Lecture 9

Patterns for Secure Middleware

References:

Fernandez-Buglioni, E., *Security patterns in practice: designing secure architectures using software patterns*. John Wiley & Sons, Chapter 13, 2013

1

Patterns for Secure Middleware

- Secure Broker
- Secure Pipe and Filter
- Secure Blackboard
- Secure Adapter
- Secure Distributed Publish/Subscribe
- Secure Model-View-Controller

2

Patterns for Secure Middleware

- Secure Middleware
  - Support developments of applications or execution
    - Providing secure services to applications
  - Include distributed aspects, e.g., secure brokering
  - May also include global security services, e.g., authentication, authorization

3

Secure Broker

- Broker architecture pattern
  - Broker component is responsible for coordinating communication
    - Forwarding request and transmitting results and exceptions
    - Basic use cases – server registration and client requests service
  - To provide secure interactions between distributed components

4

Secure Broker

- Context and Problem
  - Security threats to broker activities
    - Illegal access - from client to server and inversely
    - Message interception or replaying
      - A valid data transmission is maliciously or fraudulently repeated
    - Spoofing (forgery)
      - Client, broker, and server forgery
- Solution – Secure broker
  - Introduce mutual authentication between servers and clients
  - Provide access control to control access to resources
  - Provide cryptographic controls to prevent message attacks

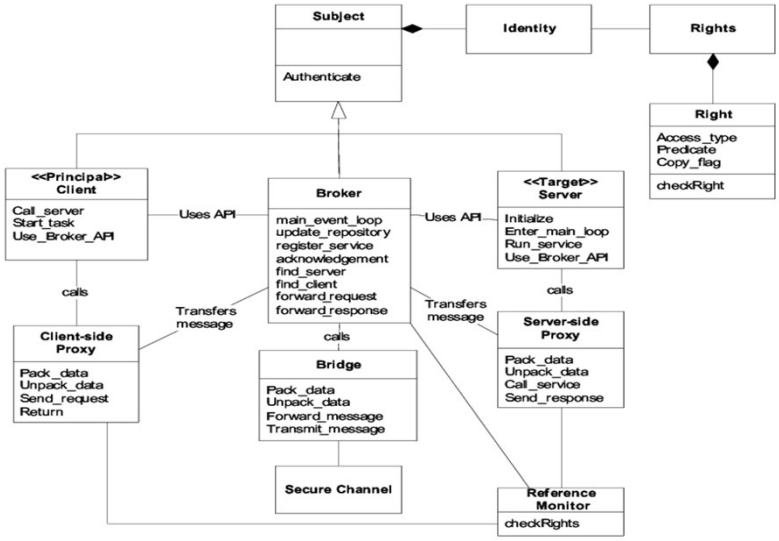
5

Secure Broker

- Structure
  - Fig. 13.2
  - Subject: Allows components to authenticate each other
  - ReferenceMonitor: Authorize requests
  - SecureChannel: Encrypting traffic between components
- Dynamics
  - Subject creation
    - Clients, servers, and brokers assigned identities and credentials
  - Secure registration
    - Mutual authentication must be done before registration
  - Secure Service Request, Fig. 13.3

6

Fig.13.2 Class Diagram for Secure Broker Patterns



7

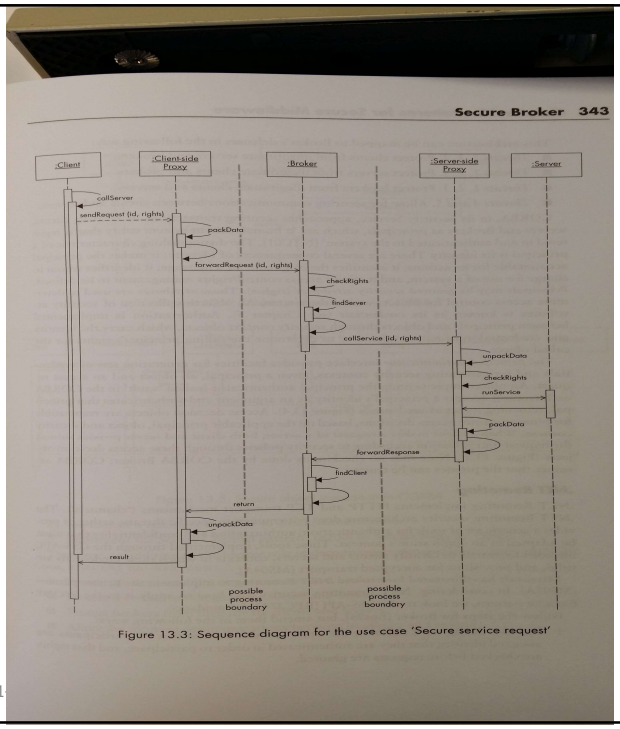


Figure 13.3: Sequence diagram for the use case 'Secure service request'

8

Pipes and Filters architecture

- Elements
  - Pipe - stream data
  - Filter - process data

- Filters are independent
  - System evolution is simple
  - Filters can be reused
  - Applicable for batch processing systems
- Example
  - Compiler
  - Filters – lexical analysis, syntax analysis, semantic analysis, code generation and optimization

9

Secure Pipes and Filters

- Provide secure handling of data streams
  - Controls the rights to apply specific transformation to the data
  - Protect the communication of data between filters
- Context and Problem
  - E.g., a company producing a product and working with multiple persons remotely in a workflow
    - Each person can apply different function to the product and update a document for the product
    - Need to control actions (functions) being performed to product
    - Also need to provide security for pipeline activities

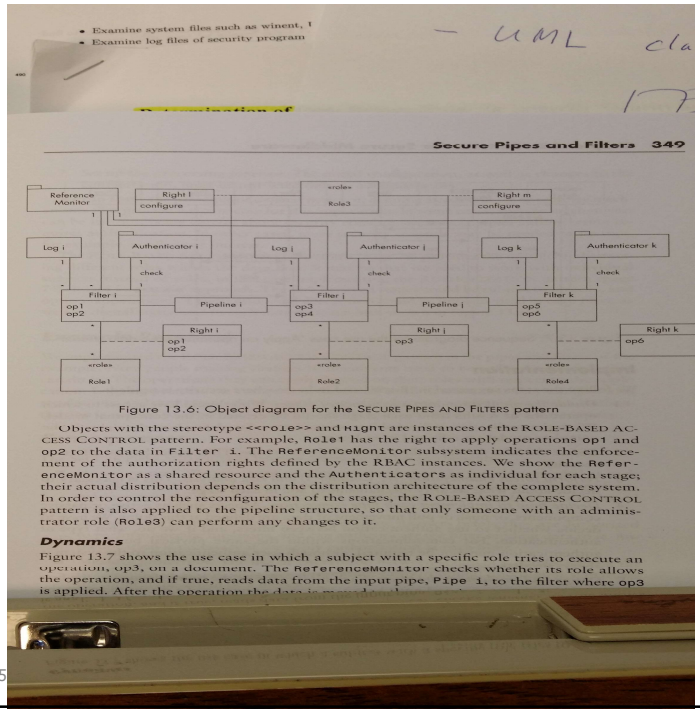
10

Secure Pipes and Filters

- Security Issues
  - Stage Control – who can do what with the data in the pipeline
  - Authenticity – for each stage
  - Message protection – eavesdropping
  - Reconfiguration control – change in workflow
  - Recoding – keep tract of any actions applied to the data
  - Transparency – security control that is transparent to users
  - Overhead – security control brought
- Solution
  - Provide authentication, authorization, information hiding and logging

Secure Pipes and Filters

- Structure
  - Fig. 13.6
  - Adopts RBAC (Role-based Access Control)
  - Authenticator
    - Allow each filter to authenticate the sender of data
  - Log
    - Keep track of any accesses to the data
  - ReferenceMonitor
    - Enforce authorization
- Dynamics
  - Fig. 13.7: Apply an operation on a data stream



- Examine system files such as winent, l
- Examine log files of security program

Figure 13.6: Object diagram for the SECURE PIPES AND FILTERS pattern

Objects with the stereotype `<<role>>` and `right` are instances of the ROLE-BASED ACCESS CONTROL pattern. For example, `Role1` has the right to apply operations `op1` and `op2` to the data in `Filter i`. The `ReferenceMonitor` subsystem indicates the enforcement of the authorization rights defined by the RBAC instances. We show the `ReferenceMonitor` as a shared resource and the `Authenticators` as individual for each stage; their actual distribution depends on the distribution architecture of the complete system. In order to control the reconfiguration of the stages, the ROLE-BASED ACCESS CONTROL pattern is also applied to the pipeline structure, so that only someone with an administrator role (`Role3`) can perform any changes to it.

Dynamics

Figure 13.7 shows the use case in which a subject with a specific role tries to execute an operation, `op3`, on a document. The `ReferenceMonitor` checks whether its role allows the operation, and if true, reads data from the input pipe, `Pipe i`, to the filter where `op3` is applied. After the operation the data is

13

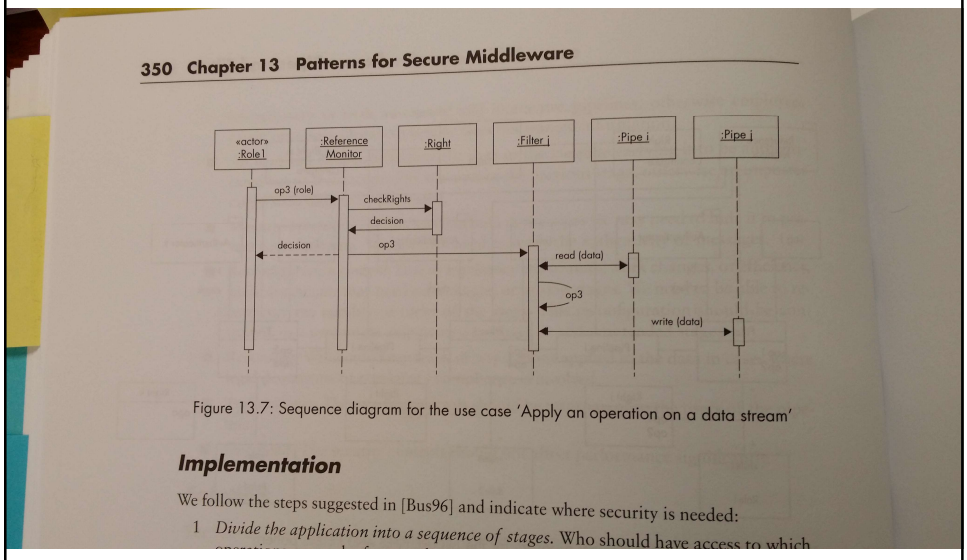


Figure 13.7: Sequence diagram for the use case 'Apply an operation on a data stream'

Implementation

We follow the steps suggested in [Bus96] and indicate where security is needed:

1 *Divide the application into a sequence of stages.* Who should have access to which

14

Blackboard (Repository)

- Two communication ways between subsystems
  - Central database or repository
  - Its own database and data passing explicitly to other subsystems
- Repository (Blackboard)
  - Central store may control the triggering of processes
  - Central store represents the system state

15

Secure Blackboard

- Context and Problem
  - Blackboard, control, and knowledge sources
  - Nondeterminism
    - Sequence of activities over data is unpredictable
  - Access Control
  - Authenticity
  - Controlled reconfiguration
    - Need to reconfigure the number of knowledge sources or their order of operation
  - Records
    - Log the actions at each update for auditing later
  - Transparency
    - Transparent to users
  - Overhead
    - Not impose a significant overhead

16

Secure Blackboard

- Solution
  - Add basic security mechanisms to the control component
- Structure
  - Fig. 13.9
  - Security Logger, Reference Monitor, Authenticator
- Dynamics
  - Fig.13.10

17

[Fer11d]. The `Reference Monitor` associated with the control indicates the enforcement of authorization (page 100). `KnowledgeSources` can be humans, not just software components. Nevertheless, either automated or human knowledge sources require that their access is authenticated by the `Authenticator` (page 52) to verify their origin. The sources belong to `Roles`, according to their functions, and their `Rights` depend on these `Roles`.

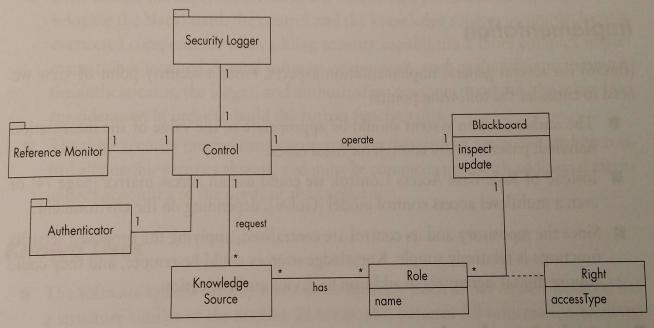


Figure 13.9: Class diagram for the SECURE BLACKBOARD pattern

Dynamics

Figure 13.10 shows a sequence diagram in which a `KnowledgeSource` (with a specific role) requests an operation on the `Blackboard`. The `Control` receives the request and in-

18

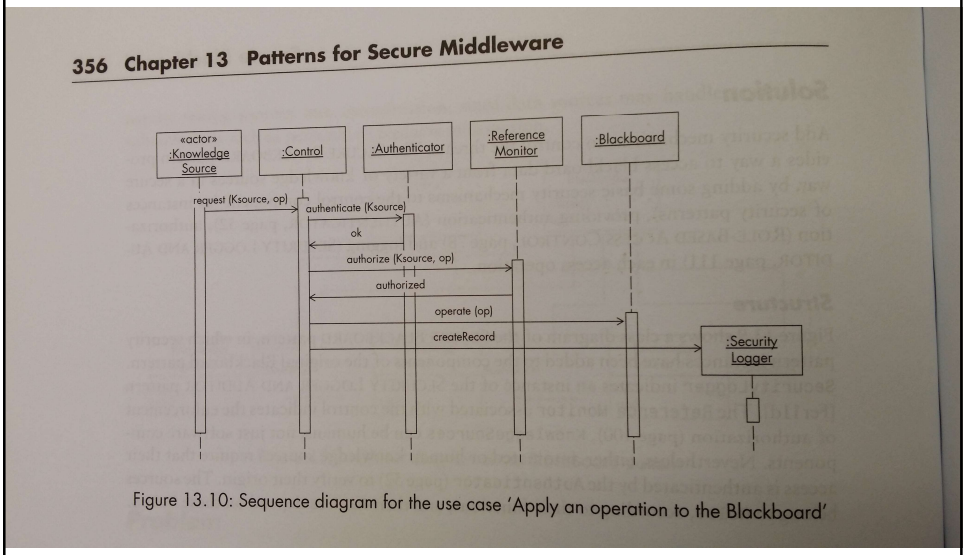


Figure 13.10: Sequence diagram for the use case 'Apply an operation to the Blackboard'

19

Secure Adapter

- Covert the interface of an existing class into a secure interface
- Context and Problem, e.g.,
  - A client requests a service using the RequestServices interface from database
  - But, in case the interface is incompatible with JDBC API
    - To adapt requests to JDBC and also converse after creating RequestServicesAdapter
- Threats
  - Database and client may be imposters
  - Client may not have permission to access
  - Data might be intercepted if the client is remote

20

Secure Adapter

- Solution
  - Authentication, Access Control, and Secure Channel between client and adapter
- Structure
  - Fig. 13.11
  - Target, adapter, adaptee (DB), Authenticator, RBAC
- Dynamics
  - Fig. 13.12
  - Request data via the secure adapter use case
  - Request is captured by adapter
  - Convert a request to a specific request for adaptee

21

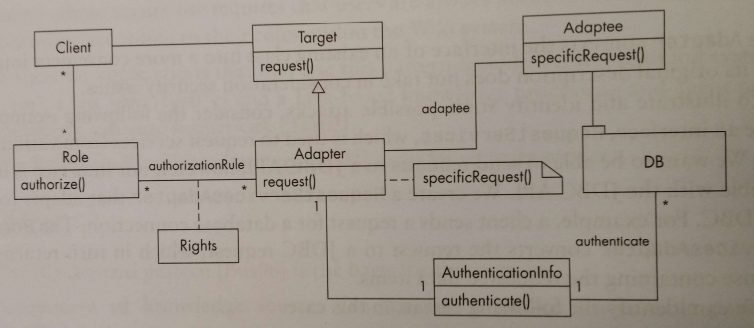
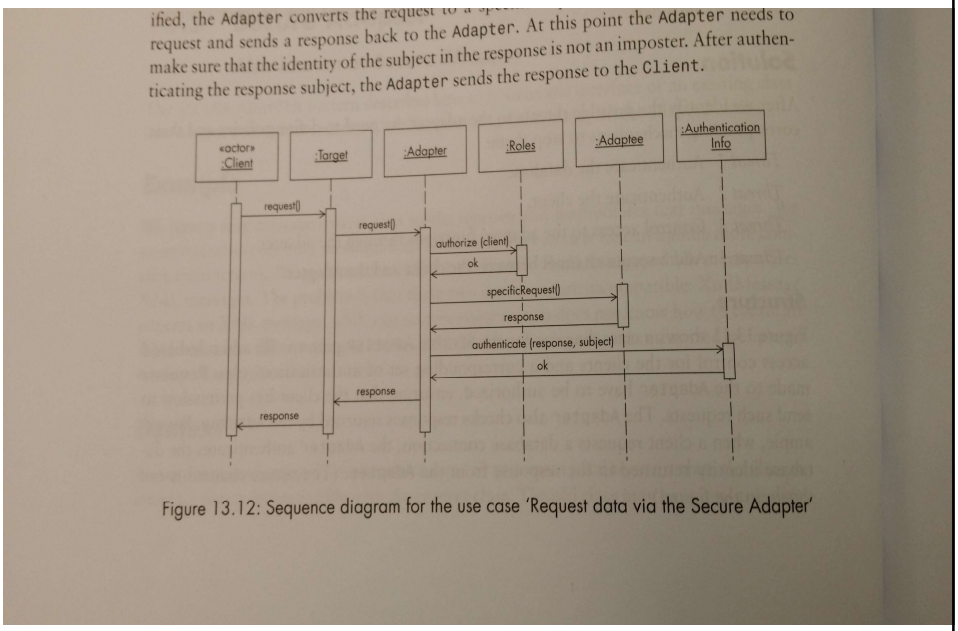


Figure 13.11: Class diagram of the SECURE ADAPTER pattern

Dynamics

Figure 13.12 shows a sequence diagram for the

22

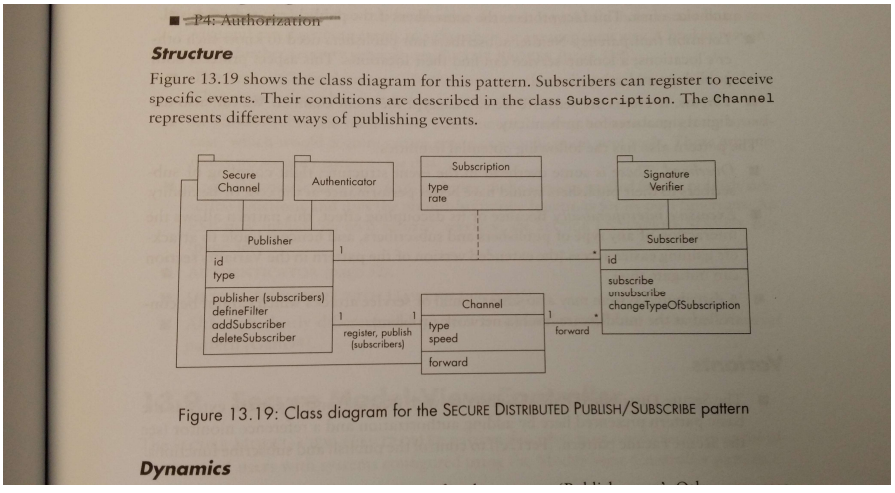


Secure Distributed Publish/Subscribe

- Subscription and publishing are performed securely
- Problem
  - Subscription
    - Imposter subscribes to receive information (authentication)
    - Publisher is an imposter (authentication)
    - Subscription message is intercepted (encryption)
  - Unsubscription
    - Imposter removes a subscriber (authentication)
  - Publish
    - Imposter receives message (authentication)
    - Imposter publishes information (digital signature)
    - Attacker reads or modifies intercepted message (encryption/integrity)

Secure Distributed Publish/Subscribe

- Solution
  - Use a secure channel
- Structure
  - Fig. 13.19
  - Secure channel, authenticator, signature verifier
- Dynamics
  - Fig. 13.20 for publish event



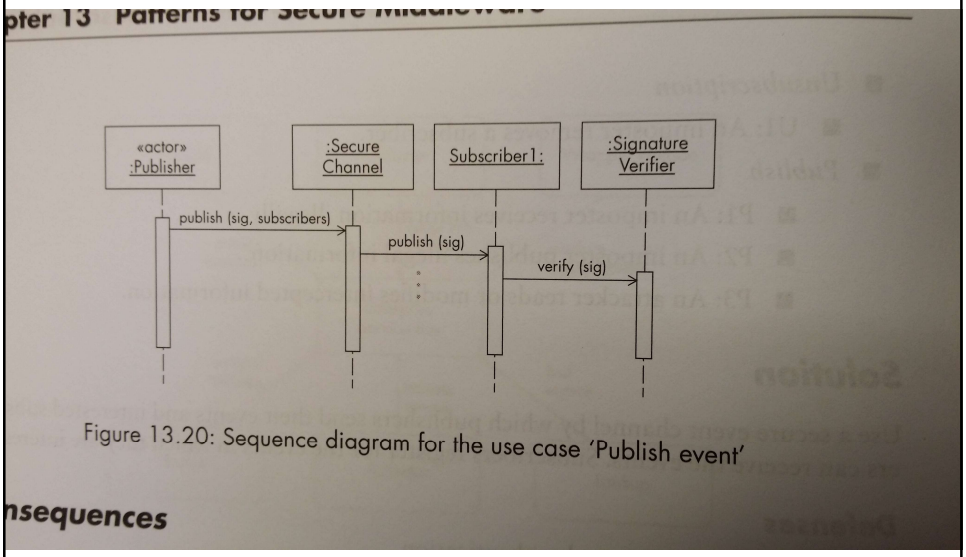


Figure 13.20: Sequence diagram for the use case 'Publish event'

Secure Model-View-Controller

- Maintain security between the model, view and control
- Context
  - Fig. 13.21 MVC pattern
- Problem
  - Authenticity for users
  - Confidentiality in transit from the model to the view
  - Integrity for changing the model
  - Records for all access to the information in the model

service-based applications to mobile web applications. Figure 13.21 shows the class dia-
gram of the structure of the MVC pattern.

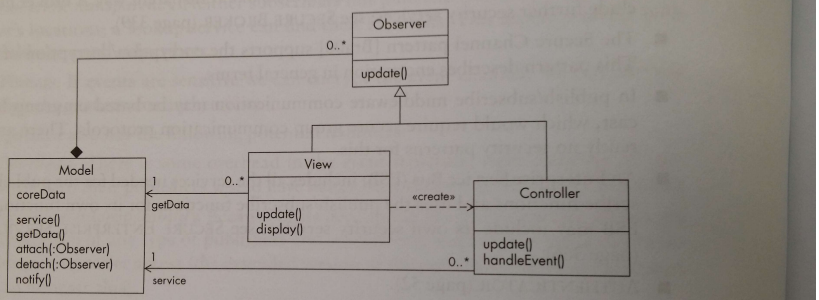


Figure 13.21: Class diagram of the structure of the MVC pattern (adapted from [Bus96])

Systems applying the MVC pattern are typically multi-user systems, and their model should be accessible and/or modifiable only by certain categories of users. At the same time, the use of the web as a transport layer has brought some new threats that must be

29

Secure Model-View-Controller

- Solution
  - Provide authentication, authorization, secure communication and logging
  - Sanitize data
- Structure
  - Fig. 13.22
  - Authenticator authenticates users accessing the Model and the Controller
  - Reference Monitor for access control
  - Secure Channel
  - Secure Logger for the model
  - Data Sanitizer to prevent malicious data
- Dynamics
  - Fig. 13.23 for Propagation of a change to the model

30

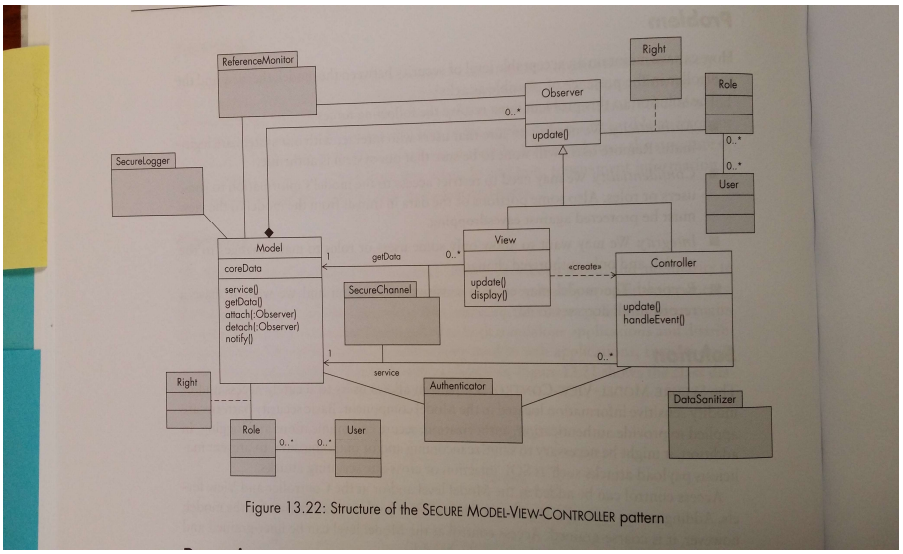


Figure 13.22: Structure of the SECURE MODEL-VIEW-CONTROLLER pattern

31

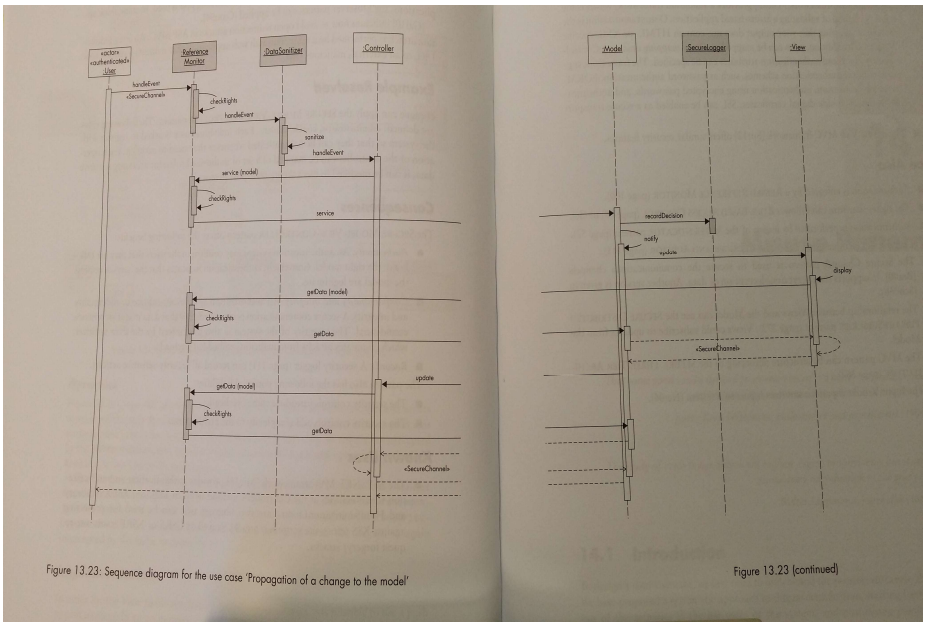


Figure 13.23: Sequence diagram for the use case 'Propagation of a change to the model'

Figure 13.23 (continued)

32