# Recipe for Service Component (R4SC)

*Realizing the recipe with SCA and SDO*

**Authors:**

Michael Beisiegel, Distinguished Engineer, IBM Software Group
Javier Garcia, IT Architect, IBM Software Group
German Goldszmidt, STSM, IBM Software Group
Carl Osipov, IT Architect, IBM Software Group
Brian Paulsen, Executive Architect, IBM Global Business Services

Draft Version 1.0
January 18, 2007

## Introduction and Intended Audience:

The Recipe for Service Component is defined at a conceptual design level, independent of product and technology decisions.  The objective of R4SC is to define a design framework that can be uniformly applied regardless of the client development and production environment technologies.  Through technique papers, we add the prescriptive detail of how to realize R4SC with specific products and technology.  This technique paper assumes you are familiar with the SOMA Recipe for Service Component (R4SC), and focuses on how you would realize the R4SC design framework with SCA's and SDO's.

## R4SC Patterns

Service Oriented Modeling and Architecture (SOMA) is IBM Global Services method and architectural approach for identifying, specifying, and realizing SOA solutions. Within SOMA, the Recipe for Service Component focuses on the prescriptive technique for taking the requirements gathered during the Identification and Specification phase of SOMA and leveraging those requirements to implement new/enhance existing Service Components. At the core of the Recipe for Service Components are two patterns:

1. the Service Component Pattern and
2. the Service Integration Pattern

The Service Component Pattern focuses on the internal structure of the Service Component, the entities that must be present to provide proper separation of responsibility in the delivery of the Service Component, and how they interact together to fulfill a Service Component.

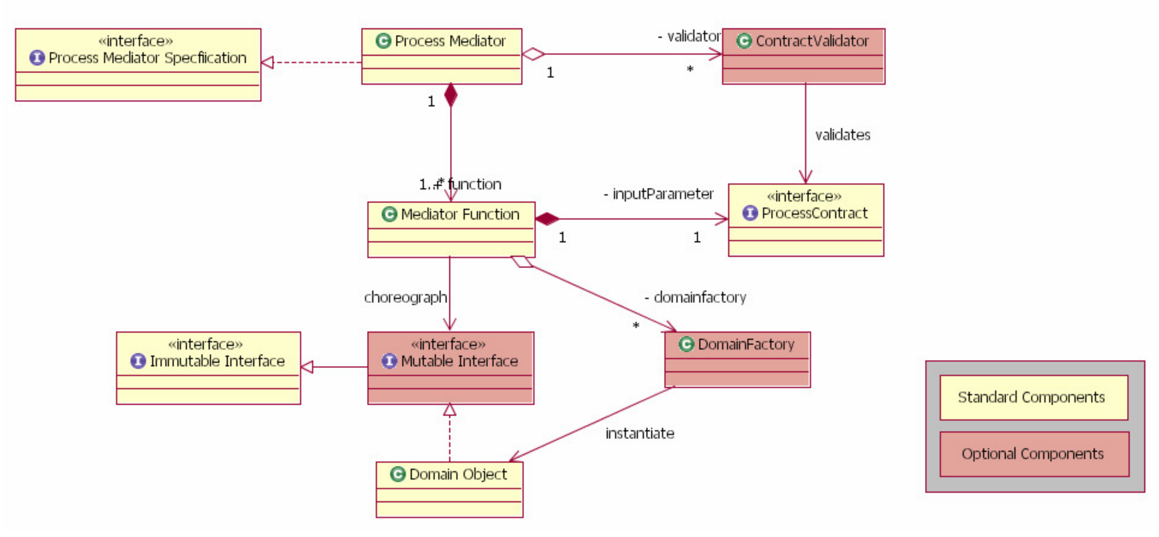

**Figure 1: Service Component Pattern**

One potential point of confusion with those familiar with Web Sphere Process Server and/or the ESB is the use of the term Mediator. As we defined the entities in the R4SC patterns, we leveraged design pattern terms to reflect the roles of each entity. Therefore, the Process Mediator and Mediator Functions are focused on the roll of the design pattern, Mediator: "Defines an object that encapsulates how a set of objects interact. Mediator Promotes loose coupling by keeping objects from referring to each other explicitly and it lets you vary their interaction independently". The Process Mediator and Mediator Function are responsible for managing the micro flow of objects within the service component for a given process, and should not be confused with the WebSphere Process Service and ESB concepts of Mediation. We will discuss how R4SC takes advantage of SCA Mediation Components later in this paper.

The Service Integration Pattern focuses on the mechanisms for a consumer to access a service component via a SOA service. From the Provider perspective, the pattern addresses the responsibilities of realizing the SOA service definition, transforming the data from the input format specified by the SOA service definition to the format required by the service component implementation, and finally transforming the output data from the service component back into the return format dictated by the SOA service definition.

From a Consumer perspective, the pattern addresses the implementation of the SOA service definition, binding the service to the integrated service component, adapting the input to the format required by the integrated service component, and adapting the resulting data from the service component to the return format dictated by the SOA service definition.

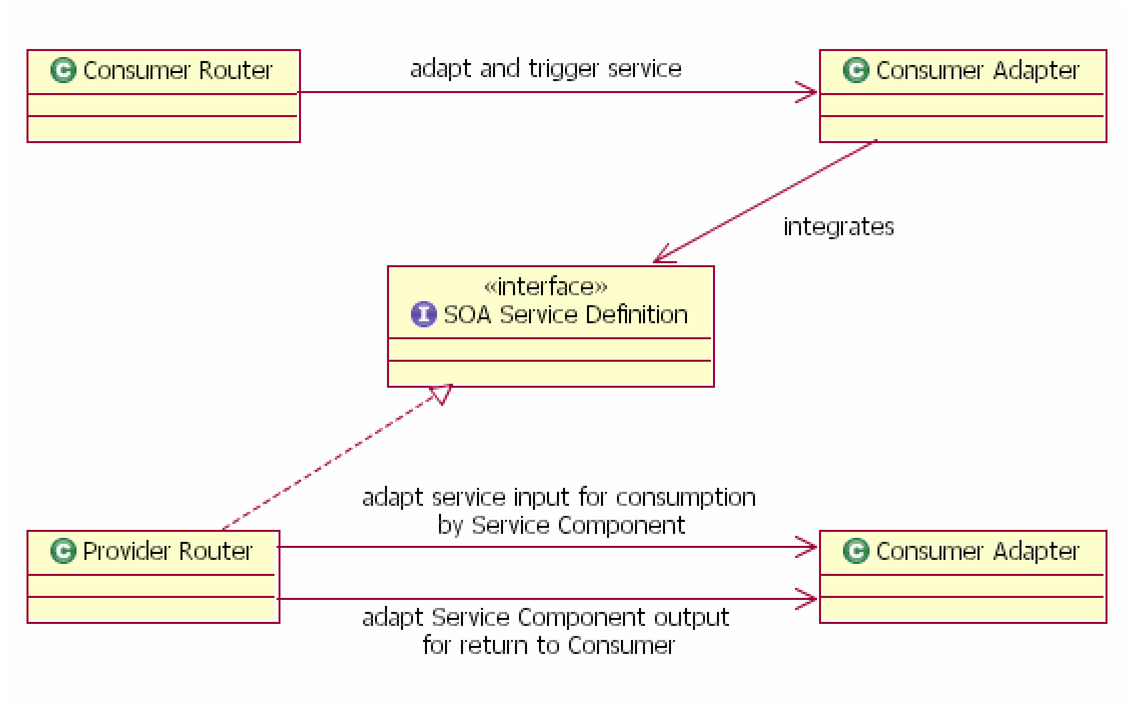
Consumer Router
adapt and trigger service
Consumer Adapter
integrates
«interface»
SOA Service Definition
adapt service input for consumption by Service Component
Provider Router
Consumer Adapter
adapt Service Component output for return to Consumer

**Figure 2 - Service Integration Pattern**

A core principle of the Recipe for Service Components (R4SC) is to remain flexible to technology and products to allow for a consistent architectural approach regardless of the client environment. For the Recipes of SOMA, there will be technique papers written to address specific technologies, specifications, and tools. From an IBM and industry perspective, two of the more significant specifications in the SOA modeling space are the Service Component Architecture (SCA) specification and the Service Data Object (SDO) specification. This paper will focus on how we incorporate SCA and SDO into a SOMA solution with the Recipe for Service Components.

## R4SC Integration of SCA:

SCA defines three major concepts of a service component implementation: the Service, References, and Properties. While the service definition is fixed, the references and properties are configurable to enable flexibility of the assembly.

In the R4SC, the key elements for implementing the SCA are the Process Mediator and the Process Router. The Process Mediator (see Figure 1, entity at the top, center of the image) and its' Process Mediator Functions embody the capabilities of the Service Component that are available for exposure as SOA services. The Process Mediator Functions contain the micro flow logic for completing these fine grained operations. The Process Router (see Figure 2, Process Router exists both on the Consumer and Provider side of the relationship) is responsible for managing the translation and binding of SOA services to these underlying Service Components.

The Process Mediator represents the Component Implementation, and an Interface with the service capabilities would need to be defined that the Process Mediator would implement. The capabilities on the Interface that would be exposed as Services are dictated by the Services that were identified and specified for exposure during the first two stages (Identification and Specification) of the SOMA method. The Process Mediator may leverage other SOA services, which would be captured as References within SCA. No standard Properties are mandated by the R4SC, but Properties may be assigned on a case by case basis, where the architect deems them valuable.

The Process Router is the focal point where SCA is applied in the R4SC. In SCA Assembly terms, the Process Router becomes the Entry Point on the provider side and the External Service on the requestor side of the SOA relationship.

Finally, SCA introduces two concepts of Service Assembly: Programming in the Large, and Programming in the Small. Programming in the Large focuses on the assembly of business solutions as networks of loosely connected services working together, while programming in the Small focuses on assembling services from closely related fine-grained components.

In the world of SOA services, one person's fine grained component is another person's large grained service, as a result we developed the R4SC patterns anticipating that when you view the structure of a "large grained service" you would see the patterns repeated at the higher order of magnitude, likely using different implementation technologies to accomplish the results. For example, you might see BPEL used to realize the patterns of the Recipe for Service Component to coordinate larger-grained, higher-order services (programming in the large), while you might see the patterns implemented in Java POJO's when you dig into the fine grained service components that are integrated to deliver the fine integrated service (programming in the small).

## R4SC Integration of SDO:

The SDO concept maps closely to the relationship between the Process Mediator and the Domain Objects of the R4SC. SDO defines a Data Object as a "generic representation of a Business Object and is not tied to a specific persistence storage mechanism". The SDO concept of a Data Graph refers to a Data Object that directly or indirectly contains all the other Data Objects in the graph. The Domain Object of the R4SC, in SDO terms, is directly realizable via the Data Object. A complex, aggregate Domain Object would represent an SDO Data Graph.

So, at a pattern level, how does the SCA relate to the SDO in R4SC? The end user makes a request for information, which is processed via the Process Router (the SCA component entry point) and forwarded to the appropriate Process Mediator Function (the SCA Component Implementation). The Process Mediator Function coordinates requests to the Domain Objects (the SDO Data Object/Data Graph) managing the logical unit of work. Domain Objects are retrieved and their Data Graphs populated from the appropriate persistence stores. The Process Mediator Function collects the Domain Objects from its micro flow and, when necessary may create a new data graph to package the requested information, which is then returned to the Process Router. It is necessary for the Process Mediator Function to create a new Data Graph in situations where the return information is not a self-contained, aggregate data object (e.g., Data Graph).

With the Data Graph returned from the Process Mediator Function, the Process Router triggers the appropriate Process Adapter to transform the data into the format mandated by the SOA service description and technology used.  The transformed data graph information is then returned to the requestor via the Process Router.  Subsequent requests to the Process Mediator Functions are managed through the Process Router to bind to the appropriate concrete Component Implementation (e.g., Process Mediator Function).  Input data is passed in according to the SOA Service interface and translated to the structure, a Process Contract, expected by the Process Mediator Function.

Keeping in line with one of the primary objectives of SCA/SDO, flexibility through decoupling, the R4SC also introduces two interfaces to decouple the solution further from the implementation of the SDO, the Immutable and Mutable Interfaces.  The Immutable Interface defines a specification, independent of concrete implementation of the SDO that allows the user to access information of the SDO, but not change its state.  The Immutable Interface is intended as the return value of the Process Mediator Function (controller) to the Process Router (View) to preserve the integrity of the Model-View-Controller framework.

The Mutable Interface extends the Immutable Interface of the SDO.  It's purpose is to provide a specification that decouples the Process Mediator Function from the concrete implementation of the SDO, while providing the ability to change the internal state of the SDO.  The Mutable Interface allows the Process Mediator Functions (the SCA Component Implementation) to perform its service without becoming coupled to a specific implementation of the SDO.  This offers the same benefit of runtime, configurable binding via dependency injection between the service component process and the service data object, as is present in SCA via the interface specification of the Service Component and the Component Implementation.

## SCA/SDO applied in the R4SC Custom Application Development Scenario

In this section, we will discuss one of the most common engagement scenarios where the patterns are pulled together to deliver an end to end application architecture.  To learn more about the architectural decisions and components of the R4SC Custom Application Development Scenario, please see the paper "SOMA Recipe for Service Component (R4SC)" section Custom Application Development Scenario.

The following image is the layered architecture for the R4SC Custom Application Development Scenario, look closely and you will see the Service Integration Pattern on top of the Service Component Pattern, which sits on top of a second application of the Service Component Pattern.

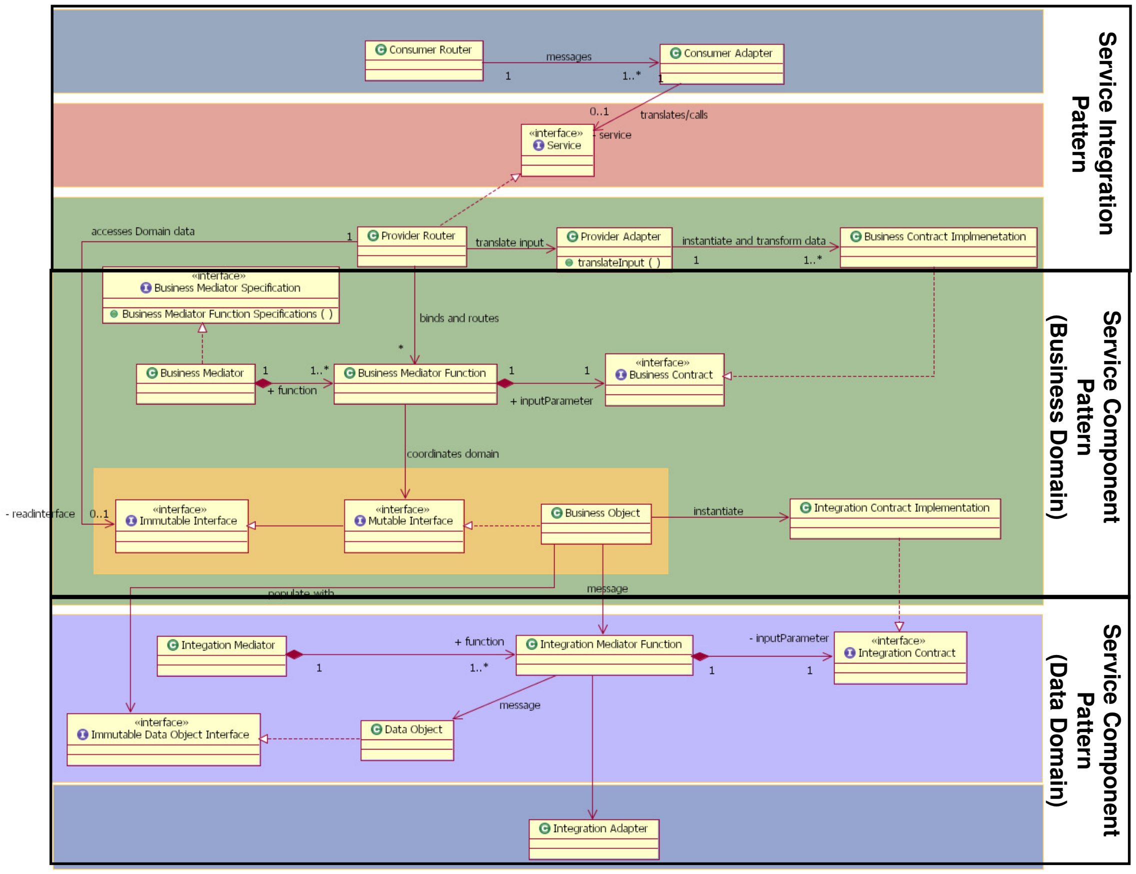
Service Integration Pattern
Service Component Pattern (Business Domain)
Service Component Pattern (Data Domain)
Consumer Router
messages
1
1..*
Consumer Adapter
1
0..1
translates/calls
«interface» Service
service
accesses Domain data
1
Provider Router
translate input
Provider Adapter
translateInput ( )
instantiate and transform data
1
1..*
Business Contract Implmenetation
«interface» Business Mediator Specification
Business Mediator Function Specifications ( )
binds and routes
*
Business Mediator
1
1..*
+ function
Business Mediator Function
1
1
+ inputParameter
«interface» Business Contract
coordinates domain
- readinterface
0..1
«interface» Immutable Interface
«interface» Mutable Interface
Business Object
instantiate
Integration Contract Implementation
populate with
message
Integration Mediator
+ function
1
1..*
Integration Mediator Function
- inputParameter
1
1
«interface» Integration Contract
message
«interface» Immutable Data Object Interface
Data Object
Integration Adapter

**Figure 3 - R4SC Custom Application Development Scenario**

Since we applied the Service Component Pattern for two domains, we have tailored the component names of the pattern to delineate their difference in role. In the Business Domain application of the SC pattern, the Process Mediator, Process Mediator Function, Mediator Contract, and Domain Object names have been changed to reflect the Business domain (e.g., Business Mediator, Business Mediator Function, Business Contract, and Business Object). The focus of the Business Domain is on business process logic (Business Mediator/Business Mediator Functions) and the Business entity logic (Business Objects). From an SCA perspective these components represent the idea of a Business Component with the Business Mediator Function performing the role as the Component Implementation.

Likewise, the component names in the Data Domain application of the Service Component Pattern reflect the focus on integration and data (e.g., Integration Mediator, Integration Mediator Function, Integration Contract, and Data Object). The Integration domain is focused on the integration logical unit of work (Integration Function), the data (Data Object), and the technology adaptation of database, middleware, operational system, and SOA service technologies, etc. (Integration Adapter). These entities work together to deliver the SCA concept of the Mediation Component. Earlier in the document, we discussed that at the pattern level the Process Mediator and Mediator Functions were not synonymous with the SCA concept of Mediation. The reason for drawing the distinction is the fact that the R4SC Service Component Pattern can be applied in different domains. It is when it is applied in the integration domain that it fulfills the role of the Mediation Component.

Both domains represent SCA/SDO combinations, one focused on delivering a business Service Component, and the other delivering a data-oriented Service Component. With the understanding of how the patterns work together for the Custom Application Development scenario, now we must explore how we take this design framework of the application and realize it with the SCA and SDO specifications.

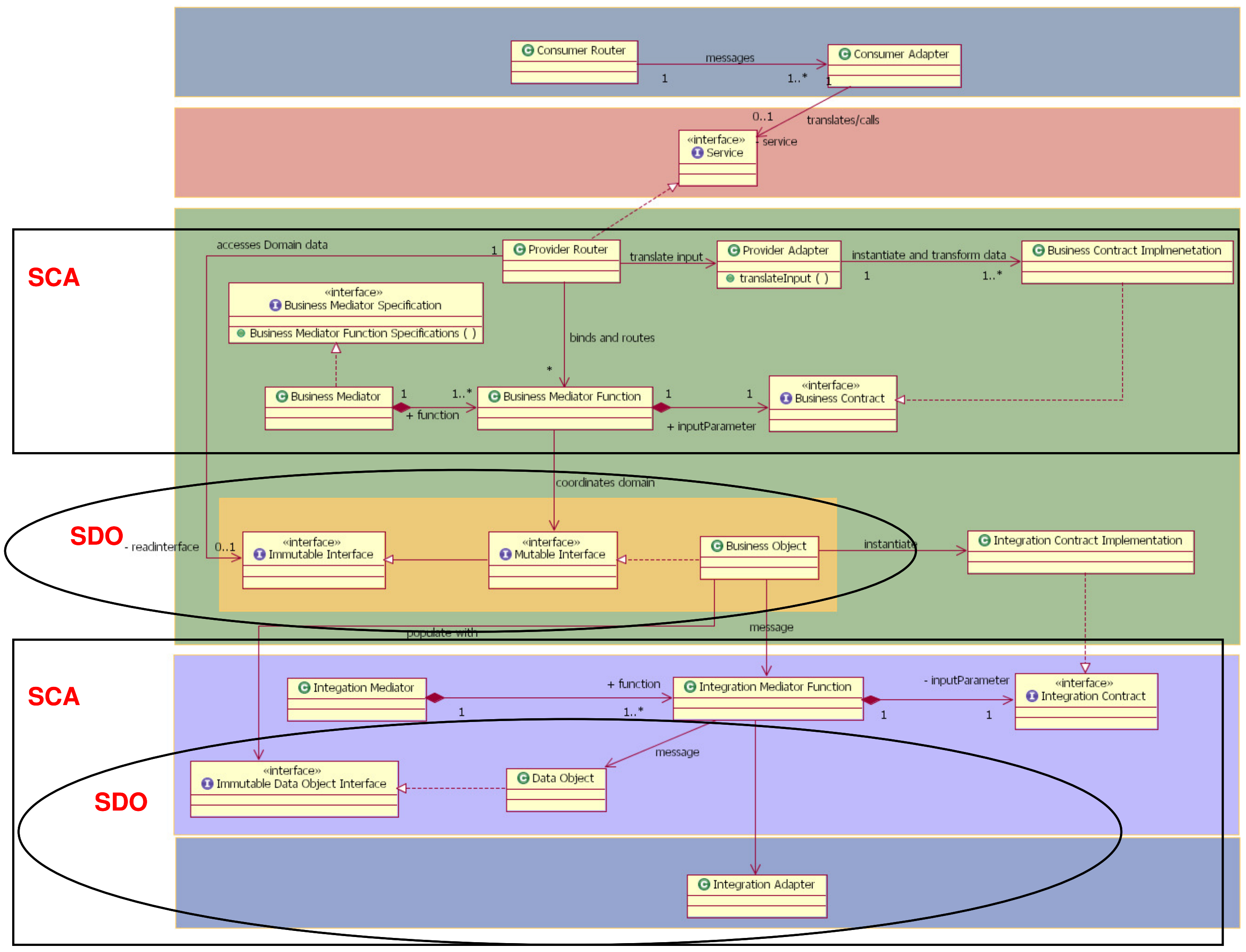

**Figure 4 - SCA/SDO integration in the R4SC Custom Application Development Scenario**

As discussed in the pattern section, the Provider Router (top, middle of the green box) represents the Entry Point to the Service Component. It is responsible for implementing the SOA service definition and managing the binding of the service to the underlying implementation of the Service Component (in this case, a Business Mediator Function). In SCA terms, the Business Mediator Function represents the Component Implementation, which has the responsibility of coordinating the Service Component activities to deliver the component service. Finally, the Business Object represents a SDO focused on the knowledge of the entities business rules and responsibilities.

The SCA box that you see surrounding the blue boxes at the bottom of the image represents a specific type of SCA called a Mediation Component. The Integration Mediator Function represents the Mediation Service (a specialized type of Component Implementation). Finally, the Integration Adapter is the implementation of the SDO concept of Data Access Service, responsible for encapsulating the technology specifics of the integrated technology (e.g., relational database, middleware, operational systems, SOA services, etc.). The Data Object is the implementation of the SDO while the Integration Data Object Interface is an interface specification independent of implementation, which is primarily used as a return value from the Integration Mediator Function preserves the MVC separation between the business and data/integration domains.

One of the potential end points integrated by the Integration Adapter is an SOA service, itself. In that case the Integration Mediator Function is also fulfilling the logical role of the Consumer Router of the R4SC Service Integration Pattern, and the Integration Adapter the role of the Consumer Adapter. From a pattern perspective, this results in the Integration Adapter being routed by the SOA service back to the top of the picture in the R4SC Custom Application Development (Figure 4, above).

References:

Gamma, E., Helm, R., Johnson, R., Vlissides, J*., Design Patterns: Elements of reusable Object-oriented Software*, Addison-Wesley, 1994.

Ang, J, Ballentine, B, Paulsen, B, *SOMA Recipe for Service Component*, September 2006

Beisiegel, M. et.al., *Service Component Architecture: Assembly Model Specification*, version 0.9, November 2005

Beisiegel, M., et. al., *Service Component Architecture: Building Systems using a Service Oriented Architecture*, version 0.9, November 2005

Beatty, J., et. al., *Service Data Objects for Java Specification*, version 2.01, November 2005