



CS544 EA

Overview

Intro: Integration and Architecture

Big Application

- A single big application is called a **monolith**.
- These often have **maintainability issues**:
 - A change in one place means recompiling the app
 - A change in one place could affect many other things (parts of the same big application)

Multiple Monoliths

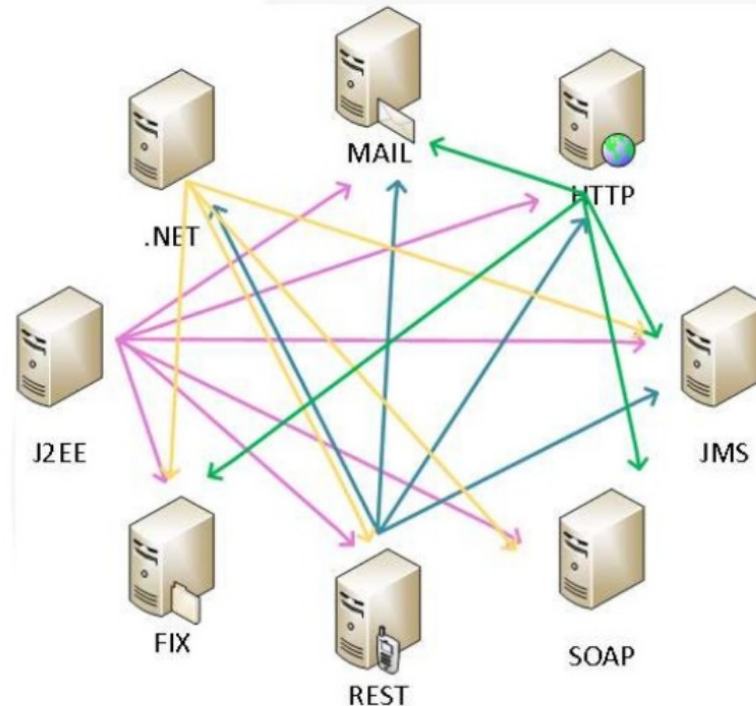
- At some point another application is made, or bought, or two businesses combine
- Whatever the reason, **integration is needed**
 - This has been true since the earliest days of Enterprise Applications

Initial Integration

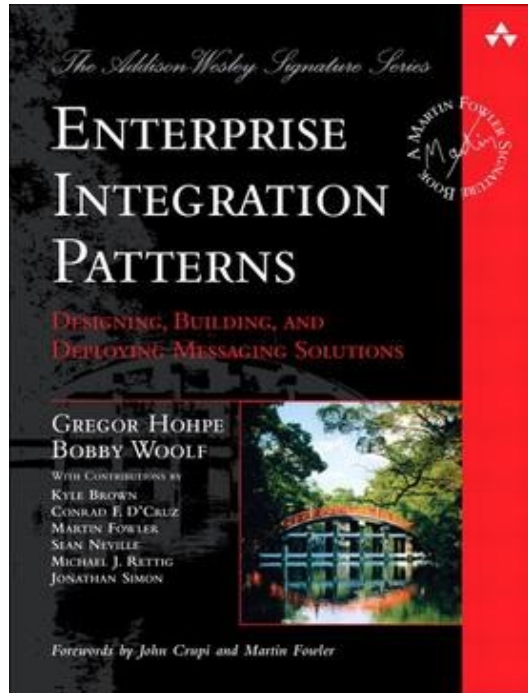
From: <https://www.slideshare.net/wso2.org/resilient-enterprise-messaging-with-wso2-esb>

Spaghetti Integration

What about maintainability, scalability, troubleshooting and governance?



Enterprise Integration Patterns



- Gregor Hohpe and Bobby Woolf, 2003
 - We will introduce (mention) some of these **patterns** at the end of this course

Book specifies 4 Ways to Integrate

- Used in the past:
 - File Transfer (leaves a lot to the developer)
 - Shared Database (does not scale as well)
- Modern approaches:
 - **Remote Procedure Invocation** (synchronous)
 - **Messaging** (Asynchronous)

Integration and Architecture

- **Integration** is not just something that can be used to connect different applications
 - It can also be an **architectural solution**
- Why have one big, hard to maintain, hard to scale, monolith when you can break it into parts?
 - Each part provides a “service”

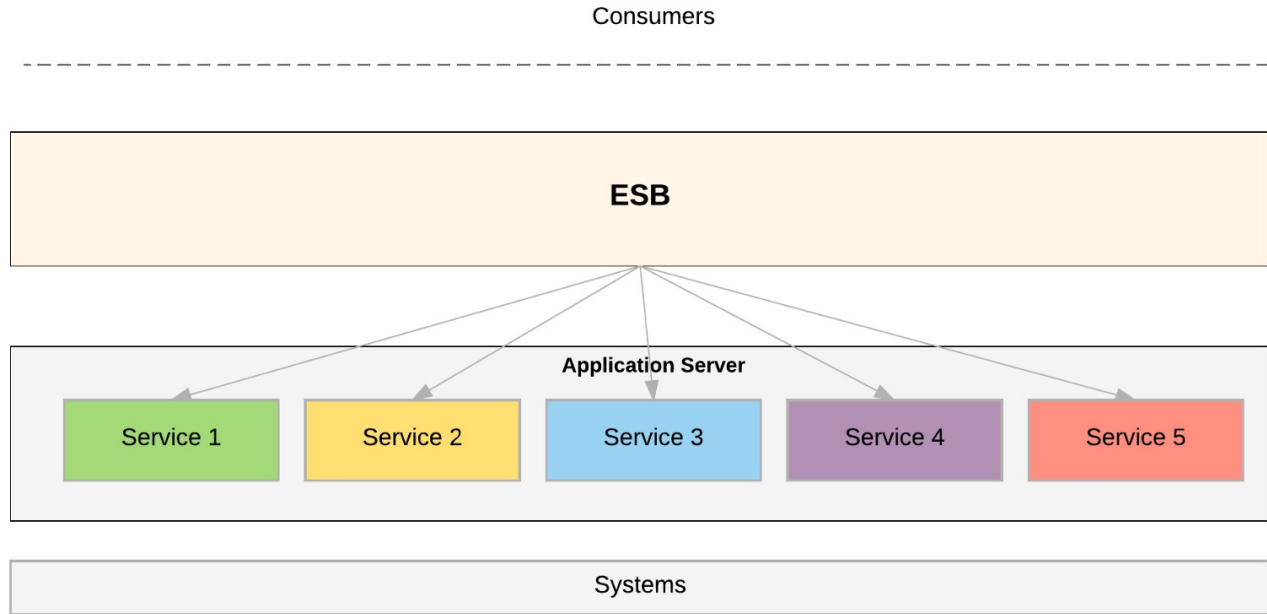
Service Oriented Architecture

- To Solve Monolith problems
 - **Separate applications** each provide their own **service**
 - The 'service layer' is the point of integration
- [Wikipedia] Each service / application:
 - Represents a business activity with a specified outcome
 - Is self-contained
 - Is a black box for its consumers.
 - May consist of other underlying services

Service Oriented Architecture

- A coherent integration plan is needed, how are the different services going to communicate?
 - How are they going to be combined?
- A common solution:
 - An **Enterprise Service Bus** (ESB)
 - Coordinates activities between the services
 - Can contain logic and combine services

Enterprise Service Bus



Similar to hardware, a channel through which all communication flows

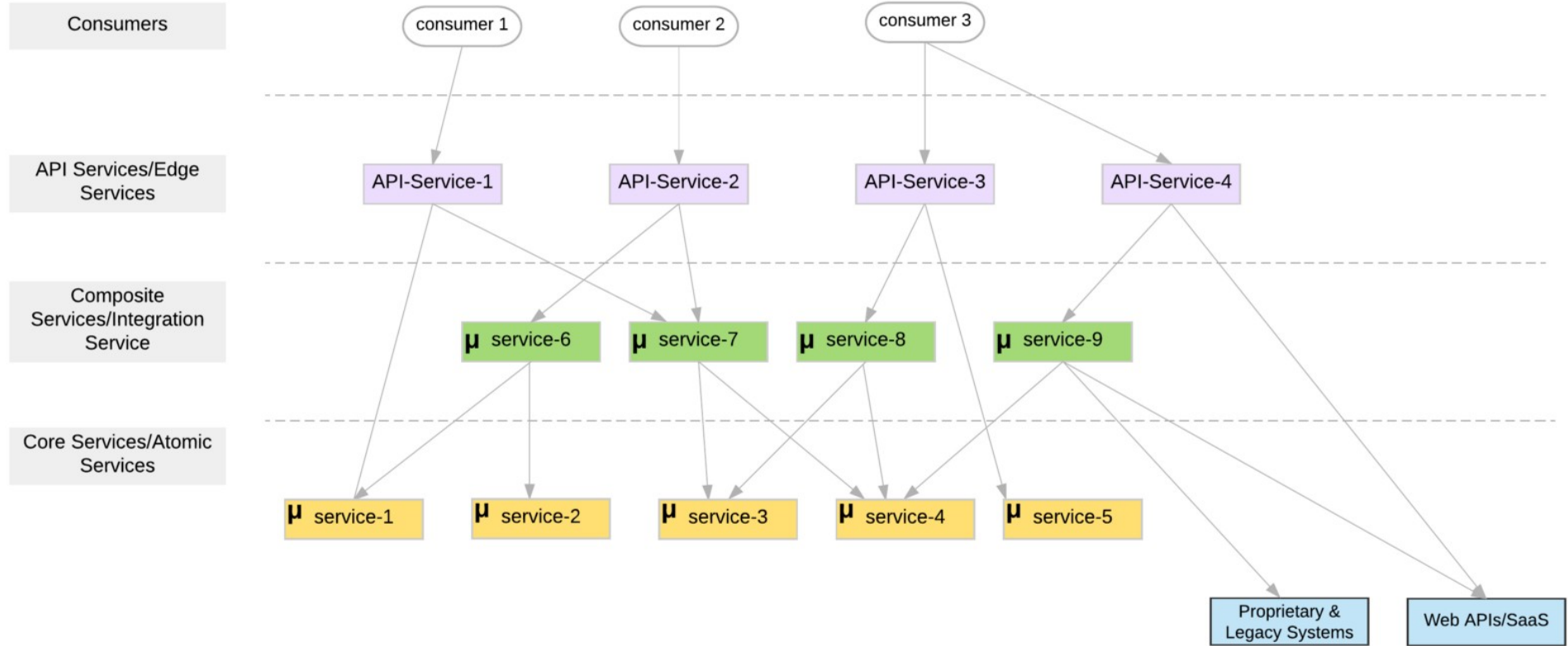
ESB

- ESBs generally **contain logic** to provide:
 - Routing, choreography (combine), transformation, business rules
- Useful, but can also be **a real problem**:
 - Business logic spread between services and ESB (what is where?)
 - ESB is a single, monolithic, center of the application
 - Single point of failure

Micro Services

- The Micro Services architectural style is often seen as a response to the use of an ESB
- **Micro Services** emphasizes:
 - Smart Endpoints (services) and **Dumb Pipes**
- Also generally smaller (more fine grained) services
 - What is or is not small is undefined

Combining Micro Services



Summary

- Once again separation of concerns is the key to maintainability and good architecture.
- By keeping each service small and simple each maintains the greatest ability adapt to change.
- Each part should do less, so that the combined whole can accomplish more.