Flexible Extensible Service Tier
An OOPSLA 2001 Design Fest Problem

Dave Busse
Sema Telecoms
dbusse@acm.org

Abstract

The problem is to design a Service Tier that provides a flexible and extensible collection of services for the Presentation Tier of a set of WEB applications. The services provided can be used independently, some of the services can be used together (a set of services) in a certain order. The problem is to design a Service Tier that allows the number and kinds of services as well as sets of services to be changed and/or extended easily.

Domain Description

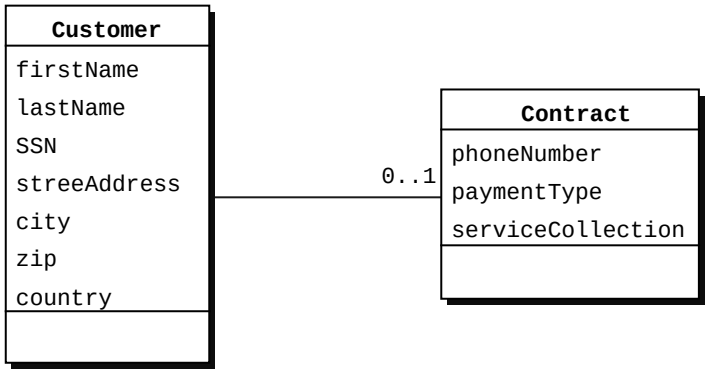
Mobile Telephone Network Operators are faced with a continuous demand for more services and increased customer convenience. A Network Operator often has deployed one or more legacy software systems that together provide the services needed for the business. These various systems have interfaces that are different from each other. We will limit our problem here to Customer Care systems. Such systems are used to add new customers and creating new contracts as well as allowing customer care representatives to answer questions for existing customers and make changes to their services. An example of a change to a customers cell phone services would be to increase the number of free minutes (for a price of course), add or drop Short Message Service, add or drop Voice Mail, etc.

There is increasing demand to provide some or all of the Customer Care software on the WEB. Such WEB applications often have architecture made up of several tiers. The Presentation Tier consists of a WEB browser and WEB server. The Service Tier consists of a program that provides services to the Presentation Tier using a protocol (such as RMI). In our case the Service Tier hides the various legacy systems and databases behind a consistent service oriented API.

In this problem, service oriented API refers to the style of the interface. In the legacy system one might need to make a dozen calls to the API to accomplish creation of a new customer. In our proposed service oriented API the Presentation Tier would make one call to our Service Tier (createCustomer).

The Data Tier contains a database and/or legacy applications that the Service Tier accesses to read and write the data the Presentation Tier needs. Some of these legacy applications provide customer and mobile phone service contract information. Some of these systems do blacklist validation (bad risk), or address validation checking.

Analysis Diagrams



Description of the Problem

Your team's task is to design a Service Tier in the form of a program that provides services that are needed by the Presentation Tier. The key element to the solution is to make it very easy to change or add individual services. Also, it must be possible to arrange individual services into sets of services that have to be completed successfully in a certain order. The details of the API that the Service Tier presents to the Presentation Tier are left to your team's discretion. However, it is desired that the API be "service oriented". For example, one call to the API to create a customer instead of many calls.

Your team can assume that there is some reasonable interface to the Data Tier and may either ignore such details, or assume whatever is convenient. The core problem is to provide a flexible, extensible set of services. New individual services are added by changing the Service Tier code. It must be easy to add a new service without disrupting the Service Tier design. The existing applications that use the Service Tier should be unaffected by addition of new services to the Service Tier. It must be easy to create sets of services. These sets of services appear to be ordinary services to the Presentation Tier. See the Detailed Requirements section for more information.

Multi-Tier systems are often designed to address scalability issues (add more instances of the Service Tier as numbers of users increase). You need not consider this issue in your design.

Detailed Requirements

- The Wireless Network Operator for whom we are building this software has a legacy Customer Care system with a rich set of APIs. In addition there is a system that does Address Validation, and a BlackList database that keeps track of former customers that are considered to be poor risks. In fact there are more services available in the Legacy System than the Marketing Department thinks they will need for the WEB applications that they are requesting. The API to the legacy system is up to your team. You can assume its CORBA IDL or RMI or some other type of interface. The details of this API are not part of this problem. You can assume whatever API detail is convenient for your solution.
- Marketing is planning a number of applications. Some of these applications have had their requirements completed and some are still in a "back of the envelope" state. The Service Tier your team is designing must serve all of the applications. We need the Service Tier to be easily extended to accommodate changes to the current application requirements and new requirements for the applications still being specified. Examples of services we know we will need include:
    - CreateCustomer

  - Create Contract for a customer
  - Modifiy Customer information
  - Modify Customer's Contract information
  - Search for a Customer by name, address, etc.
  - Customer Address Validation (is this a real address)
  - BlackList Validation (is the customer in the BlackList database)
- A customer must be created before a contract can be created for her. A contract defines the telephone number, the kinds of services that the telephone number has (call forwarding, caller ID, voice mail, number of free minutes per month, carry over of unused free minutes, etc) and how the customer will pay for the contract.
- A customer can have zero, one or more that one contract. There is one cell phone (and thus one phone number) per contract.
- A contract has to be created before it can be added to the customer. A CreateContract service would take a phone number, and a collection of services and define how the contract is to be paid for (credit card, direct debit).
- The legacy system has an API that returns a random available phone number, or will return a list of available phone numbers given an area code and prefix.
- All services provided by the Service Tier can be called individually by the Presentation Tier.
- All services that will be provided by the Service Tier take some number of parameters. The number of parameters depends on the service. For example the Create Customer service takes many parameters including customer name, address, etc. The Customer Address Validation service takes far fewer parameters (just an address).
- All services that will be provided by the Service Tier will return a result. The exact contents of the result will depend on the service. For example, Create Contract service will return success or failure. In the case of failure an error message is also returned. A Search For Customer service will return a collection of Customer matches.
- Services can be assembled into sets. A set of services can be requested in just the same way as a single service. In fact the Presentation Tier can not tell the difference between calling a set of services and calling an individual service.
- The services in a set must be executed in a particular order defined by the set (an ordered set).
- Services can be thought to have a required property. If a service is used individually, the property has no meaning. If a service in a set has its required property set to false the service does not have to succeed for the set to succeed. For example, if the Create Customer Set is a set of services that includes some validation services in addition to the individual Create Customer service, it is possible that some of the validation services can fail, and it is still permissible to create the customer. In some applications, the Address Validation service can fail (it's used for informational purposes only) and the Customer can be created. In some applications, it is required that the Address Validation succeeds for the set to succeed. This property can be turned on or off at program initialization time.
- Services can be thought to have an execute property. If a service is used individually, this property has no meaning. If a service in a set has its execute property set to false, then when a service set is executed, that service is skipped. The execute property for a service in a set can be set at run time by the Presentation Tier of an application via the Service Tier API.

- Sets of services can be nested. For example, a Validation Set can be made up of a number of validation services. A Create Contract Set can be made up of a Validation Set plus and individual Create Contract Service.
- Individual services are defined at compile time. It must be easy for the Service Tier programmers to add new individual services.
- Service Sets can be defined at application start-up time. The Service Tier will provide an initialization method in its API that allows service sets to be defined by reading a parameter file. The parameter file format can be XML or some other format. This detail is left to the Design Team.
- Sets of services have optional and execute attributes the same as individual services.

Use-Cases

The following use-cases illustrate how the applications will use the Service Tier.

1. The application starts with its parameter file that defines a set of services that includes Address Validation, Black List Validation, and Create Customer, in that order. Address Validation is not required.
2. This use-case extends use-case 1. The application creates a customer with a bad address. The Presentation Tier is informed of the Address Validation failure as well as the success of Black List Validation (the customer name and SSN is not in the Black List database) and the customer is created.
3. This use-case extends use-case 1. The application searches for the customer just created and displays it.
4. This use-case extends use-case 3. After finding and display the customer, a contract is created and added to the customer.
5. This use-case extends use-case 2 by changing the customer or contract information displayed and updating the system with these changes.
6. This use-case extends use-case 1. The application has an GUI that allows the use to turn Address Validation and BlackList Validation on or off by using check boxes. The user un-checks both boxes and then creates a customer. The Service layer skips both validations and creates the customer.
7. The application starts with its parameter file that defines a set of validation services. It also defines a set of services that includes first the validation set and then the individual create customer set.
8. This use-case extends use-case 7. The application creates a customer with a bad address. The Presentation Tier is informed of the Address Validation failure as well as the success of Black List Validation (the customer name and SSN is not in the Black List database) and the customer is created.

Extending the Design

The key issue in this problem is to make a flexible, extensible design. Test your design by adding a new service, Credit Check. How minimal a disruption to the original design is this? How would the already existing applications be effected?

Another test would be to create some more use-cases that require various nested sets of services to be created and see if your design would accommodate these use-cases.