

Integration between services and mediation

Oxford University
Software Engineering
Programme
December 2019



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>



Through 2020, integration work will account for 50% of the time and cost of building a digital platform.

Use a Hybrid Integration Approach for Digital Transformation

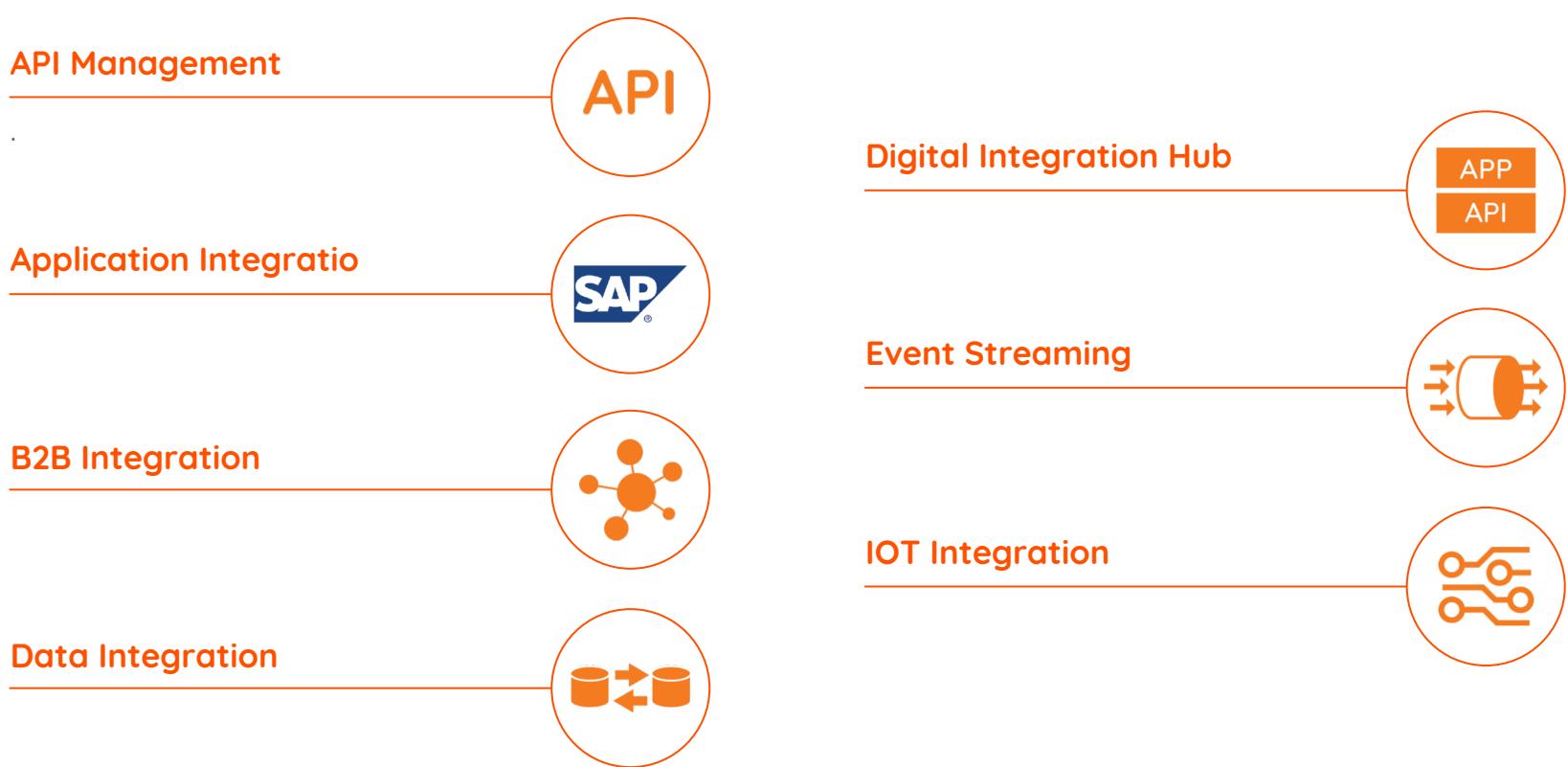
© 2018 Gartner, Inc. and/or its affiliates. All rights reserved.

Smarter With  Gartner

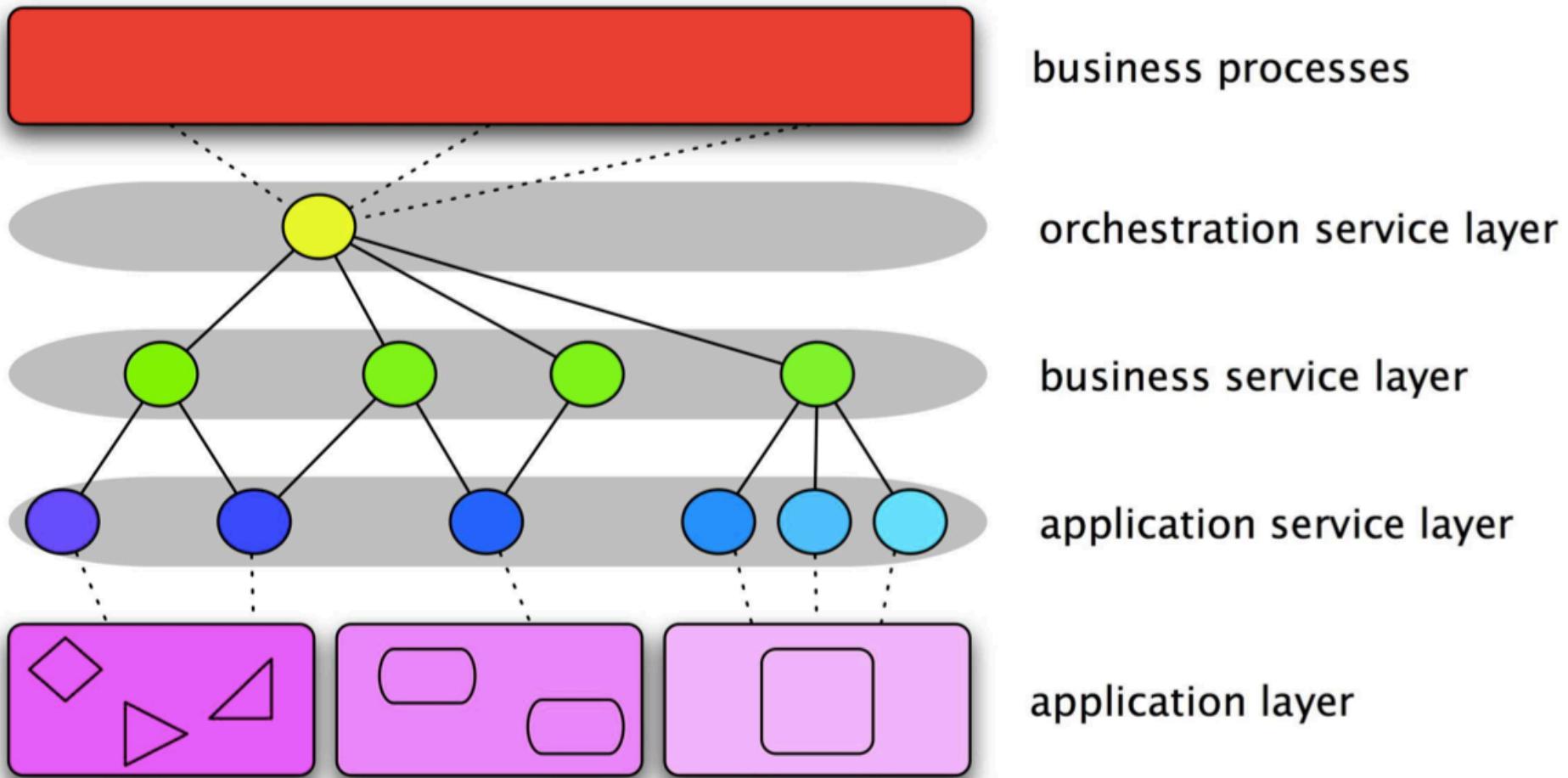


© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Seven Cases of Integration (from Gartner)



Recap on SOA model

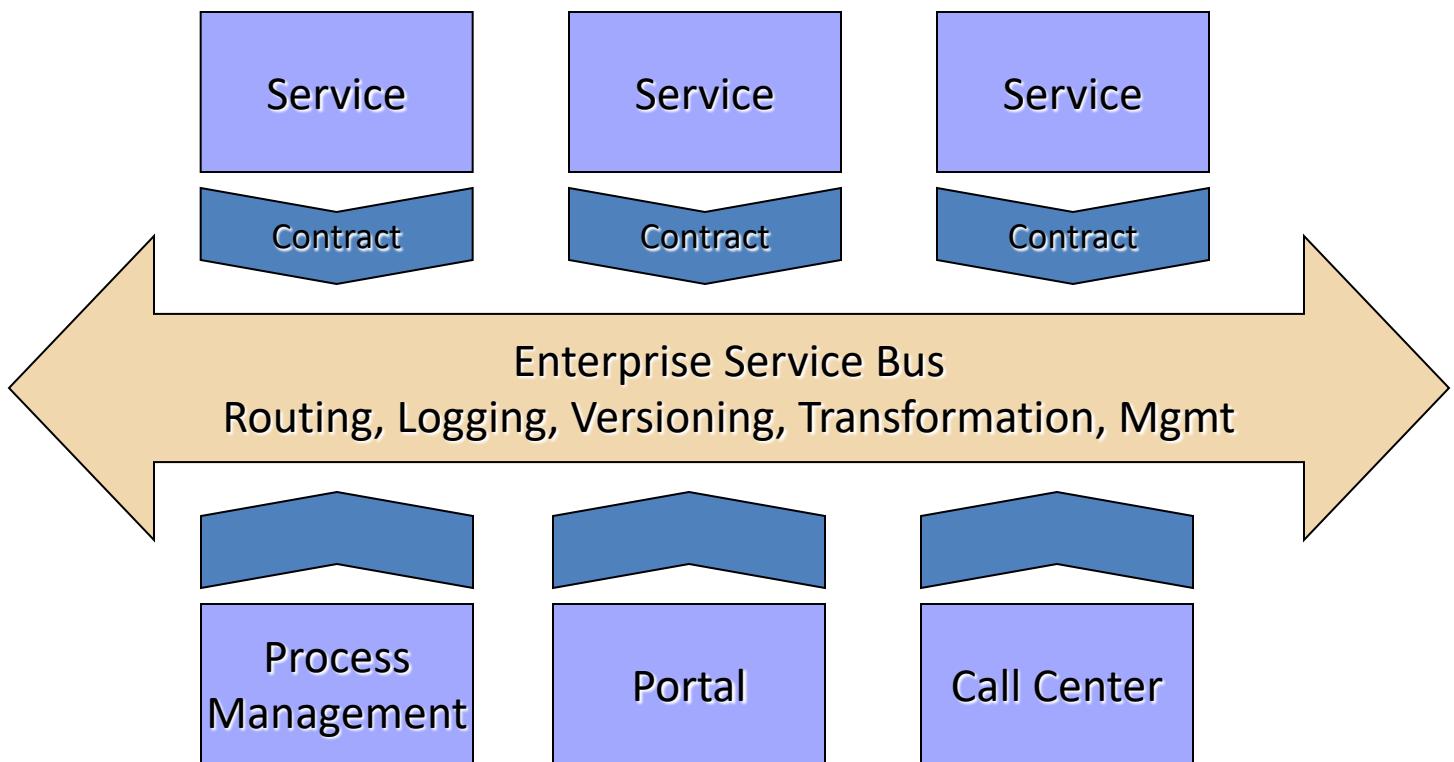


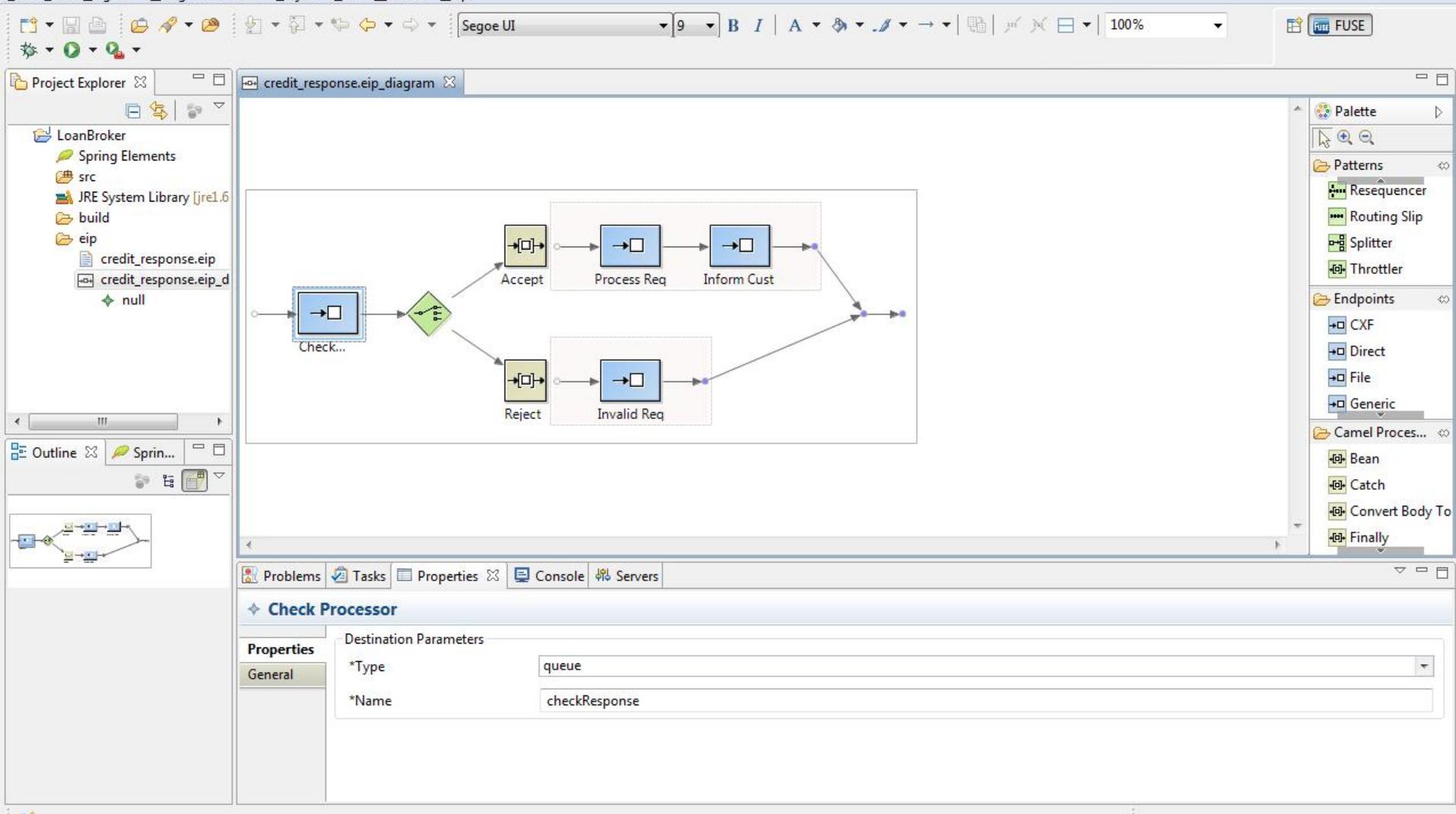
Enterprise Service Bus (ESB)

- A software architecture
 - A logical intermediary through which every message flows
 - Offers a policy based approach to decide what to do to each message or interaction
- The benefits of the gateway model
 - Without a physical hub and spoke
- Many vendors offer ESB products
 - Often a layer over an existing messaging framework



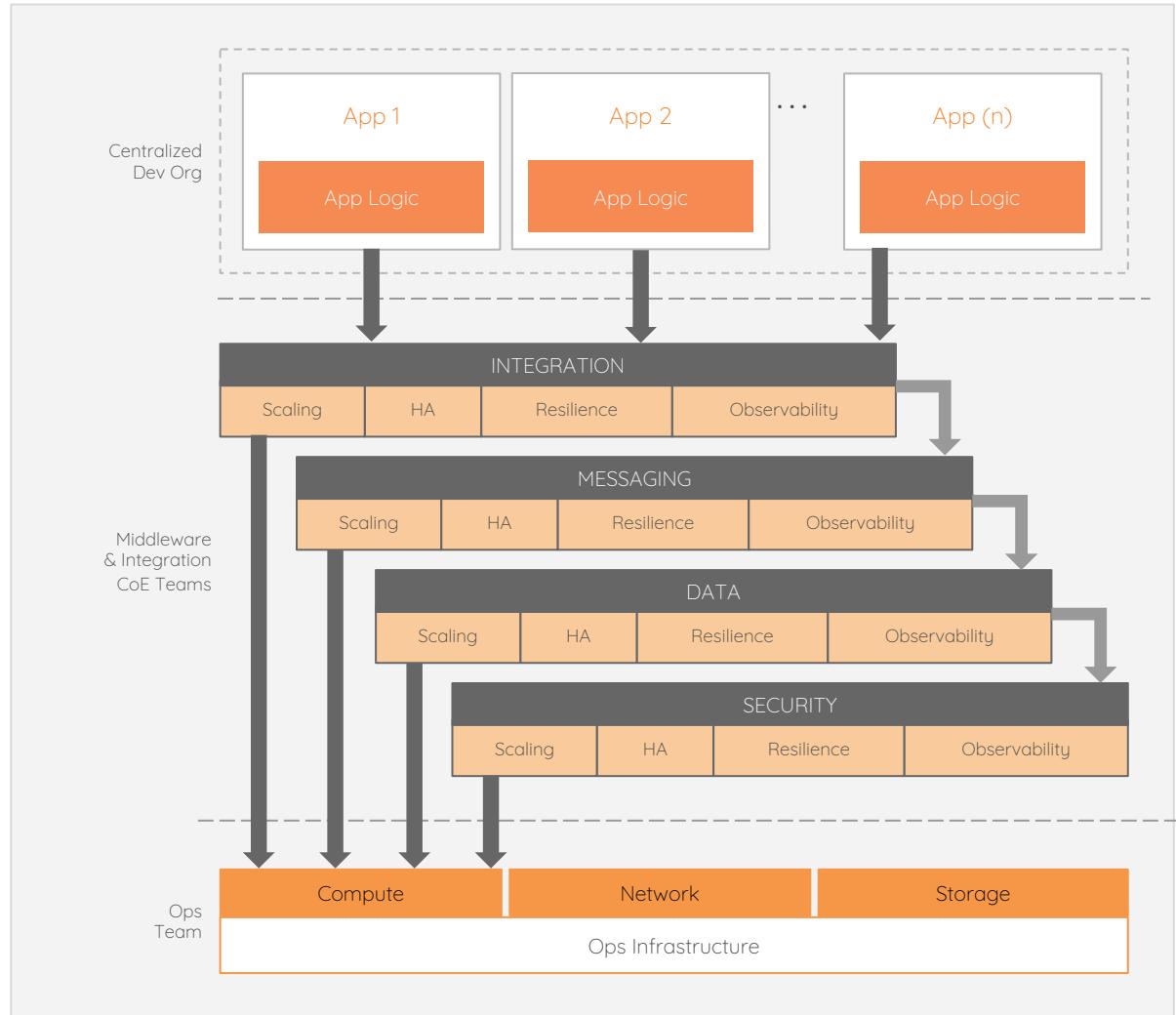
ESB as an implementation of SOA





Fast Waterfall

“Wagile”
“Fagile”



Enterprise Integration Patterns

Screenshot of the Enterprise Integration Patterns website (www.eaipatterns.com) displayed in a web browser.

The browser window shows the following tabs:

- Home - Enterprise Integrati...
- Offline Mail
- Inbox (124,820) - pe...
- WSO2
- WSO2, Inc. - Calenda...
- + bitmark
- Shorten with bit.ly

The main content area displays the following sections:

- Enterprise Integration Patterns** (Logo)
- Home**
- HOME • PATTERNS • RAMBLINGS • ARTICLES • TALKS • DOWNLOAD • LINKS • BOOKS • CONTACT**

Ramblings

My ongoing thoughts about the present and future of integration, SOA and Web services. [[see all](#)]

[DDD - Diagram Driven Design](#)
(March 22, 2010)

[What Does It Mean to Use Messaging?](#)
(Feb 17, 2010)

[A Chapter a Day...](#)
(Feb 1, 2010)

Upcoming Events

Articles & Interviews

[Conversations Between Loosely Coupled Services](#)
(Video on [InfoQ](#))

[Developing in a Service-oriented World](#)
(Video on [InfoQ](#))

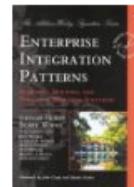
[SOA Patterns - New Insights or Recycled Knowledge?](#)
(Whitepaper)

Patterns and Best Practices for Enterprise Integration

This site is dedicated to making the design and implementation of integration solutions easier. The solutions and approaches described here are relevant for integration tools and platforms such as IBM WebSphere MQ, TIBCO, Vitria, SeeBeyond, WebMethods, or BizTalk, messaging systems such as JMS, WCF, or MSMQ, ESB's such as Sonic, Fiorano, ServiceMix, Mule, Apache Synapse, or WSO2, and SOA and Web-service based solutions.

All content on this site is original and is maintained by [Gregor Hohpe](#). I have been building integration solutions for large clients for many years and enjoy sharing my findings with the community. I hope you find this material insightful and useful. Please [contact me](#) if you have suggestions or feedback.

Enterprise Integration Patterns - The Book

 Enterprise integration remains harder than it really should be. While integration is inherently complex, I felt that one of the major stumbling blocks is the lack of a common vocabulary and body of knowledge around asynchronous messaging architectures used to build integration solutions. Under the guidance of Martin Fowler and Kyle Brown, I teamed up with Bobby Woolf to create such a language in the form of 65 [integration patterns](#) (see the pattern links on the right).

The book *Enterprise Integration Patterns* provides a consistent vocabulary and visual notation to describe large-

Integration Patterns

- [Integration Patterns Overview](#)
- [Table of Contents](#)
- [Revision History](#)
- Introduction**
- [Preface](#)
- [Introduction](#)
- [Solving Integration Problems using Patterns](#)

Integration Styles

- [Introduction](#)
- [File Transfer](#)
- [Shared Database](#)
- [Remote Procedure Invocation](#)
- [Messaging](#)

Messaging Systems

- [Introduction](#)
- [Message Channel](#)
- [Message](#)
- [Pipes and Filters](#)
- [Message Router](#)
- [Message Translator](#)
- [Message Endpoint](#)

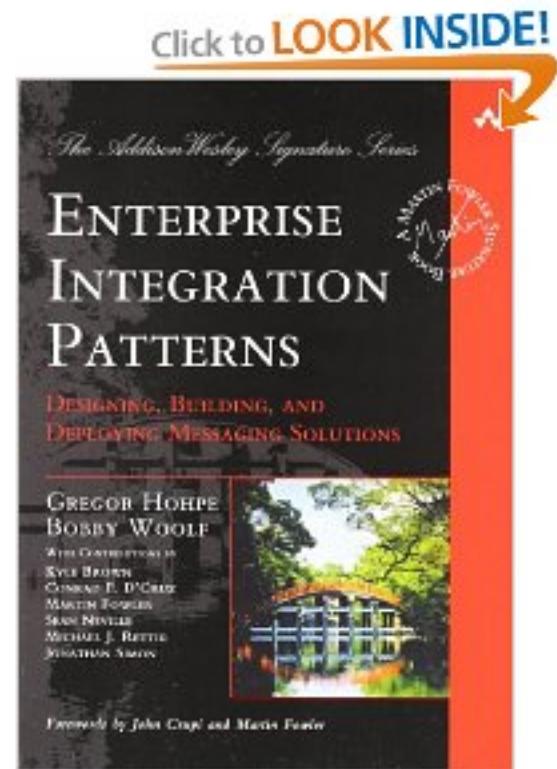
Messaging Channels

- [Introduction](#)
- [Point-to-Point Channel](#)



Enterprise Integration Patterns

- <http://www.eaipatterns.com/>
- The book
 - Enterprise Integration Patterns
 - Gregor Hohpe, Bobby Woolf



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

How does mediation / integration fit into Microservices / Containers?

“Smart Endpoints and Dumb Pipes”

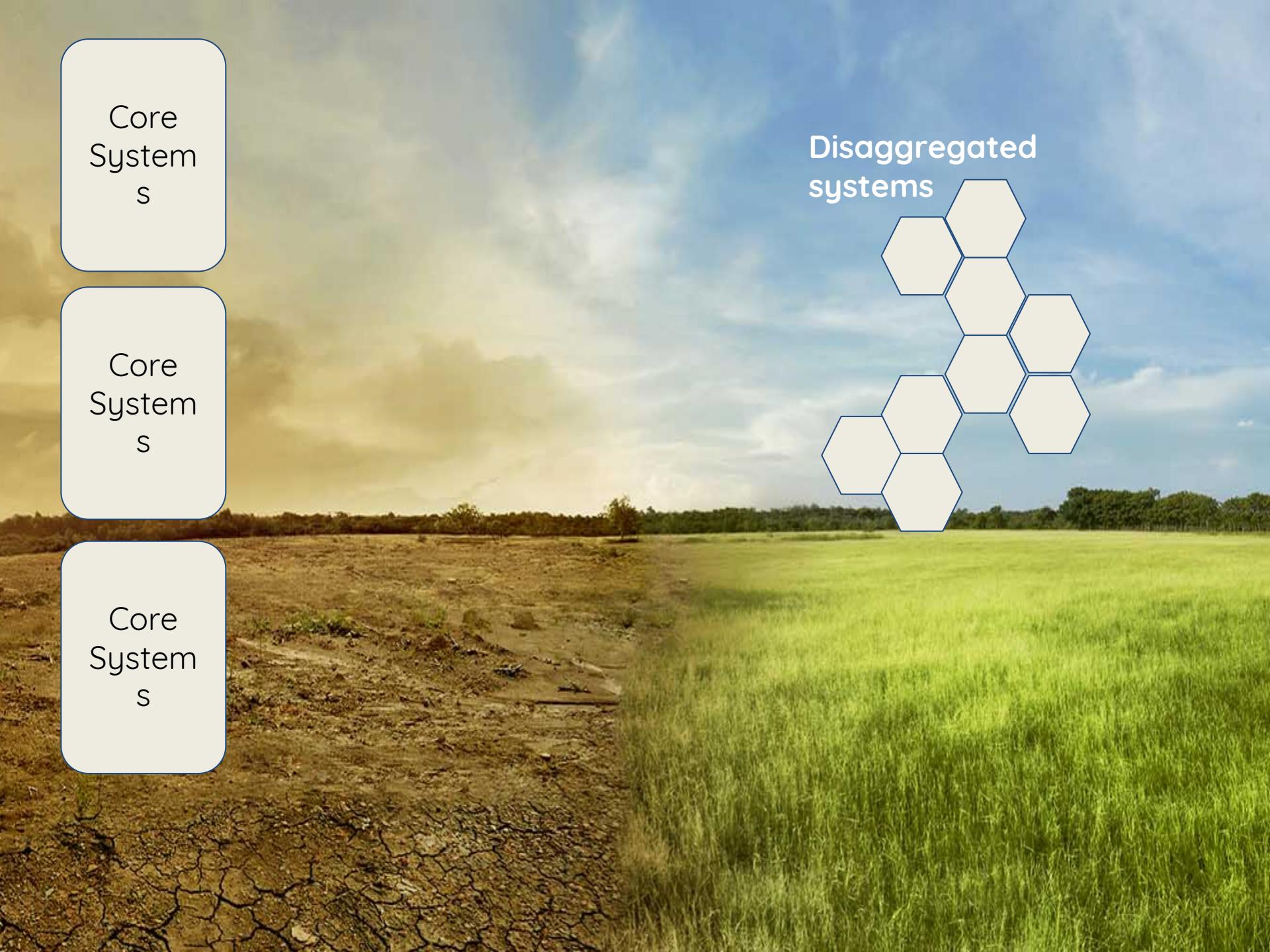


© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>



Brownfield

Greenfield

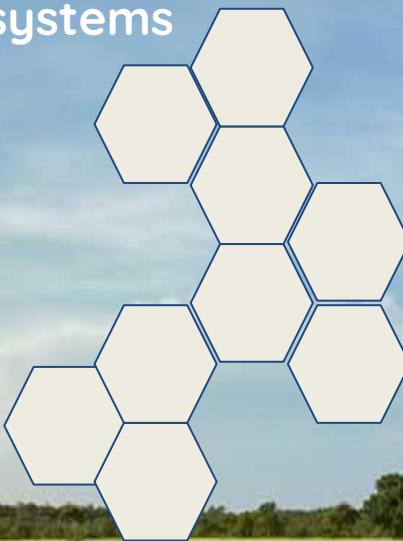


Core
System
s

Core
System
s

Core
System
s

Disaggregated
systems



Ballerina Language

- A new integration language and framework for Microservices, RESTful SOA
- Based on Swagger and Sequence Diagrams
- Textual and graphical are 100% interchangeable
- <https://ballerina.io>



© Paul Fremantle 2016 except where credited elsewhere. This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License
See <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Ballerina diagram and language

The screenshot shows the Ballerina IDE interface. On the left, the code editor window titled "demo.bal" displays the following Ballerina code:

```
18
19     @http:ServiceConfig {
20         basePath: "/"
21     }
22     service<http:Service> hello bind listener {
23         @http:ResourceConfig {
24             path: "/",
25             methods: ["POST"],
26             body: "js"
27         }
28         hi (endpoint caller, http:Request request, json js) {
29             string title = check <string>js.title;
30             string content = check <string>js.content;
31             github4:Issue iss = check
32                 | git->createIssue("pzfro", "btest", title,
33             json resp = {
34                 key: "value",
35                 id: untaint iss.id
36             };
37             _ = caller -> respond(resp);
38         }
39     }
40 }
```

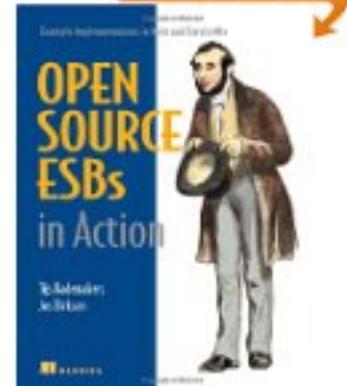
On the right, the "Ballerina Diagram" tab displays a sequence diagram titled "hello <http:Service>". The diagram shows interactions between a "caller" endpoint and a "git" service. A "default" lifeline is also present. The sequence of messages is:

- A synchronous message "request(js)" is sent from the "caller" to the "default" lifeline.
- An asynchronous message "createtissue("pzfro", "btest", title, content, id)" is sent from the "default" lifeline to the "git" service.
- A synchronous message "resp" is sent from the "git" service back to the "default" lifeline.
- The "default" lifeline sends a message "respond(resp)" back to the "caller".

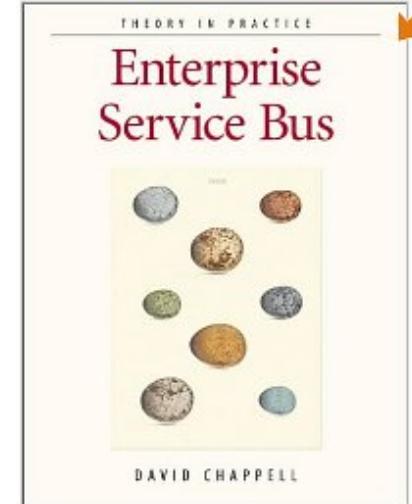
The bottom status bar indicates the code is at Line 33, Column 13, with 4 spaces, using UTF-8 encoding, and is a Ballerina file. There are 1 notification and 1 smiley icon.



LOOK INSIDE!



Click to LOOK INSIDE!



- Wikipedia!
 - http://en.wikipedia.org/wiki/Enterprise_service_bus
- Books
 - David Chappell: ESB
 - Open Source ESBs in Action
- Open Source
 - synapse.apache.org
 - wso2.com/products/enterprise-service-bus
 - servicemix.apache.org

