linnéuniversitetet
Kalmar Växjö

Assignment 3

Quality Attributes, Patterns, & Tactics

Författare: Jonathan Alklid, Helena Tevar
e-mail:
Examinator: Jesper Andersson

Termin: VT 20
Ämne: Software Architectures
Kurskod: 2DV604

View: Flexible Communication

Overview

Our system general architecture pattern is “Model-View-Controller” (MVC from now on). The elements interact with each other by passing data from one component to another. This pattern is flexible and helps possible modifications. Our system components should interact with each other in a way that they remain independent, decoupled, available and maintaining modifiability.
In order to achieve these concerns, a broker sub-system will be placed as middleware and include a subscription manager for several clients. The clients subscription and the filtering will be managed by a content pool. When our system invokers send events to the broker, it will filter them and broadcast the results. Afterwards, it will log the event to the logger component.

This view will solve the concern of a flexible communication system that provides aid for three scenarios. The changes on data in our system will be the stimulus that triggers a callback-response to the observers of each scenario.

- Scenario 1: Spectators can receive notifications on competitions and events, results and favorite gymnasts.
- Scenario 2: Judges get information on their schedule and schedule changes.
- Scenario 3: The internal logging mechanisms requires a flexible internal and external messaging system.

Patterns Used

The pattern “Publish-Subscribe” (PS from now on) is able to give information or access components in runtime by broadcasting or filtering to specific components, while remaining decoupled. This gives the system the possibility of being scaled while this functionality is kept encapsulated.

- Scenario 1: When the sub-system receives a stimulus, the data will be *broadcasted* to our subscriber pool, that will determine what data is interesting.
- Scenario 2: When the sub-system session receives a stimulus, it will filter the data and broadcast it to the interested subscribers.
- Scenario 3: When the system session receives a stimulus, the data will be sent to the logger component.

Audience

Stakeholder	
Architect	Responsible of the development of the architecture
Project Manager	Responsible for the planning and allocating
Designer	Responsible of designing and meet architectural requirements
Implementer	Responsible for the development of elements
Maintainer	Responsible for fixing bugs and providing enhancements
Tester	Responsible for test and verification

Models

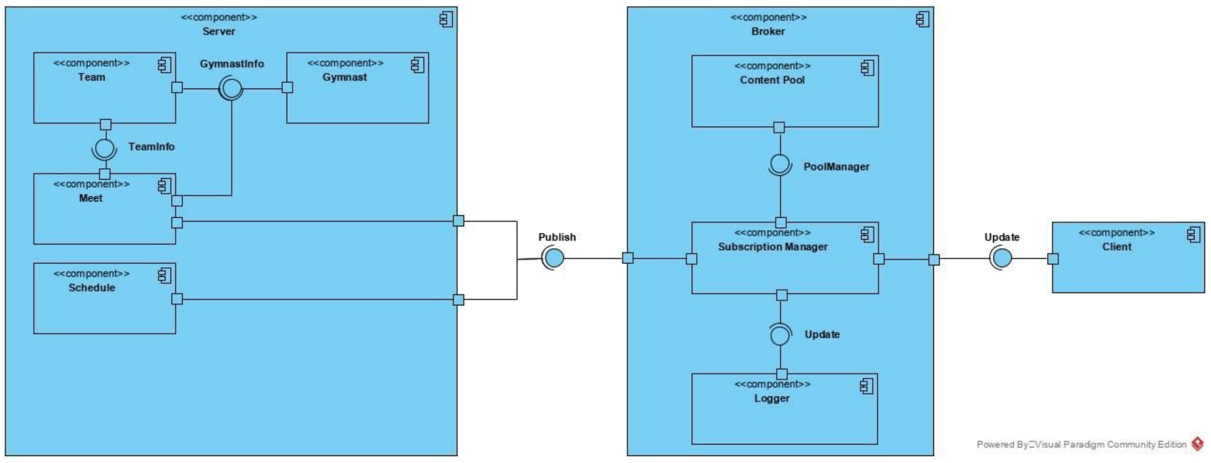
This section contains the models provided below:

Model
Component diagram
Sequence diagram

Component Diagram

Presentation

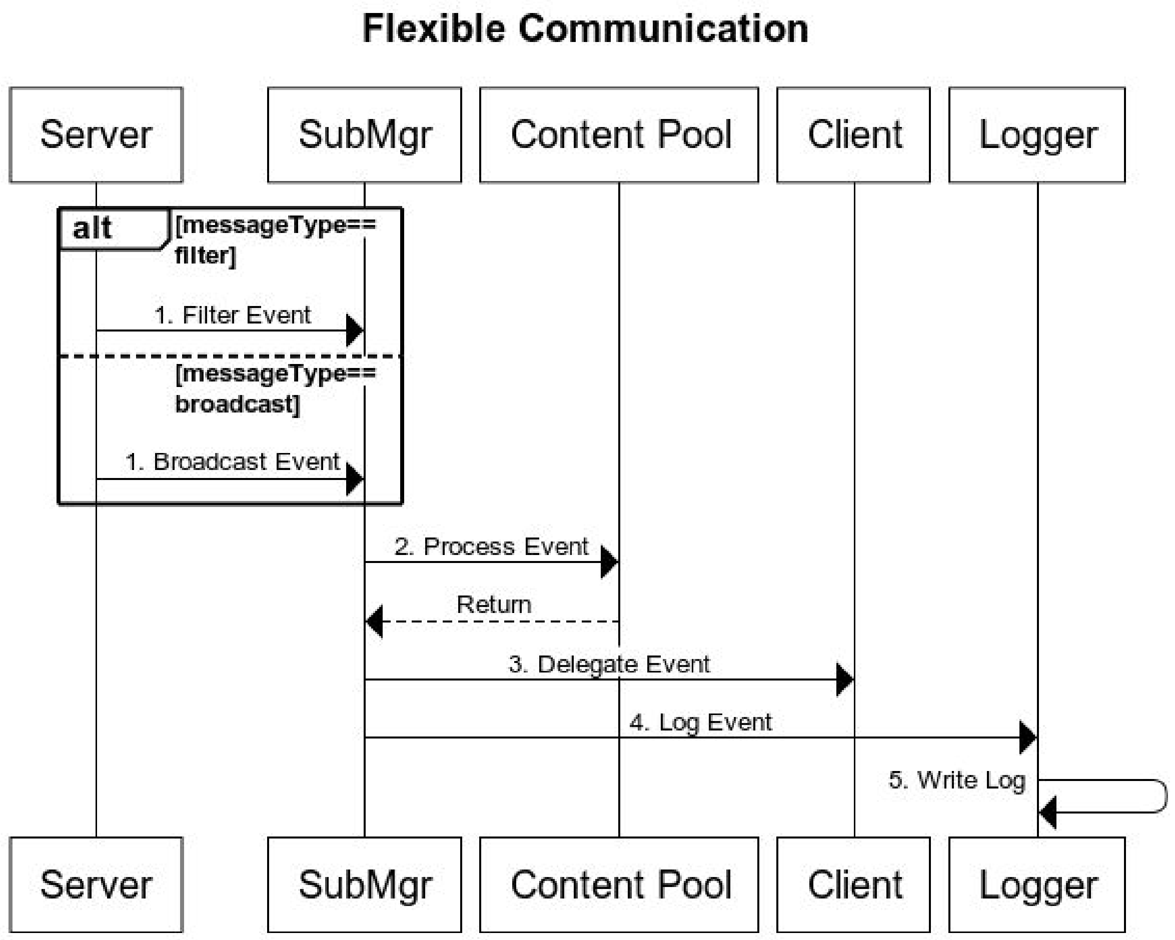
The model below is meant to be a general overview of the different components that have a main participation on communicating information with flexibility.



Sequence Diagram

Presentation

The model below focuses on the communication between components that are relevant for this view. It is meant to be a representation of event routing by broadcasting, topic based messaging and the logger functionality.



Element Catalogue

In this section it will be described the elements that compose the view and their relationships. Both diagrams employ the same elements and relations and therefore these sections have been merged into one.

Elements

The main elements included in this views are:

- Client: One or many clients will be subscribed to our broker component and wait for information from it. The clients would be from web-based applications loaded in different browsers.

- Server: Our system will provide services. In early stages, it will be allocated in a single server but it has to be developed to maintain scalability. The internal workings of the server have been omitted due to the scope of this view.
- Broker: While not visible in the sequence diagram (as the internal communication was deemed more important for this view), it represents a middleware sub-system between our server and clients to provide subscription manager services that will take events invoked from the server and transmit them to our clients. The broker subsystem and the server subsystem could be allocated to the same hardware platform in early stages, but could be escalated to different servers with multiple brokers.
  - SubMgr: the Subscription manager as seen in the previous diagram. This is responsible for delegating events from the server to interested clients.
  - Content Pool: allows for processing and temporary storage of incoming events, ensuring that the correct client gets the correct event.
  - Logger: saves data to be used for maintenance sub-systems.

Relations

The invoker/publisher elements are attached to the subscriber elements.

Linnéuniversitetet
Kalmar Växjö

Tactics versus Patterns motivation

Tactics are the most atomized level from patterns that are aimed to solve a single quality attribute response, mostly. In this particular case we could use the PS pattern as a tactic for a small component. For instance, our assumptions point that our system is using a MVC pattern in high level, but the particular request on communication between components is solved by using the PS pattern. That could be understood as a tactic inside our system architecture.
However, a pattern defines a set of elements and relationships that solve a particular problem from some perspective. Using this definition it would be clear that the PS pattern is indeed, and as its name proclames, a pattern that is used to solve a problem, in this case a communication problem that includes many quality attributes, while using a broker pattern as a tactic. For that reason our subsystem is not enough atomized to be named a “tactic”.