Component Specification Technique Paper

Draft

Version 0.2

SOMA Core Team

Document History

Document Location

Revision History

Date of this revision:	Date of next revision *(date)*

Revision Number	Revision Date	Summary of Changes	Author
0.1	7-12-2008	Created	V. M. Ramos
0.2	10-16-2008	Updated to align with method content for SOMA 3.2	Shuvanker Ghosh

1. Introduction

This technique paper will describe the service component specification activity of the specification phase of SOMA.

Definition of the various types of components:

1. Service Component - Is a component which exposes a service interface. A service component may also be a technical or functional component. A service component may be just a wrapper for a technical or functional component or it may provide some control / coordination / choreography between components.
2. Functional Component - is a component that implements part or all of a use case.
3. Technical Component - is a component that implements infrastructure functions such as: logging, error handling, queuing, message routing, etc.

Any component may be implemented by custom code, package, frameworks, legacy, etc.

Service, functional & technical components are essentially the same type of thing, only differentiated by specialty (service exposure, infrastructure or use case functionality).

From the Architecture Description Standard: A component is a large grained thing that offers interfaces via contract. Ideally, a component should have cohesive functionality.

2. Purpose

The purpose of Service Component Specification is to design and specify the service components that were identified / created during subsystem analysis. Component Specification is an activity that creates the details of the (Service) Component Model. Designing and specifying a service component includes identifying the functional and technical components (Which may be done during the Subsystem Analysis activity) as well as the interaction and the data messages that will be passed between the components.

The Component Model will be further refined during SOMA Realization phase, where decisions about haw a component will be implemented will be made.

3. Impact of Not Using This Technique

If this technique is not used:

- The developers / assemblers will not know what to code.

- Redundant functionality may be left in or built into the applications.
- Risk of developing monolithic, non-cohesive applications
- The development team may deviate from the architecture.

4. Description

Component specification describes detailed attributes, internal flow, structure and variations of service components, including their constituent functional and technical components. After the boundaries of the service components, their functional and technical components and services have been identified through subsystem analysis, the detailed structures of service components are specified.

5. Usage

5.1 When to Use It

Service Component modeling is used when a new service component is identified to support a business process, or the underlying functional or technical components are modified

5.2 How to Use It

A Service Component will have a set of attributes that need to be defined at the specification level. In this step we will define the details of each Service Component, including attributes, services, policies, and rules. Figure 5-1 Component Specification tasks shows the activities and tasks for developing a component specification.

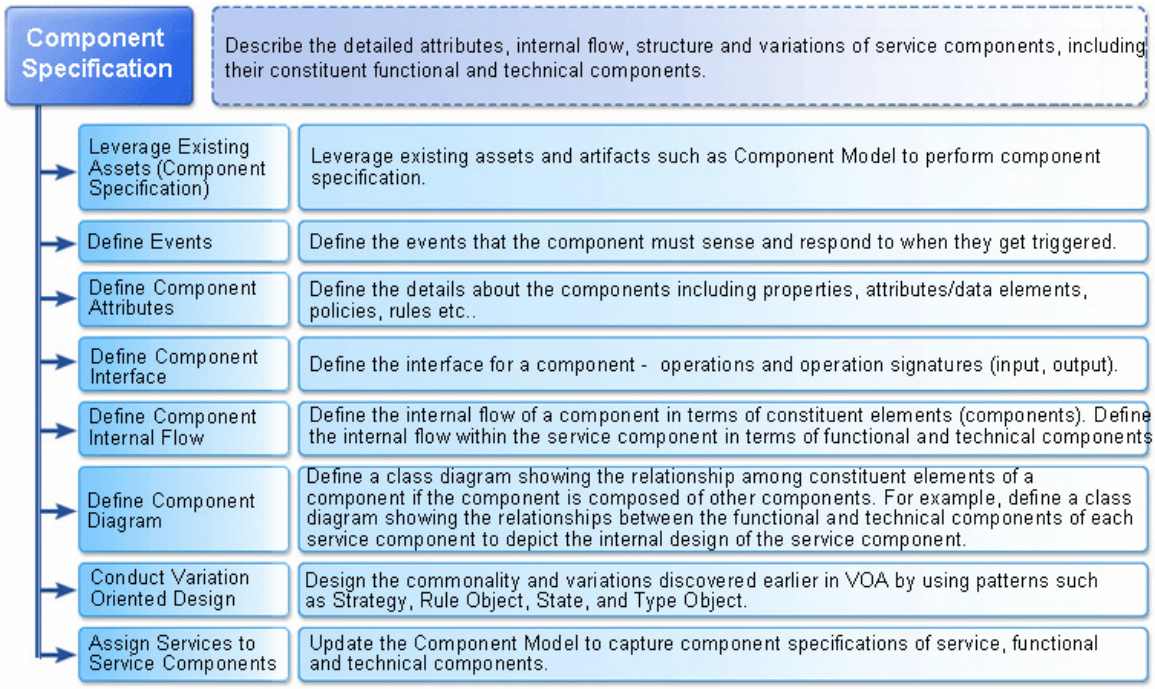


Figure 5-1 Component Specification tasks

5.2.1 Leverage Existing Assets (Component Specification)

Potential reusable assets include existing component models (static & dynamic), interface specifications, variation oriented analysis & design documents and business entity class models.

5.2.2 Define Events

During this activity, we identify the events that the component must sense and respond to when they get triggered. In-coming and out-going component messages are also specified.

Examples of events:

- Customer makes a deposit
- Customer makes a withdrawal
- Customer transfers between accounts

Depending on the project, it may also be helpful to collect the frequency, criticality and priority of the events. For example, it would be helpful to know that an event from a patient medical monitoring system happens infrequently, but when it does, it has extremely high criticality and priority.

5.2.3 Define Component Attributes

Define the attributes/data elements inside the component. (See Figure 5-4 Example of a Banking Application Class Model for examples of the attributes found).

5.2.4 Define Component Interface

An interface specifies one or more operation signatures and is both offered and used by a component. If an interface is offered by a component then the component is making that interface available so that other components can use the services provided through that interface. If an interface is used by a component then the component is using that interface to access the services of another component. A component can offer more than one interface (but must offer at least one) and more than one component can offer the same interface. A component may use the interfaces of several other components and several components can use the same interface of another component.

An operation signature specifies the method that is called to access a service of a component. Each operation signature has a name and a signature, where the signature consists of an optional return type, and a list of zero or more typed parameters.

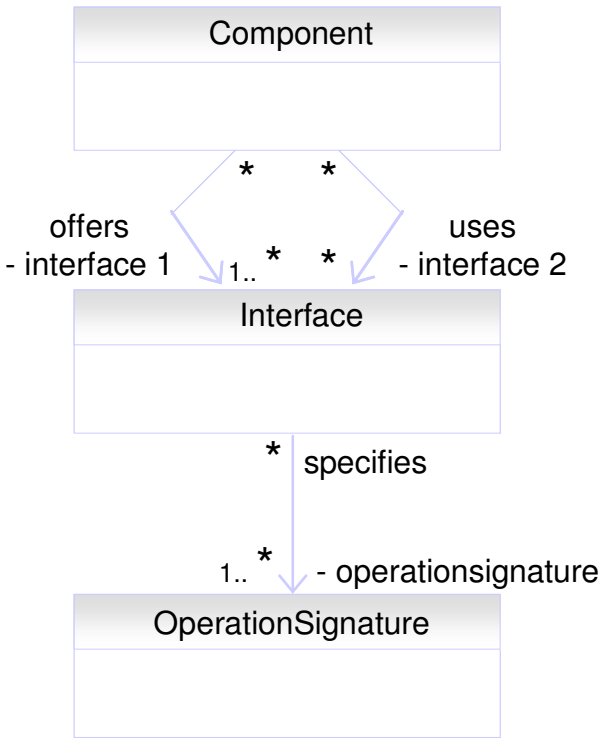


Figure 5-2 Defining Component Interface

5.2.5 Define Component Internal Flow

When services are executed by service components input messages are received processed and output messages are produced. This task describes the details of such internal flows.

This task develops a dynamic model (component interaction diagram or component interaction graphs) that shows the internal flows or control as well as the messages that are passed between the entities within the component. Component internal flows are not visible to the service consumers. The diagram notation is identical to component or object interaction diagrams.

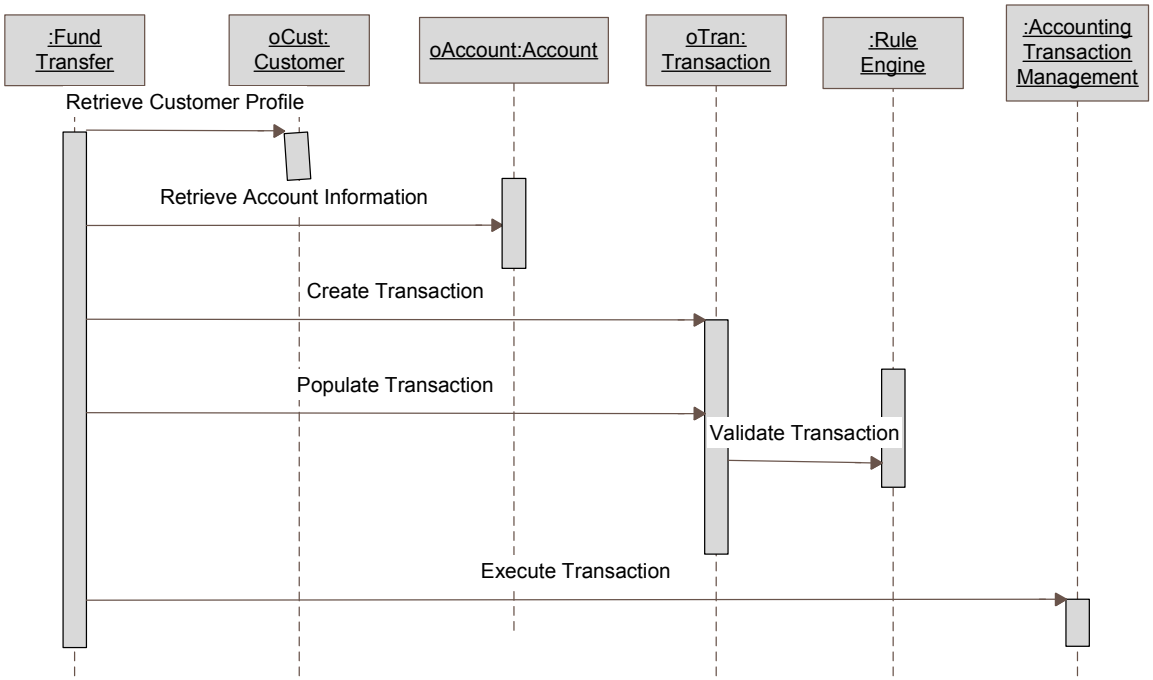


Figure 5-3 Example Component Internal Flow Diagram

5.2.6 Define Component Diagram

The class diagram depicts the internal component design of constituent elements and their relationship. Figure 5-4 Example of a Banking Application Class Model shows and example of a banking class model.

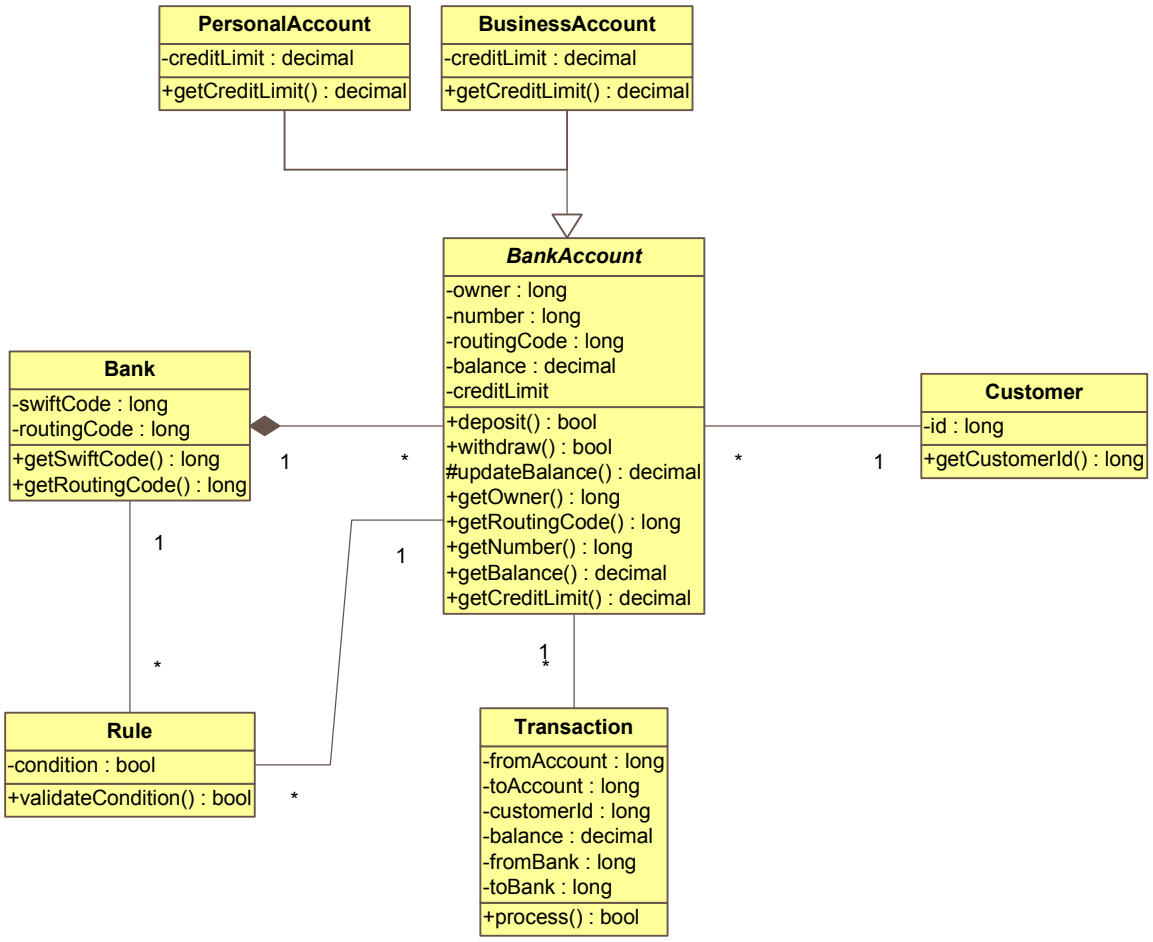


Figure 5-4 Example of a Banking Application Class Model

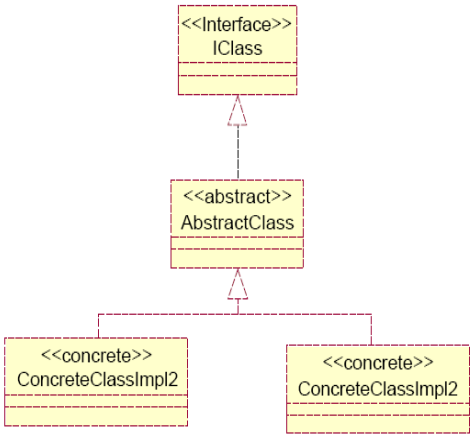
5.2.7 Conduct Variation Oriented Design

In this task, by using the output of the Variation Oriented Analysis as input, we design the mechanisms for dealing with the variations in process, data and rules.

Identify commonality and focus on building pluggable variations. Use the six principles that help separate the changing from the less changing aspects of software systems, and isolate and encapsulate the changes:

1. Separate and model changing from non-changing aspects of the domain: identify, separate, encapsulate, and externalize increasing variations.
2. Create type hierarchies for each variation point.
3. Assign rule types to each variation type.
4. Implement three-levels of abstraction; use aggregate inheritance meta-pattern.

5. Start from reuse levels higher than objects and build assets at each reuse level; build small frameworks around variation points. In general, each framework should have no more than 7+-2 classes.
6. Each reuse element has its own behaviors. Externalize behavior as configurable data that can be read into the application to allow soft-wiring.



aggregate inheritance meta-pattern

Figure 5-5 Designing a Structural Variation

Base on the work and information done in this activity, it may be necessary to refine the component model, by adding any new components found, restructuring the static or dynamic component models if needed and perhaps applying design patterns such as the Enterprise Component Pattern shown in Figure 5-6 The Enterprise Component Pattern Please note there is a technique paper available on applying the ECP.

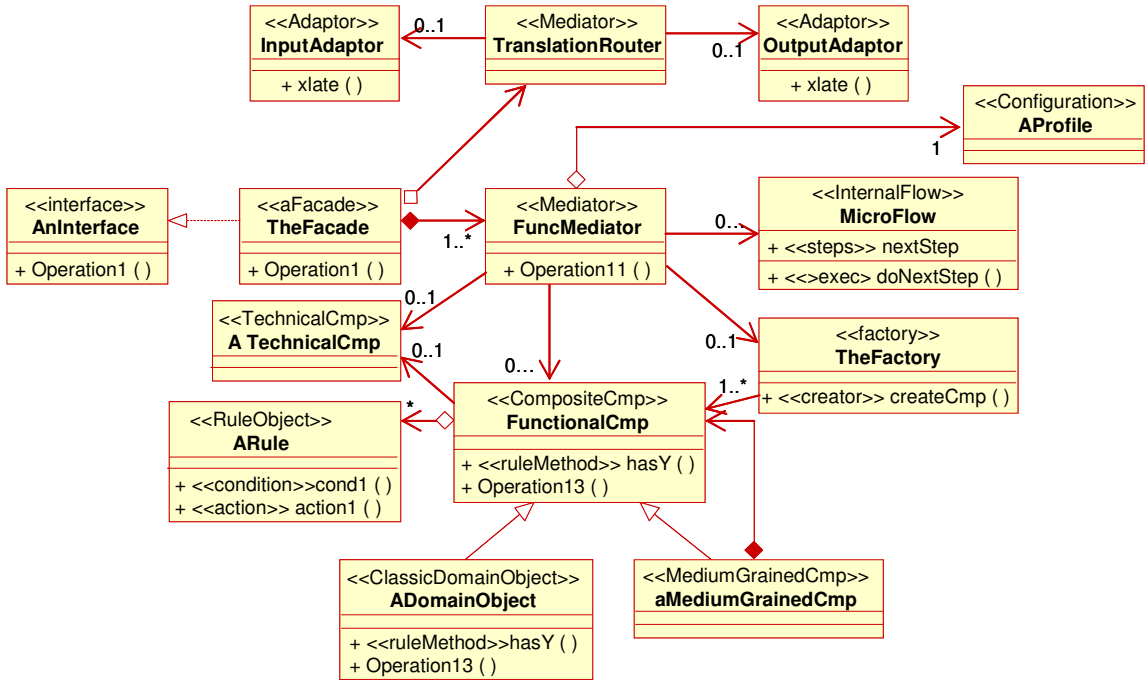


Figure 5-6 The Enterprise Component Pattern

5.2.8 Assign Services to Components

In this task we allocate services from our services model to components.

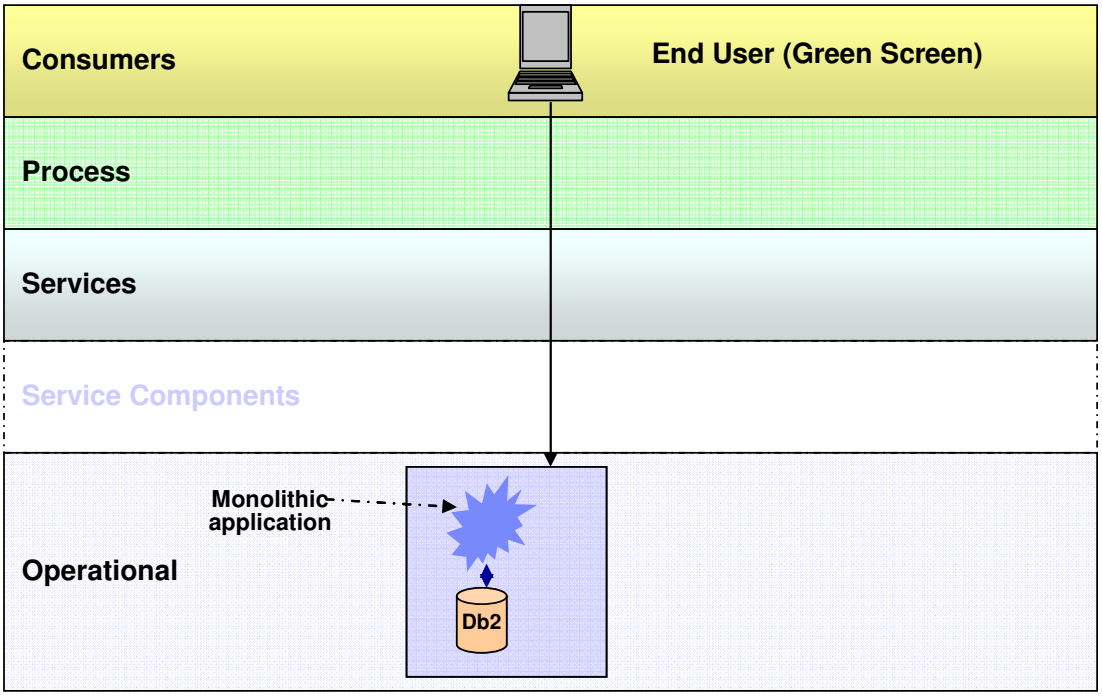
Service	Service Component
1.2.4.1.1.1 Identify Customer ID	Customer Relationship Management
1.2.4.1.1.2.1 Get Source A/C	Account Management
1.2.4.1.1.2.2 Get Eligible Destination A/C	Account Management
1.2.4.1.1.2 Retrieve Eligible Customer A/C	Account Management
1.2.4.1.2.1 Pre-populate Info	Account Transaction Management
1.2.4.1.3.1 Validate Selected Destination A/C	Account Transaction Management
1.2.4.2.1.3 Approve Transaction	Financial Transaction Management
1.2.4.2.1.3A Check Available Balance	Account Transaction Management
1.2.4.1.1.2.1 Update Transfer Status	Financial Transaction Management

Figure 5-7 Example of Assignment/Allocation of Services to Components

6. Example

The following example starts with Service Identification through Component Specification.

Figure 6-1 Current ABC Insurance Application Architecture shows the current situation at ABC Insurance which is a green screen directly attached to a legacy CICS application. The client would like to modernize their systems using the principles of SOA.



Note: For simplicity vertical layers are not shown

Figure 6-1 Current ABC Insurance Application Architecture

1. SOMA Identification was performed and resulted in the following services (partial list):

- getCustomer
- getCurrentClaims
- getPastClaims
- getPolicy

After applying the SLT, the following services have been decided to be exposed:

- getCustomer
- getCurrentClaims

2. Functional Area Analysis was performed and the following functional areas and service components were identified:

- Customer Information
- Claim Information

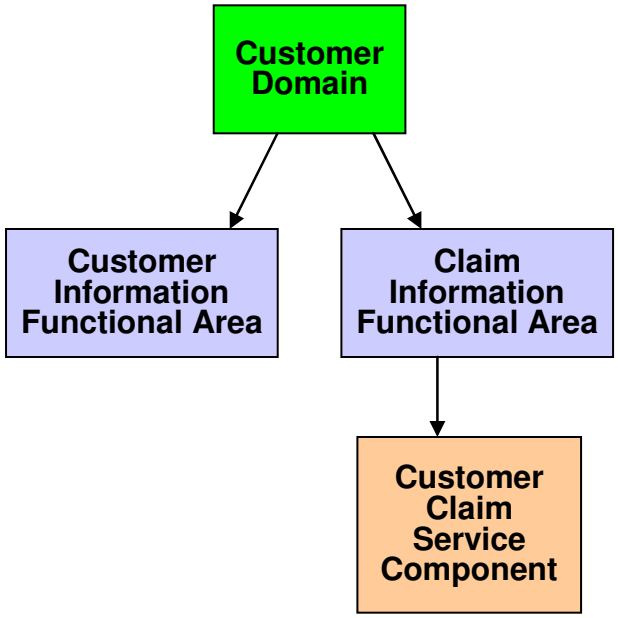


Figure 6-2 Functional Areas and Service Components

Claim Information FA maps to:

- Customer Claim Subsystem / Service Component

3. Applying the Enterprise Component Pattern to design the new service component for ABC Insurance

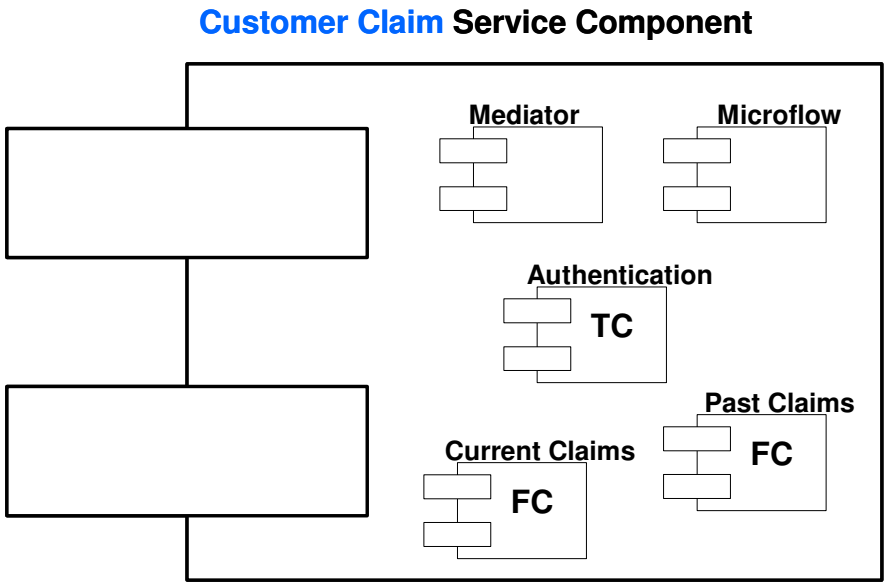


Figure 6-3 Service Component Design

The Current Claim functional component interacting with the legacy system would look like Figure 6-4 Functional Component accessing Legacy System.

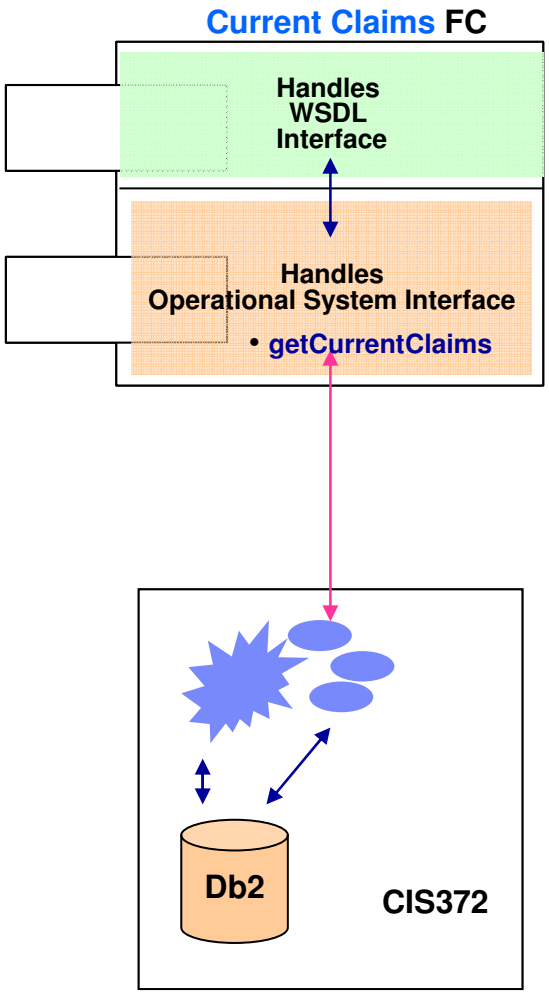


Figure 6-4 Functional Component accessing Legacy System

The next diagram shows the service component and functional components in the appropriate layers of the S3 model.

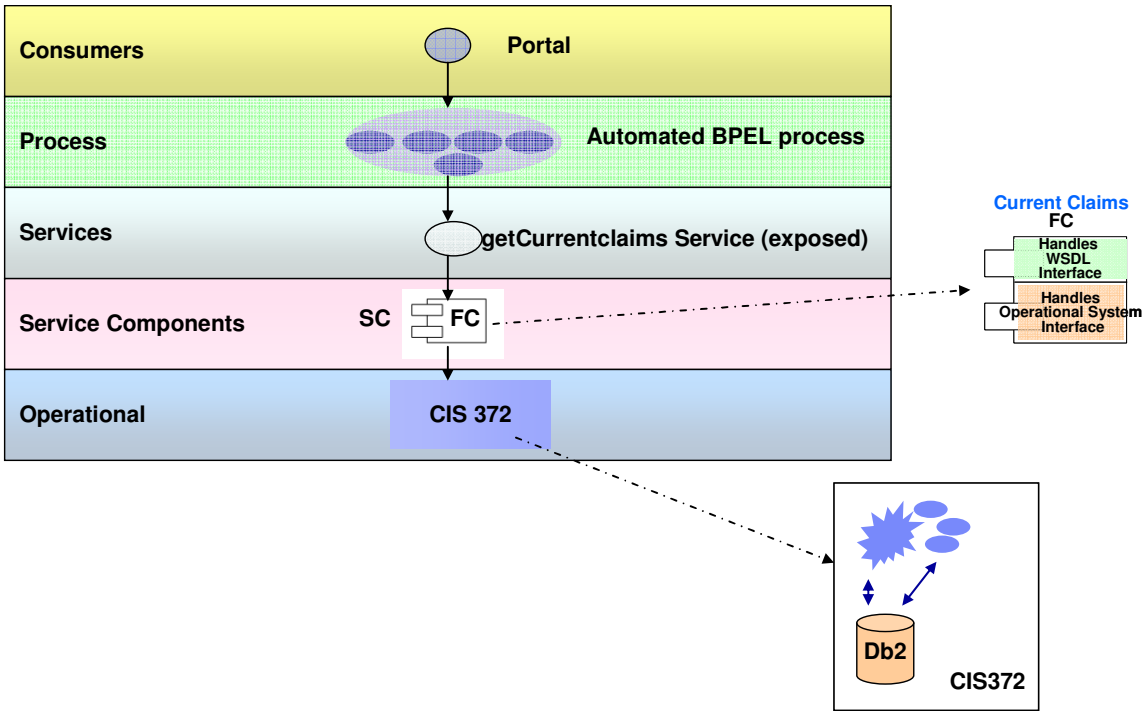


Figure 6-5 Service Component and Functional Component in SOA Architecture

7. Best Practices

8. Key Considerations

A functional or technical component may be used in multiple service components

9. Estimation Considerations

Some estimation considerations are:

- The number of subsystems
- The number of service components
- The number of functional and technical components needed
- The number of services exposed by a functional component

10. Related Techniques

- Functional Area Analysis
- Subsystem Analysis

11. Roles, Input & Output Work Products

Roles:

- Application Architect
- Application Developer
- SOA Architect
- Business Analyst
  Business Area Expert
  Business Design Consultant
  Data Architect
  Designer
  Infrastructure Architect
  SOA Architect
  Software Architect

Input & Output Work Products:

- Architectural Decisions
  Business Rules Catalog
  Class Diagram
  Component Model
  Deployment Unit
  High-Level Design Specification
  High Level Gap Analysis
  Infrastructure Gap Analysis
  Interaction Diagram
  Interface Specifications
  Non-Functional Requirements
  Operational Model
  Process Definition
  Requirements Specification
  Service Model
  Subsystem Analysis
  System Context
- Use Case Model
- Variations Model

12. Resources

13. Frequently Asked Questions

14. Other References