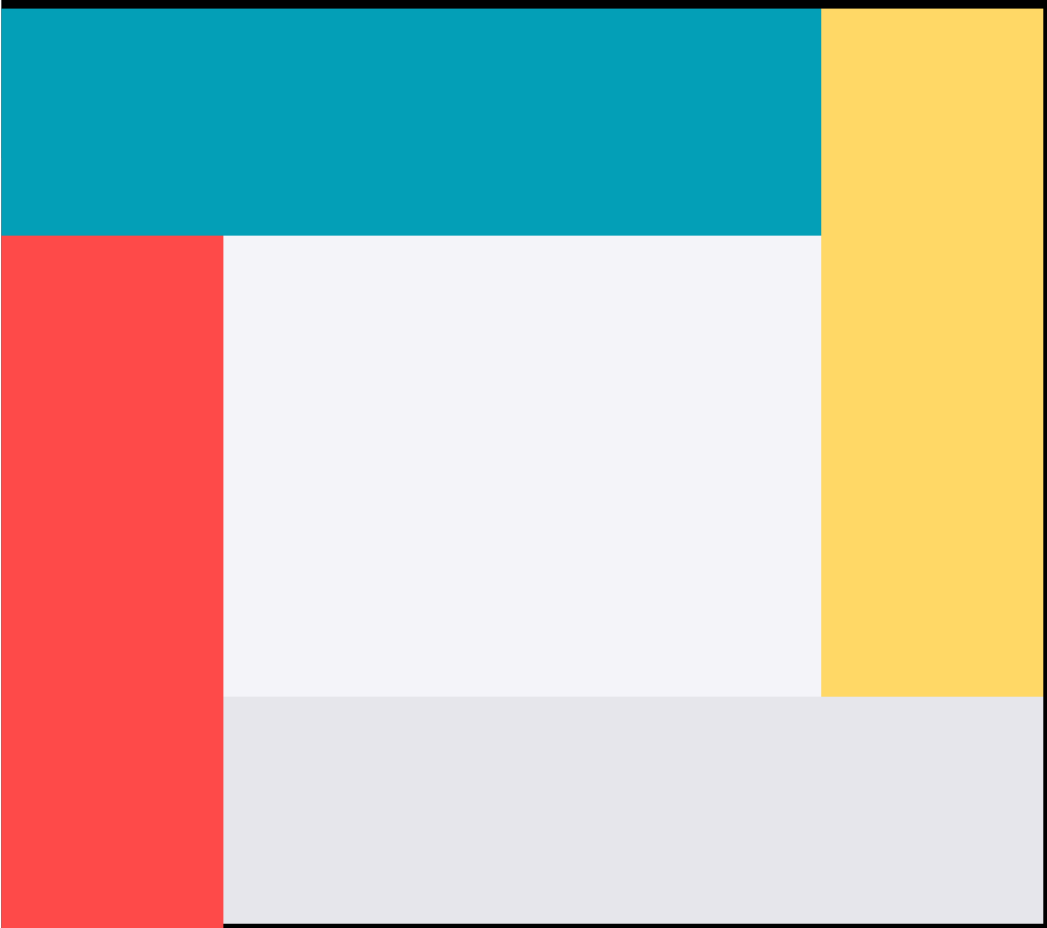
Building Software Platforms

A Guide to SaaS Transition with AWS



Pablo Bermejo

With foreword by Bill Ohnemus

Building Software Platforms

A guide to SaaS transition with AWS

Pablo Bermejo

This book is for sale at http://leanpub.com/software-platforms

This version was published on 2022-05-22

Tweet This Book!

Please help Pablo Bermejo by spreading the word about this book on Twitter!

The suggested hashtag for this book is #softwareplatforms.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#softwareplatforms

To Mom and Dad

Contents

Foreword

> en·tro·py | \ ˈen-trə-pē : a process of degradation or running down or a trend to disorder – chaos.

We who work in information technology could perhaps be forgiven for believing that software development is utter chaos – at times. When I was young, back in the sixties and seventies, I desperately wanted to know how everything worked. I would often take apart my toys and other possessions (like a green transistor radio I had). You might say I was acting as an agent of entropy. I was always fascinated with the parts and how they functioned together. During high school, small personal computers began to become available. Commodore came out with the Vic 20 and Commodore 64. You could buy kits to build a computer that probably was less functional but seemingly, far more intricate and interesting.

As we moved forward, computers became less of a curiosity for tinkerers and more of a tool to be used to generate some actual value - and not just for science and engineering. This happened because, slowly but surely, the intricacies and details of what went *into* the computer faded into the background while the things it enabled became far more important. This same evolution occurred with programming languages and software built from them. We went from machine code to assembler to first and second and third-generation languages. We went from data entry and batch processing to on-line-real-time processing, green screens to GUIs, and more. In an odd way, all this evolution led to a sort of reversal of entropy – *negentropy* – everything was and is tending to become *more* organized and the complexities are becoming less visible to the people most impacted.

Thus it is with enterprise software and the systems built from it. The sole purpose of enterprise software for most companies is to generate

business value at the lowest possible cost. That value isn't measured in complexity or intricacy or how amazing the algorithms are at its heart. The value is measured in how it helps people get things done. For a large enterprise, all this must be done quickly, robustly, safely, and securely. Getting all that is not easy. It is not simple, and it involves a lot of work to build and manage complexity in a way that hides it entirely while still delivering value.

For those of us who have been around a while in the information technology business change and evolution are nothing new. However, all that change wasn't random and did have (does have) a discernable trajectory. That trajectory is toward a functional commoditization of nearly everything done with a computer. In fact, that evolution is toward computers themselves simply fading into the background – at the consumer level and at the enterprise level.

Back when "the cloud" was first created, it was really just a bunch of servers running virtual machines with virtual networking on which you could place workloads without needing to buy that hardware. It was just a place to put the same software you would install on your local servers, but now you didn't need to buy the hardware or manage it. It was essentially virtualization on-demand over the nascent Internet. Over time, though, all that has changed. All the details about how and where to "build" that virtualized infrastructure has led to "infrastructure as code" or the software-defined data center. Beyond even that is the idea that you just define workloads and push them out to the cloud to run it without regard to infrastructure. That is, to build serverless solutions to enterprise demands.

In this book, you'll see building a *platform* based on Software-as-a-service and serverless concepts can relieve developers of the need to understand the complexities of running and managing software and allow them to focus on delivering greater value to their customers. You'll learn about how and why a platform enables, as Pablo puts it, "serviceful" development to deliver that value that is so important. You'll learn what a platform is and why you should want one. You'll see how the trajectory of the past has led us to this point and how,

while complexity still exists, it fades into the background (into the platform) and we, as developers, can focus on what is important to our customers – delivering valuable *services* that make their lives better.

Bill Ohnemus, MSIS
DXC Distinguished Engineer

Preface

What's in this book?

The idea of leveraging tools and frameworks to create better software is not new. For example, in traditional product-centric approaches, vendors pushed developers to build services that orbited around a smart integration bus with the vain hope of making the software more robust. Constrained by these proprietary technologies, this approach was far from ideal for developers to design and build their applications. Nowadays, with the emergence of standards-based computing utilities (serverless) and the novel software engineering practices that evolved with them (NoOps), we can look at these internal tools from a more developer-centric point of view.

Technology leaders worldwide keep rolling out incredibly complex product-centric infrastructure schemes to deal with technical debt rather than addressing it at its core. Instead, this book will discuss how to deploy a safety net for those brave software leaders that want to embrace the cloud as a *serviceful* platform, invest in developers, and make their organizations evolve with them. These are leaders who don't love failure, but they tolerate the process. That's why there is a bit of technology strategy in this book, too, based on Wardley maps[1].

You no longer need to buy a product to cover all the cross-cutting and foundational software needs. Building your stuff in-house is much better, especially if what you are creating is core to your business and sets you apart from the competitors in the marketplace. That is why internal software platforms are built in-house and invisible to the eyes of the final customers. These end-users receive the value that platforms help to unleash in the form of better functional software.

For all these reasons, this book is called *Building Software Platforms* and not *Installing Software Platforms* or *Configuring Software Platforms*.

Terminology

Even at the risk of being too dogmatic, I have tried to be very cautious with the terms I used to refer to some specific concepts. Being picky with words in a language that is not your mother tongue is the next level of pedantry!

Nonetheless, words do matter. A careful selection requires intentionality, especially when you want to discontinue meaningless terms attached to old ways of thinking. To that end, the following list contains a few examples of phrases and expressions that I carefully employed to express a few ideas captured in the book:

- I tried to use "managed cloud services" instead of "cloud-native." The reason is that, if we look closer, there is nothing new in those technologies that makes them innate to the cloud. On the contrary, they are based on ages-old standards!
- When I described the relationship between services and the internal software platform, I suggested that services run "with" the platform instead of "on" the platform to ward off layered architectures. The use of the former preposition helps to visualize that idea.
- I tried not to use "microservices" as a generic term to refer to any service running with the platform. Instead, I utilized the word "serviceful" when I referred to the characteristics of true service-oriented architectures. To me, there is a big difference, and the "micro" prefix denotes a distinct architectural style.
- I refrained from using old concepts like "System of Integration" or "Systems of Engagement." The word "system" in this context denotes some monolithic thinking and even layered architecture. As I unfold the characteristics of *serviceful* architectures across

> the first chapters, I change those terms to "integration services" and "engagement services," respectively.

As you go through the different chapters in this book, please give me feedback, and I'll correct it if you find that I have not been honest with my own rules.

Wardley maps

Technology executives use strategy tools to help their organizations move forward. Methods like SWOT analysis, flow charts, or linear roadmaps help them make decisions and answer questions. But are they going in the right direction? Movement is fine, but you also need position when it comes to designing a game plan.

I am not a strategist, but I understand the importance of using the right tool for the right job. I found that Wardley maps are an excellent technique for understanding your context, developing situational awareness, interpreting the competitive landscape, and visualizing the path forward. As with any other map, it helps you determine movement and position.

Wardley maps are two-dimension diagrams that combine a value chain axis (from less visible to more visible to the user) and an evolution axis (from unchartered territories to fully industrialized). Then, when you position the business elements you are modeling on the map, you unveil a visualization of where your business is at in a given moment in time, how the competition is making these elements evolve, and your paths forward.

This tool is used across the book to represent the role of internal software platforms in the organization and how this architectural style can help you transition to SaaS. The good thing about Wardley mapping is that all the official documentation has been released under the creative commons share-alike license by its creator, Simon Wardley. Consequently, and although I have tried to explain every map included in the book in detail, you will be able to find a lot

of materials out there that will help you understand this technique better.

AWS references

As we can learn from this Forbes article[2], the most significant factor in a startup's success is timing, according to Bill Gross (founder of Idealab). Gross studied over 200 companies to come to the following conclusions:

- As per this study, in 42 percent of the cases, the timing was determinant for success.
- Team and execution were critical factors in 32 percent of the companies he studied.
- The idea accounted for 28 percent of the cases.
- The business model meant approximately 24 percent.
- Funding represented only 14 percent.

Okay, but what does this have to do with AWS? The answer is that AWS masters the timing success factor exemplary, releasing their services precisely at the right time, meeting their customers where they are without being condescending about the technology. Not too soon, not too late. Just right on time.

Being conscious of timing and empathizing with the people around you has another name: *zeitgeist*. AWS understands and respects our *zeitgeist* like nobody else. For example, AWS Lambda has dominated the market since 2014 thanks to an incrementally richer set of features and a decreasing billing granularity. Due to this, this cloud giant is now the number one provider of serverless computing services.

But it wasn't AWS that pioneered the serverless market among the top cloud providers. Introduced in 2008, Google App Engine disrupted the infrastructure services market by offering runtimes as a service. The idea was compelling, the product was well delivered, the business

model is proven to work, it was funded by one of the world's largest companies, and the team was brilliant. Why didn't Google App Engine succeed? Because it was released too soon. It was too disruptive, revolutionary, and advanced for the developers back then, for whom Google App Engine didn't resolve any of their problems. Google didn't consider the *zeitgeist* when the product was released.

For this and other reasons, I like the catalog of AWS services. Therefore, this book contains an assortment of AWS innovations to help you design many of the ideas and principles portrayed across the different chapters. By using AWS's managed cloud services in the book examples, the goal is that you have more tools in your toolbox to implement an internal software platform strategy successfully.

Who should read this book?

The strategies, methodologies, references, and principles documented in this book are aimed to guide software companies to transition to a service-based economy. These are companies that build software, regardless of whether they license it or not.

Cloud infrastructure utilities priced per-use (IaaS and serverless) enable new economic models for software (SaaS), allowing end consumers to subscribe to on-demand products and pay only for their use.

How do internal software platforms help with that?

The answer is by empowering developers to build business services that maximize the value of their customers' subscriptions. Platforms do so by curating experiences and offering them self-service to the developers, who can now focus on creating domain-specific functionalities their customers will love to use. To that end, internal platforms introduce a new architectural style aimed at helping developers build software with efficiency and effectiveness. Be prepared because this theory is repeated *ad nauseam* during the following chapters!

This book is for anyone who loves building quality software, from developers to product managers. However, I have identified two clear target groups:

- First, this book is a must-read for software engineering and architecture leaders who work in software companies moving from a product-based economy to a service-based one and want to thrive on that change.
- Second, this is a refreshing manuscript for technology leaders in IT organizations who build in-house software systems sitting at the center of their businesses and are looking for new architectural styles to do it with efficacy.

There is still the question of what type of software is better suited for the tenets of the new platform-based economy. And there is a straight answer: the strategies and principles outlined in this book apply to any kind of software. However, I can observe a more compelling business case for sizeable enterprise software suites composed of heterogeneous legacy systems.

What will you learn?

By the end of this book, you will take the following insights away with you:

- Understand the challenges of building modern software products and primarily on-demand software in a post-pandemic world.
- Unfold the secrets of internal software platforms and why adopting such an architectural style can help software organizations ship better software and gain a competitive advantage.
- Refresh the architectural and technological principles for building, testing, and releasing internal software platforms and business services running with them.

- Discover how AWS cloud managed services, focusing on serverless, can help product engineering teams build secure, resilient, and scalable internal software platforms.
- Articulate the properties, advantages, and benefits of internal and on-demand software platforms so that you can build a compelling business case for your organization.

Acknowledgments

I want to thank the most incredible group of women and men who helped me write this book. I owe them an enormous debt.

I am starting with **Bill Ohnemus**, a fellow architect and tech honoree at DXC, who was kind enough to do a technical review and write a lovely foreword to this manuscript. Also, I have to give special thanks to **Martin Bartlett**, Chief Engineer at DXC, for his technical thought leadership. He is responsible for pushing me to break with those old approaches to software development. And, of course, to my boss **Brian Bacsu** for being a role model and leading the way to change how our large company builds, tests, and releases quality software.

I want to give a special appreciation shout-out to my *Architecture and Engineering* teammates, who happen to be friends too and with whom I spent countless fun hours talking about software philosophy. Many of the opinions and suggestions you will find in this book are the results of these discussions, so I want to give them the credits they deserve. Thanks, **Enrique Riesgo**, **Pablo Castaño**, **Jesús Suárez**, **Aitor Echevarría**, and **Rubén López**.

Also, to my friends and connections who helped me in many ways: showing support, sharing tips, or giving me early review feedback. Thanks, **Diego Alonso**, **Marina Cortés**, **Joaquín Peña**, **José Juan Mora** and **Elena Cid**.

And finally, thanks to my family. Without their support, this book

would have been another idea lost in an ocean of miscellaneous unfinished projects.

Here we go!

This book is all about recognizing the need for change in software development, looking at it in the eye, and embracing it fearlessly. There are stories in this book that will help you thrive on this change, especially around reinventing yourself, appreciating the mundane, designing for the people, holding fast, and staying true. You don't need to follow them verbatim; just think about them as a north star.

What has driven and inspired me to write *Building Software Platforms*? Well, other books did. I have received incredibly good vibes and brilliant insights from other technology authors after reading their texts, which I strongly recommend:

- *Team Topologies: Organizing Business and Technology Teams for Fast Flow*, by Matthew Skelton and Manuel Pais.
- *Good Services*, by Lou Downe.
- *Extreme Programming Explained (second edition)*, by Kent Beck.
- *Ask your developer*, by Jeff Lawson.

If this book leaves you with more questions than answers, I would call it *mission accomplished*. I firmly believe that good software can make the world be a better place, and we can do it healthily, without burning out our human assets or causing any harm to our environment. I want to implant a seed of curiosity in our community to build better software. I want every software engineering and architecture leader to be thoughtful about their decisions instead of choosing technology for the sake of it.

When we are building software, we are standing on the shoulders of giants. And this book is a physical manifestation of this idea.

Enjoy!

Pablo Bermejo
DXC Distinguished Technologist

PART I: STRATEGY

Software strategy includes understanding the playing field around customers, partners, competitors, and employees.

First is crucial to understand your customer's buying patterns. When and why do they buy? When and why do they build themselves? The answer to these questions will help you design your software products accordingly to those patterns. For example, are you designing your commercial off-the-shelf (COTS) to customize your customer needs fully, or do you prefer them to use your default industry expertise? Are you aggressively factoring in your vision in the software design, or do you choose to meet customers where they are and guide them toward your vision progressively?

Figuring these things out is essential.

You must develop situational awareness and see how to move from a product-based market to a services-based market, or at least how to challenge those playing in the latter. That is where software is leading, and if you don't do it, some of your competitors will. During this journey, you don't have to build everything yourself. Leaning on 3rd party products and commodities from partners will be pivotal in your strategy and allow you to focus on what is important to differentiate yourself from the competition.

Ensure you have the proper methodologies, talent, and technologies to build this vision. Developers are first-class citizens in modern software organizations and must be treated as creative problem solvers. Empowering autonomy, mastery, and purpose in these development teams will set your organization for success.

Chapter 1: Transitioning to SaaS

Reinventing yourself

Funnily enough, the first story in a book about software like this one is not about technology but gastronomy. Concretely speaking, it is about a restaurant, not just any but the best restaurant of all time known as *El Bulli*. Located in Roses (a charm-filled town in the Spanish *costa brava),* the place had been awarded already with one Michelin star a couple of years before chef Ferran Adrià joined the staff in 1984. With Ferran's and his brother Albert's ideas, the restaurant's popularity rocketed thanks to the vast number of innovations they brought into the *cuisine,* obtaining two additional Michelin stars in 1990 and 1997.

With Adrià already at the front of the team, during the late 90s, El Bulli helped put the new Spanish gastronomy on the map and changed the status quo of the gastronomic industry with new technologies such as the kitchen siphon and new techniques such as deconstructions. All in all, Adrià's *Molecular Gastronomy* concept was disruptive enough to put his restaurant in a very prominent position in an already competitive list dominated by French chefs with their *Nouvelle Cuisine*.

And this is when the story starts to get interesting.

In 1998, and with El Bulli already enjoying a leading position among the best places in the world, Adrià took a decision that surprised the entire gastronomic industry (and the whole world, as a matter of fact). As he expressed in many interviews[3], he felt exasperated about going from congress to congress and watching other chefs

sharing the results of their ideas while keeping the secrets, and most importantly, the recipes to themselves. The intensity in how those colleagues guarded their creations triggered Adrià's creativity.

As a professional chef, Adrià was committed to improving continuously and changing the world's perception of this type of cuisine as a banal activity carried out by many pompous elitists who were only interested in recognition. He held an alternative vision that went beyond making El Bulli a mere lucrative restaurant and another mission than just running a business. He wanted to make a positive impact on people's lives through the experience of having food and, as a consequence, leave a legacy. At one of those conferences, in San Sebastián, he proclaimed one of his most famous statements that kickstarted a whole movement within the industry:

> *"The time of keeping recipes in the drawer is over."*

As he said that, he started to share all his famous recipes and techniques publicly, congress after congress. Yes, precisely this is what you are thinking. To get better, Adrià open-sourced El Bulli's recipes as he understood that the best way of protecting an idea is sharing it immediately. The restaurant was good, but they wanted to become the best. They quickly recognized that the only way of getting there was by accelerating innovation and sharing knowledge with the community to compete in identical product and practice conditions. By releasing their secrets in the open, they transformed their technologies and techniques into a quasi industry standard adopted by many other restaurants across the planet that were pursuing the same kind of success.

What happened was that the gastronomic industry was starting to compete in a commoditized market. Molecular gastronomy technologies and techniques at their genesis in the early 90s became industrialized by the early 2000s and accessible by all kinds of restaurants. Molecular gastronomy became so popular and handy that all good restaurants included quite a few dishes using this technique in their

primary menu. If we look at our kitchens now in the 2020s, we will likely find some sophisticated pieces of equipment such as siphons in a shameless attempt to impress our friends with a taco foam next time they come and visit us for dinner.

What happened after this master move won't surprise you.

Adrià and his team outperformed themselves to the point that *Restaurant* magazine awarded El Bulli as the best restaurant in the world from a global list of 50 places in 2002, 2006, 2007, 2008, and 2009, setting a record high of five times. Again, they were competing in a highly commoditized market where most restaurants worked with the same products and techniques based on Molecular Gastronomy. Not surprisingly, El Bulli's chefs were again leading in a very competitive market and differentiating themselves from their competitors by using their techniques to produce exclusive and unique dishes (Figure 1). This approach helped Adrià set a new best practice in the gastronomy industry.

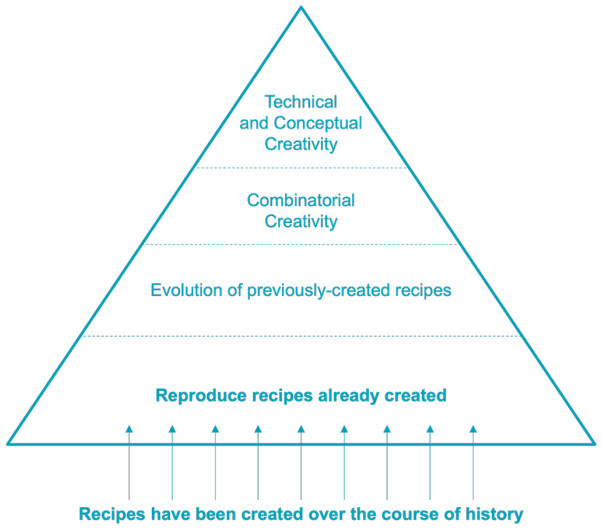


Figure 1: Adrià's creativity pyramid

However, little did he know that they were about to die of success except for one person: the visionary chef.

Adrià started to witness the first signals of burnout in his increasingly mature team. More importantly, he was very frustrated that he had to reject customer after customer because the restaurant's waiting list extended for months-worth (and even years-worth) of reservations. The team even had to cancel lunchtime service to focus on one service per day at dinner time, which raised many questions and concerns to the management team, severely impacting El Bulli's revenue. Adrià estimated that the team couldn't physically and mentally make it beyond 2012, so in an unexpected anticipation maneuver, he closed the restaurant in July 2011.

Although El Bulli as a restaurant has closed forever, its core concept, together with Adrià's vision, didn't stop there. Right after he announced that the most iconic restaurant in modern history was closing at its peak, the Spanish chef revealed the details about how he wanted to build his legacy. Therefore, and in the spirit of the experimentation and innovation that led El Bulli to become the number one restaurant, Adrià's next project consisted of spinning up a foundation to foster gastronomy innovation, learning, and sharing experiences with the community. The new entity is known as *El Bulli Foundation,* and it is aimed at creating and delivering new culinary experiences for its guests.

When this book was written, all we knew was that this master chef wanted to persevere in realizing his vision of leaving a legacy to the world and disrupting the gastronomy industry once again. Indeed, Adrià and his team are closing the cycle of evolution, making the pendulum swing back to genesis to reinvent themselves, formulate uncertain things again, work with poorly understood concepts, and start to compete in an undefined market. They don't love failure at this new stage, but they tolerate the process to create innovations that will help them differentiate with unique gastronomy products in the future. This idea is presented in the following Wardley map (Figure 2)

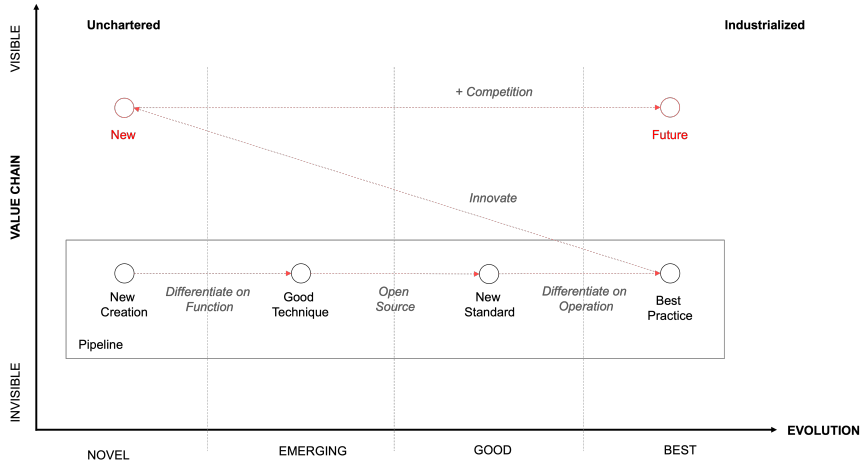


Figure 2: Representation of the evolution of El Bulli's practices using Wardley maps

There is a hidden lesson for the software industry despite this story's evident survivorship bias and the manifestation of privilege that entails closing a wealthy business at its peak.

We are moving toward a world where technology is so abundant that companies need to quickly evolve from product differentiation to service differentiation. To make the jump, many of these companies also need to understand that the only way to make a product better is to transform it into an industry standard. In technology, that could happen by open-sourcing it.

After that, companies compete in a services market that fosters innovation to gain new competitive advantages. It took El Bulli, a company in the gastronomy business, more than three decades to preserve a leading position by going through this magical cycle, something that technology companies are doing in a decade or less. This is how fast technology is going these days, so be prepared for a bumpy ride.

Software as a Service (SaaS)

> *"Any X as a Service simply refers to the commoditization of the computing stack from a product-based economy to a service one. End of story." - Simon Wardley*

How is it that every incumbent business is a software company nowadays?

We have heard the prophecy that every company is a software company[4] for the last ten years approximately. Still, it is only during the previous five that we witnessed more clear evidence of its realization. To name just a few examples, we don't put our savings in banks anymore; we use FinTechs. We don't protect our properties by contracting insurance carriers; we subscribe to InsurTechs. Our people don't learn at traditional universities; they join online sessions through EdTechs. We don't drive cars anymore; we drive *software on wheels*.

The software buyer's perspective

As Jeff Lawson, software developer and CEO of Twilio, tells us in his book *Ask Your Developer*[5], "*the category of problems we can solve with software is exploding*." It is as if most of humanity's problems are resolvable by software these days, or at least with a "*software development mindset*," as Lawson likes to call it.

This mentality is all about embracing change. With it, business models will benefit from the continuous improvement spirit of modern software development, which is based on rapid increments and fast feedback loops (Figure 3). This agile paradigm is easier to follow when technology is at the center of the business, not just a means to support it. This approach is nothing but applied science, a clear manifestation of the hypothesis-driven scientific method, backed by Agile software development doctrines such as Extreme Programming.

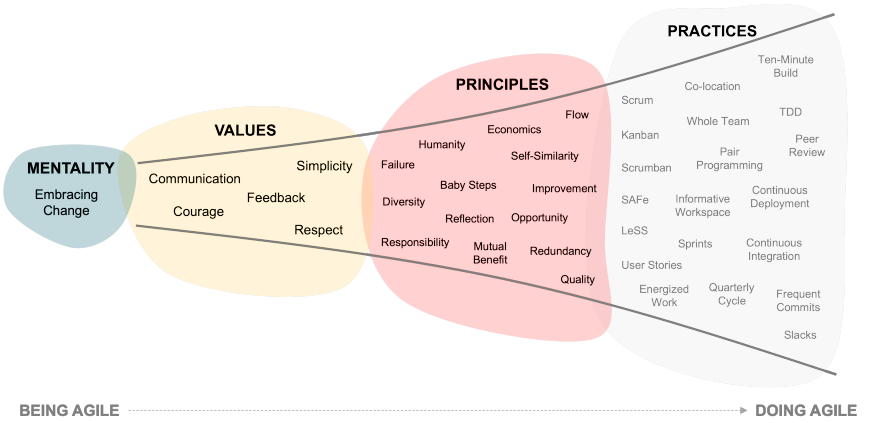


Figure 3. The tenets of Extreme Programming

Without having to wait for perfection to get going, this lean mentality is changing how companies from all industries look at *Build vs. Buy* patterns. These companies are more willing to build the core competencies that set them apart from the competition, only acquiring non-core to their business and not critical to their capital flow. Furthermore, they don't buy software for the set of functionalities initially included in it but for the features that the software will eventually gain through totally transparent update processes.

Every sensible company consuming off-the-shelf software adapts their non-core business process to the product and not the other way around. From a buyer's perspective, this is an exciting way of looking at software not as an asset they can capitalize on to support their business but as a means to operate it. And that's where software as a Service (SaaS) comes into play.

What systems and processes are core and non-core to these companies, then?

It depends on the industry but as a general rule, whatever makes them obtain better business outcomes. For example, for an insurance company, better results mean selling more policies, whereas for a car manufacturer, it translates into selling more cars. Complementary, for a SaaS company, a core system is whatever helps obtain more

customer subscriptions. Everything else is the cost of doing business.

Remember: your customers don't buy software anymore; they now subscribe to it.

The software provider's perspective

As a software organization, you are in a race against time, as your competitors and new entrants in the technology market are raising the innovation bar quickly. To stay competitive and maintain an edge, software organizations must reinvent their way of working and embark on a journey to find solutions that empower them with the right capabilities to build modern software.

And deliver it as a service. And do it continuously.

Today's software engineering and architecture leaders are privileged witnesses of the emergence of new economic models at a pace that nobody experienced before. The industrialization of development tools and the commoditization of computing are catalysts for the shift in the expectations of software consumers, who are discovering a new world full of possibilities.

At a high level, the following sections summarize the top five challenges that you, as a software leader, and your organization will have to face in the transition to on-demand software to stay competitive.

Managed services

Transitioning to on-demand software (or SaaS) will help you in moving your customers away from a capital expenditure (CAPEX) toward operational expenditure (OPEX) or outcome-based pricing models. It means your customer's IT department will depend less on executive approvals for products like yours, paying only for what they need and adding capabilities as required.

You need to find a solution for your customers aligned with the principles of managed services. Hence, the first place to look at is the consumption of managed cloud services for all your infrastructure needs. As an on-demand software provider, you will own the cost of the solution in its totality, so you want to optimize your spending and focus on your core business. As a result, your software organization must remain aware of the evolution of the surrounding technology ecosystem and be open to externalizing any non-core business capability to third parties who can offer those as managed services, including other internal groups.

It's a *Build and Buy* strategy.

Cost control

In line with the principle above, it is crucial to be aware of the cost implications of your technology decisions with a special focus on understanding the pricing models of the cloud provider of choice (with niche companies dedicated only to cloud bill optimizations). Cost implications may vary not only depending on the type of managed service you choose (i.e., cloud DB clusters may be expensive while functions as a service are highly cheap across different cloud providers) but also on your architectural choices (i.e., a purist services-oriented approach would require one DB cluster per service, incurring in higher costs, while taking a CQRS approach may result in lesser costs)

Live upgrades

As SaaS providers, organizations pursue an evergreen, always-on status for their services even after patches, updates, or maintenance work. This challenge is enormous for some industries, such as insurance or banking, where core systems still need to go down overnight to do batch processing or file transfers. Technology teams must research and build services, patterns, and tools for developers to create business functionalities that cater to live deployments without impacting service availability or business continuity.

Low operations

No technology organization wants to see its system engineers wasting time provisioning virtual machines on the cloud, handling complex container deployment schemes, or applying security patches. Instead, they want them to move up the value chain, become the voice of the customer, participate in feature contributions to the product roadmap, and help the company's SaaS strategy move forward.

To do that, you must choose an aggressive (yet very conscious and reasoned) approach to leveraging managed cloud services as much as possible. It would be more beneficial if you don't force the development teams to use them in every scenario just for the pleasure of using the latest, most extraordinary technology. It means that instead of finding the technical justification to select a managed service by default, it would be best to identify the business justification not to use it. If there aren't any, then yes, go ahead with your cloud service.

User experience

As your software organization transitions from a core systems vendor to a SaaS provider, you will also be shifting focus and investment toward a reimagined integrated suite of applications. Many studies and analyst reports show that a compelling User Experience is increasingly becoming a critical decision factor for businesses to subscribe to a particular SaaS solution over another.

For example, in the article "Transforming life insurance with design thinking,"[6] McKinsey describes how the insurance industry must adopt Design Thinking to foster engagement amongst the youngest population, such as millennials and Gen Y. As stated in this article, users now have greater expectations *"for highly interactive digital experiences as well as fast and even instant delivery."*

To that end, any engagement services built and delivered to your consumers should be designed with a set of professional design principles that embed your brand identity and culture. This is how

you provide them with a family of solutions that feature consistent and compelling user experiences.

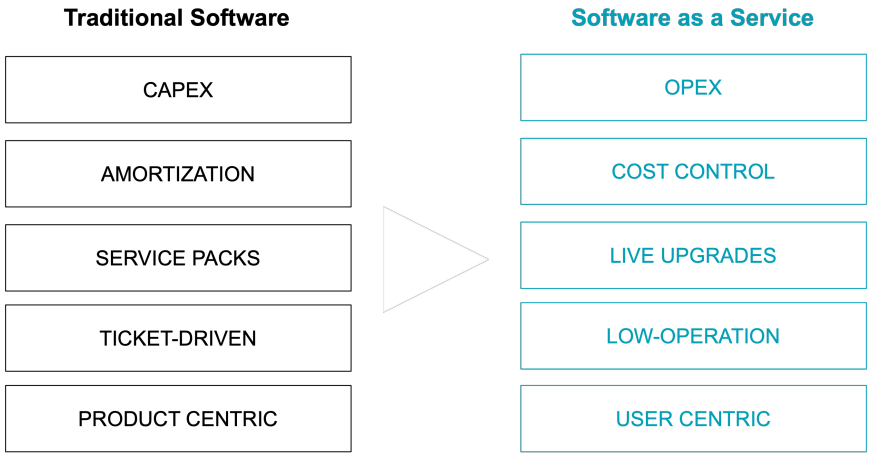


Figure 4: The characteristics of traditional software compared to on-demand software

Building and buying software

The heterogeneous space of problems that companies from all sectors face these days as they progress in their digital transformation journeys may result in mutually exclusive requirements and expectations.

Being pulled in many directions is not new in the development of software products, where you are in charge of designing your abstractions and baking in your domain expertise. However, delivering them as a service has relevant and unprecedented implications, primarily related to the challenges outlined before.

What do customers want?

To start with a straightforward example, let's imagine you have a customer seeking outcome-based software subscriptions as they want to optimize their spending, so they like SaaS. At the same time, you have another customer who needs to strictly conform all the user

interfaces to their corporate branding guidelines, so they seek high fidelity. Finally, a third one needs to configure their business rules as per their core processes, so they want low-code tools for building new apps.

You may even have a type of customer who wants everything!

Well, as anticipated earlier, these are mutually exclusive requirements, and all these use cases would have been a challenge even for *on-prem* software. This challenge comes with additional ramifications for SaaS engagements, mainly because your customers do not own the software asset. You do. It means that these companies have higher expectations regarding configurability, speed, and self-service. Hence, they are unwilling to hire extra professional services on top of their subscription and consumption fees.

The table below summarizes this dilemma and depicts the challenges that SaaS providers need to address:

	You Build	**Customer Builds**
Low Fidelity / Low Cost	Pay per use	No-code
High Fidelity / High Cost	Professional services	Custom development

In short:

- Customers want no-code, but they want high brand fidelity
- Customers want high brand fidelity, but they want low cost
- Customers want low cost, but they want high customization
- Customers want high customization, but they want to pay per use

Therefore, it is vital to consider the best business model and strategy for on-demand software and whether your customers build themselves or buy from you. Or, most probably, a combination of both[7].

This idea is depicted below (Figure 5) and explained further in the subsequent sections.

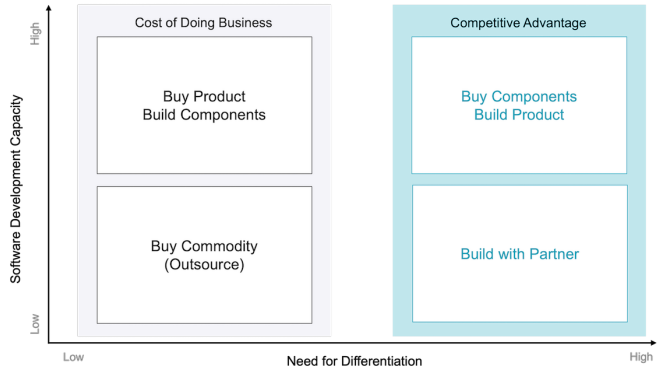


Figure 5. Build and Buy strategies based on the need for differentiation and development capacity. (Credits: Celent Insights)

Customers buying your software

If several companies subscribe to your SaaS solution, what's the differentiation and competitive advantage between them, especially in well-established and highly regulated industries like real estate, insurance, or banking? One could think that good on-demand software should include sophisticated visual solutions to cater to extreme levels of customization so customers can factor in whatever makes them different, whether it is User Experience (UX) or business processes.

However, be careful with that decision.

It is important to remember that configuration tools for on-demand software come with limitations and tradeoffs since they offer speed of development at the expense of flexibility. To keep this scale tipping toward velocity, low-code tools can't make every single software parameter configurable. Still, if you cross that line, the resulting system may become too complex for a business user, so they will need to hire professional services to extend, adapt, or customize the software to their needs.

In any case, there is a sweet spot for on-demand software products. Especially in mature (and regulated) industries, this type of software is more suitable for essential yet undifferentiated workloads such as process-heavy back-office applications that don't need customizations because they are not strategic. In other words, things that are just a cost of doing business.

This idea is totally in line with what Jez Humble (co-author of software development best-sellers and SRE at Google) expressed on many occasions[8] via Twitter:

> *"COTS is for business processes that aren't strategic to your organization. So you should modify your business process to fit what the software does out of the box."*

This is important because, as stated by Chris Swan (engineer and former Fellow at DXC Technology) in his article "Industry best practice as expressed in software,"[9] if you allow your software to be highly customized, you are opening the door for your customers to move backward in terms of evolution (Figure 6). Consequently, if you keep improving your software to become a leading service, your customers won't be able to tap into all the new benefits due to the impossibility of upgrading.

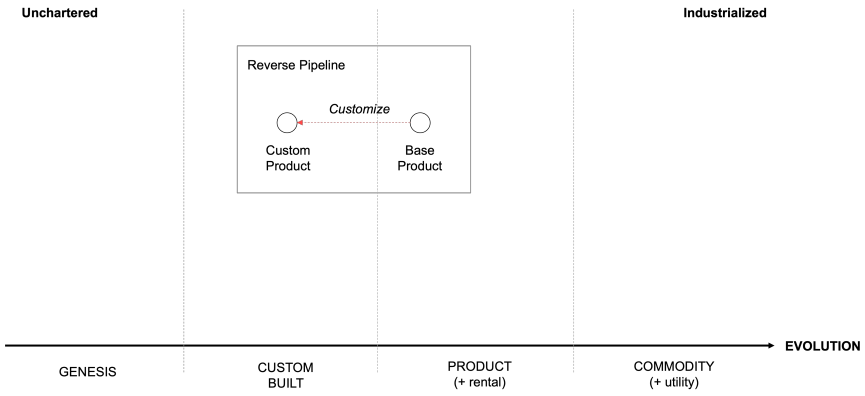


Figure 6: A visual representation of backward evolution due to excessive customization

Following this line of thought, it is entirely acceptable to make your SaaS configurable to meet your customers' needs but within reason. It is essential to make this configurability lightweight, so the software does not become so unrecognizable that it turns into a ball of mud. It has to be large enough to let your customers adapt some non-critical parameters and build small integrations independently. But they must be small enough so they can still do it self-service and at speed.

Suppose your customers can't use these configuration features to adapt your software to their needs. Then, as Jez Humble mentions, it is preferable to alter a business process to fit a software product than to have edgy low-code customization capabilities. Technical debt can pile up in many forms, and complex configurations stored in a database are among them.

In summary, you need to pay attention so that your customers don't trade efficiency for effectiveness when using configuration tools.

Customers building their software

We know that leading companies compete on service rather than on product, which applies to all sorts of firms from all industries. As we have learned from the McKinsey report mentioned earlier, user experience is a crucial differentiator element to compete in the services market.

It matters—a lot.

Your final customers are willing to spend money on building custom user experiences that give them a competitive advantage, so a configuration capability would not suffice. Instead, you may need to provide them with the necessary building blocks and components to build these applications at speed on top of your software and do it programmatically (e.g., a business-oriented UI component framework including API consumption).

Twilio Flex[10] is an excellent example of this. We can see how developers are directly in charge of the extensibility and customization of the system through code, which is the highest level of customization possible. Solutions like this exist because there is a point at which adding extra configurability is detrimental to the system. The main reason is that the overhead introduced by excessive configurations exceeds the benefits that otherwise could have been obtained more quickly through coding.

Most companies are better at dealing with technical debt than avoiding it because they don't know how to move forward in terms of evolution.

Evolving toward a service economy

Do companies know what makes them different? Do they know what is core for them and what is not? Have they developed a situational awareness of the market they are competing in? Does context drive them? It is hard to tell, but, as usual, the answers depend on to whom you are talking.

That is why having a good Build and Buy strategy is key to success, and that is why connecting with the right person is so important. Although this book does not have all the answers and will probably leave you with more questions, it aims to give you a decision framework so when you are involved in that type of conversation (either with your customers or your stakeholders), you can ask the right ones.

This is of utmost importance for software engineering and architecture leaders like you. As Simon Wardley mentions in his article "Here comes the farmer ..."[11], which dates back to early 2008:

> *"A key part of this transition is that we move to a service economy that competes on service rather than on product."*

It would be best to have situational awareness and understand the market context in which you compete. If your product is unique and you are pioneering in a new field, you compete on the product. Eventually, more and more companies will pop up with functions like yours that only flatten the competitive terrain, so you will need to start looking for another differentiator. In other words, when software products compete in a commoditized market, the best service prevails.

Thirteen years after Wardley's blog post about the topic, McKinsey also emphasizes the importance of moving to a service economy combined with other technology enablers, as described in the article "SaaS, open source, and serverless. A winning combination to build and scale new businesses"[12]:

> *"Using SaaS and serverless to free IT from infrastructure management greatly reduces the complexity of app development and deployment. This, in turn, allows tech teams to organize around products—for example, cards or loans—which brings code closer to the business."*

Transitioning to SaaS is vital: if you don't do it, one of your competitors will. But you have to be careful. Software companies keep mistaking *transitioning to the cloud* with *becoming the cloud.*

- The former just means taking your existing on-prem, highly-customizable software and moving it to a fancy data center (i.e., a cloud platform) to integrate better with your customer's cloud platform of choice.
- The latter, instead, implies rethinking your software products entirely to offer them as a service to users. Evolve, industrialize, and provide your software resources as commodities with sprinkles of self-service and evergreen updates. That's the cloud. That's what a SaaS company is.

You don't need to own a data center and expose your infrastructure resources through APIs to be a cloud company.

The industrialization of computing services, along with the co-evolution of other technology practices and the impact on team dynamics, paves the way for internal software platforms as an enabler for on-demand software. The rest of this book dives deeper into all these concepts to help software engineering and architecture leaders like you thrive on this exciting transition.

Chapter 2: Internal Software Platforms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Appreciating the mundane

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Serverless: the modern cloud

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Managing complexity

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The economic forces of serverless

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Building the future

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS serverless catalog

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Empowering developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Helping developers with the right tools

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Building tools for developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Internal software platforms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

What is an internal software platform?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The mission of an internal software platform

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Experiences captured in internal software platforms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Serverless software platforms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Serviceful software platforms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform components

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Funneling innovation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The business overlay

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Build and Buy patterns for the business services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platforms as internal products

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Going off-platform

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The benefits of internal software platforms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The internal software platform business case

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The world's most mission-critical platform

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Conclusion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Chapter 3: Platform Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Designing for the people

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Core Infrastructure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The Platform Architectural Style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Service-Oriented Architectures (SOA)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Microservices Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

A developer-centric approach

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Concerns with the platform architectural style

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Lock-In

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Cost-driven micro-optimizations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Service meshes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS reference architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Isolating environments with VPCs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Lifecycle Automation Management

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Road to continuous delivery

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Automation in a platform-based architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

DevOps Principles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Provisioning serverless environments

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Lifecycle automation with AWS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Evolving the platform's lifecycle automation management with AWS services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Core Enablement Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Just enough encapsulations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Simplifying developer experiences

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS Examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Event Management Core Service

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Identity Management Core Service

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform Utility Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform Design System

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Ecosystems and application marketplaces

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Client frameworks

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The pendulum swings back and forth

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Choosing the right technology

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

UI architectures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The frontend developer role

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Configuration vs. Customization

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Application development tooling

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Configure an application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Write an application

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Danger zone

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform Management Consoles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Promoting configuration data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform Microfrontends

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Build-time microfrontends

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Hyperlinked microfrontends

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Embedded microfrontends

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Chapter 4: Platform Teams

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Holding fast, staying true

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

What is a platform team?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

It is all about the developers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Structure of an internal software platform team

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Specialized skills

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

How do platform teams fit within the organization?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Engineering Managers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The pillars of high-performance teams: autonomy, mastery, and purpose

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Product Managers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Software Architects

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The Software Architect Elevator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Software Engineers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Managed Services Thinking

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

What value do these teams create?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Build and Buy patterns for the software platform

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The Operational Experience: toward NoOps

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The Developer Experience: first-class citizens

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The User Experience: Programmable Engagement Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

What does that mean to you as a software house?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

How can you do that?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Why is this important?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Interacting with platform teams

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Mastering industrialization

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Chapter 5: Platform Adoption

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Leapfrogging

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The Digital Platform economy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Legacy replacement

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

External Business Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Managed Business Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform-Native Business Services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

A multi-competence strategy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The innovator's dilemma

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Outsourcing practices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Lean practices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Agile practices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Building platform-based business services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Designing for user impact

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Factoring in your organization's domain expertise

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Building Minimum Viable Products

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Whole teams

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS Lambda Reference Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

API-driven

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Event-driven

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Data-driven

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform buy-in

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform adoption priorities

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Platform adoption attitudes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Skeptical users: The *farmers*

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Neutral users: The *campers*

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Ambassadors: The *pros*

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Internal software platforms and the Theory of Constraints

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

PART II: PRINCIPLES

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Chapter 6: Technical Architecture Principles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Favor *serviceful* platforms over monoliths

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Backward compatibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Favor iterations over big up-front designs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Solving real problems

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Handholding with AWS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Evolving business service development with AWS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Favor asynchronous integrations over synchronous ones

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS internal asynchronous architecture

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Favor elimination over re-engineering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Problem restatement

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Holding on to the doubt

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Favor re-engineering over multiplying

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Automate everything

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Remove technical bebt

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Improve performance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Favor duplicity over hasty abstractions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Let it spring and then harvest

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Identifying platform core services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

References over templates

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Chapter 7: Technology Principles

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Make your platform reachable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Define everything as code

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Debugging in a serverless environment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Use the cloud platform as a programmable system

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Embrace the cloud fully

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Serverless is a non-zero-cost tradeoff

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Working with AWS services

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Make your platform transparent

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Secure your platform access and data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

AWS security model

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Authentication

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Authorization

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Data Security

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Observability and visibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Use open standards

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Chapter 8: Serverless-first Software Engineering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Building portable software

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

The Provider Pattern

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Hexagonal Architectures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Conclusion and sample implementation using AWS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

A serverless-first approach with AWS

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/software-platforms.

Notes

Preface

1 https://medium.com/wardleymaps

2 https://www.forbes.com/sites/bernhardschroeder/2019/09/23/what-is-the-most-important-element-of-a-successful-startup-hint-its-not-the-idea-team-business-model-or-funding-dollars/

3 https://www.laverdad.es/sociedad/despegue-gastronomico-20180705004051-ntvo.html

4 https://www.forbes.com/sites/techonomy/2011/11/30/now-every-company-is-a-software-company/

5 https://www.askyourdeveloper.com/

6 https://www.mckinsey.com/industries/financial-services/our-insights/transforming-life-insurance-with-design-thinking

7 https://www.celent.com/insights/375184678

8 https://twitter.com/jezhumble/status/1422924763647778821

9 https://blog.thestateofme.com/2020/03/23/industry-best-practice-as-expressed-in-software/

10 https://www.twilio.com/docs/flex

11 https://blog.gardeviance.org/2008/04/here-comes-farmer.html

12 https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/saas-open-source-and-serverless-a-winning-combination-to-build-and-scale-new-businesses