

Solving Feature Interaction Problems Using Java Filter Objects

Anshu Veda(04329022)
KReSIT, IIT Bombay

Prajakta Kalekar(04329008)
KReSIT, IIT Bombay

April 20, 2005

1 Introduction

In current-day scenarios where each system is expected to support multiple features, feature interaction is inevitable. If this feature interaction is conflicting in nature, one may not be able to use one service because of the other. Most of the ongoing research has been focusing on the detection of Feature Interactions. However, a neat solution which would enable us to use conflicting features in a system is certainly desirable.

The recently introduced *Java Filter objects* [2] seem to have the potential for solving such problems.

2 Feature Interaction

A feature is an observable, relatively independent behavior or characteristics of software. Some important things to note about them are

- Software requirements only exist in problem domain, while features exist in both problem domain and solution domain. One requirement may produce one or more features. And there is no strict correspondence between problem-domain features and solution-domain features.
- A software component may implement one or more features, while a feature may be implemented by multiple components.
- A feature may be a functional unit or some non-functional property.

2.1 Types of feature Interactions

Feature interactions can be classified as

- Shared Trigger Interactions These basically refer to those features that are triggered by a common cause and may occur simultaneously. For eg. Call Busy and Call forwarding both of which are triggered by line.
- Sequential Action Interactions Herein one feature triggers the second, second triggers third and so on.
- Looping Interactions Looping Interactions (LIs) occur when multiple features, or even multiple instances of a single feature, operate in concert to cause a cyclic sequence of events to occur.
- Missed Trigger Interactions Interactions where the presence of one feature in the system prevents the second feature from operating at all are termed Missed Trigger Interactions

Sequential Interactions do not pose much of a conflicting problem. however, other are highly probable to interact negatively under which circumstances, only one of the conflicting features could be used.

3 Java Filter Objects

Java Filter Objects [2], can be defined for different methods. In a sense they perfectly model the pre-and post conditions for a method being invoked. Java Filter objects can be easy plugged and used to monitor the behavior of methods. In other words, when a method is invoked these filters can transparently act as middlemen between the method and the caller and hence prevent conflicting features to occur only in constrained situations.

4 Approach

We have been trying to simulate some of the common feature interaction problems using Java Filter object (TJF).

5 Problems Studied and Designed

5.1 Shared Interaction

5.1.1 Problem Description

The scenario under consideration is that of a service provider who charges different rates for domestic and commercial use of the video on demand service (the commercial rate is higher). If a pub/bar owner is already subscribed to the domestic VoD service, he may wish to use the call forwarding service to forward video streams from his domestic VoD service to his pub/bar to provide in-bar entertainment. This would make financial sense if using the domestic VoD with call forwarding was more cost effective than using the standard commercial VoD service.

5.1.2 Design

The problem solution can be to use a filter object before the video forwarding function, which does not let one forward to a commercial users.

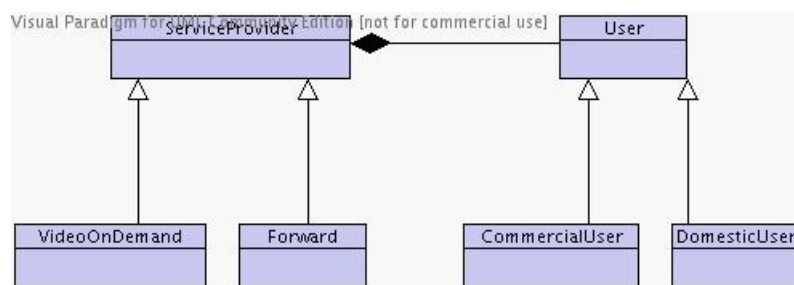


Figure 1:

5.2 Looping Interaction

5.2.1 Problem Description

The vacation program service (VPS) automatically replies to any received e-mails with a user-defined message. Its purpose is similar to that of an answering machine. The service is useful if the subscriber goes on holiday and wishes to inform email senders of this, thus preventing his e-mail

box from filling up with repeated messages from the same e-mail senders while he is away. The mailing list service (MLS) allows users on a mailing list to mail the other users on the list by simply e-mailing the list server. If the return address for e-mails sent on the mailing list is the address of the list server (this is usual), and one of the subscribers on the mailing list enables his vacation service, the following feature interaction scenario can occur. When the vacation program receives an e-mail from the mailing list server, it will automatically send back a user-defined message to the server, thus causing the server to send it another message which the vacation program also replies to. This situation will continue until the vacation program until the user's e-mail box overflows or the list server is disabled.

5.2.2 Design

Restricting the list server to not distribute messages to the sender of the e-mail prevents this FI scenario for a single vacation program subscriber, but it does not prevent the case when two or more list subscribers have activated the vacation program. So the approach taken was to apply a filter on the vacation program, that on replying a mail keeps record of all the users been intimated and never mails back a user till its memory is flushed. The memory of the program is cleared as soon as the program starts.

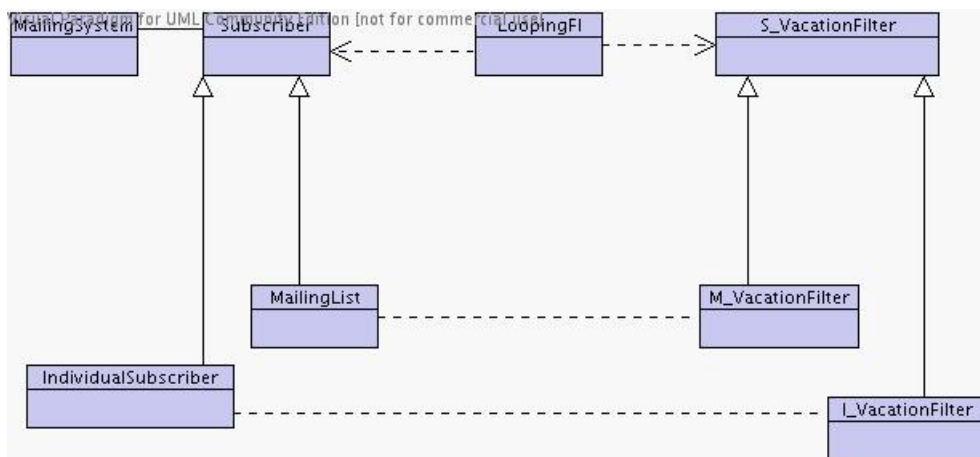


Figure 2: Looping Interaction

5.3 Triple Feature Interaction Problems

A Feature Interaction which involves three features is called a Triple Feature Interaction. In other words, in order to be called Triple Feature Interactions, three features need to be active in a call and the interworking of all three of them needs to cause undesired behavior. Imagine the scenario where two telephony users subscribe to following services:

- User A: Calling Number Delivery Blocking,
- User B: Automatic Callback and Itemized Billing

An interaction wherein following sequences of steps occur

- A calls B, B is busy
- Once B is idle, B calls A back
- A's number appears on B's bill

In this scenario, the A s subscribed feature is getting violated.

5.3.1 Design

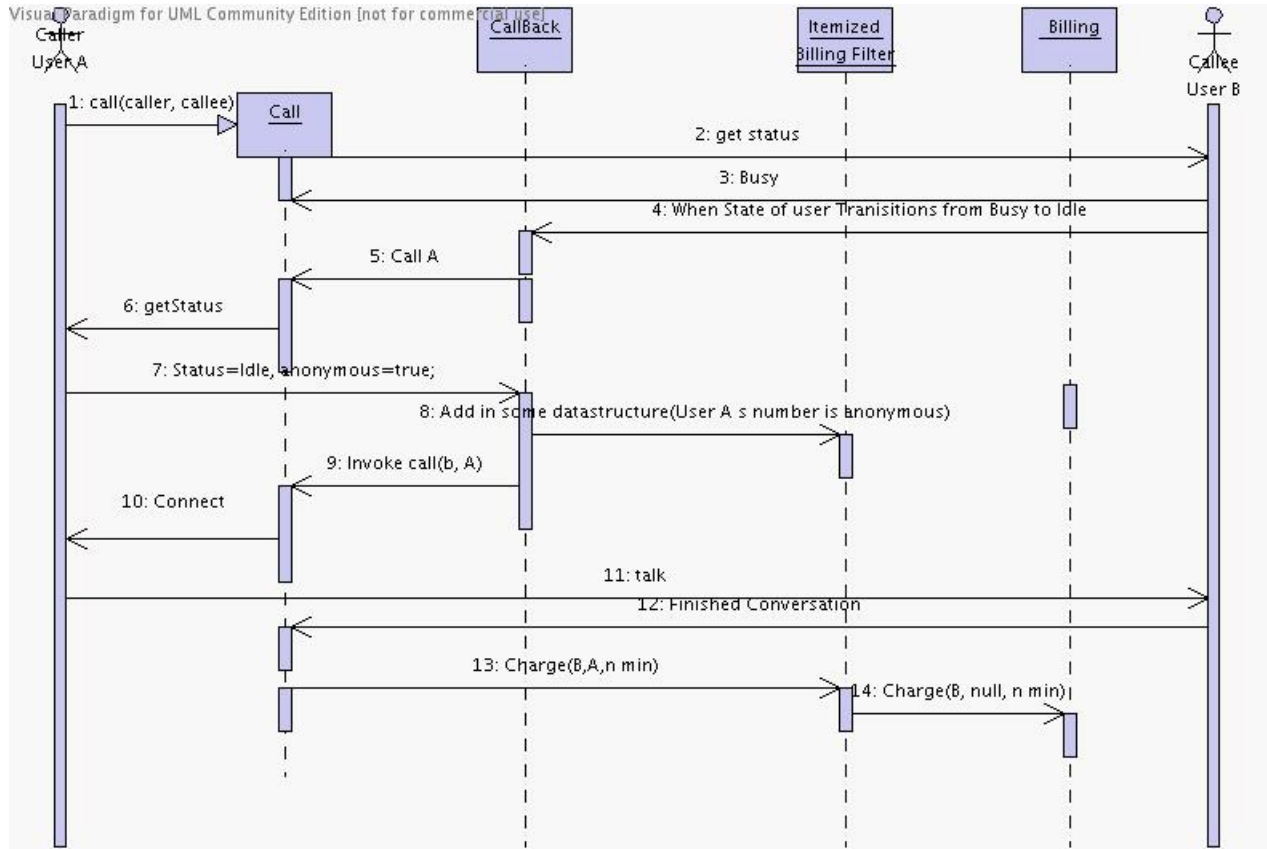


Figure 3: Triple Feature Interaction

An approach taken to this problem was to apply filter on the Itemized Billing class. Now when a callback occurs, a filter on callback, checks the CNDB status of User being called. If its to be blocked, it reports to the itemized billing filter, which stores some state till the call conversation finishes and call ends. The BillingFilter when invoked for Billing will check for the status and correspondingly, make the value of number as null.

References

- [1] Simon Tsang and Evan H. Magill *An Investigation of the Feature Interaction Problem in Networked Multimedia Services*
- [2] Rushikesh K. Joshi, Maureen Mascarenhas and Yogesh Murarka *Filter objects for Java*
- [3] Muffy Calder, Mario Kolberg, Evan Magill, Dave Marples, Stephan Reiff-Marganiec *Hybrid Solutions to the Feature Interaction Problem*