



[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.

[\[see all\]](#)

[What is in a Name?](#)

(Mar 5, 2006)

[Of Boxes and Lines](#)

(Feb 25, 2006)

[Revenge of the Nerds - OOPSLA 2005](#)

(October 23, 2005)

[Popular Articles](#)

[Developing in a](#)

[Service-Oriented World](#)

(Whitepaper)

[Integration Patterns with BizTalk Server 2004](#)

(Whitepaper)

[An Asynchronous World](#)

(Software Development Magazine, July 2003)

[Test-Driven](#)

[Development in](#)

[Integration Projects](#)

(Whitepaper)

[XML Abuse](#)

(Software Development Magazine, Dec 2002)

[Upcoming Events](#)

[Forrester's IT Forum 2006: GigaWorld](#)

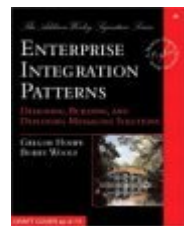
April 6

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 valid for most integration tools and standards such as IBM WebSphere MQ, TIBCO, Vitria, SeeBeyond, WebMethods, BizTalk, JMS, MSMQ, Web Services etc.

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.

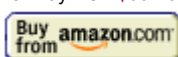
Enterprise Integration Patterns - The Book



[Enterprise Integration Patterns](#)

Gregor Hohpe

Best Price \$32.99
or Buy New \$39.73



[Privacy Information](#)

For quite a while I have been feeling that 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.

[Martin Fowler](#) noticed the same when he wrote *Patterns of Enterprise Application Architecture*. As a result, back in 2001 Martin and [Kyle Brown](#) started a series of discussions on messaging patterns. In early 2002 I joined the discussions and started collecting more patterns based on my ongoing client work. I teamed up with Bobby Woolf to document 65 [integration patterns](#)

[Integration Patterns](#)

[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](#)

[Publish-Subscribe Channel](#)

[Datatype Channel](#)

[Invalid Message Channel](#)

[Dead Letter Channel](#)

[Guaranteed Delivery](#)

[Channel Adapter](#)

Las Vegas

[SpringOne](#)

June 15-16

Antwerp, Belgium

[TheServerSide Java Symposium Europe](#)

June 21-23

Barcelona, Spain

News

["Starbucks Does Not Use Two-Phase Commit" selected for "The Best Software Writing"](#)

[Presentation Downloads](#)

TheServerSide

[Software Visualization and Model Extraction](#)

[PDF]

SD West

[Conversations Between Loosely Coupled Systems](#) [PDF]

SD Forum

[Developing in a Service-Oriented World](#)

[PDF]

JAOO

[Enterprise Integration Patterns](#) [PDF]

• [Patterns Roundtable](#)

[MSDN TV](#)

[Roundtable on .NET Patterns](#)

(see the links on the right). After multiple rounds of reviews, writer's workshops and conference presentations, the hard work bore fruits and the patterns catalog is now (Oct 2003) available in book form from Addison-Wesley, titled *Enterprise Integration Patterns*.

The book provides a consistent vocabulary and visual notation framework to describe large-scale integration solutions across many technologies. It also explores in detail the advantages and limitations of asynchronous messaging architectures. You will learn how to design code that connects an application to a messaging system, how to route messages to the proper destination and how to monitor the health of a messaging system. The patterns in the book are technology-agnostic and come to life with examples implemented in different messaging technologies, such as SOAP, JMS, MSMQ, .NET, TIBCO and other EAI Tools. [Download a sample chapter!](#)

"If you are involved with the operation or development of an enterprise application, there will doubtless come a time when you will need to integrate your application with another using the emerging preferred approach of messaging. When that time comes, this book will be your most valuable reference."

--Randy Stafford, Chief Architect, IQ Navigator
[\[More Testimonials\]](#)

Why Do We Need Integration?

Today's business applications rarely live in isolation. Users expect instant access to all business functions an enterprise can offer, regardless of which system the functionality may reside in. This requires disparate applications to be connected into a larger, integrated solution. This integration is usually achieved through the use of some form of "middleware". Middleware provides the "plumbing" such as data transport, data transformation, and routing.

What Makes Integration so Hard?

[Messaging Bridge](#)

[Message Bus](#)

Message Construction

[Introduction](#)

[Command Message](#)

[Document Message](#)

[Event Message](#)

[Request-Reply](#)

[Return Address](#)

[Correlation Identifier](#)

[Message Sequence](#)

[Message Expiration](#)

[Format Indicator](#)

Interlude: Simple Messaging

[Introduction](#)

[JMS Request/Reply Example](#)

[.NET Request/Reply Example](#)

[JMS Publish/Subscribe Example](#)

Message Routing

[Introduction](#)

[Content-Based Router](#)

[Message Filter](#)

[Dynamic Router](#)

[Recipient List](#)

[Splitter](#)

[Aggregator](#)

[Resequencer](#)

[Composed Msg. Processor](#)

[Scatter-Gather](#)

[Routing Slip](#)

[Process Manager](#)

[Message Broker](#)

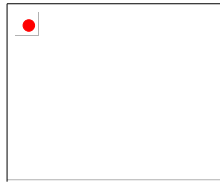
Message

Transformation

[Introduction](#)

[Envelope Wrapper](#)

[Content Enricher](#)



Architecting integration solutions is a complex task. There are many conflicting drivers and even more possible 'right' solutions. Whether the architecture was in fact a good choice usually is not known until many months or even years later, when inevitable changes and additions put the original architecture to test. Unfortunately, there is no "cookbook" for enterprise integration solutions. Most integration vendors provide methodologies and best practices, but these instructions tend to be very much geared towards the vendor-provided tool set and often lack treatment of the bigger picture, including underlying guidelines, principles and best practices.

Asynchronous Messaging Architectures

Asynchronous messaging architectures have proven to be the best strategy for enterprise integration because they allow for a loosely coupled solution that overcomes the limitations of remote communication, such as latency and unreliability. The trend towards asynchronous messaging has manifested itself in a variety of EAI suites as well emerging standards for reliable, asynchronous Web services. Unfortunately, asynchronous messaging is not without pitfalls. Many of the assumptions that hold true when developing single, synchronous applications are no longer valid. What is needed is vendor-independent design guidance on building robust integration architectures based on asynchronous messaging.

How can Patterns Help?

Patterns are a proven way to capture experts' knowledge in fields where there are no simple "one size fits all" answers, such as application architecture, object-oriented design, or message-oriented integration. Each pattern poses a specific design problem, discusses the considerations surrounding the problem, and presents an elegant solution that balances the various forces or drivers. In most cases, the solution is not the first approach that comes to mind, but one that has evolved through actual use over time. As a result, each pattern incorporates the experience base

[Content Filter](#)

[Claim Check](#)

[Normalizer](#)

[Canonical Data Model](#)

[Interlude: Composed Messaging](#)

[Introduction](#)

[Synchronous \(Web Services\)](#)

[Asynchronous \(MSMQ\)](#)

[Asynchronous \(TIBCO\)](#)

[Messaging Endpoints](#)

[Introduction](#)

[Messaging Gateway](#)

[Messaging Mapper](#)

[Transactional Client](#)

[Polling Consumer](#)

[Event-Driven Consumer](#)

[Consumer](#)

[Competing Consumers](#)

[Message Dispatcher](#)

[Selective Consumer](#)

[Durable Subscriber](#)

[Idempotent Receiver](#)

[Service Activator](#)

[System Management](#)

[Introduction](#)

[Control Bus](#)

[Detour](#)

[Wire Tap](#)

[Message History](#)

[Message Store](#)

[Smart Proxy](#)

[Test Message](#)

[Channel Purger](#)

[Interlude: Systems Management Example](#)

[Instrumenting Loan Broker](#)

[Integration Patterns in Practice](#)

that senior integration developers and architects have gained by repeatedly building solutions and learning from their mistakes. This implies that we did not “invent” the patterns; patterns are not invented, but rather discovered and observed from actual practice in the field.

The patterns on this site are by no means a complete treatment of all things integration. We focused on developing a cohesive set of patterns that would make a well rounded book. We continue to discover new patterns as part of our daily client work and hope to find the time to document them in the future.

[Case Study: Bond Trading System](#)


[Concluding Remarks](#)

[Emerging Standards](#)


[Appendices](#)

[Bibliography](#)

What am I Reading Right Now?

 [Essential Business Process Management](#)
Michael Havey, 2004

This one is hot of the press and part of O'Reilly's *Theory in Practice* series. Should you read another book on BPM? I think so, if your work is related to the world of SOA, Web services and you are tired of hearing people wax about pi-calculus, BPEL, CDL etc. This book is an easy read but contains sufficient theoretical background to debunk many of the myths around BPM and orchestration these days. Also, it goes beyond mere descriptions and includes process patterns based on the [work by Wil van der Aalst et al.](#) Having Wil as a reviewer pretty much guarantees technical correctness and ensures no handwaving occurs.

 [Web Services](#)
Alonso, Casati, Kuno, Machiraju, Springer Verlag, 2003

Somehow this book managed to get to market without me noticing. Springer Verlag is known for its in-depth academic books. This one is no exception with three Ph.D.'s among the authors. Nevertheless the book is pleasant to read and is chock full of real information on distributed systems architecture, with a particular focus on Web services. The book explains the rationale behind many of the Web services standardization

efforts. For example, the book dedicates a whole chapter each to service coordination and service orchestration. The book is on the higher end of the price scale but the amount of content it is definitely worth it.

© 2005 [Gregor Hohpe](#) • All rights reserved.
