how to decide functional responsibilities for SOA services?

Asked 10 years, 11 months ago Modified 10 years, 6 months ago

Viewed 280 times

5

I've been searching (mostly google) to try an find tools or methodologies that I can use to identify the functional responsibilities for SOA services. My searching hasn't really come up with anything.

Currently, the approach I use for deciding the functional responsibilities is adhoc and is really just gut feel, e.g.

- all customer related functionality goes into a customer service
- all payment related functionality goes into a payment service
- etc

Reflecting on other approaches used in the software design/architecture world:

1. Object Oriented Analysis has the concept of Class Responsibility Collaboration (CRC) models to decide the responsibility of classes.
2. From what I understand, Domain Driven Design (DDD) has the concept of bounded contexts to logically partition a domain.

3. In traditional software architecting:

    - the book Process of Software Architecting has a similar approach to CRC that is used for identifying software components and their responsibilities.
    - the Architecture Based Design process has steps for dividing responsibilities info functional areas.

Question: What tools/methodologies does the SOA architect have that allow them to determine the functional responsibilities of a service?

architecture domain-driven-design soa crc-cards

Share

Improve this question

Follow

edited Jun 2, 2014 at 14:16

asked Jan 7, 2014 at 14:53

Chris Snow

24.5k ● 37 ● 154 ● 326

1 Answer

Sorted by: Highest score (default)

4

Your approach and analysis seems reasonable keep a service functionality around the business problem it is trying to resolve. Customer related functionality goes into the customer service etc. However you need to solidify your decision making process and remove the gut feel approach.

What you are referring to is called SOA design time policies part of a larger topic called SOA governance. Remember just having a bunch of web services does not make your architecture SOA rather service based so it is essential that you go through the pain of setting up your SOA Governance before jumping into a SOA architecture. Please note I am making the assumption that this is not in place.

You SOA governance policy will specify how to design your services. A typical SOA design time policy might look something like this around service design.

Design for reusability.

When you design a service, it would be nice if this service could be reused by other services and consumers.

If you look at a SOA system and all of the services it provides you'll probably notice that services provide different levels of granularity. You can have anything from services that allow you to modify a single property of an entity to services that allow you to apply for a a loan. Now

why is this granularity important? The granularity of a service defines how easily it can be reused.

Fine-grained services can often be more easily reused than coarse-grained services. Here is a example of how you can break up this granularity:

- Process services: Process services are the coarsest-grained services. These kinds of services most often offer services or products to their consumers. A typical process service example would be something like the sale of a car. In this scenario the sales system needs to be updated, the inventory system needs to be updated, and a lot more systems are involved in this transaction. A process service will call other services to accomplish its task. Typically when you do orchestration between a lot of services you are dealing with a process service
- Business services: A business service provides a single, specific business function for a system. For example updating the sales system with the invoice and tax documentation around a sale of a car.
- Technical services: The finest-grained services are the technical services. A technical service provides a small piece of specific functionality to other services. An example of this would be a service that updates the inventory of cars on the floor i.e. mark the car in the database as sold other example would be to send an email, or call a legacy backend.

You should also keep a catalog of services. This catalog must be kept up to date with the services and their functionality to eliminate duplication of services.It will also allow you to determine where a services functionality must be defined.

Using a catalog of services and design time policies will allow you to decide where functionality should lie.

I would recommend this book as it deal with the whole management around SOA. As far as which formal methodology to use I would suggest try them and keep the one that works for you. Keep in mind that it helps to have business people on your decision making team as they understand the business they could probably give you insight into where the functionality should lie. Just formalize it in your SOA governance policies and review those policies every 8-12 months to ensure they are still relevant and are working for you.

Share Improve this answer

Follow

answered Jan 7, 2014 at 22:21

Namphibian

12.2k ● 8 ● 49 ● 77