Service-Oriented Architecture (1)

Hui Chen [a]

[a]CUNY Brooklyn College, Brooklyn, NY, USA

April 5, 2022

Outline

Outline

Software Design

- Design starts mostly from/with requirements – evolving mostly from functionalities and other non-functional characteristics
  - In the waterfall model Design generally occurs after Requirements
  - In agile, design is performed during in each iteration
- To answer: How is the software solution going to be structured?
  - What are the main components – (functional composition) often directly from requirements' functionalities (e.g., use cases, user stories, scenarios)
  - How are these components related? – Possibly re-organize the components (composition/decomposition)
- Two main levels of design:
  - Architectural (high level) design
  - Detailed design
  - Different design concerns at different abstraction levels (e.g. classes vs. modules vs. entire system)
- How should we depict design – what notation/language?

Outline

Outline

- Service-Oriented Architectures (SOA)
  - Architectural style (high-level design)
- Implementing SOA
  - Using RESTful Web Services
- SOA history lesson
- Demo (Part 2)

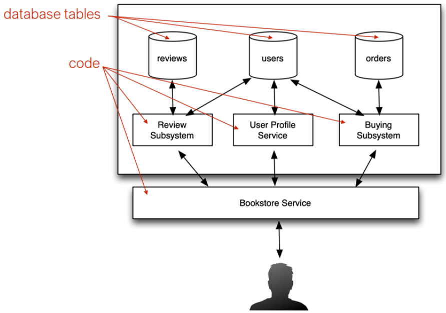
Service-Oriented Architecture

- Technology that makes it easy to recombine independent services to offer many different apps
  - a.k.a. modules
  - a.k.a. components
  - The power of composition
- Architectural style

Example: Bookstore Application

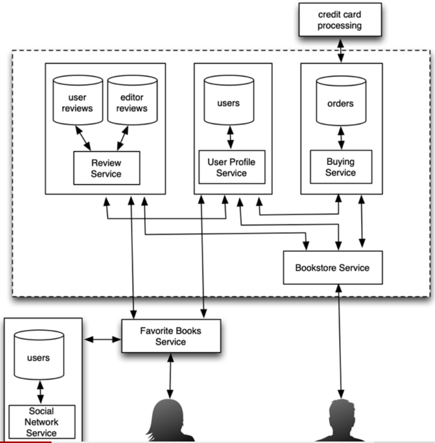
Source: Figure 1.2, Engineering Software as a Service by Armando Fox and David Patterson, 2nd Beta edition, 2013.

- Internal subsystems can share data directly
    - Review subsystem accesses user profile
- All subsystems inside single API (“Bookstore”)



Example: Bookstore Application: SOA

- Subsystems independent, as if in separate datacenters
  - Review Service access User Service API
- Can recombine to make new service (“Favorite Books”)



Why SOA Works?

- networked – a common assumption is that services can exist anywhere (on the same machine or over a network)
- independently deployable – each service is independently allocated to hardware
- heterogeneous – allows composing services written in language or framework X or Y to others
- encapsulation – data is hidden and can only be accessed/modified via well defined and interfaces
- abstraction – the details of each service are hidden
- service contract – the service will adhere to what it has promised

Modern Apps Use SOA

Example Reference

https://aws.amazon.com/lex/

Micro and Macro SOA

- Macroservices
  - Composing substantial applications into a larger application
  - e.g. integrating with Amazon, Google, Facebook, ... services
- Microservices
  - Each application can be a suite of small services

Outline

Implementing SOA

Most popular approach is using Web Services. Two choices, in general:

- REST-ful
  - i.e. using REST
  - on top of HTTP (protocol)
- REST-less
  - using WSDL (interface definition language) and SOAP (protocol)

RESTful

REST (Representational State Transfer) – R. Fielding, 2000

- Effectively maps a URL into a method or action
  - Stateless, so transfer the state
  - Map resources + HTTP requests to actions and functions
- A service (in the SOA sense) whose operations are like this is a RESTful service
- Ideally, RESTful URIs name the operations

REST API Examples

- Github
- Stackoverflow
- ...

Statelessness of REST

- Advantages
  - simplicity – RESTful services can treat each method request independently
  - caching – RESTful services can save results and provide them to similar requests
  - scalability – RESTful services can be replicated or relocated as needed
- Disadvantages
  - need to transfer state

REST-less

Stateful

- WSDL defines an endpoint
- SOAP is a common protocol, built on top of XML, for transferring data to and from this endpoint

```xml
<message name="getTermRequest">
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">
  <operation name="getTerm">
    <input message="getTermRequest"/>
    <output message="getTermResponse"/>
  </operation>
</portType>
```

SOA $\geq$ Web Services

- All Web Services are SOA
  - but the reverse does not hold!
  - Not all SOA uses Web Services
- Many other technologies exist to implement SOA
  - e.g. using messaging: ActiveMQ, JMS, RabbitMQ
  - e.g. using components: OSGI, Spring Framework

Outline

SOA is Not New

"On the Criteria to Be Used in Decomposing Systems into Modules"
David L. Parnas. Communications of the ACM. 1972

- This paper suggested some criteria which can be used in decomposing a system into modules
- Benefits of modular programming:
    - Managerial: development time should be shortened because separate groups would work on each module with little need for communication
    - Product Flexibility: it should be possible to make drastic changes to one module without a need to change others;
    - Comprehensibility: it should be possible to study the system one module at a time.

Jeff Bezos: Amazon shall use SOA

“All teams will henceforth expose their data and functionality through service interfaces.

Teams must communicate with each other through these interfaces.

There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team’s data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.”
– Jeff Bezos

Jeff Bezos: Amazon shall use SOA

"It doesn't matter what [SOA] technology you use.

Service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.

Anyone who doesn't do this will be fired.

Thank you; have a nice day!"
– Jeff Bezos

Questions

- Overview of SOA
- Implementing SOA: REST-ful and REST-less
- Some history
- Demo next class
- Consider to use it in your project

Outline

"Engineering Software as a Service" by Armando Fox and David Patterson (2nd Edition)
"Essentials of Software Engineering" by Frank Tsui, Orlando Karam, and Barbara Bernal(4th Edition)