Services that reach from the inside out

July 19, 2006

Jim Tyhurst, Ph.D.
Consultant
Tyhurst Technology Group LLC
Portland, Oregon
jim06@tyhurst.com

Outline

- Architectural context
- The problem
- Additional forces to consider
- A solution
- Design details
- References

Architectural context

- Enterprise applications
- Integration of distributed systems
- Distributed services:
  - Developed locally; or
  - Maintained elsewhere in the enterprise; or
  - Provided externally by third party.

Sample services

- E-Commerce
  - Authorize credit card purchase
  - Request shipment of items from inventory
- Telecom
  - Provision a service
- Banking
  - Retrieve check images from archival service
- Insurance
  - Retrieve credit report
  - Retrieve motor vehicle report
- Enterprise applications
  - Authenticate login and authorize usage based on LDAP data

The problem

How can you encapsulate the details of the communication protocol for an external service, so that internal application components can focus on the business content of requests and responses?

Forces

- Asynchronous vs. synchronous
- Communications protocol
- Mapping request and responses
- Statefulness
- Audit trail
- Caching responses
- Error handling
- Unit testing
- Hot spots
- Reusability
- Wrapper on a wrapper on a wrapper on a …

Asynchronous vs. synchronous

- Not an issue specifically for this pattern
- Asynchronous requires a bit more complexity to track requests and corresponding responses
- For this paper, assume asynchronous response from external service as the general case

Communications protocol

- Encapsulate calls to external service
- Enables external service to be accessed through internal protocol
- Wrap content of request
- If needed, enhance request with:
  - Authentication parameters
  - Connection parameters

Mapping requests/responses

- Transformation of data content: format, field lengths, enumerated types, …
- Transformation of request structure
  - API differences
    - “granularity” of messages
    - sequence of messages
  - One-to-many
    - Might use multiple external services
    - Aggregate responses from one service or several
  - Many-to-one
    - Hold requests for batch to external service

Statefulness

- Especially for stateful requests that run for a long period of time:
  - Monitor completion status
  - Poll external service, if needed
  - Raise time-out event
- May need to persist changes to state
- For one-to-many request: raise event when all external responses received

Audit trail

- Persist request info
- Track source of requests for:
  - Security
  - Charge-back
- Log external response time
  - Diagnose overall system response problems
  - Confirm SLA

Caching responses

- Motivation: avoid call to external service
  - Improve response time
  - Reduce cost
- For data that changes infrequently
- Might use persistent store to hold values
- Need policy for removing "old" items from the cache

Error handling

- Translation of external error codes
- Service not available … wait and retry?
- Time out

Unit testing

- Isolate each aspect of the interaction: message translation, sending request, handling response, error handling
- Use mock objects to provide stable, isolated environment
- Do as much testing as possible outside of container

Areas of variability

- "Hot spots" [Pree 1995]: areas where change is likely, so flexibility is required
- Consider components within the Gateway that are likely to change:
  - From one deployment to another
  - From one implementation to another
  - Over time as:
    - requirements change
    - environments change

Reusability

- Component provides reusable, single point of contact
- Encapsulate the complexities of calling the external service
- Provide an interface that fits the business application

Too many layers

- Wrapper on a wrapper on a wrapper on a …
- Introducing another remote call will add to response time
- Trade-offs for simplicity for client component vs. additional layer of messaging
- One more unit to be deployed, configured, monitored, …

A solution

- Create a component that encapsulates the communication details
- Expose a business interface for internal components
- “Gateway” to the external service is built from components that isolate variable behavior

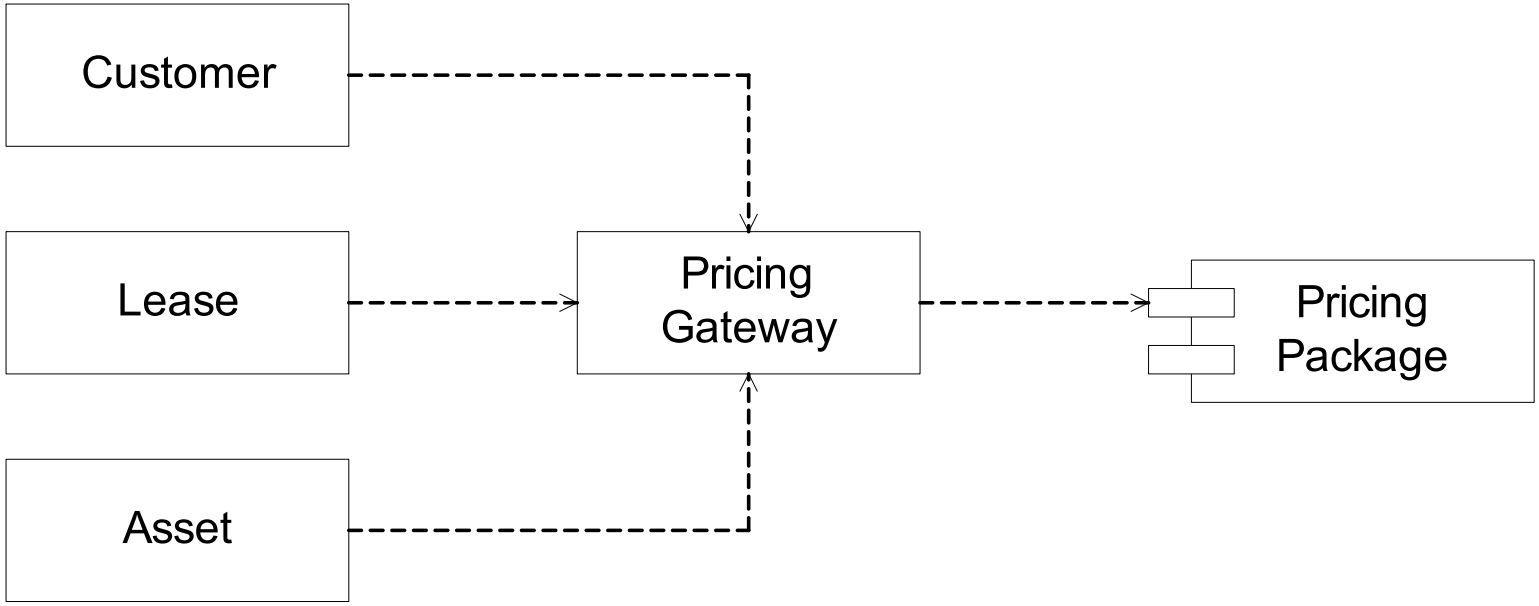
Gateway pattern

Fowler (2003):

An object that encapsulates access to an external system or resource.

- Similar to:
  - Façade simplifies a complex API
  - Adapter alters one interface to another

…Gateway pattern



Legend:

class component dependency

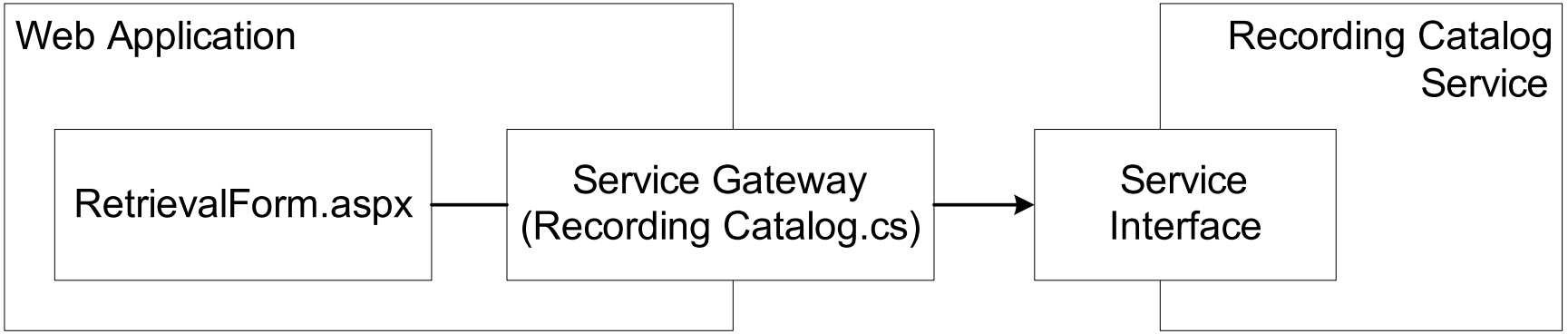
Fowler (2003)

Service Gateway pattern

Trowbridge et al. (2003) extend the concept to SOA:

Service Gateway encapsulates the details of connecting to the source and performing any necessary translation.

…Service Gateway pattern



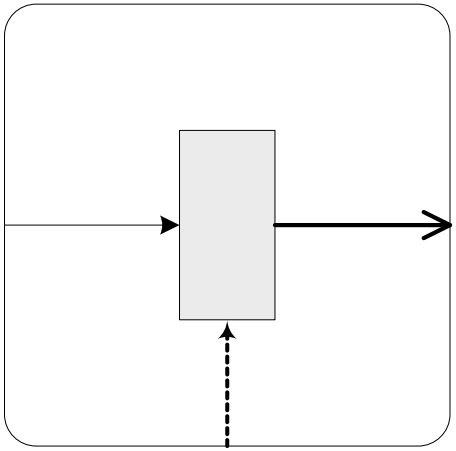
Trowbridge et al. (2003)

ESB Gateway pattern

Hutchison et al. (2005)

- Enterprise Service Bus (ESB)
- Sophisticated protocol switch
- Incorporates transform and monitor mediations:
  - encryption
  - logging
  - auditing
- Aggregate and disaggregate messages in a one-to-many relationship
- Single point of contact for multiple services

…ESB Gateway pattern



Legend:

protocol switch

enrichment mediation

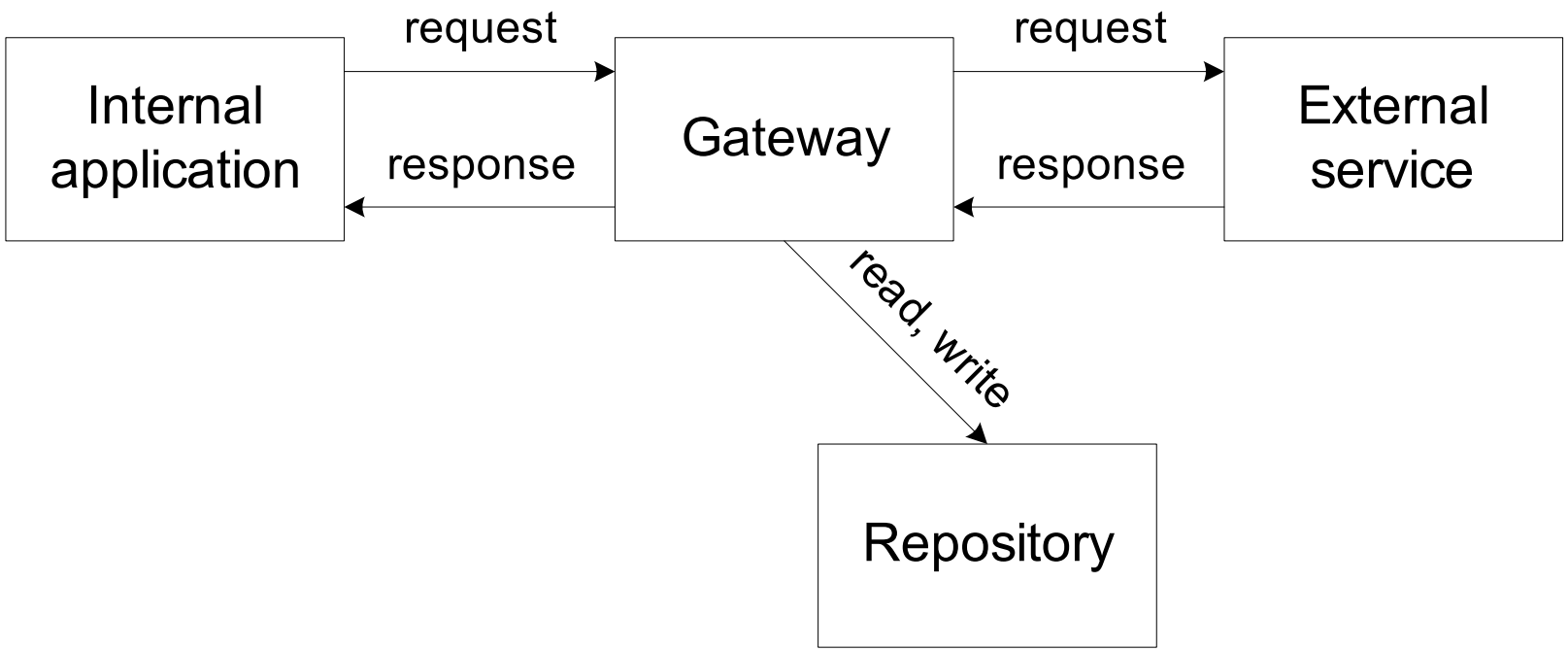
Hutchison et al. (2003)

Sample context

- Asynchronous messaging
- One internal request requires multiple external requests
- External requests are independent
- External responses are aggregated into a single internal response

Architecture

- Simple block diagram shows major components and interactions
- Diagram ignores communication details



Requests and Responses

- Interface to internal application
  - InternalRequest
  - InternalResponse
- Gateway domain model
  - Request holds List of IndividualRequest
  - Response holds List of IndividualResponse
- Interface to external service
  - ExternalRequest
  - ExternalResponse

Gateway structure

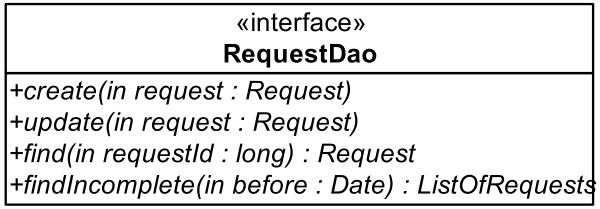
- Interfacers to messaging systems
- Message handlers (controllers)
- Data mappers
- Information holders

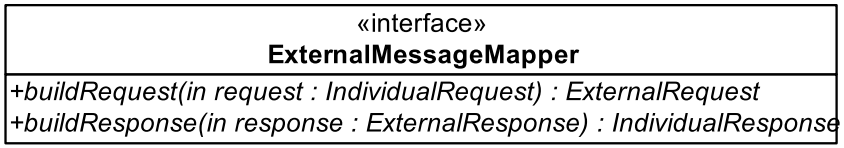
Request Gateway

Interfacers

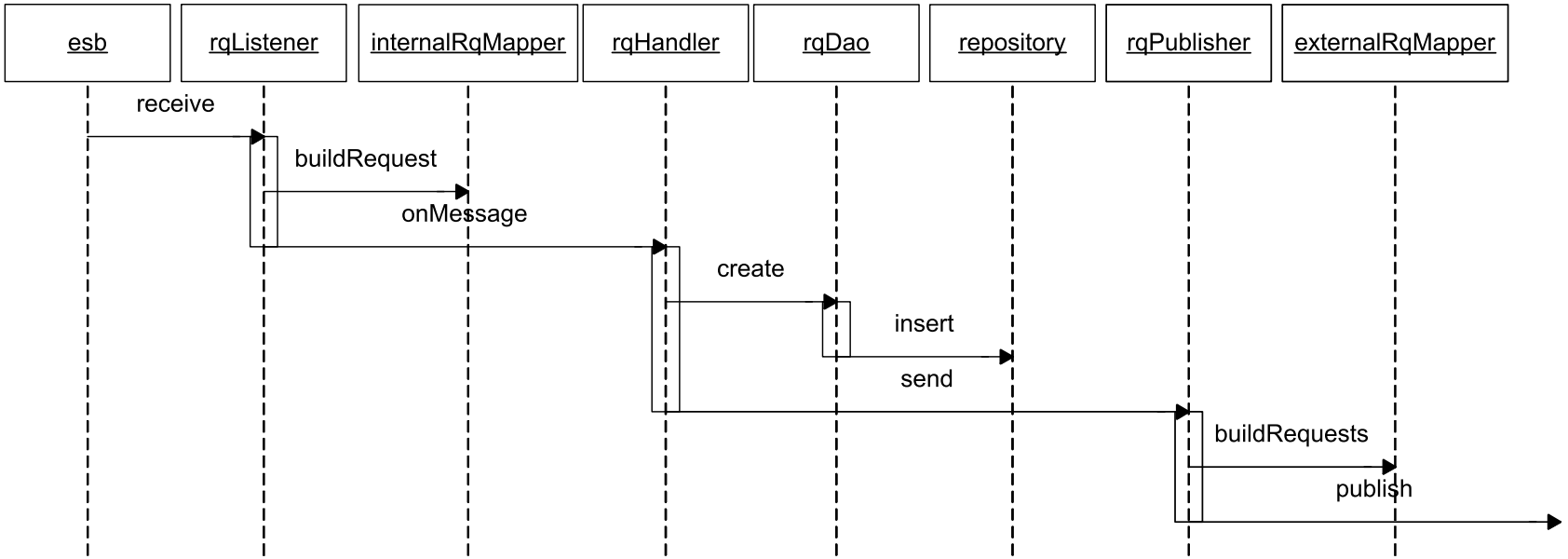
Controllers

Data Mappers





Request



UML sequence diagram:
Simplified view of receiving internal request that results in sending external requests while saving state in the repository.
'rq' abbreviates 'request'.

Response Gateway

Interfacers

«interface»
InternalResponsePublisher
+send(in response : Response)

«interface»
ExternalResponseListener
+receive(in response : ExternalResponse)

Controllers

«interface»
ResponseHandler
+onMessage(in response : IndividualResponse)

Data Mappers

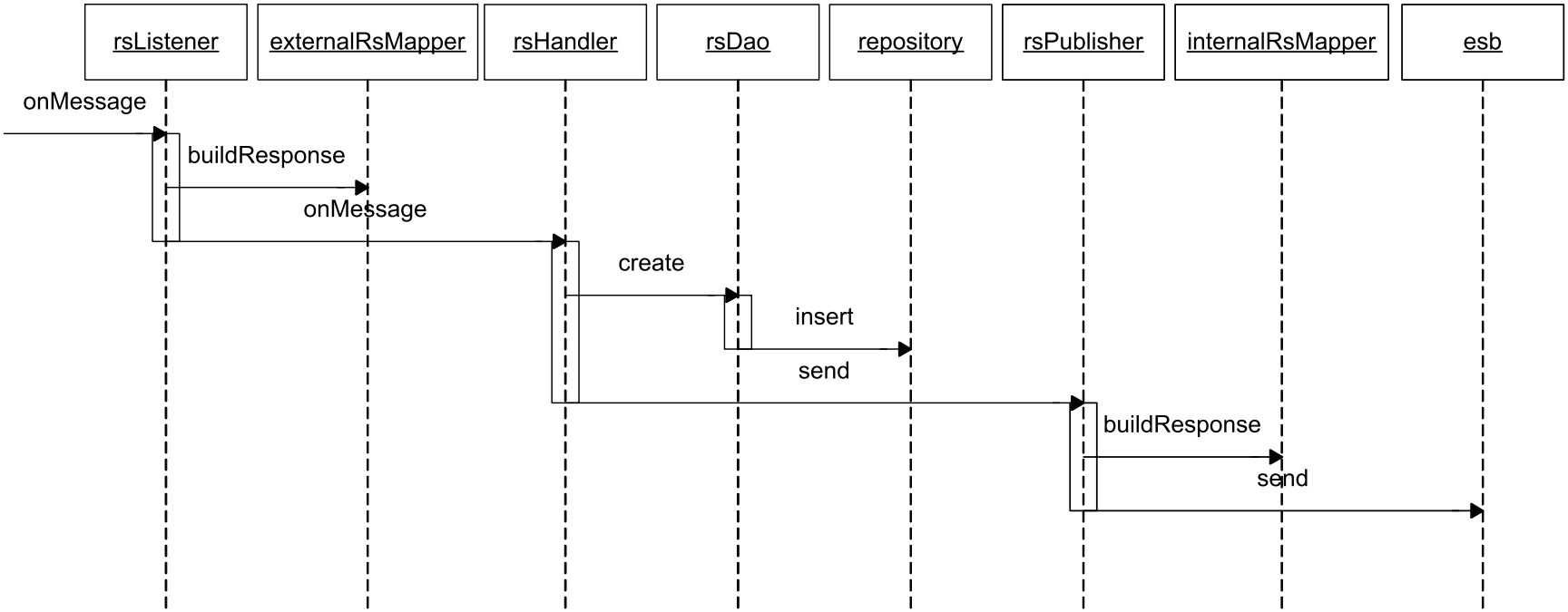
«interface»
RequestDao
+create(in request : Request)
+update(in request : Request)
+find(in requestId : long) : Request
+findIncomplete(in before : Date) : ListOfRequests

«interface»
ResponseDao
+create(in response : Response)
+update(in response : Response)
+find(in requestId : long) : Response

«interface»
InternalMessageMapper
+buildRequest(in request : InternalRequest) : Request
+buildResponse(in response : Response) : InternalResponse

«interface»
ExternalMessageMapper
+buildRequest(in request : IndividualRequest) : ExternalRequest
+buildResponse(in response : ExternalResponse) : IndividualResponse

Response



UML sequence diagram:
Simplified view of receiving external response and then sending internal response. Does not show flow to determine if original request is complete.
'rs' abbreviates 'response'.

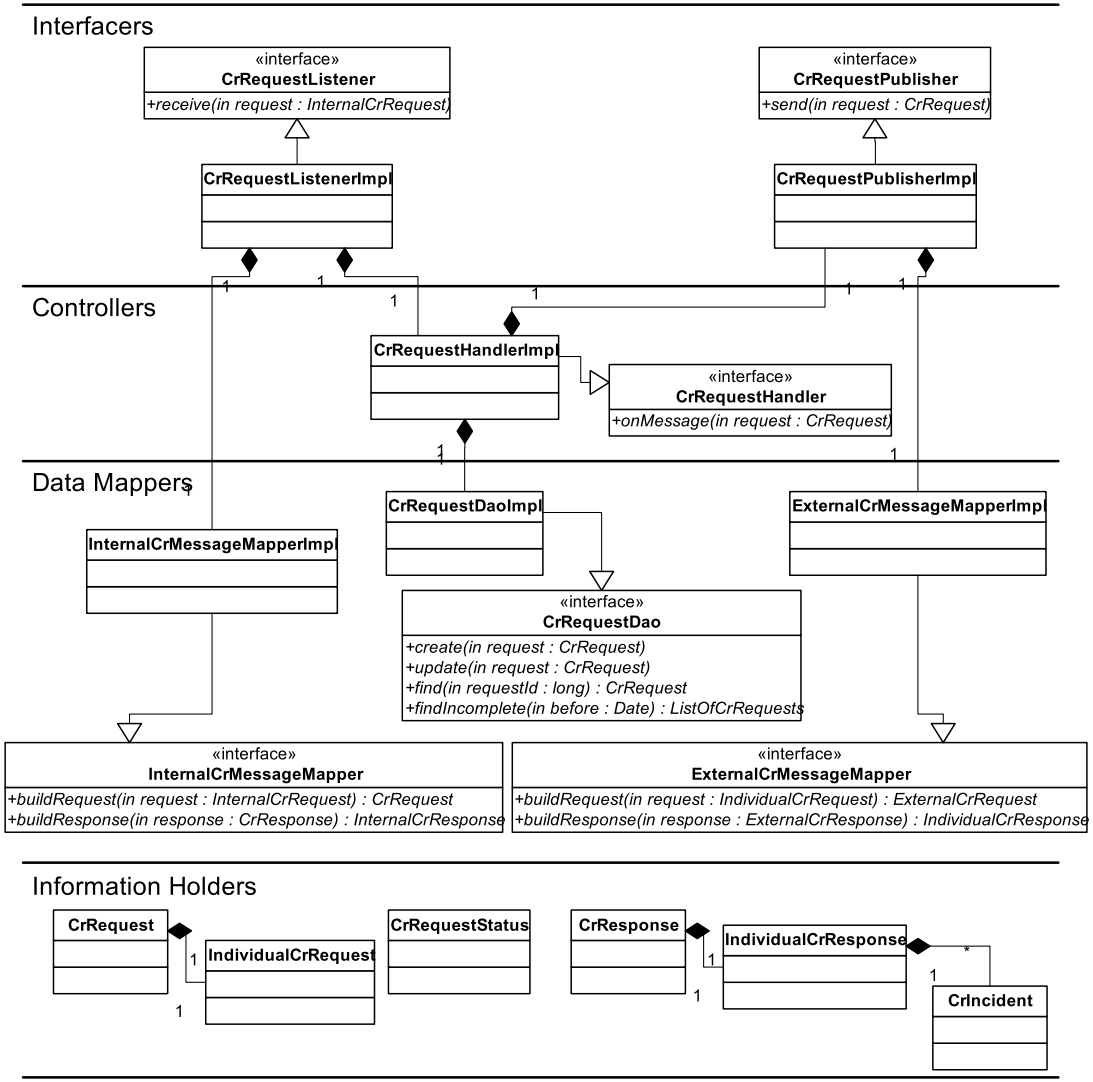
Canonical domain model

- Standard representation of requests and responses
- Usually neither internal nor external format
- Isolates handlers from data not needed by the service
- Isolates handlers' logic from message formats

Code to write

- Interfaces
- Classes
- Unit tests
- Mock Objects

Implementation classes

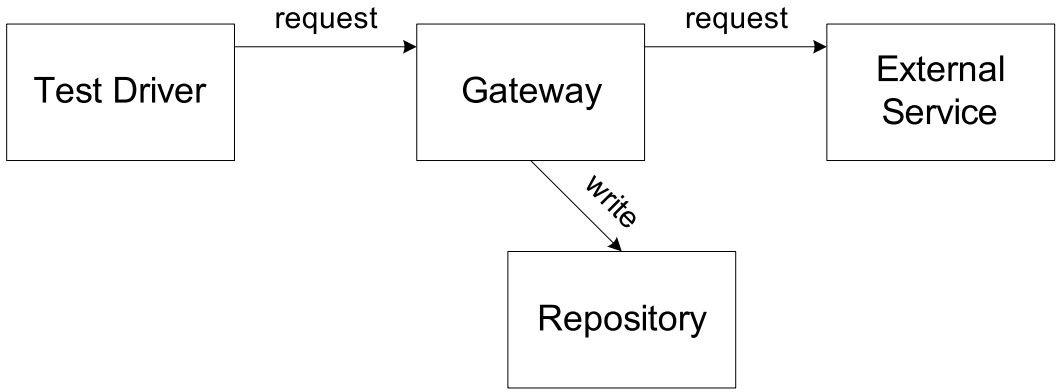


Unit testing

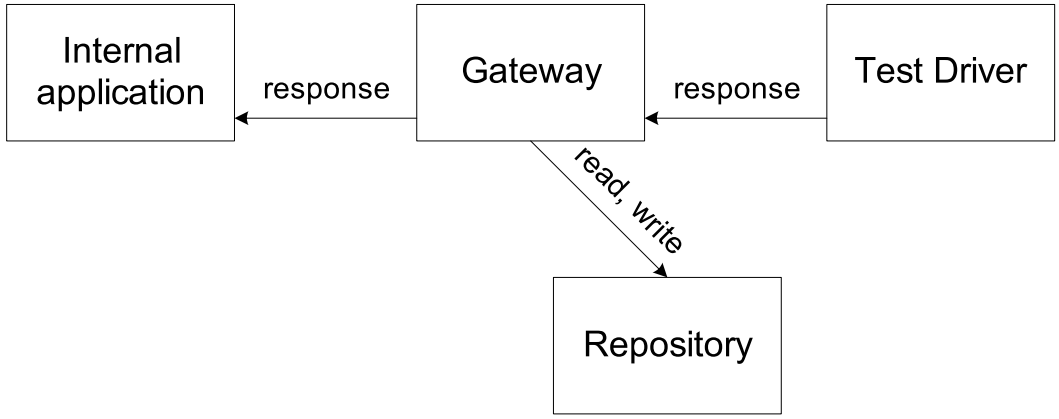
- Each unit of functionality specified with an interface
  - Replace dependencies with mock objects (or mock services) for unit testing
- Controllers, mappers, and information holders are independent of any container
  - Test in a bare JVM

Integration testing

Testing outgoing messages



Testing incoming messages



Error handling

- Same issues as any service
- Distinguish business errors:
  invalid zip code
  from infrastructure errors:
  ill-formed request
  service not available
- Should translate external error codes/ messages to internal format

Caching

If caching is appropriate for the domain …

- Gateway provides a place to cache results from the external service
- Probably improves performance
- Avoids repetitive calls to external service, which might charge per request

Advantages

Gateway provides central location to:

- Manage a long-running process
- Translate messages
- Encourage separation of business logic from request/response protocol
- Encapsulate connection information

…Advantages

Gateway provides central location to:

- Collect data to audit SLA with outside vendor
- Collect data for charge-back to client organizations
- Cache responses to:
  - improve performance
  - save money by reducing usage of external service

Advantages of this design

- Separation of concerns:
  - Sending/Receiving
  - Message mapping
  - Managing the workflow
- Flexibility for hot spots
- Small, independent classes may be unit tested
- Much of it is testable outside a container

Disadvantages

Implementing a Gateway as a service:

- Increases layers
- Another trip on the network
- Another location that marshals, unmarshals, and translates
- Another distributed component to deploy, configure, and monitor

References

- Fowler, Martin. 2003. Patterns of enterprise application architecture. Addison-Wesley.

- See patterns:
  - Gateway
  - Service Interface
  - Service Stub
- Brief descriptions are available online: www.martinfowler.com/eaaCatalog/

…References

- Gamma, Erich, et al. 1995. Design patterns: Elements of reusable object-oriented software. Addison-Wesley.

- Gateway pattern is similar to several other patterns:
  - Adapter
  - Façade
  - Mediator

…References

- Hutchison, Beth, et al. (2005). SOA programming model for implementing Web services, Part 4: An introduction to the IBM Enterprise Service Bus.
- See discussion of Gateway pattern and types of mediation.
- Available online at:
  www.ibm.com/developerworks/webservices/library/ws-soa-progmodel4/

…References

- Trowbridge, David, et al. 2003. Enterprise solution patterns using Microsoft .NET: Patterns and practices, Version 2.0. Microsoft Corporation.
- Chapter 6 includes discussion of Service Interface pattern and Service Gateway pattern.
- Available online at: www.microsoft.com/downloads/details.aspx?FamilyId=3C81C38E-ABFC-484F-A076-CF99B3485754&displaylang=en

…References

- Wahli, Ueli, et al. 2005. WebSphere version 6: Web services handbook: Development and deployment. IBM Redbook SG24-6461.
- Chapter 22 “Web services and the enterprise integration bus” provides an architecture to expose external web services on an ESB.
- Available online at: www.redbooks.ibm.com (search for “SG24-6461”)

…References

- Wirfs-Brock, Rebecca, and McKean, Alan. 2003. Object design: Roles, responsibilities, and collaborations. Addison-Wesley.
- General discussion of OO design principles and “responsibility-driven design”.