# Books for Great Software Architects

It has been common to see posts and articles talking about what a software architect must read and what your book library, as an architect, should include. However, no one explains how to read them or if there is a specific sequence you must follow.



Great journey for great architects

Personally, and through my reading journey, I found a link between these books. This link defines which book is considered a pre-requisite for another and which book’s content can make sense when read after another and which one should be the next to gain the full picture.

Without knowing this link, the reading will be rough and frustrating, consuming a lot of your time and effort. For example, and in many cases, I had to read the same book twice; the first time was before reading its pre-requisites, which was boring and confusing. On the second time, it was after reading the right pre-requisites, where everything suddenly became crystal clear.

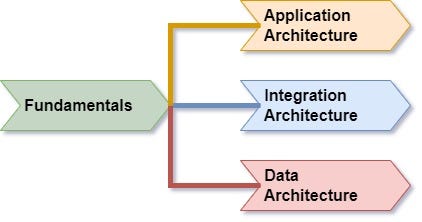
In this article, I will show and explain the right reading sequence for the most important books in the software architecture world in the form of a roadmap composed of multiple paths, which is supposed, if followed, to streamline your reading process and make your journey smoother (hopefully smoother than mine) and to become a great software architect faster.

Note: In this article, it’s assumed that you have a good experience in one or more programming languages and you are interested in entering the software architecture world.

# Overview

As explained in my previous article “[Types of Technology Architectures](https://vocal.media/01/technology-architecture-types-and-related-architectural-styles)”, there are many different types of software architectures that address different types of specializations, such as Application Architecture, Integration Architecture, Data Architecture, etc.

The reading roadmap in this article follows the same and it will have different paths according to the different types of software architectures, as depicted below:



Reading roadmap

As per the roadmap, no matter which architecture type you are interested to be specialized with, there is a fundamental path that must be followed before going into your own specialization.

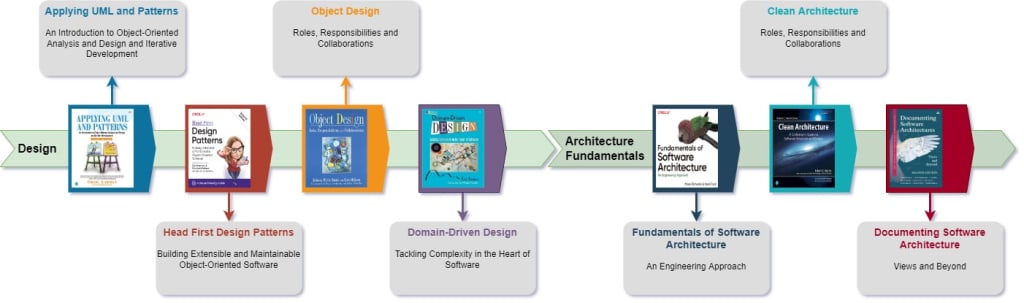
Accordingly, the article is divided into 4 parts:

* [Part 1 covers Fundamentals Path](https://vocal.media/01/books-for-great-software-architects-fundamentals-path)
* [Part 2 covers Application Architecture Path](https://vocal.media/01/books-for-great-software-architects-application-path)
* [Part 3 covers Integration Architecture Path](https://vocal.media/01/books-for-great-software-architects-integration-path)
* Part 4 covers Data Architecture Path (still under progress)

**Don’t forget to follow me to get notifications for my new articles and please clap for the article if you liked it.**

# Books for Great Software Architects - Fundamentals Path

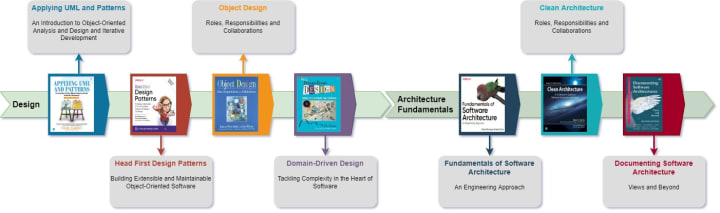
## Your first steps into Software Architecture



As we have seen in the [Overview](https://vocal.media/01/books-for-great-software-architects), the reading roadmap starts by the fundamentals path which will be explained in this story:

## Fundamentals Path

The fundamentals path is actually composed of two sub-paths; Design Path and Architecture Fundamentals Path.

Fundamental Path

**Design Path**

Before starting your journey to the software architecture you must be equipped with the basic skills, techniques and tools required through out the journey and this is exactly the purpose of the design path.

In this path, you will gain a lot of skills (such as development methodologies, design approaches, modeling techniques, design principles and patterns) that will support you through the subsequent paths.

By the end of this path, you will gain deep knowledge in the following topics:

* Responsibility-driven design (RDD) which is the base for Domain-Driven Design
* Object-Oriented (OO) Analysis and Design
* SOLID principles
* Modeling using UML
* GRASP principles
* Basic Agile Development concepts
* Object-Oriented (OO) Design Patterns
* Domain-Driven Design (DDD)

**Book 1: Applying UML and Patterns**, by Larman

This is a must read book and it must be your first book, because it will provide you with all you need to comprehend the subsequent books correctly. In this book you will learn:

* Object-Oriented Analysis and Design
* Modeling using UML (which is required while reading all the upcoming books)
* SOLID principles (I found it useful to learn about SOLID principles before reading any of the OO Design Patterns)
* Responsibility-Driven Design (RDD) and GRASP principles which are considered the base for Domain-Driven Design (I found it important to understand RDD before reading DDD)
* Development using Agile methodology
* Few design patterns

**Book 2: Head First Design Patterns**, by Freeman

This book must be your second book to learn about OO Design Patterns. In this book, you will learn, in a very simple, yet comprehensive, way, the context of each , the problem it addresses and the solution it proposes.

The thing I liked most in this book is the focus on explaining the relationship between the design patterns and SOLID principles (SOLID principles covered in Book 1) and how the design patterns' solutions are implemented to enforce the different SOLID principles.

**Book 3:Object Design: Roles, Responsibilities and Collaboration**, by Ivar

This book is taking the RDD (RDD basics are covered in Book 1) to the next level and taking it into more details.

The thing I liked most in this book is the differentiation between role and responsibility, roles stereotypes and CRC cards that can be used to capture early design ideas.

**Book 4: Domain-Driven Design Tackling Complexity in the Heart of Software**, by Eric

This book is the last in the design path and it is also must to read. This book is following RDD principles (RDD covered in Book 1 and Book 3) to build domain model using techniques that aligns with the business domain and business stakeholders.

**Architecture Fundamentals Path**

While the design path was the preparation stage, Architecture Fundamentals is actually your first steps into the software architecture where you will learn about the main architectural concepts, principles, best practices and patterns and you will also learn about architecture documentation.

**Book 5: Fundamentals of Software Architecture**, by Mark Richards

In this book, you will learn the basic architectural concepts, elements, characteristics, rules, principles, measures and styles. It also shows the most important skills that an architect must have.

**Book 6: Clean Architecture**, by Robert Martin

This book is building on the previous book by focusing more on the architecture structure and the relationships between the different parts to come up with clean, maintainable and extendible solution.

I found Component Principles (Reuse/Release Equivalence, Common Closure, Common Reuse, Acyclic Dependencies, Stable Dependencies, Stable Abstractions principles) very useful.

**Book 7: Documenting Software Architecture**, by Paul Clements

By now, you should be able to build an architecture based on Books 5 and 6 and you need now to document them. This book explains how an architecture must be described and documented, although I didn't like the proposed template, but the book gives you all the techniques you need to build your own template and eventually your architecture documents.

## Bonus Reading

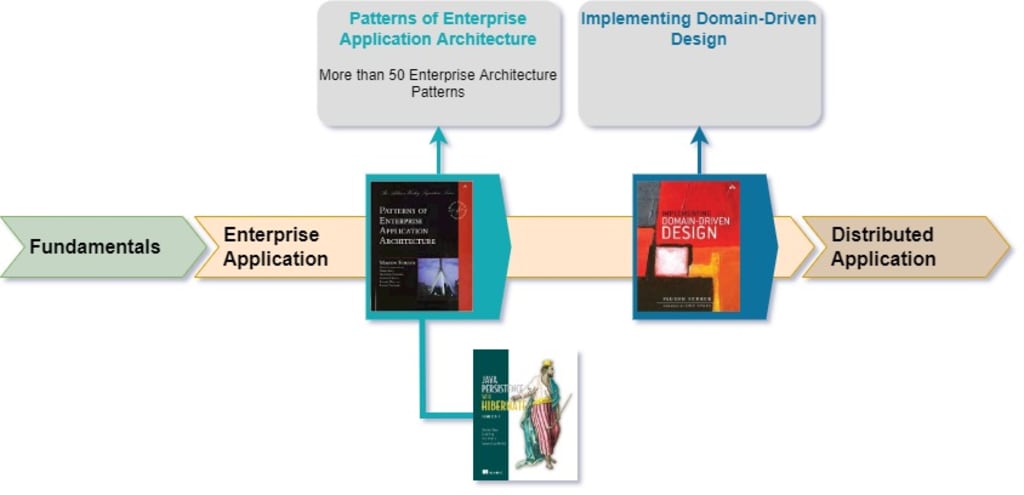
If you are involved in projects delivery and you are a part of a DevOps process, I would recommend for you to read a book called "Continuous Architecture in Practice, by Murat Erder"

## Summary

In this part, we started the reading journey by learning about the fundamentals path. Now we are ready to move on into; [Application](https://vocal.media/01/books-for-great-software-architects-application-path), [Integration](https://vocal.media/01/books-for-great-software-architects-integration-path)and/or Data Architecture paths.

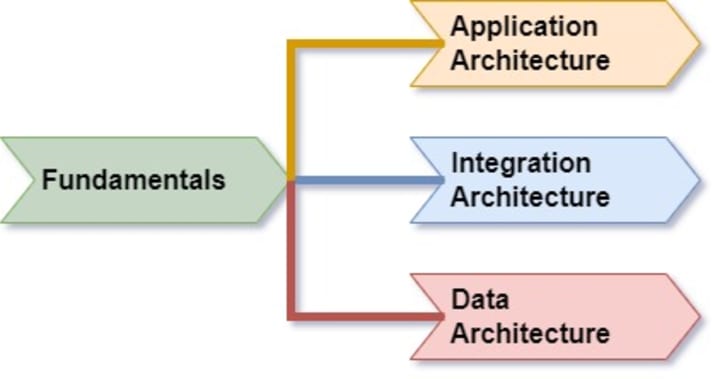
# Books for Great Software Architects — Application Path

## This story will draw the reading path for Great Application Architects



Application Path

In the [Overview](https://vocal.media/01/books-for-great-software-architects), we have seen the high-level roadmap and in [Part 1](https://vocal.media/01/books-for-great-software-architects-fundamentals-path), we went through the Fundamentals Path that covered the first steps into the software architecture which is also considered as a pre-requisite for the specialized parts, 2, 3 and 4.

Reading Roadmap

By completing [Part 1](https://vocal.media/01/books-for-great-software-architects-fundamentals-path), you will be ready to get specialized in one of the architectural domains. In this part, we will draw the reading path for Great Application Architects.

## Application Architecture Path

Application Architecture defines the software structure by describing its elements and the relationships between these elements. To learn more about the applications elements (Module, Subsystem and Component), please read my article “[Main Concepts in Software Architecture](https://vocal.media/01/main-concepts-in-software-architecture)”

In this path, we will focus mainly on the enterprise application architecture and distributed application architecture.

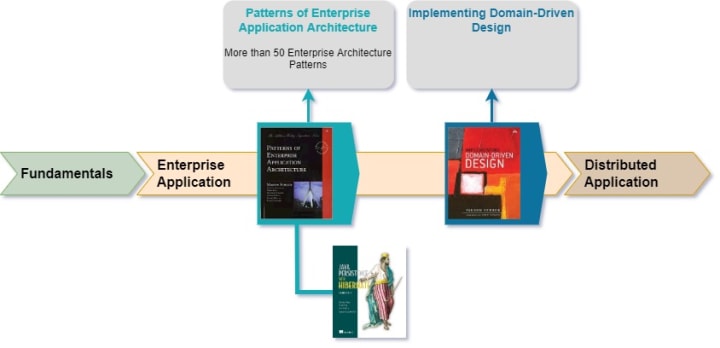
Application Architecture Path

Distributed application is a special type of the enterprise application where its elements and parts are distributed into multiple hosts (physical servers, virtual machines, or containers).

**Enterprise Application Architecture Path**

Enterprise Application is an Application that supports concurrent access through APIs and/or UI screens and requires heavy involvement for data persistence. Enterprise Application is also known as Information System.

In this path, I found the following two books are the most valuable in this domain:

Enterprise Application Architecture Path

**Book 1: Patterns of Enterprise Application Architecture**, by Martin Fowler

This book provides more than 50 patterns addressing multiple issues and problems related to data mapping, domain logic, web presentation, session management and concurrency.

I found the ORM-related patterns (more than 15 patterns) useless as they are already implemented as out of the box capabilities by nowadays ORM frameworks. My recommendation is to read one of the existing books specialized in the ORM rather than reading them here. If you are a Java developer, my recommendation is to go with: [Java Persistence with Hibernate](https://www.manning.com/books/java-persistence-with-hibernate-second-edition).

The most interesting patterns are the one related to the concurrency patterns and base patterns

**Book 2: Implementing Domain-Driven Design**, by Vaughn

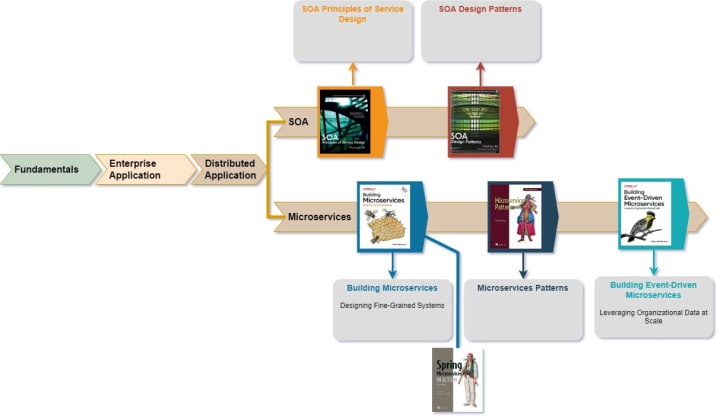
This is one of my favorite books as it capitalizes on the principles and patterns discussed in the books mentioned in this part and the [previous part](https://vocal.media/01/books-for-great-software-architects-fundamentals-path) (especially the books; Applying UML and Patterns, Head First Design Patterns, Domain Driven Design, and Patterns of Enterprise Application Architecture), and provides a consolidated, yet practical, implementation for those principles and patterns.

Note: it is very important to read DDD by Eric first (the blue DDD) before reading this book, otherwise you will be fully lost and confused.

**Distributed Application Architecture Path**

Here, we will be building on the previous path to learn about the distributed application architecture.

As of today, the most common and most important distributed architectural styles are SOA and Microservices and accordingly the distributed application path is also built to support these two styles:

Distributed Application Architecture Path

Although the path is segregated into two parallel paths one for SOA and another for Microservices, I found it useful to read SOA path first before going into Microservices where a lot of Microservices principles and patterns are inherited from SOA and it will give a great idea about the differences and similarities between them.

If you are interested in Microservices and not welling to go through SOA, I would recommend for you to read my article “SOA and Microservices Full Comparison”.

**Book 3: SOA Principles of Service Design**, by Erl

This is one of the most important books in the SOA world. It lists and explains the different principles that when followed, you will get the real value from it.

**Book 4: SOA Design Patterns**, by Erl

This book provides about 85 design patterns (basic and compound) grouped mainly into inventory and inventory layering patterns, service design, composition and implementation patterns, service contract patterns, governance and security patterns.

I found this book interesting as most of the patterns covered here are either supporting API designs patterns or Microservices patterns (you will see a lot of Microservices patterns are actually inherited from SOA patterns).

Don’t be confused with another book called SOA Patterns by manning, the later is weak and shallow.

**Book 5: Building Microservices**, by Newman

This is one of my favorite books. In this book, the author tries to highlight the main challenges you are going to face while transforming to Microservices and he also tries to address these challenges by providing the right techniques and suggesting different frameworks, tools and libraries to support you through the transformation process.

Note: It will be highly valuable if you can empower what you have got from this book with an implementation hands-on. If you are a java developer, I would recommend for you “Spring Microservices in Action”

**Book 6: Microservices Patterns**, by Richardson

In this book, you will formalize what you have learned so far into a set of patterns that links the challenges with solutions and gives them a name that is supposed to streamline your solutioning skills and build strong architectural vocabulary.

**Book 7: Building Event-Driven Microservices**, by Adam

Microservices prefers choreography over orchestration for higher isolation between the services. This book builds on the previous books to implement choreography-based microservices using event-based platforms.

## Bonus Reading

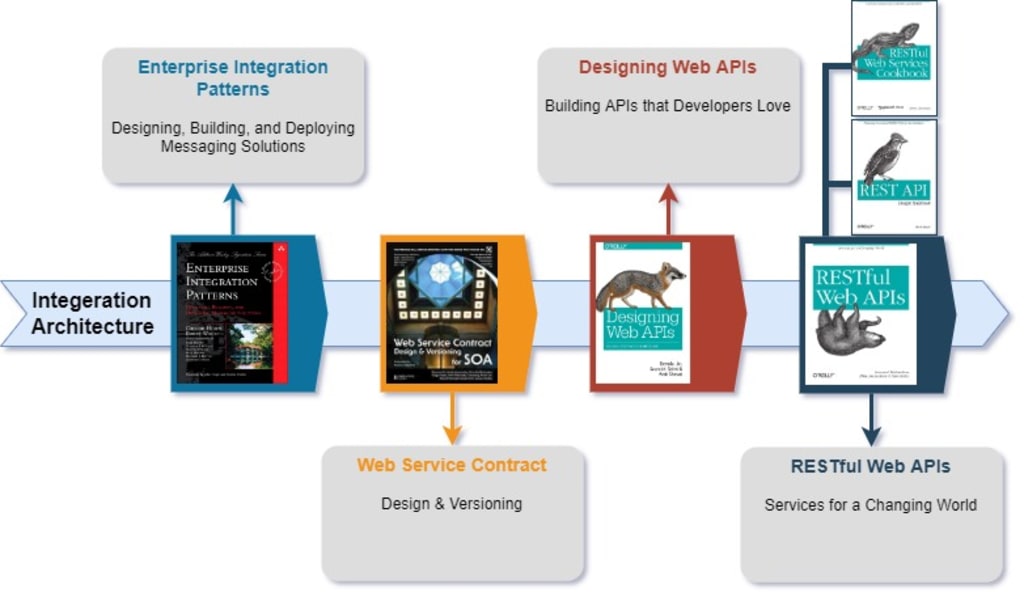
What have been introduced so far is enough for you to have all the skills needed to be a great application architect. But if you feel like you need to read more, I would recommend for you a book called “Building Evolutionary Architectures, by Neal Ford”.

## Summary

In this part, we have built on [Part 1](https://vocal.media/01/books-for-great-software-architects-fundamentals-path) and provided a specialized reading path for application architecture.

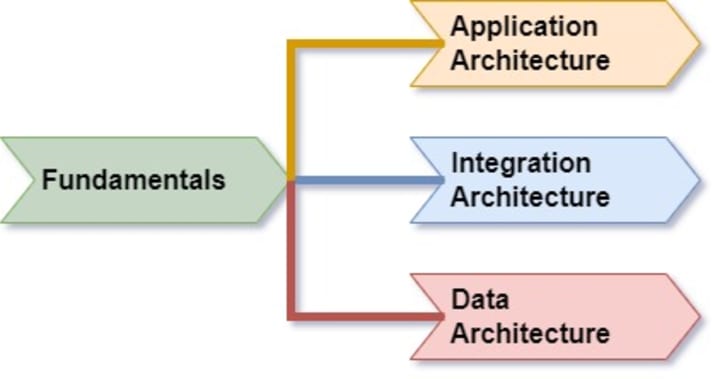
# Books for Great Software Architects — Integration Path

This story will draw the reading path for Great Integration Architects



Integration Path

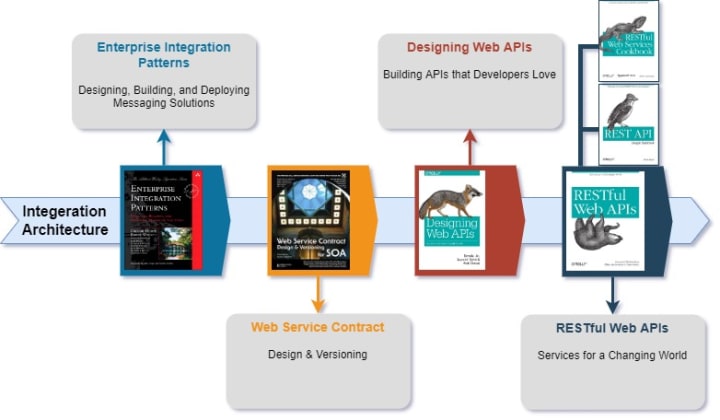
In the [Overview](https://vocal.media/01/books-for-great-software-architects), we have seen the high-level roadmap and in [Part 1](https://vocal.media/01/books-for-great-software-architects-fundamentals-path), we went through the Fundamentals Path that covered the first steps into the software architecture which is also considered as a pre-requisite for the specialized parts, 2, 3 and 4.

Reading Roadmap

By completing [Part 1](https://vocal.media/01/books-for-great-software-architects-fundamentals-path), you will be ready to get specialized in one of the architectural domains. In this part, we will draw the reading path for Great Integration Architects.

## Integration Architecture Path

Integration Architecture path focuses on the messaging, SOAP-based Webservices design, Web APIs design as well as REST API design.

Integration Architecture Path

**Book 1: Enterprise Integration Patterns**, by Hohpe

This book is a must for every integration architect and it should be your first book as well before moving forward in your reading journey. It is impressively comprehensive book with about 55 patterns and with many examples showing when and how to use the different patterns.

**Book 2: Web Service Contract Design & Versioning**, by Erl

Although the RESTful webservices are dominating nowadays, but it is still common to see and implement SOAP-based webservices especially in the financial and banking systems. For those who are still working on SOAP-based webservices, this book is a must for you. It is a comprehensive guide to design, version and govern SOAP-based webservices contracts.

Although the title is directed specifically to SOA, but it is still useful and valuable for non-SOA integrations.

**Book 3: Designing Web APIs**, by Jin

This book is focusing on the Web APIs, listing the different APIs paradigms (RPC, REST, GraphQL, Event-Driven), and it is also defining how to choose the right paradigm. This book is also providing a set of best practices that need to be followed while implemented and documenting a Web API.

To learn more about the different types of APIs and APIs history, please read the following article “[API Definition and API History](https://vocal.media/01/api-definition-and-history)”.

The consolidated list of Web API design rules discussed in the above books are covered in the following article “[Consolidated Design Rules and Standards for Great Web API](https://haitham-raik.medium.com/consolidated-design-rules-and-standards-for-great-web-api-e3c3df6a3dac)”

**Book 4: RESTful Web APIs**, by Richardson

This book is building on the previous one by focusing on designing RESTful APIs. The book is focusing on the best practices, especially the usage for hypermedia which is absolutely good, but the thing that I didn’t like about this book was the trying to take the hypermedia into a more dynamic level which is over-complicating the implementation on both the service provider and consumer.

It is good if you can empower what you have read in this book with another two related books:

* REST API Design Rulebook which provides a set of rules that enables you to build consistent webservices interfaces. In this book there are parts talking about something called WRML which is something invented by the author and has no practical implementation, so whenever you see WRML, just skip it.
* RESTful web services cookbook: this book provides more than 100 recipes that address complex scenarios and provide you with the best designing options.

The consolidated list of REST API design rules discussed in the above books are covered in the following article “[Consolidated Design Rules and Standards for Great REST API](https://haitham-raik.medium.com/consolidated-design-rules-and-standards-for-great-rest-api-3f9451d49e19)”

## Bonus Reading

It will be good to read more about event-driven architectures and for that purpose, I would recommend a book called “Building Event-Driven Microservices, by Adam”

## Summary

In this part, we have built on [Part 1](https://vocal.media/01/books-for-great-software-architects-fundamentals-path) and provided a specialized reading path for integration architectures.