# Reuse in Oracle SOA Suite 12c: Templates, Libraries or Services?

**Ronald van Luttikhuizen**
**eProseed**
**The Netherlands**

**Keywords:**

Oracle, SOA Suite, Service Bus, 12c, Template, Pattern, Library, Microservice, Service, Subprocess, BPEL, Spring, Maven, Reuse, Framework

## Introduction

Most people in IT agree: you want to reuse code, rather than type the same thing twice. Choosing the right type of reuse can greatly speed up your development, decrease maintenance efforts, and reduce the number of errors by eliminating similar code.

Of course you can copy something, and reuse it that way. However, this is hard to do, especially if somebody changes the original code. In general, copying code is considered bad practice. In SOA Suite 12c there are several ways to reuse code. These options vary from project skeletons using Maven poms to composite templates and creating services. In this article we will explain the differences between a pattern, a library and a (micro) service and illustrate this with features from SOA Suite 12c. These Oracle SOA Suite 12c features will be explained with use cases.

At the end of this session you will be familiar with the facilities that SOA Suite 12c offers for reuse and you will be better equipped to choose the appropriate reuse method (library, pattern or service).

## Patterns, Libraries and Services

Typically you can reuse functionality or logic in three different ways:

1. Reuse of a general idea; this is often referred to as a pattern.
2. Design-time reuse of a component; this is often referred to as a library.
3. Runtime reuse of specific functionality; this is often referred to as a service.

These types of reuse are not only applicable in software, but can also be applied in other areas of our life such as construction. Let’s examine these three different types of reuse and apply this to housing.

## Patterns

There are several patterns that we apply when building a house. For example, a house always has a roof to prevent that we and the furniture are getting wet. Another patterns has to do with the location of parts: houses often have a front door facing the street for easy access.

These *general* patterns can be designed in different ways, as you can see in the picture below.

*Illustration. 1: use of a pattern*

Every house in the picture has a different colour. Some have two stories, others are three story buildings, the interior is different, and so on. However, all these houses have a roof and a front entrance facing the street for easy access. The general ideas or patterns are reused for every house.

**Libraries (design-time reuse)**

When a contractor is building a set of houses, he might decide to order the garage doors from a single garage-door manufacturing company. These doors can all have the same design and functionality. It is built in a specific way and is built into every single house. When one door breaks, other people don't have that issue (unless it was badly built or designed of course) and people can take out the door and replace it with something else or give it a different colour.

*Illustration. 2: design-time reuse*

The doors are all the same at design time, but during the life cycle of the garages, the different garages can change the doors at a different pace and deviate from the original design.

**Services (runtime reuse)**

In a gated community, a gate is created to protect the residents from unwanted visitors and to guard the houses. The gate is shared by all residents. There is only one gate or one 'runtime instance' for the entire community. If there is a problem with the gate, everybody has a problem. For example, if there are many people that want to go through the gate, there will be a line and waiting times. If we want to change the colour of the gate, all the residents will get the change at the same time. The gate is reused at runtime.

*Illustration. 3: use of a service*

**Patterns in software**

Apart from patterns in our daily lives, patterns are also very common in software. Grady Booch et. al. wrote a famous book: Design Patterns. They identified different types of patterns, well known to most people in the IT industry:

- Creational patterns, for example Factory;
- Structural patterns, for example Adapter and Composite;
- Behavioural patterns, for example Mediator and Publish-Subscribe;
- Concurrency pattern, for example Scheduler.

A lot of these and other patterns are implemented in tooling. For example the patterns listed above are implemented in Oracle SOA Suite, literally. If you want to enforce the use of specific patterns, you can write rules and regulations (like housing regulations), or you can enforce them by using templates.

Using patterns is good, but you should not go overboard with them. Some people feel that if you need to build a pattern yourself, it is sign that the tooling you use is lacking a feature. Apart from that, Incorrect or redundant use of patterns leads to complexity. For example having a Factory is more complex than just instantiating an object, so you should only use it when needed.

**Libraries (design-time reuse) in software**

Design-time reuse in software is accomplished by creating and using libraries. There are two types of libraries:

1. Built-in libraries, for example as part of the programming language library (java.util, java.math) or as part of the application server (JPA).
2. Manually added libraries. You can add a library to your own application or software that you build. These can be commercial libraries, open source libraries or your own domain-specific libraries.

**Services (runtime reuse) in software**

Services enable runtime reuse in software, they are like the gate in our gated community. Service-orientation is an architectural style. You can read more about this in our book, SOA Made Simple. A variation on this theme are microservices. This is very nicely described by Martin Fowler in http://martinfowler.com/articles/microservices.html.

The table below summarizes and shows the differences between the types of reuse we discussed:

| **Pattern** | **Library** | **Service** |
|---|---|---|
| Reuse of a general idea | Design-time reuse | Runtime reuse |
| Quickstart development | Deploy with your solution every time | Deploy once for all consumers |
| Consumer is responsible for QoS | Shared responsibility for QoS | Provider is responsible for QoS |
| Can be “reapplied” | Can be upgraded | Changes on service impact all consumers |

**SOA Suite 12c**

Oracle SOA Suite 12c supports all these concepts of reuse:

- Patterns are built-in the platform: Mediator, Scheduler, Adapters and so on. Besides these out-of-the-box patterns it is also possible to enforce patterns you come up yourself by creating SOA Suite Templates and Pipeline Templates for Service Bus.
- Libraries are supported in two ways: you can create Service Component Templates or BPEL Templates and provide these as reusable design-time components or reuse libraries by wrapping them in a Spring component.
- Services are of course supported by SOA Suite: you can build (composite) services, call services and use Adapters to expose functionality as a service.

In the next paragraphs we will take a closer look at these types of reuse that are supported by Oracle SOA Suite.

**Project quick start: Maven**

Maven provides a quick-start mechanism for generating the structure of SOA Composite and Service Bus applications and projects through so-called "archetypes". Archetypes help to enforce naming conventions and folder structures. SOA Suite 12c introduced Maven-support for SOA Composite and Service Bus applications and projects, including the archetypes.

**Templates to implement patterns and create libraries**
Design time reuse and patterns are both implemented as templates in SOA Suite 12c.

- A particular pattern can be enforced by providing a standard structure or part of the implementation. For example, a standard way of logging messages can be captured in a template.
- Design can be reused by creating a template including implementation details. This template can then be imported and reused in your project. For example, you can create a template providing the implementation of a call to a specific database package in e-Business Suite using the EBS or Database adapter.

The templating-mechanism is provided by SOA Suite, the developer is responsible for creating and packaging the templates.

**Using libraries in your composite**

Apart from using templates to make a library of your component or BPEL scope, you can also warp Java code in a Spring component; Apart from using the Spring component you invoke Java code directly from BPEL using the embedded Java activity. In this case the JAR file or Java classes need to be copied to a specific extensions folder on the WebLogic Server that runs SOA Suite. Java libraries can also be used to implement custom XSLT functions that can be used from the design-time XSLT

Mapper in JDeveloper, and at runtime when executing XSLTs in your SOA Composites and Service Bus projects.

Java libraries can of course also be deployed with other Java code and be reused in Web Services and Web Applications. SOA Suite can then call that functionality by referencing it.

**Runtime reuse**

SOA Suite itself is a platform dedicated to implementing, consuming and providing services. Using SOA Suite it is easy to both invoke (external) services at runtime as well as providing them to service consumers using a variety of supported interfaces such as REST/JSON, SOAP/XML, JMS, RMI, and so on.

When it comes to reuse, SOA Suite 12c introduced the BPEL subprocesses feature to support "internal reuse" of BPEL snippets within SOA Composites.

The session at DOAG will dive into all of these mechanisms for reuse in SOA Suite 12c.

**Summary**

We have seen that we can differentiate between the following types of reuse: patterns, libraries and services. The right choice for reuse depends on how specific the solution is and how specific the problem is. It differs per type of reuse what the responsibilities are for quality-of-service aspects and what impact updates have. Finally, the article discusses how SOA Suite 12c supports all these types of reuse.

| Type of reuse | What is reused | Support in SOA Suite 12c | QoS |
|---|---|---|---|
| Pattern | General idea | Maven Archetypes (template)<br>SOA Suite Project Template<br>Service Bus Pipeline Templates | Consumer is responsible for QoS |
| Library | Design and code | Java libraries in Spring component<br>SOA Suite Templates (Component and Custom activity)<br>Service Bus Pipeline Templates | Shared responsibility for QoS |
| Service | Runtime | Create in Java, SOA Suite or PL/SQL<br>Publish service on Service Bus | Provider is responsible for QoS |

**Contact address:**

**Ronald van Luttikhuizen**
eProseed Netherlands
Agnes v Leeuwenberchstr 15
3515 AX Utrecht
The Netherlands

Phone: +31(0)30-8080175
Email ronald.van.luttikhuizen@eproseed.com
Internet: www.eproseed.com