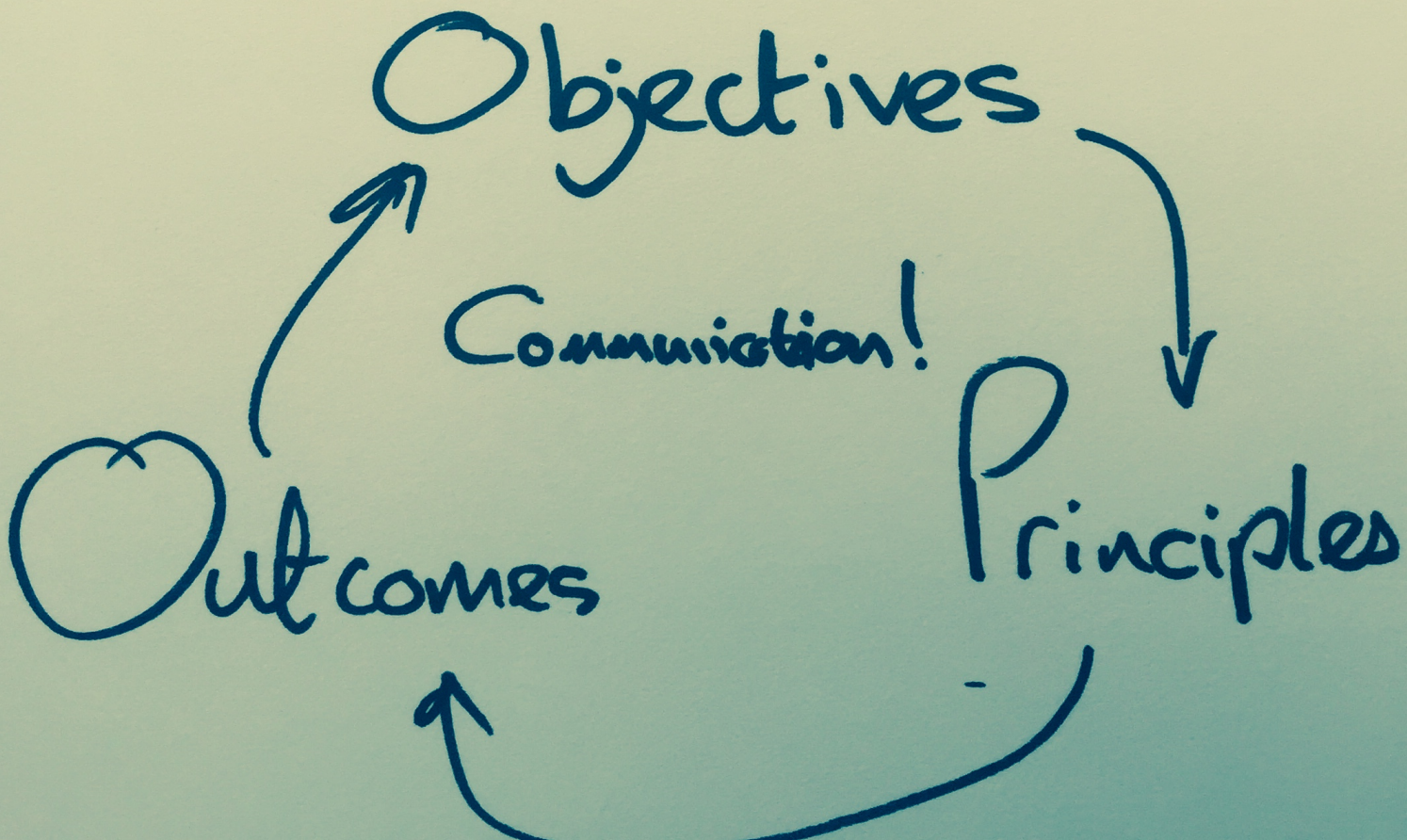
Guidebook for Software Architects



Guidebook for Software Architects

Russ Miles

This book is for sale at http://leanpub.com/guidebookforsoftwarearchitects

This version was published on 2014-12-16

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Also By Russ Miles

Antifragile Software

Contents

Introduction

It's a Saturday morning, 22nd November 2015 and I'm up early. The combination of excitement and jet-lag from an amazing week in Abhu Dhabi the key contributors to my sudden morning insomnia.

It's a great moment though. I'm about to embark on an experiment that even Eliyahu Goldratt would be proud of; I'm going to apply the theory of constraints and write a book in a single day. I'll be doing this entirely alone while continuing to do the normal things I would do on this Saturday.

My chosen topic for this day of madness? Software Architecture. Or, more accurately, the sorts of things a modern software architect needs to consider. In fact, the title says it all, this is a guidebook for how to take on the role of what I consider to be a modern software architect.

It's going to be tough, life cannot stop while I do this… but I hope you'll enjoy coming on this ride with me. At the very least it should be fun and something to tell my kid… although of course that will likely result in the bored, "Daddy, can we do something else?", reaction I usually get, but if you can't do things in your life that you feel the urge to tell your kids, and that they will promptly find the most boring stories in the world, then what are you doing with your life?

Why?

The first task when starting anything is to ask the question, "Why are you doing this?". So here are my reasons.

The main reason is the first one really, I feel that a modern software architect is slightly confused as to their role. In a world where we actively try and reduce the amount of "Big Decisions" that need to be made, this traditional stomping ground of someone who has the role of architect (or team lead, or anyone who guides and leads … see Princple 1 in the main content of this book!) is looking decidedly shaky.

This book aims to help anyone leading software development, particularly those who have to think long-term, in how to approach the job. What they need to think about, some ideas for how to go about it, and what outcomes might result.

It's an iterative process

Admittedly, today's book will likely be a first draft and I'm sure I'll be tweaking right up until midnight and beyond. The aim here is to force myself to write while also continuing on with my day as per usual.

As such it's likely to be a very iterative process. Today's draft will form the backbone for further writing, more pictures and graphics, and likely more refinement generally. Since this is how modern software is crafted it feels strangely authentic to write a book that guides software architects in much the same way. Pay heed dear readers, even the way the book is being written is a lesson in the challenges a modern architect will face and how to overcome them… I think that's sufficiently mysterious for now!

Trains, Tools, Cafes and Pubs

Mostly this book had to be simply written quickly and well. For that I chose the following tools to write the book, and to publicise the fact that I was doing something this crazy and fun.

Thanks to all of the tool vendors mentioned in the following sections; you're doing great things so please keep it up!

Hardware

- **Seawhite of Brighton** notepads for diagrams.
- **Sharpie Pens** for diagrams.
- **Apple iPhone 6** for photos of diagrams and other as the day progressed. Also for taking shots of the day as I journey from Eastbourne to London.
- **Mac Air** is my trusty typing friend.

Software

- **Hipstamatic** for making it easy to take interesting photos of the diagrams for the book, as well as the settings of where the book was written.
- **Instagram** for dropping photos to social media as my writing journey plays out.
- **IA Writer** for being fantastically minimal as I wrote at pace on this book. The best tool I've found for working with Markdown so far!
- **Git & GitHub** for being fantastic tools for version control for helping me not lose old changes and roll back when I needed to.
- **LeanPub** for being the best tool for rapidly delivering books. Still in love with this approach and, although not for everyone, certainly works for me!

Logistics & Other

- **Urban Ground** for coffee in Eastbourne
- My **Hot Tub** for just being there and being awesome.

As the book writing has progressed I have visited these wonderful places as the Biker Software Monk:

- Malaga (November, 2014)
- Eastbourne (November, 2014)
- London (December, 2014)
- Prague (December, 2014)

Apologies for any errors

Since I'll be writing this book alone there will be no copy editor and I feel I should apologise for that in advance. Copy editors, and editors and publishers in general, do an amazing job of turning disparate rantings of authors into something the public will understand and, most importantly, possibly pay for.

I'm not trying in any way to demean that role in creating great books. You only have to look at the editing of the Montaigne's Essays throughout his life and even after his death to see the refinement that is necessary even to those books we consider 'great' to begin with.

No, the difference here is I'm doing this in one day and I can't possible inflict that on anyone else.

This also means that since I'll be on the run (I'm looking right now at the clock and knowing I have to de-camp to a cafe in a few minutes) throughout my day of writing this book, there are likely to be LOTS of TBD entries for images and such until I can navigate to those WiFi enabled islands on my journey.

All I ask is that you bear with me as I go through the day. Sometimes the book will be published in rapid succession and at other times we might be looking at a few hours between updates as I attempt to test my phone's 4G connection to upload and publish. Regardless, part of the fun will be the constraints of the day itself and you can be sure I'll be adding to the appendix of this book and my own blog to describe my experiences of trying to do this crazy thing today!

Please send me your feedback!

If you're 'lucky' enough to have bumped into this book on 22nd November 2014 then please fire over your feedback at any time and I'll incorporate what I can in the book as I write it today. Best way to get me your feedback is to pop me an email at russ@russmiles.com.

This book is also going to continue to evolve, so if you're coming to it after November 22nd, 2014 and so you've missed the initial madness of the "Book in a Day" effort then don't worry! My intention is to write this book within 24 hours; technically a day. So chances are I'm working on it now so it's not too late!

Please still send your feedback to me at russ@russmiles.com, I want to hear from you because my books always continue to change as I meet and discuss things with new people. The worse that can happen is that I make a new friend, so your feedback is always entire welcome!

It's so not "Thanks for all the fish"

This book is just the first step on a journey. Going forward Ill be updating the book I'm sure and so when you've bought it you'll, as is the "LeanPub" way, be getting those updates also.

But it's not just about the book. Today is not just about starting a journey for the reader who is interested in how to take on the aspects of the role in modern software development that we call an 'architect' but also the start of an actual, physical journey in 2015…

2015: The Year of the "Geek on a Harley" tour

Next year I'm planning, every month, to visit a part of the world either on my own motorbike. On each mini-journey, I'm planning to visit as many individuals and companies as possible to talk software process, architecture, design and coding. The full topic, no topics not allowed. The title of this escapade is:

"Geek on a Harley: Will Architect for Food; will Design for Wine"

[TBD] insert picture here.

Stealing this idea completely from something similar that Corey Haines completed a while back, this I feel is a fun way to get out there, meet lots of people in very different contexts, put a human face on software development through a video-blog, and at lowering the financial bar to consultancy from me. Plus I'll have fun, so it's win-win as far as I can see!

At the same time, I'm planning on doing these trips for charity as well. Any income I get from the businesses I visit I will take 20% of the profit and donate it to YoungMinds http://www.youngminds.org.uk

Doing Good: 100% of this book's (modest) price going to Young Minds

Similar to the donations I make from my book "Antifragile Software" and am planning to donate as part of my "Geek on a Harley" tour, 100% of this book will also be donated to YoungMinds as well.

Time to get started

I think that's enough precursor for now, it's time to get rolling. First stop is a brief explanation of what this book is all about: Objectives, Principles and Outcomes.

The Three Guides of an Architect: Objectives, Principles & Outcomes

> *Desirable developer skills:*
>
> 1 Ability to ignore new tools and technologies
>
> 2 Taste for simplicity
>
> 3 Good code deletion skills
>
> 4 Humility
>
> *By @codeinthehole on Twitter*
>
> ***Desirable architect skills:***
>
> 1 Ability to ignore new tools and technologies
>
> 2 Taste for simplicity
>
> 3 ***Be Scientific with Objectives, Principles & Outcomes***
>
> 4 Humility, please
>
> *By Russ*

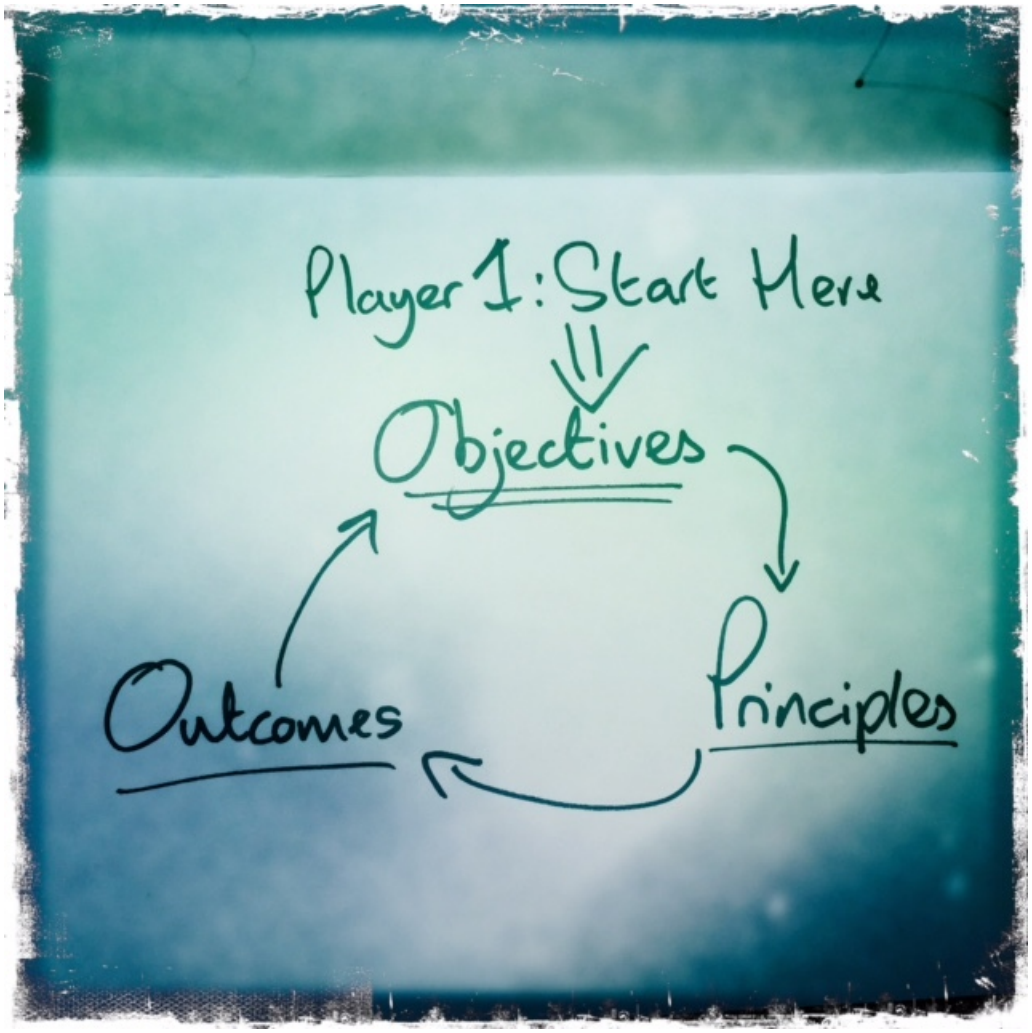
When you take on the responsibility of software architecture there is a lot of vague and blurry definitions of what you should be doing and thinking about. Is it about the "Big Decisions", or mandating the usage of a particular framework or library in the service of "Consistency through Governance"? Maybe.

The problem is that those tasks can be very dependent on your context, and the values of your organisation. What would be helpful is an active, simple process for approaching and executing architectural work that was not context-specific but allowed you to tailor things to your own context.

That process is what we gain by thinking in *Objectives*, *Principles* & *Outcomes*.

Why Objectives, Principles & Outcomes?

For me, this is the heart of what an 'architect' is thinking about, the triad of architectural thinking in software development:



The Software Architect's Triad

Great architects in software development that I've known tend to think in this triad and work an iterative process around the details in each of the waypoints.

Also I was particular inspired by a talk given by Adrian Cockroft of, at the time, Netflix. Adrian explained how he, as Cloud Architect, would go about capturing the overall organisations goals, how that could be communicated and enforced through principles, and how he could then look for and adjust for resulting outcomes.

It was such a neat, pragmatic, and even empirically scientific explanation of the process of architectural guidance that I couldn't help but follow Picasso's advice and 'steal' from Adrian to set the structure for this guidebook.

Dangers of "Vacuum Thinking"

There is a very real danger when writing any book on software architecture of ending up writing "from the rarified air of the unreal mountaintop". Many books, and speakers, fall foul of this one. The problem is that as a set of ideas is made more generically applicable, you lose the context entirely and in fact completely dilute the meaning and power of what you're trying to put across.

Software architecture is probably the easiest place in software to fall make this mistake.

"Context is King" is my own mantra as a software consultant and developer, and I'd certainly advice my readers to consider this too. The other mantra I have is "We don't know what we're doing", but dealing with that is reserved for a different book (Antifragile Software[1]). With all of these mantras, I speak to myself a lot … that's possibly not a good thing but there you go!

The point here is that all of the sections of this book should be broken out of the rarified, vacuum-like surrounds of these pages and carefully assessed as to their important to *your given specific context.* As always, there are no general silver bullets, so apply what you learn about here with care of your context. Be scientific…

Iterative, and Scientific

The job of the modern software architect, first and foremost, is to be scientific about things. To do that you need to have the triad on a in a continuous-replay, feedback loop:

TBD Diagram of three things.

As architects we are responsible for setting principles within the scope of desirable, hypothetical objectives, and then to measure the outcomes and relate those back to the objectives. Constantly, iteratively, refining as we go.

Taking the scientific process, this is how things map across:

TBD Diagram of how hypothesis, method and findings map to triad.

Providing the momentum in this empirical, practical process is the main part of the modern software architect's responsibility.

Communication, communication, communication

In order to turn principles into any measurable outcomes, we need to communicate them.

Anyone taking on the responsibility of thinking about the software architecture is therefore going to need to be able to get *out there* and communicate those principles that are forming the outcomes that are to be compared against the general and specific objectives of your organisation.

[1] TBDlink

Empower, and Lightly Enforce

Communication can only go so far… Sometimes you need something more active to attempt to make sure that those principles you've selected are being understood applied, otherwise misapplication of principles could completely kill the validity of any outcomes that you measure.

Luckily we have tests and *monkeys*. I am of course referring to the example of the Simian Army (TBD link) in the latter, not raiding the local zoo.

One of the key jobs of the architect responsibility is to come up with active participants that stress the resulting designs and implementations of the software, looking for deviations from the principles that are being applied.

These *stressors*, a force on a system that provides an opportunity for the system to gain from that force, at design, build and operational times can come in many flavours. I have had most success with is by applying development and build-time tests and operational stressors from my own army of simians to show that the system exhibits the ability to deal with these stressors that imply strongly that the principles are being followed.

Also, to be honest, it's fun to create software that breaks software when it doesn't follow a given principle, or maybe that speaks to my own sadistic tendencies … the serious point is that this turns architectural principles into something actively applied and enforced, which is of huge benefit when assessing the outcomes of our architectural experiments.

Let's get started

It's not time to begin your journey towards adopting the Empirical Architect responsibilities by looking first at some candidate general objectives.