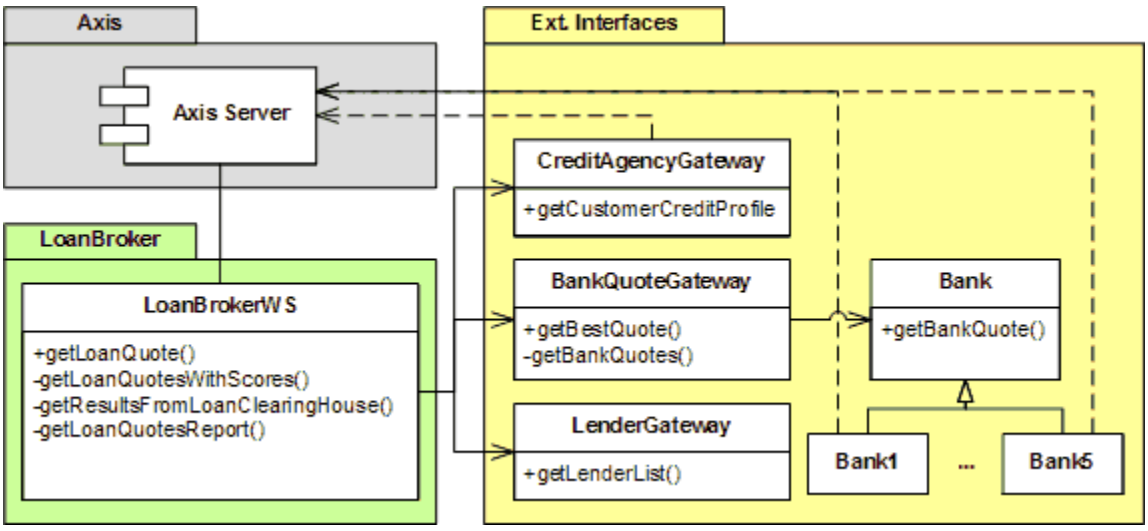
THE LOAN BROKER APPLICATION

The figure below shows the class diagram for the loan broker component. The core business logic for the loan broker is encapsulated inside the LoanBrokerWS class. This class inherits from a class provided by the Axis framework that implements a *Service Activator* to invoke a method in the LoanBrokerWS class when a SOAP request arrives. The LoanBrokerWS references a set of gateway classes that implement details of interfacing with external entities, such as the credit agency and the banks. This logic is encapsulated inside the classes CreditAgencyGateway, LenderGateway, BankQuoteGateway.



Loan Broker Class Diagram

The only service interface from the loan broker to the outside world is the message endpoint for the client to access the loan broker service.

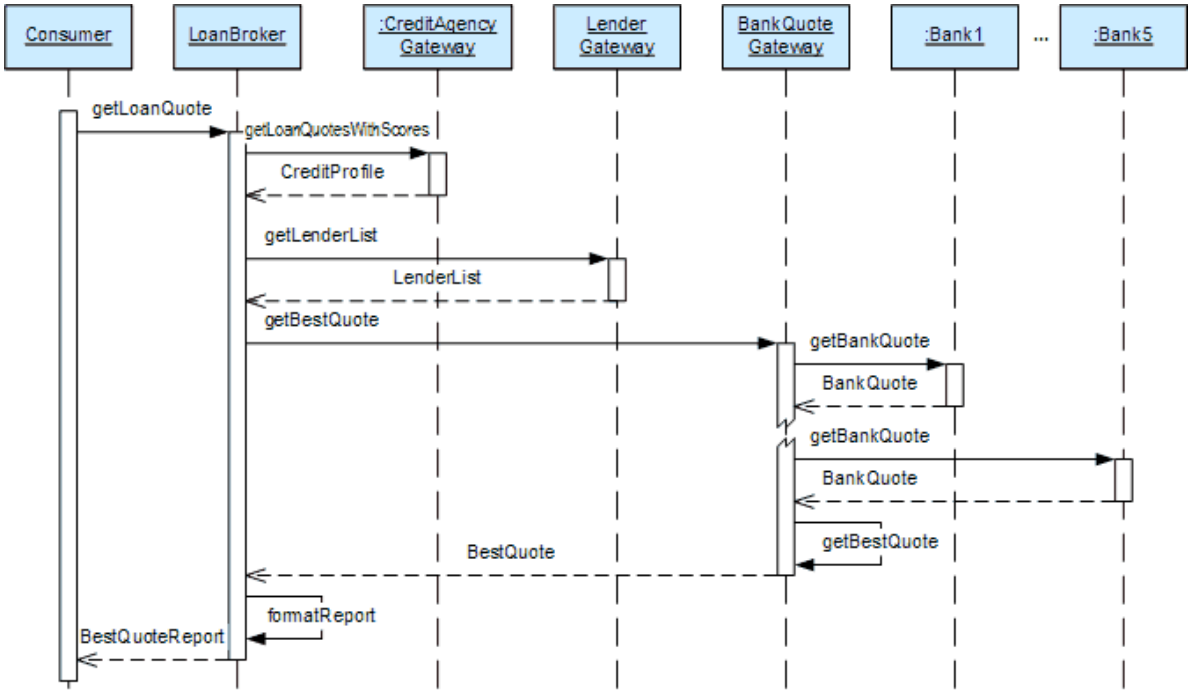
There is no need to define a *Messaging Gateway* since the Axis framework acts as the gateway on behalf of the application.

The following sequence diagram illustrates the interaction between the components for the synchronous predictive Web service example. The loan broker first calls the credit agency gateway component, which enriches the minimum data provided with a credit score and the length of credit history for the customer. The loan broker uses the enriched data to call the lender gateway. This component implements the *Recipient List* that uses all the data provided to choose the set of lenders that can service the loan request.

The loan broker then calls the bank quote gateway. This component performs the predictive routing operation by calling each bank in turn. The bank components model the interface to a real-world banking operation. For example the Bank1 class is the interface to the Bank1WS.jws Web Service that models the banking operation. When a loan request comes in, there is some amount of clerical work performed before generating the rate quote based on all the parameters. In this example, the rate quote is generated by a banking Web service.

The bank quote gateway aggregates the responses from the banks and chooses the best quote from all the quotes received.

The response is sent back to the loan broker, which formats the data from the best quote and returns the report to the client.



Loan Broker Sequence Diagram

We have only shown two of the five banks in the sequence diagram in the interest of keeping the picture manageable. Based on the *Recipient List* generated for a particular request, the loan broker could contact exactly one bank or all five banks to service the client request.

The diagram highlights the sequential processing of the quote requests to each bank. The sequential processing has some advantages because we can run the application in a single thread calling each bank in turn.

The application does not need to spawn threads to request a quote from each bank and so we don’t have to worry about concurrency issues. However, the total time to get the response quote is high since the loan broker has to wait for a response from each bank before submitting a request to the next bank in the lenders list. The net result is the customer will have to wait a long time to get the result quote back after submitting the request.

Source:

http://www.enterpriseintegrationpatterns.com/patterns/messaging/ComposedMessagingWS.html