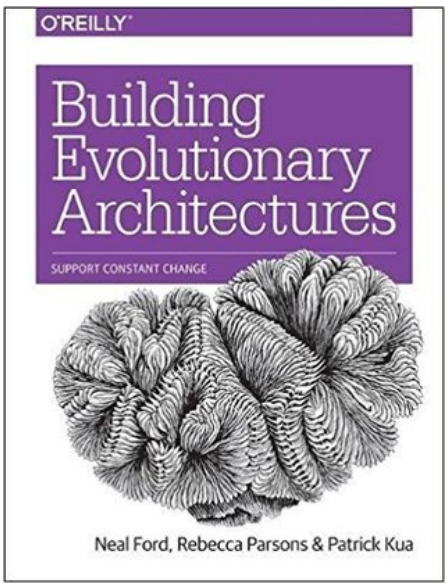

Building Evolutionary Architectures: Support Constant Change

Neal Ford , Rebecca Parsons , Patrick Kua

Building Evolutionary Architectures: Support Constant Change

Neal Ford , Rebecca Parsons , Patrick Kua

Building Evolutionary Architectures: Support Constant Change Neal Ford , Rebecca Parsons , Patrick Kua
The software development ecosystem is constantly changing, providing a constant stream of new tools, frameworks, techniques, and paradigms. Over the past few years, incremental developments in core engineering practices for software development have created the foundations for rethinking how architecture changes over time, along with ways to protect important architectural characteristics as it evolves. This practical guide ties those parts together with a new way to think about architecture and time.

Building Evolutionary Architectures: Support Constant Change Details

Date	:	Published October 13th 2017 by O'Reilly Media
ISBN	:	9781491986363
Author	:	Neal Ford , Rebecca Parsons , Patrick Kua
Format	:	Paperback 190 pages
Genre	:	Science, Technology, Computer Science, Programming, Architecture, Software, Nonfiction

From Reader Review Building Evolutionary Architectures: Support Constant Change for online ebook

Vinayak Hegde says

This is quite a good book written by someone who has deep experience in building several systems. Many of the ideas as well the experiences in building systems resonated with me. The idea of fitness functions was good. Also liked the transpose of different types of architecture and their pros and cons.

Also one important idea that is not expressed in software enough is the coupling of architecture, culture and team structure. This is dealt at length here. However some of the prose felt repetitive and there were not enough examples of change management (the few examples that were there were good, but the book could have done with more of the same). IMHO the fictional made-up example of widgets was the Achilles heel of otherwise good book.

Also it is one of those books that are relevant at a point in time. If you are more experience, many of the ideas seem obvious. If you are less experienced then there is not enough context to appreciate the ideas in the book. Overall a decent read and would rate it 3.25 stars.

Franck Chauvel says

Neil explains how to use continuous delivery and other modern engineering practices to build architectures that better sustain time and avoid rotting.

I think this text explains well how various practices and tools fit together (DDD, CD, testing, etc.) but does not dive into any. I don’t know if the analogy with evolutionary computing and its fitness functions really helps those unfamiliar with it. What I found less clear is how to decouple those fitness functions from the code so as to enable evolution.

Jared Davis says

I do not enjoy writing bad reviews, and less so for a book that I've only partially read, but I believe it's warranted in this case. Lots of people will encounter *Building Evolutionary Architectures* in their professional life -- either as current or as aspiring software architects and/or team leads -- and they'll make decisions based on what they learn (or misunderstand). So I think it's fair to pan the book with the understanding that I just want people building software and managing projects do a great job and don't create more confusion than already exists out there.

I have a simple complaint: the author does not seem to know much about evolutionary theories. I am not an expert either, but I do understand three things:

1. an evolutionary theory explains (with predictive power) the effect of reproduction on a species' population
2. an evolutionary theory explains (with predictive power) the effect of survival of an individual has on a species' population

3. an evolutionary theory explains (with predictive power) the effect of mutation (or adaptation) on a species' population

A "fitness function" basically encapsulates some interpretation of those three aspects of evolutionary theories. The output basically predicts growth or decline of a species population given adjustable parameters.

So what's an evolutionary theory of software architecture? What's a "species" in terms of software architecture? What does it mean for an architecture to "reproduce", or "survive", or "mutate"? The book does a woeful job settling these terminological points.

One might defend the book's approach by suggesting that evolutionary theories serve only as a metaphor, and that a reader should not look for a close correspondence between the practices the authors describe and the theories that inspire those descriptions. But I am not so sure.... why reach for a highly technical (and not-at-all-unified-yet), deeply researched collection of biological theories as the basis for a business-process metaphor if you're not going to be precise? It's too easy to be mistaken about the science in such a way that the business-practices become pathological... hence, one star.

Sten Vesterli says

Even though the main idea of this book is fatally flawed, it is still a good book on IT architecture.

The flawed concept is the idea of a "fitness function" that automatically calculates the quality of an IT architecture. If it were possible to build such a function, it could be included in your build pipeline, you would notice and could prevent your architecture from degrading. The authors present the idea and benefits, but fail to argue convincingly how this can be done in practice.

The good part is a well-argued discussion of architectural coupling and what causes architecture to atrophy. The chapter on how to build evolvable architecture contains many good guidelines and there is a lot of knowledge in the architecture antipatterns chapter. Finally, the chapter on putting evolutionary architecture into practice is also full of actionable advice, even though implementing the book's main idea of a "fitness function" is not practicable.

Maria Duffy says

I'm not 100% who the target audience is for this book. I was also very disappointed by some of the terrible scientific analogies, in particular when they use quanta!

But anyway, aside from that I appreciated the writing. There was a lot of spelling mistakes but I noticed that an effort had been made to use gender neutral language. I also though that including examples at the end of each chapter was a good idea.

On the whole though, aside from giving concepts new names I'm not sure how much I learned from this book.

William Anderson says

While I agree with everything discussed, and what is outlined, I believe this book has spectacular title but more encompasses what has become standard practices in monitoring, analytics, and dev / devops. Taking to heart practices from eXtreme Programming, (re)reading "The Pragmatic Programmer: From Journeyman to master" and having an established and consistent CI and moving towards CD are the real recommendations. Ultimately then defining Evolutionary Architectures as ones that change based on information derived off of these metrics/considerations.

Ricardo Cavalcanti says

The book builds on top of ideas from Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation and Building Microservices: Designing Fine-Grained Systems, feeling almost like a continuation.

This book is an agile take on classic software architecture work. The idea of fitness functions as a way to evaluate the architecture as it is being built is very interesting, although not new.

It feels a bit short. I'm looking forward to a "Building Evolutionary Architectures: Fieldbook" with more real life examples.

Adam Sipos says

A couple of good ideas but I was expecting more.

David Edwards says

This is an excellent book that crystalizes definitions and approaches to designing software systems that are amenable to change. Many of the concepts captured in this book have been practiced by experienced software engineers for many decades, but the discipline of evolvable software architecture has been lost since the explosion of the Internet ushered in fast and furious development. The authors do a great job of steering clear of discussing technology and instead focus on the essence of architecture and articulating the tradeoffs that come with the decisions we make.

Nadim Rafehi says

Well written with a lot of useful and insightful information, however it's somewhat lacking in exploring concrete examples of fitness functions.

npacenop says

Let me first say that I've still got a deep respect for the authors, even though I didn't particularly enjoy this book.

I've been tracking the work of the authors (and their peers) for some time now and am somewhat invested in the field of evolutionary architecture myself. That being said, I was somewhat disappointed that there were rather only a few new ideas in there for me. The whole book felt more like a unstructured taxonomy on current buzzwords.

Koen Wellens says

Building Evolutionary Architectures is criticised because of its overuse of biological terms. I get that. They do it a lot. However, if you understand what they’re going for, it doesn’t botter you that much. What did bother me was the fact that at some points the book was selling stuff. Either other books from the authors or even products from the company they work. That was a bummer.

There was also a lot of repetition in this book. In the end, there wasn’t that much in it that was new to me. I do like the idea of fitness functions and making them upfront. A lot of things that are explained are logic, but the diagrams provided help understand the impact. Like, what is a plugin architecture? Okay, it’s logic. There’s plugins. But when you see it, you completely understand how it works.

Read the full review at my blog.

Steve Fenton says

Firstly, this is an interesting read and there are several concrete ideas that I know I'm going to use. The writing style makes it easy to read (see below) and the concepts are supported by clear diagrams and descriptions.

I recommend this book to any developer interested in the bigger picture.

I can't end this review without mention of the disastrous proof reading in this, the first edition (October 2017). It is practically impossible to catch every mistake when preparing a book for release, but the number of basic mistakes is truly exceptional. Having to re-read so many sentences to "make them work" drove me to distraction.

I still recommend this book, but be aware that there are a goodly number of sentences you'll need to rewrite mentally as you go.

Please fix this in the second edition!

Emre Sevinç says

This book is a very high-level overview of the current thinking and challenges in modern software architecture and design. If you can stand the overuse and overstretching of terms from physics and biology such as "quantum", "quanta", "fitness function", etc. you can have a general idea about what it means to design for changeability.

If you subscribe to the idea that the software architecture means "the parts that are difficult to change", you already know how important to it is to design for relatively easy change, because change is one thing that will certainly happen to any software-intensive system that can survive at least a few months.

Some parts of the book is like time travel where the authors compare the architectural patterns from the end of 1990s and beginning of 2000s, and how things changed as of 2018: within the space of few pages they compare and contrast enterprise service bus based architectures with microservices.

The downside of the book: if you aren't already familiar with most of the concepts and techniques, it'll be difficult to understand what the book is all about. And if you're already familiar, then you know that for many of the topics mentioned, there are dedicated and well-written books.

If you want a very broad and high level overview, this book might serve you for a short period of time. If you want to go into more detail, you'll have to dive into titles such as Infrastructure as Code: Managing Servers in the Cloud and Architecting for Scale: High Availability for Your Growing Applications.

Tbueno says

The book contains good content, but it feels much more like a compendium that a book with new ideas.

This book is basically a summary of https://www.goodreads.com/book/show/1... and https://www.goodreads.com/book/show/8.... If you have read both these books, "Building Evolutionary Architecture" will bring no novelty to your knowledge about modern software architecture.

The premise of "Evolvability " as one of the "ities" (scalability, security, etc..) we should be concerned with in an architecture is interesting, though.
