The InfoQ eMag / Issue #80 / December 2019

InfoQ

VOLUME 1

The InfoQ Software Trends Report

An opinionated guide to the professional software landscape

DevOps and Cloud | **Culture & Methods** | **Architecture and Design** | **JavaScript and Web Development**

The InfoQ Software Trends Report - Volume I

IN THIS ISSUE

PRODUCTION EDITOR Ana Ciobotaru / COPY EDITORS Lawrence Nyveen & Susan Conant / DESIGN Dragos Balasoiu
GENERAL FEEDBACK feedback@infoq.com / ADVERTISING sales@infoq.com / EDITORIAL editors@infoq.com

CONTRIBUTORS

Abel Avram

has been involved in many InfoQ editorial activities since 2008, enjoying writing news reports on Mobile, HTML, .NET, Cloud Computing, EA and other topics. He is co-author of Domain-Driven Design Quickly.

Arthur Casals

is a Computer Science researcher working in the area of Artificial Intelligence / Multi-agent Systems. He has been developing software for 20+ years, in different markets/ industries. Arthur has also assumed different roles in the past: startup founder, CTO, tech manager, software engineer. He holds a B.Sc. degree in Computer Engineering and an MBA degree.

Ben Evans

is a co-founder of jClarity. He is an organizer for the LJC (London's JUG) and a member of the JCP Executive Committee, helping define standards for the Java ecosystem. Ben is a Java Champion; 3-time JavaOne Rockstar Speaker; author of "The Well-Grounded Java Developer", the new edition of "Java in a Nutshell" and "Optimizing Java"

Ben Linders

is an Independent Consultant in Agile, Lean, Quality and Continuous Improvement, based in The Netherlands. Author of Getting Value out of Agile Retrospectives, Waardevolle Agile Retrospectives, What Drives Quality, The Agile Self-assessment Game, and Continuous Improvement. Creator of many Agile Coaching Tools, for example, the Agile Self-assessment Game. As an adviser, coach and trainer he helps organizations by deploying effective software development and management practices.

Charles Humble

took over as editor-in-chief at InfoQ.com in March 2014, guiding our content creation including news, articles, books, video presentations and interviews. Prior to taking on the full-time role at InfoQ, Charles led our Java coverage, and was CTO for PRPi Consulting, a renumeration research firm that was acquired by PwC in July 2012.

Chris Swan

is Fellow, VP, CTO for the Global Delivery at DXC.technology, where he leads the shift towards design for operations across the offerings families, and the use of data to drive optimisation of customer transformation and service fulfilment.

Craig Smith

has been a software developer for over 15 years, specialising in a large number of technologies in that time. He has been an Agile practitioner for over 10 years, is a Certified Scrum Master and Certified ICAgile Professional and a member of both the Scrum Alliance and Agile Alliance and currently works as an Agile Coach

Daniel Bryant

works as an Independent Technical Consultant and Product Architect at Datawire. His technical expertise focuses on 'DevOps' tooling, cloud/ container platforms, and microservice implementations. Daniel is a Java Champion, and contributes to several opensource projects. He also writes for InfoQ, O'Reilly, and TheNewStack, and regularly presents at international conferences such as OSCON, QCon, and JavaOne. In his copious amounts of free time, he enjoys running, reading and traveling.

Dustin Schultz

is a Lead Software Engineer, a Pluralsight Author, and a technology evangelist at heart. He holds a Bachelors and a Masters in Computer Science and has been developing software for over 15 years.

Dylan Schiemann

As CEO of SitePen and co-founder of Dojo, Dylan Schiemann is an established presence in the JavaScript and open source communities.

Erik Costlow

is a software security expert with extensive Java experience. He manages developer relations for Contrast Security and public Community Edition.

Helen Beal

is a DevOps consultant, coach, trainer, speaker and writer. Her focus is on helping organisations optimise the flow from idea to value realisation through behavioural, interaction based and technology improvements. Fond of llamas.

Jan Stenberg

is working as an IT consultant since more than 25 years in northern Sweden, experienced in building systems on both .Net/C# and JVM/Java platforms. His experiences range from large distributed and service based systems through web based and rich client applications down to hardware related software.

Jonathan Allen

got his start working on MIS projects for a health clinic in the late 90's, bringing them up from Access and Excel to an enterprise solution by degrees.

Manuel Pais

is a DevOps and Delivery Consultant, focused on teams and flow. Manuel helps organizations adopt test automation and continuous delivery, as well as understand DevOps from both technical and human perspectives.

Rafiq Gemmail

is a freelance technical coach, teacher and polyglot who has coached the principles of fast-feedback with leading firms in New Zealand. He is a passionate advocate for mob programming, having supported cross-functional teams through over a year of mobbing at New Zealand's largest news site.

Rui Miguel Ferreira

Rui's great passion is the team. He spent the major part of his career influencing teams with a strong, self-motivated and positive attitude, always supporting a winning culture. He truly believes that following an Agile approach as Scrum, he can get the most of a team.

Shaaron A Alvares

is a News Reporter and Editor for DevOps, Culture and Methods at InfoQ and works as an Agile Transformation Coach and Trainer at T-Mobile in Bellevue, Washington. She is Certified Agile Leadership, Certified Agile Coach from the International Consortium for Agile, and Agile Certified Practitioner, with a global work experience in technology and organizational transformation.

Shane Hastie

leads the Culture and Methods editorial team for InfoQ.com where he hosts the weekly InfoQ Culture Podcast. He is the Director of Agile Learning Programs for ICAgile and is the founding chair of the Agile Alliance New Zealand.

Steffen Opel

is managing partner at Utoolity, a provider of tools for cloud computing operations and software development processes. With a formal education in C++, and an early focus on rich client technologies, he joined the paradigm shift to RESTful web service architectures early on.

Susan McIntosh

is an agile coach and scrum master with a background in programming. A former teacher and consultant, she has been drawn to agile practices, especially the training and change management that are a part of transformations

Thomas Betts

is the Lead Editor for Architecture and Design at InfoQ, and a Principal Software Engineer at IHS Markit Technology, now a part of Informa Tech. For over two decades, his focus has always been on providing software solutions that delight his customers.

Werner Schuster

Werner Schuster (murphee) writes software and writes about software. After a day of JavaScript wrangling, he prefers Mathematica, Clojure & Co.

Save time learning about new technologies

QCon London 2020

Conference for Professional Software Developers

Shortcut your way to the newest trends in the software industry, attend QCon London (March 2-6) the conference that teaches what you don't know as a software developer, architect or team lead.

QCon London 2020 tracks include:

- Next Generation Microservices: Building Distributed Systems the Right Way.
- Leaving the Ivory Tower: Modern CS Research in the Real World.
- JavaScript: Pushing the Client Beyond the Browser.
- Full Cycle Developers: Lead the People, Manage the Process & Systems.
- Modern Compilation Targets.
- Building High Performing Teams.
- Machine Learning: The Latest Innovations.
- Growing Unicorns in the EU: Building, Leading and Scaling Financial Tech Start Ups.

Take a look at the rest

QCon LONDON by InfoQ

Get 75GBP off
with code **INFOQ20**

A LETTER FROM THE EDITOR

Charles Humble

took over as editor-in-chief at InfoQ.com in March 2014, guiding our content creation including news, articles, books, video presentations and interviews. Prior to taking on the full-time role at InfoQ, Charles led our Java coverage, and was CTO for PRPi Consulting, a renumeration research firm that was acquired by PwC in July 2012.

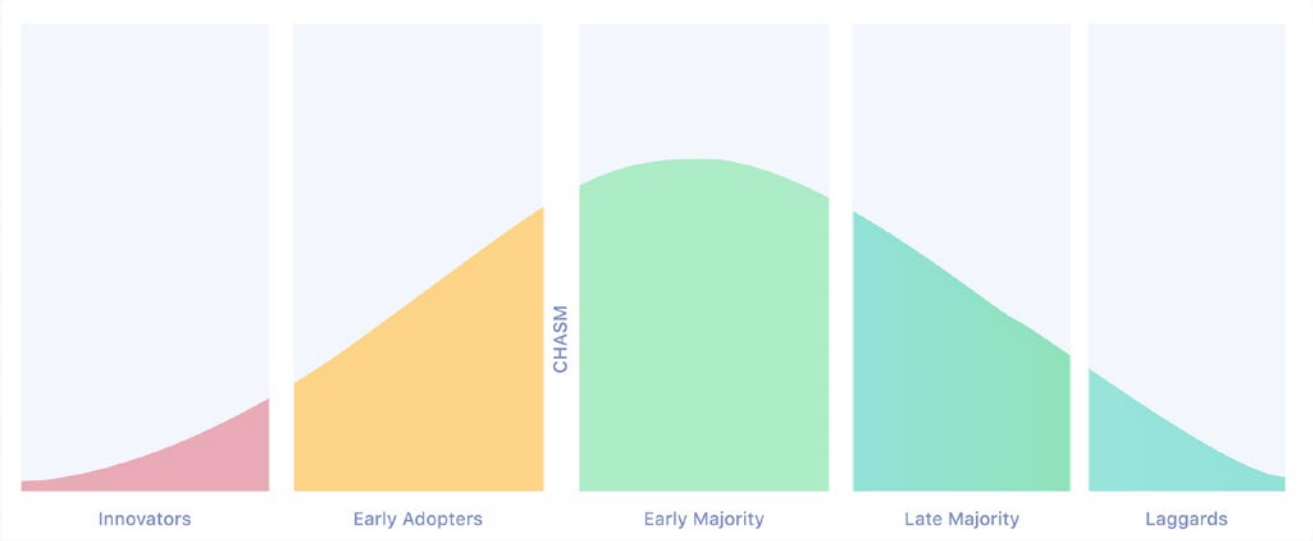
At InfoQ we are passionate about software. Our team of regular editors and contributors all have full time jobs in the software industry, building software and managing software teams. Our mission is to facilitate the spread and change of innovation in professional software development. We do this through the content we publish online via InfoQ and a number of other platforms including YouTube, Apple News, Alexa, as well as in person through the various QCon conferences we run around the world.

To help guide our content, groups of our editors meet regularly to discuss the global content strategy for InfoQ and the technology trends that significantly impact our industry. During these discussions we consider the state of practice, emerging ideas and things we hear within our network and at meetups, conferences, analyst events, etc. We also take into account traffic patterns on the site and attendance at sessions at QCon and other industry conferences as well as, where possible, publicly accessible surveys, our own reader surveys and other data.

The output from these discussions is a series of topic graphs, based on a technology adoption curve, which we use to inform our content policy.

The model we use is loosely derived from Geoffrey Moore's book "Crossing the Chasm".

This model shows a skewed bell curve where the divisions in the



curve are roughly equivalent to where the standard deviations would fall. That is, the early majority and late majority fall within one standard deviation of the mean, the early adopters and laggards within two and the innovators at three. It is important to note that something's exact position on the adoption curve can vary by location and industry. We do try to normalize for this as far as we are able.

In general, when considering this model from a company perspective we believe that those looking to adopt technology in the innovator and early adopter stages are seeking competitive advantage, whilst those who adopt later - at early majority and beyond - are typically seeking proof of the effectiveness of a technology or strategy prior to adoption.

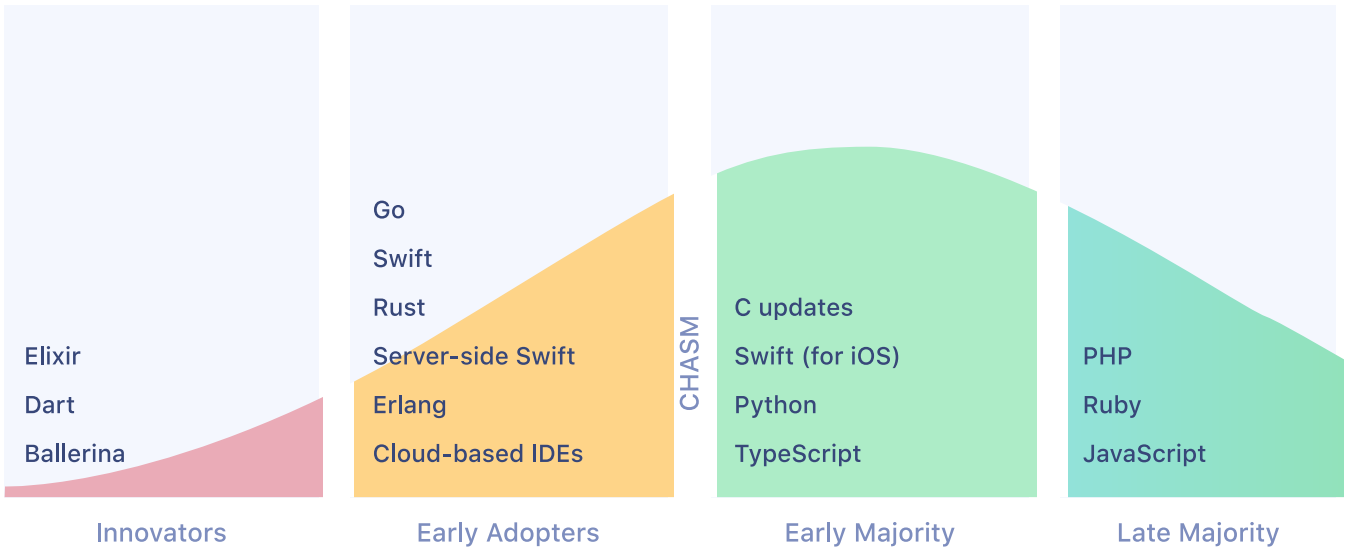
From an InfoQ content point of view, if a topic is on the right-hand part of the graph, you will probably find lots of existing content on InfoQ about it – we covered it when it was new, and the lessons learned by the innovators and early adopters are available to help guide individuals, teams and organizations as they adopt these ideas and practices.

The topics on the left-hand side of the curve are the ones we see as emerging now, being used by the innovators and early adopters, and we focus our reporting and content on bringing these ideas to our readers' attention so they can decide for themselves whether they should be exploring them now, or waiting to see how they unfold.

Since early 2018 we have been publishing these discussions on InfoQ to provide readers with an "InfoQ view of the world". The InfoQ trend reports represent an opinionated guide to the various software trends that we are tracking.

This eMag brings together the complete set of reports from the last 12 months and as such represents various points in time. We hope that this format provides InfoQ readers, from developers to CTOs, with a concise summary of the professional software landscape. We encourage you to explore these technologies for yourselves.

Software Development
Programming Languages 2019 Q3 Graph
http://info.link/proglang2019



InfoQ Trends Report - October 2019

Programming Languages

What are the programming language trends for 2019 and beyond? This report aims to assist technical leaders in making mid- to long-term technology investment decisions, and to help individual developers identify popular programming languages and choose where to invest their valuable time and resources in terms of which new programming languages to learn and skills to develop.

The report provides a summary of how the InfoQ editorial team currently sees the adoption of technology and emerging trends within the programming language space, with the exclusion of Java / the JVM and JavaScript / Web Development which are reported on separately. This is an opinion piece that is formed from extensive internal discussions with our editors, who are active software engineers themselves, reviews of external public survey results, and reviews of internal or private survey and analysis results. Extracts of the various editorial conversations can be found below.

Both InfoQ and QCon focus on topics that we believe fall into the innovator, early adopter, and early majority stages. What we try to do is identify ideas that fit into what Geoffrey Moore referred to as the early market, where "the customer base is made up

of technology enthusiasts and visionaries who are looking to get ahead of either an opportunity or a looming problem." We are also on the lookout for ideas that are likely to "cross the chasm" to broader adoption. It is worth saying that in this context a technology's exact position on the adoption curve can vary. As an example, Rust may be widely adopted at this point among San Francisco Bay Area companies but may be less widely adopted elsewhere.

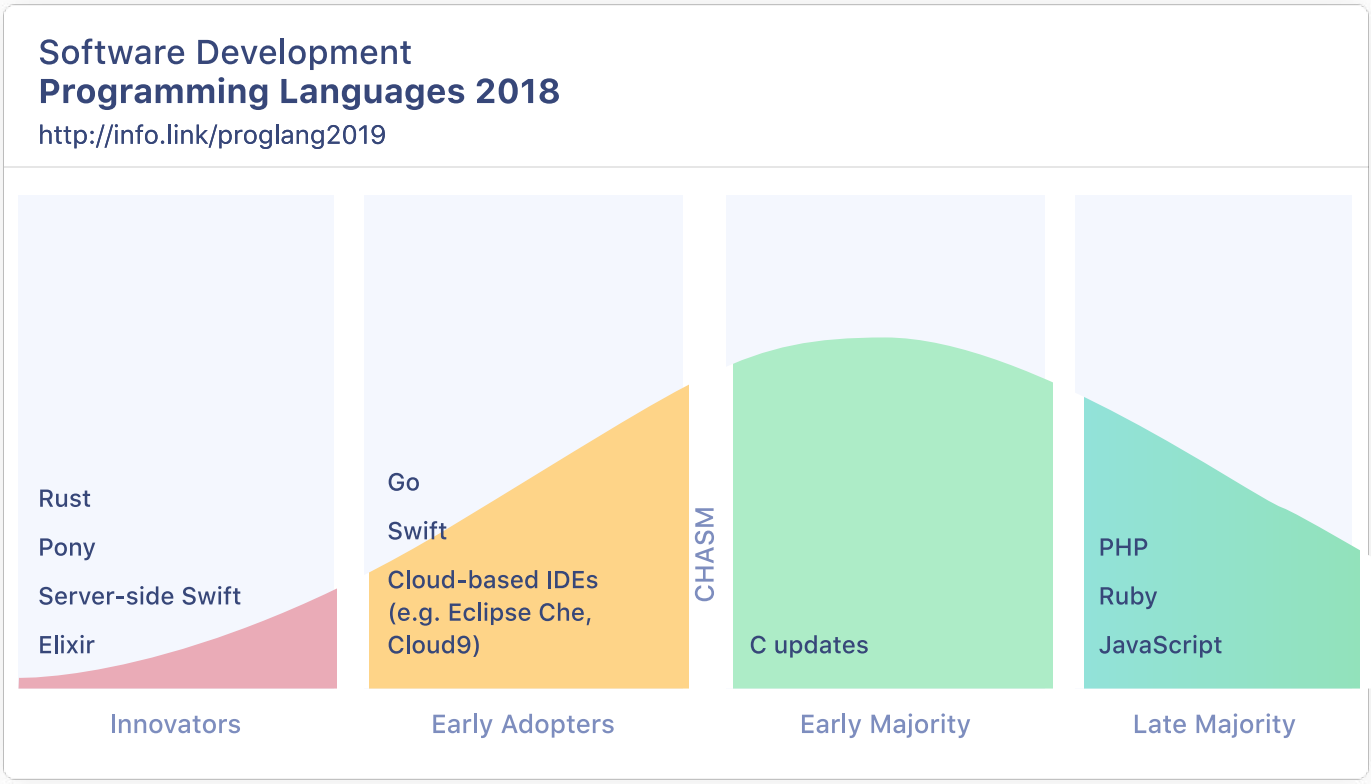
Notable changes since our internal 2018 programming trends report was produced include the addition of several languages that, although popular, we were not publicly tracking on the adoption curve, such as Python, Ruby, and PHP. Although technologies and factors relating to these languages were being discussed internally, we have now rectified their absence on our trend graph.

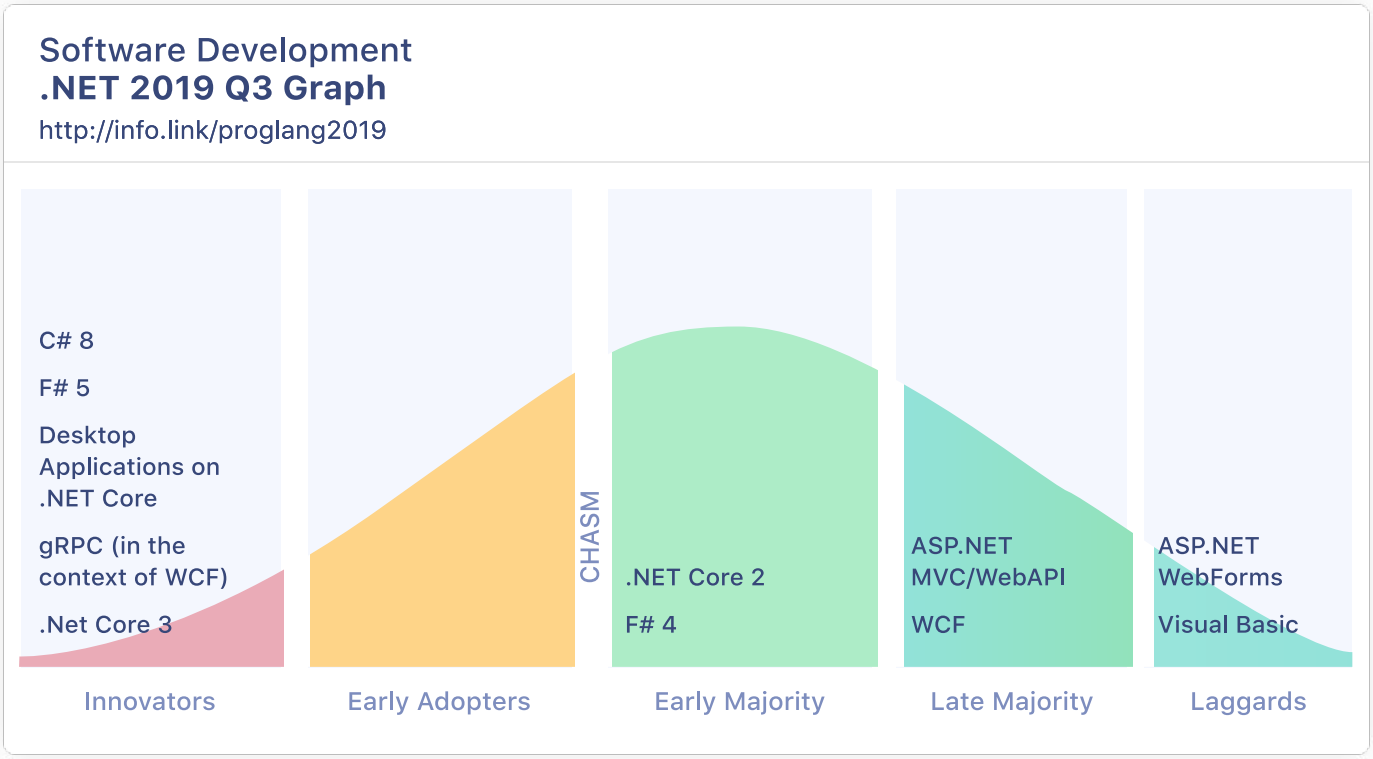
Programming Languages on the move: Elixir, Rust, and Swift

Elixir — a functional, concurrent, general-purpose programming language—has entered the trend report at the innovator adoption phase. Elixir builds on top of Erlang (which we have placed within the early adopter category) and runs on the Erlang virtual machine. Elixir and Erlang share the same abstractions for building distributed, fault-tolerance applications.

We believe that Rust has moved from innovator to early adopter, driven largely by its uptake within the infrastructure and networking data plane space — for example, Habitat and Linkerd 2.0. The language is also emerging as a natural partner for WebAssembly, which is helping drive awareness. In addition we note that Facebook has chosen to implement its Libra cryptocurrency using Rust.

Swift for iOS development has moved to early majority, primarily because of the popularity of iOS as a mobile application runtime. Although we are not tracking Kotlin within this trend category, the InfoQ Java and JVM trends report has seen the popularity of Kotlin increase as this is now the default language for developing Android apps.





Languages of Infrastructure: Ballerina, Pulumi, Dark

We are seeing increased interest related to infrastructure/cloud-targeted languages, DSLs, and SDKs like Ballerina and Pulumi. We're also tracking Dark, which is currently in private beta and so doesn't quite make it onto the graph yet. This category of technologies is currently firmly planted within the innovator adoption phase, but we are watching this closely, and we had a track dedicated to "Languages of Infrastructure" at QCon San Francisco in November.

For context, here is what our internal topic graph looked like for the second half of 2018. The 2019 version is at the top of the article.

.NET Q3 2019

For .NET we see a lot of interest in .NET Core, and with .NET Core 3 having just arrived, we expect this will continue. We've split .NET Core into the 2x branch and 3x branch on the graph since they are at different points in terms of adoption. Regarding .NET languages, we're expecting C# 8 will be adopted rapidly. We continue to track F# with interest, but feel that complexity and a lack of strong support from Microsoft means it will unlikely gain broader adoption.

We have ASP.NET Core in early majority. At this point we believe most new web work is being done on ASP.NET Core, but not necessarily .NET Core. Some existing applications are being ported, but most will not. WCF is still really important in the enterprise, but MS isn't properly supporting it in .NET Core. We've moved Visual Basic to laggards, and really see it as a language for hobbyists at this point.

The following is a series of lightly-edited and aggregated excerpts from the corresponding internal conversations between several of the InfoQ editorial team, which provides for more context for our recommended positioning on the adoption graph.

Abel Avram, Assistant Professor at the Computer and Automatics Faculty of the Technical University of Timisoara, and InfoQ Editor:

JetBrains published their The State of Developer Ecosystem 2019, polling 7,000 developers and coming to the following conclusions:

- Java — the most popular primary programming language
- JavaScript — the most used overall programming language
- Go — the most promising programming language
- Python — the most studied language

Go started out with a share of 8% in 2017 and now it has reached 18%. In addition, the biggest number of developers (13%) chose Go as a language they would like to adopt or migrate to.

StackOverflow had this to say about Python this year: "Python, the fastest-growing major programming language, has risen in the ranks of programming languages in our survey yet again, edging out Java this year and standing as the second most loved language (behind Rust)."

Arthur Casals, Computer Science researcher working in the area of Artificial Intelligence / Multi-agent Systems:

From what I've been seeing/reading lately, Rust seems to be picking up the pace in terms of adoption. It's still (somewhat) niche though, and seen as "a replacement for C++." However, with increasing numbers of available libraries and more adoption by major players, I think it's a good bet for "early adopter."

Elixir seems to be gaining some momentum especially in Brazil — there's the fact that the language's creator is Brazilian, and at least one of the new unicorns there (Movile) is openly using it (along with other trending companies such as Quero, in education).

I've seen some chatter about Elixir recently — especially after they had someone from Quero as a panelist, telling the story of how they adopted Elixir since the beginning, etc. There's also a nice use case from the engineers at Discord, who recently wrote about using Rust and Elixir combined to provide backend support for 11 million concurrent users. This is a great real-world scenario involving both languages and a widely-used application, which definitely adds to their momentum.

Dylan Schiemann, CEO of SitePen and co-founder of Dojo:

Rust is in the early adopter stage, and in my experience its adoption is gaining in large part because of WebAssembly and perhaps Servo.

While Go has been interesting for a while, I would still classify its adoption as early adopter. Languages have a much slower lifecycle than frameworks for example, and I really wouldn't say that Go has reached mainstream adoption yet. Similarly, Elixir and Elm would rate as early adopters.

In my opinion, Pony is a language I rarely hear mentioned, and remains an innovator at this time.

Swift has probably reached early majority due to it being the default entry point for iOS applications.

For Cloud-based IDEs, CodeSandbox and StackBlitz have gained significant popularity over the past year due to their impressive feature set for browser-based editing, as well as integration with local development tools.

After stagnating for a couple of years, Dart seems to be having a resurgence in large part because of Flutter.

Charles Humble, Editor in Chief at InfoQ:

I'd move Swift to early majority status — it seems to me that is continuing to pick up steam, and I think, judging from some of the announcements at WWDC — noticeably Swift UIKit — that we're entering the "Swift era" for Apple's platforms.

I think we can also move server-side Swift to early adopter. This is anecdotal, but talking to some people who are close to it, they tell me it's seeing steady growth, and there is a lot of good stuff that has been driven by the open-sourcing of swift-nio, which has in turn increased the performance of several server-side frameworks that have moved on to it.

Rust definitely seems to have seen growth in the last year, and I'd move it from innovator to early adopter. They've been good at positioning it as a partner with Wasm, which is helping, I think.

I'm not sure about Go; it is still growing, but my sense is that it hasn't quite reached the mass adoption point yet; possibly controversial, but I'd leave it at early adopter.

I'm hearing a bit more about Elixir, but I think it still stays at innovator; it's still quite niche.

We also should put Python on here, and probably within early majority. It's continuing to grow in popularity, largely driven by its popularity among data scientists, and it is a language I think we need to talk about more.

I don't think Pony is likely to move beyond Innovator; it's a shame. I like the language, although I found the type system rather complex, and I do think it is an impressive piece of work, but the community for it is still very small. I would drop it from the tracker at this point.

Finally I'm hearing more about Dark, although it is in private beta. I find that interesting in the context of languages for infrastructure more broadly.

Werner Schuster, Software Plumber at Wolfram:

On Swift: it's really early days, but Swift for TensorFlow (S4TF) might become a big selling point for Swift and a potential competitor to Python (at least in that area).

Here's a very thorough and interesting document on why Swift and not other languages. It has Chris Lattner behind it who works on this now at the home of TensorFlow; and Chris is up to his old tricks, creating IRs all over the place (MLIR)

The S4TF name is kinda boring and hides all the interesting bits (AD, etc).

On Python: I'm kind of intrigued where Python goes; it clearly has benefitted hugely from having been the glue language of choice for lots of data science/ML native libraries—something Ruby didn't figure out a decade ago, to its loss.

On the other hand, Python has been remarkably resistant to any sort of modernization of its runtime (anyone remembers Unladen Swallow?) not to mention the decades long project(s) for de-GIL-ing the interpreter (none of which have gone anywhere).

I think Jython is dead too. Last time I checked the website, it hadn't been updated in years and isn't anywhere near Python 3.x.

Infrastructure languages: there seems to be a new trend of coupling languages (some new ones) to deployment.

- Pulumi & co — Typescript
- Ballerina
- Unison Language — A new functional language, by ex-Scala/Haskell folks, but nicely free of Monads; it is

still very much early days here

- Dark Language — not public yet

I'm curious where all of these are going to go; might be an overly coupled flash in the pan, or the first step to language features that will seem indispensable in 15 years (like when subroutines became language features).

We're potentially looking at a new, unencumbered Wasm future (browser and server side with WASI etc.) as well as a newfound appreciation of native (with VMs, containers, serverless) which reduces the appeal of the JVM/bytecode and allows everyone to just use the native versions of languages (the canonical versions that have been under development for decades). To be clear—none of what I said above really concerns JVM native languages like Clojure, Scala, or Kotlin, etc.

Ben Evans, Principal Engineer and JVM Technologies Architect at New Relic, Inc.:

We have a small amount of stuff in Elixir. I am not particularly a fan — it's a nice enough language but unless you need the actual strengths of the BEAM VM (fault tolerance, massively distributed simple things), then it doesn't seem to add much.

Most of the fans of the language seem to like it because it's the "new shiny," and/or they have Java Allergy (usually based on a view of the platform that's >10 years out of date), but ... I haven't seen anything compelling in it that couldn't also be done fairly easily in JVM tech (albeit with potentially a bit more ceremony) — and the JVM has much better, broader integrations with libraries.

Dustin Schultz: Lead Software Engineer, Pluralsight Author, and technology evangelist

I've seen quite a rise in Python in the past years. As others have stated, I think this is largely due to data science, teaching it schools/universities, and lighter frameworks like Flask (vs Django).

I don't have anything against Python (I actually enjoy writing it), but I think it's popularity does not represent usage for enterprise use cases. I rarely see large, distributed, enterprise applications written in Python that stay in Python. They may start out in Python, but they eventually switch to something else because of performance.

Charles Humble: Editor in Chief at InfoQ

In terms of the core framework it seems to have had incremental improvements since about 2012 — I think a lot of effort is focused on .NET Core. My sense of this is that .NET Core has moved from innovator to early adopter. I wonder if we should split it when .NET Core 3 lands and perhaps have .NET Core 3 back in innovator and .NET Core 2 in early adopter.

Likewise I think C# should probably shift to early majority — to be honest, I'm not sure why it isn't.

I think F# is following something of a similar path to Clojure on the JVM — it's a lovely language in my view, but I don't see much sign of it expanding beyond its current size. Equally I don't see use declining much either.

Jonathan Allen, Software Architect for KPMG and Lead .NET Editor for InfoQ:

Talking to the community, F# has two problems:

It doesn't have strong support from Microsoft

- The leadership is focused on chasing C# compatibility at the cost of ease of use
- It is continuing to grow more and more complex, which the Computer Scientists like. But, the trainers actually teaching F# are mainly focused on non-programmers who need a scripting language. So, the additional complexity is making their job harder.

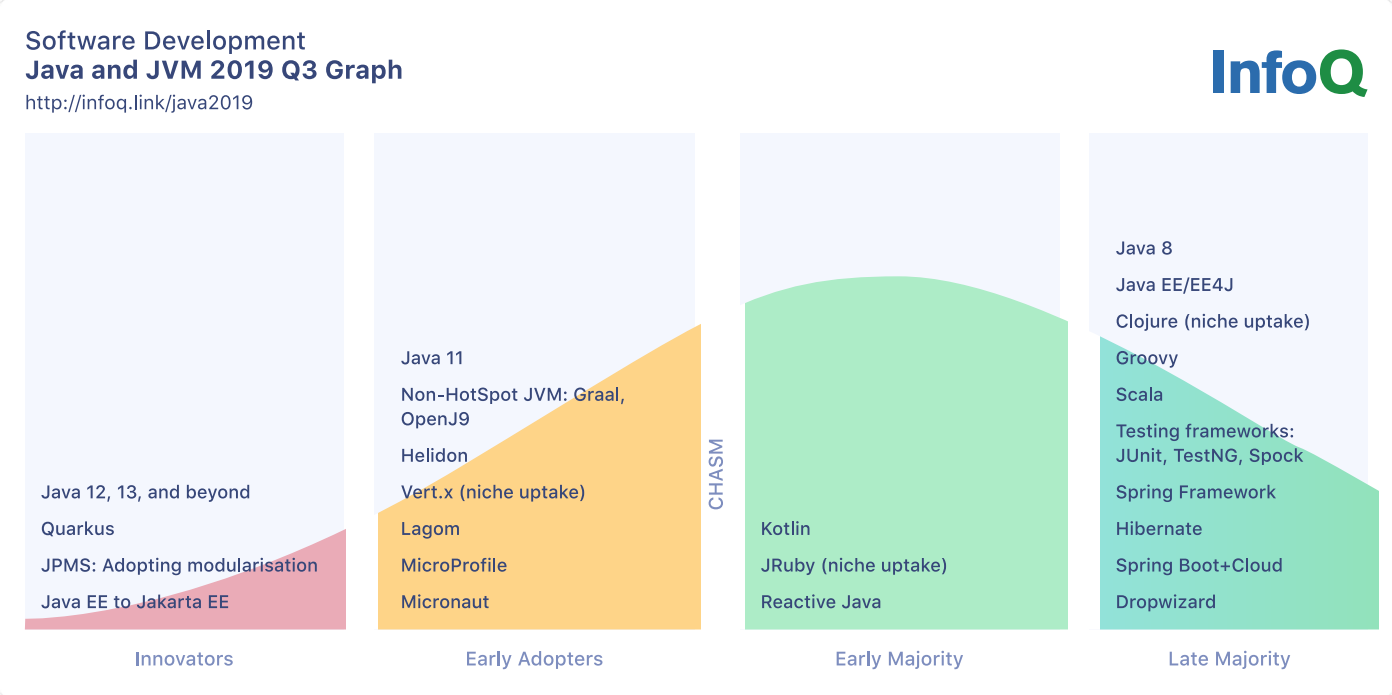
I also think C# 8 is going to reach early adopter fast. People have been asking for nullable reference types since...well, .NET 1 to be honest.

Additional publicly available research and surveys that were discussed included: "The RedMonk Programming Language Rankings: June 2019" and the IEEE Spectrum "The Top Programming Languages 2019."

Let us know which of the programming languages you have used in the last 12 months or which ones you are planning to use. Fill in the survey.

TL;DR

- Elixir has entered the trend report at the innovator adoption phase. It is a functional, concurrent, general-purpose programming language that runs on the Erlang virtual machine.
- We are seeing increased interest and innovation related to infrastructure-aware or cloud-specific languages, DSLs, and SDKs like Ballerina and Pulumi.
- We believe that Rust has moved from the innovator to early adopter phase, driven largely by its uptake within the infrastructure and networking data plane space—for example, Habitat and Linkerd 2.0.
- Python is continuing to gain in popularity, largely thanks to its roles within data science and teaching.
- Swift for iOS development has moved to early majority, primarily because of the popularity of iOS as a mobile application runtime. Kotlin, although tracked separately in the JVM trend report, has seen similar movement to early majority in relation to Android app development.
- For .NET we see a lot of interest in .NET Core, and, with .NET Core 3 having just arrived, we expect this will continue.



InfoQ Trends Report - July 2019

Java

This article provides a summary of how the InfoQ editorial team currently see adoption of technology and emerging trends within the Java space, which focuses not just on Java the language but also on related languages like Kotlin and Scala, the Java Virtual Machine (JVM), and Java-based frameworks and utilities. We discuss trends in core Java, such as the adoption of Java 11 and 12, and also the evolution of web-based frameworks like Spring Boot and MicroProfile.

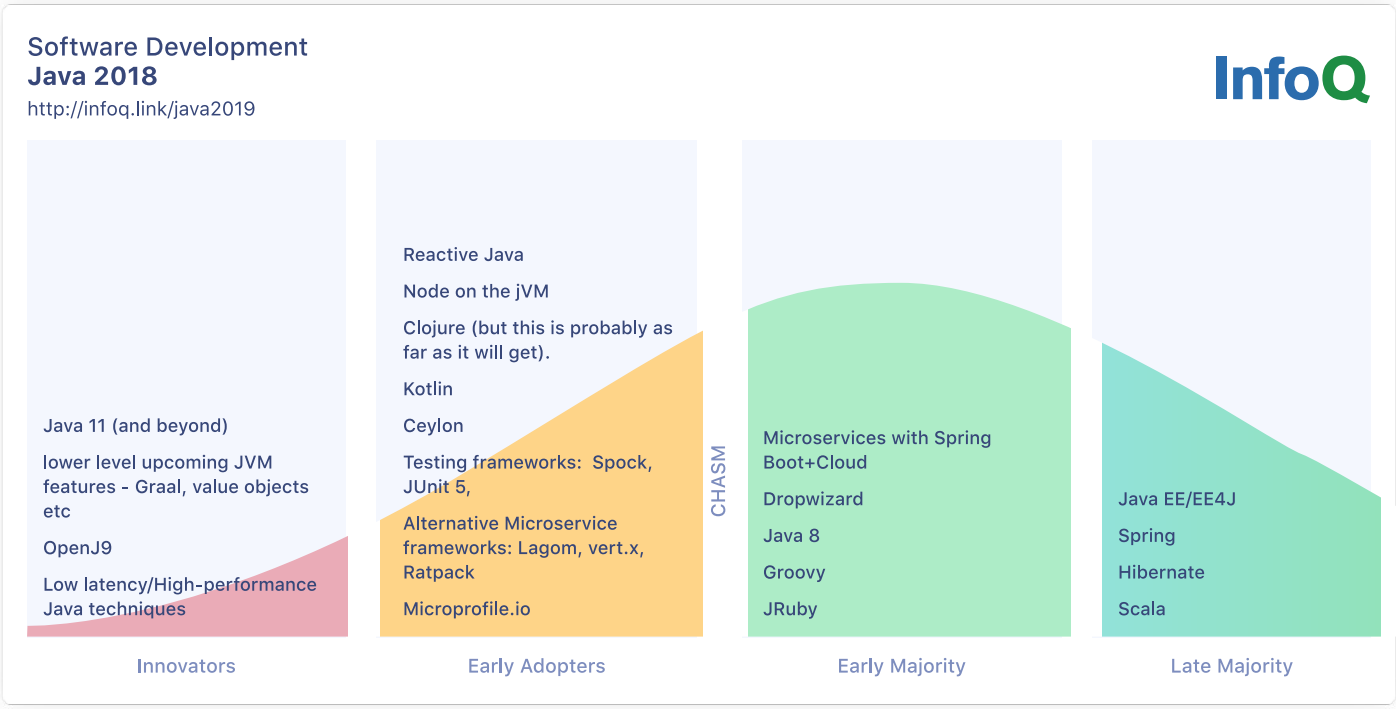
This report aims to assist technical leaders in making mid-to-long term technology investment decisions, and also help individual developers in choosing where to invest their valuable time and resources for learning and skill development. This is our first published Java trends report, although the topic has received ample coverage since InfoQ launched in 2006, and we have been tracking Java and JVM trends internally for many years.

Both InfoQ and QCon focus on topics that we believe fall into the “innovator, early adopter, and early majority stages”. What we try to do is identify ideas that fit

into what Geoffrey Moore referred to as the early market, where "the customer base is made up of technology enthusiasts and visionaries who are looking to get ahead of either an opportunity or a looming problem". We are also on the lookout for ideas that are likely to "cross the chasm" to broader adoption. It is perhaps worth saying, in this context, that a technology's exact position on the adoption curve can vary. As an example, Java 11 may be widely adopted at this point amongst San Francisco Bay Area companies but may be less widely adopted elsewhere.

Notable changes since our internal 2018 Java trends report was reduced include the addition of Java 13 (a non-LTS release) within the innovator stage of adoption. This has caused a cascading effect of moving Java versions along the adoption curve, with Java 11 (an LTS release) moving to early adopter (EA), and Java 8 (which is now officially tagged as end-of-life for commercial support from Oracle) moving to late majority (LM).

We are seeing increased adoption of non-HotSpot JVMs, and we believe OpenJ9 is now within the early adopter stage, and we have also introduced Graal to the graph. We believe that the increasing adoption of cloud technologies within all types of organisation is driving the requirements for JREs that embrace associated "cloud-native" principles such as fast start-up times and a low memory footprint. Graal in itself may not be overly interesting, but the ability to compile Java application to native binaries, in combination



with the support of polyglot languages, is ensuring that we keep a close watch on this project.

There has been a shuffle of Java microservice frameworks, with Spring Boot and Spring Cloud moving into late majority. It's worth remembering that this means that the rate of uptake is slowing (due to market saturation), and not that developers are moving away from the framework. Helidon has joined MicroProfile within EA, and we believe that vert.x will not progress past the EA phase, due to the relatively niche appeal.

For context you can see what our graph looked like for the second half of 2018 on the previous page. The 2019 version is at the top of the article.

The following is a lightly-edited and aggregated summary of the corresponding internal email conversation between several of the InfoQ Java editors, which provides for more context for our recommended positioning on the adoption graph.

Daniel Bryant, Independent Tech Consultant, Product Architect at Datawire, and InfoQ News Manager:

So, it's time to update the Java trends graph! I'm keen to explore if all of the topics in our internal 2018 tracker are located in the correct adoption phase, and also identify any topics to be removed or added.

For a starter for ten, I would suggest changing in innovator:

- Java 11 -> Java 12/13
- Move OpenJ9 to early adopter (EA)
- Add Adopting Modularisation
- Add Quarkus

In EA:

- Move Spock to late adopter (LA)
- Move microservice frameworks to LA, with the exception of Lagom and MicroProfile?
- Add Helidon

I'm happy to see the late adoption and laggard phases remain the same.

Erik Costlow, Software Architect, focused on security and Java:

Here's my take:

- Move Java 8 to Laggards. It was released in 2014 and Oracle ended public support.
- LTS Java (JDK11) is still early-adopter, I think
- non-LTS (JDK 13?) is Innovator, rotating whichever is current.
- Graal is not key in itself; people are really wanting two things:
  - Statically compiled native Java applications (native_image)
  - Polyglot Java (e.g. Ruby Truffle)
  - There is an outstanding issue of if Graal native_image is Java, or if other vendors can use the same closed-world assumption for static compilation. The issue is that native_image does not pass the TCK. The key question is are other JVM vendors permitted to do this or what would happen if they did?
- Graal and OpenJ9 should just become non-HotSpot JVM since we have Excelsior, Azul, Liberica, Corretto, Alibaba

Dragon, AdoptOpenJDK, RedHat, and so on.

- Node on the JVM is dead.
- Move Testing Frameworks into Early Majority
- Move Spring Boot into Late Majority, or I see this a lot and it's a safe choice.
- Add "Migrating Java EE to Jakarta EE" as Innovators. This will be interesting and I hope it picks up.

Right now there are too many Java distributions and releases. I would like to see an Innovator talk about managing this at scale, or how people can actually do this to operate between teams, systems, and other things where some applications or services change and others don't. If an application is native-compiled, who becomes responsible for patching that "JRE" and what does that mean?

Dustin Schultz, Lead Software Engineer, Pluralsight Author, and technology evangelist:

+1 on all your suggestions for innovator (technically Quarkus is vert.x and MicroProfile, but I agree that it's innovator.)

Personally, I don't think Spock has ever (or will ever) make it past EA. I haven't seen tons of adoption.

I'd remove all alternate microservices frameworks except Vertx from EA ... potentially could leave Lagom. Not sure that either will make it past this though.

It may be worth moving Groovy to late majority since its pretty flat and will probably decline.

Charles Humble, Editor in Chief at InfoQ:

Agree with most of this. "Adopting Modularisation" is interesting. I'm sure the OSGI folks would have something to say about having it in innovator, but in truth it is probably correct.

I think we should move Clojure to late majority. It's still used, of course, but I sense its use is in decline, and it was always quite niche.

I would remove Ceylon. It never got much traction, and I don't think moving to Eclipse has helped it much; there's very little activity on the project as far as I can see - roughly 10 commits in the last 6 months.

With testing frameworks that picture looks quite stable to me at this point. Should we drop it? I'd roughly rank in order: Junit, TestNG/Cucumber, and finally, Spock in terms of use/adoption, but I don't think it's changing much?

SmartBear recently acquired Cucumber which might give it a boost, but even so, I'm inclined to drop tracking these.

Does node on the JVM get much interest? I'd remove this I think.

I would be tempted to move Kotlin to Early Majority. It seems to be the JVM language that I hear the most about these days, thanks mainly to its popularity for Android development I think, and I would say it has crossed the chasm, although I would guess it lags someway behind something like Groovy, or, say, Swift or Go.

I think Scala can stay in late Majority - use is declining I would say, but very slowly.

I'm not quite sure how to handle the Java version situation. My reading is that the non-LTS releases have pretty limited adoption and maybe all live in innovator, but that's kind of weird when some of them are also EOL. Still, in terms of graphing it, I guess Java 8 is still late majority, Java 11 in EA and Java 13 in innovator?

Ben Evans, Principal Engineer & JVM Technologies Architect at New Relic, Inc.:

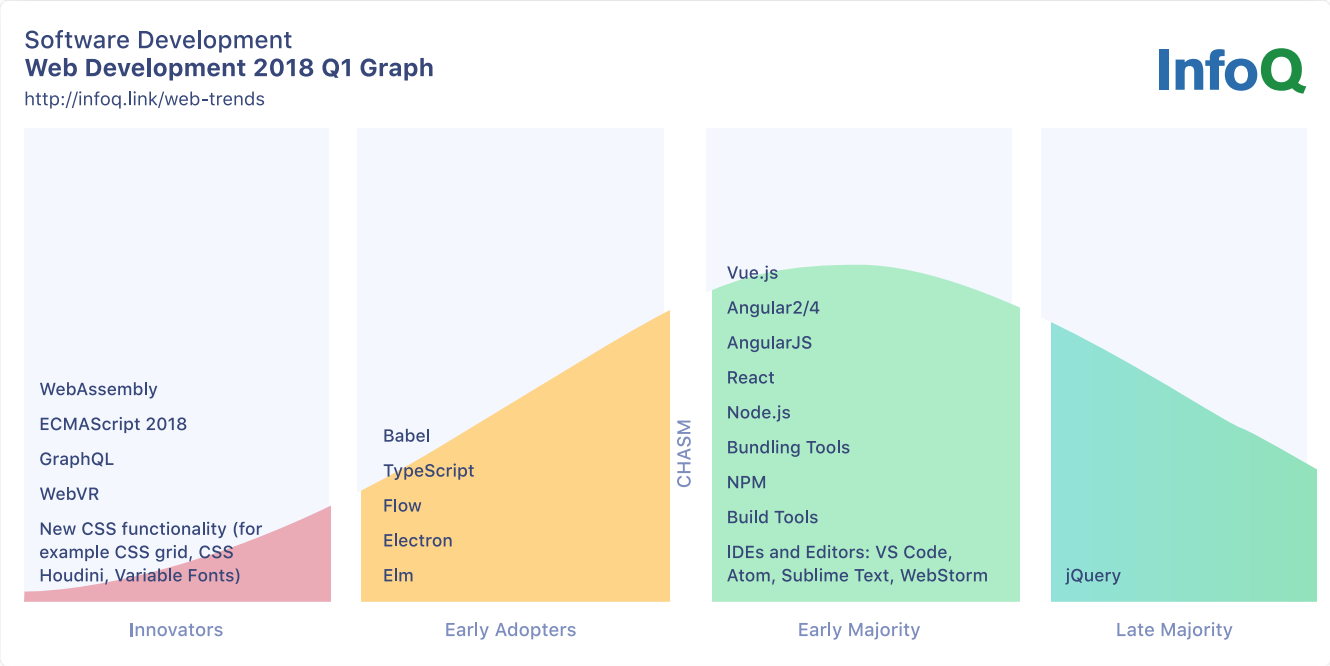
Agree completely about non-LTS releases of Java. The data from production deployments that I see has a stable and rising 5+% of Java 11 deployments so that feels like EA to me.

Definitely remove Ceylon - it never got enough traction and its lunch was basically eaten by Kotlin.

Java 8 should stay in a majority column - it still has significant amounts of new services being written in it, and there are still major libraries (e.g. Cassandra) that are not compatible with *any* post-8 version. We haven't yet seen a major library to drop Java 8 support yet (I believe ElasticSearch will be the first).

TL;DR

- The soon to be launched Java 13 (non-LTS) and the recently released Java 12 (non-LTS) have been added to the innovator stage of adoption within our Java and JVM trends graph.
- Java 11 (LTS release) has moved to early adopter (EA), and Java 8 -- which was marked as end-of-life for commercial use in January 2019 -- has moved to late majority (LM).
- Non-HotSpot JVM are seeing increased adoption, with OpenJ9 moving to EA, and the addition of Graal to our adoption graph. We believe that the increasing adoption and importance of cloud computing is driving organisations to utilise Java Runtime Environments (JREs) that embrace "cloud-native" principles such as a fast start-up time and a low memory footprint.
- Graal in itself may not be overly interesting to regular developers, but the ability to compile Java application to native binaries, in combination with the support of polyglot languages, is ensuring that we keep a close watch on this project.
- There has been a shuffle of Java microservice frameworks, with Spring Boot and Spring Cloud moving into late majority, as these are trusted choices for creating Java microservices within the enterprise. Helidon has joined Microprofile within EA, and we believe that vert.x will not progress past the EA phase, due to it's relatively niche appeal.



InfoQ Trends Report

JavaScript and Web Development

At InfoQ we periodically update our topics graph to show where we think different topics are in the technology adoption curve. When we do so, we consider the state of practice, emerging ideas and things we hear within our network and at meetups, conferences, analyst events, etc. We also take into account traffic patterns on the site and attendance at sessions at QCon and other industry conferences.

If a topic is on the right-hand part of the graph, you will probably find lots of existing content on InfoQ about it – we covered it when it was new, and the lessons learned by the innovators and early adopters are available to help guide individuals, teams and organizations as they adopt these ideas and practices.

The topics on the left-hand side of the curve are the ones we see as emerging now, being used by the innovators and early adopters, and we focus our reporting and content on bringing these ideas to our readers' attention so they can decide for themselves

whether they should be exploring them now, or waiting to see how they unfold.

This month we're looking at JavaScript and Web Development as this rapidly changing industry requires an update more than once a year. This is how the graph looked for us in Q1 of 2018, when we last reviewed it.

Below is the revised version from Q4 of 2018

The web development space is always an interesting one for us, with new JavaScript frameworks launched almost daily. Trying to decide which ones to focus on and which ones to ignore is particularly challenging. Developers can learn and gather inspiration from interesting approaches even if they do not currently use them in their daily development efforts.

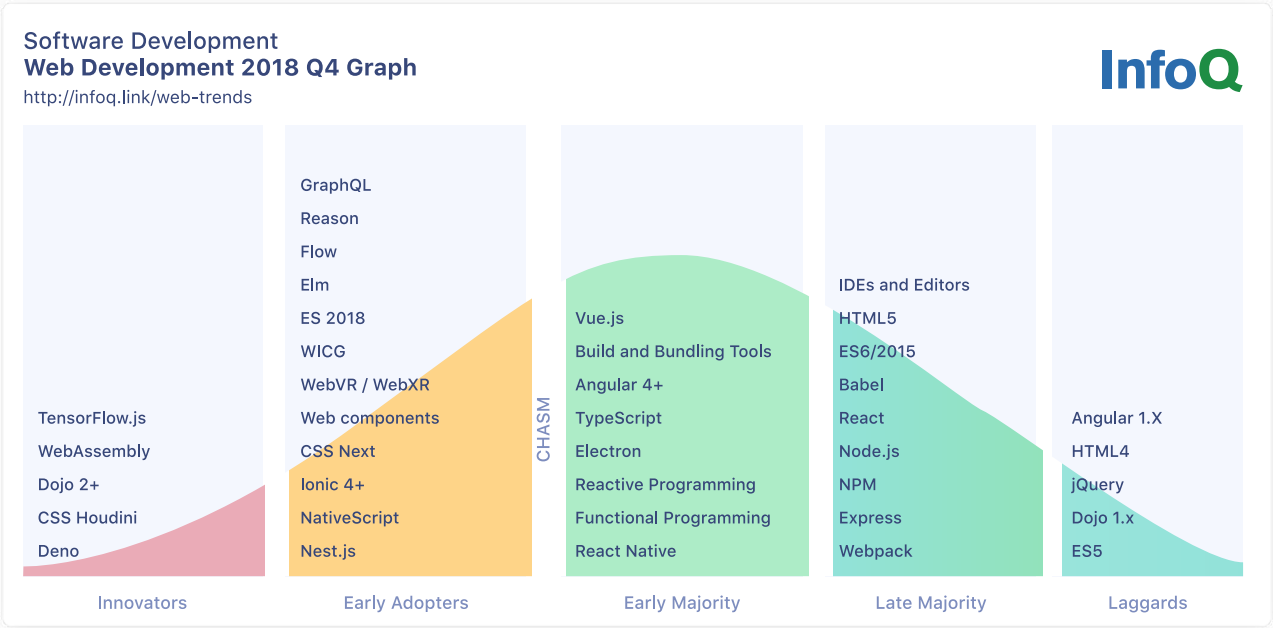
Innovators

Looking at the trends from the last few months, interest in WebAssembly continues to grow as browser vendors move beyond the minimum viable version and strive to improve performance and fill gaps.

CSS Houdini has led to a flurry of activity in the realm of CSS, and tools like PostCSS are challenging incumbent CSS preprocessors like SASS, LESS, and Stylus. CSS Houdini is an area which we're continuing to follow with interest.

Deno is a new project from the original creator of Node.js, authored in TypeScript, which attempts to resolve the challenges with Node.js. It's a very early stage project but shows significant promise.

Dojo, one of the original JavaScript toolkits dating back to 2004, released version 2 this year (and subsequently versions 3 and 4!). This represents a major rewrite using TypeScript, that turns Dojo into a modern, reactive,



standards-aligned, virtual DOM-based framework. The new version exhibits significant performance and developer ergonomic improvements over its peers but is very early in its adoption cycle.

TensorFlow.js has emerged as the foundational library for JavaScript-based machine learning. We track this library as well as projects leveraging it to advance the state of machine learning with JavaScript.

Early Adopters

Looking at our early adopters, we continue to track Flow, a static type checker for JavaScript, and Elm, an alternative language to JavaScript for generating JavaScript-based web apps.

We have promoted a few technologies to this category since our last report due to increased industry uptake:

- GraphQL continues to be a trend we follow, with several established libraries including Apollo, Vulcan.js, and urql.
- ECMAScript 2018 is now final, and work is well underway for the 2019 edition.
- WebVR and its future replacement WebXR provide virtual, augmented, and mixed reality capabilities with HTML, CSS, and JavaScript. There is significant churn currently as the standard broadens from VR to XR, but there is substantial interest in this space within the browser and JavaScript. Libraries such as A-Frame and React 360 have gained popularity by leveraging Three.js.

We have added a few early adopters as well:

Like Elm, Reason is yet another alternative to JavaScript, also offering type safety and streamlined transpilation to JavaScript.

WICG, the W3C's Web Platform Incubator Community Group, has been working to bring a wide variety of useful standards to fruition, ranging from finalized standards like Intersection Observer and Resize Observer, to more experimental techniques like picture-in-picture and WebUSB.

Web components have evolved over the past few years and are now supported natively in Chrome and Firefox, with partial support in Safari, and efforts underway in Edge. Numerous frameworks and libraries also support web components natively including Angular, Dojo, Ionic, Stencil, Svelte, and Vue.js.

CSS Next encompasses a collection of CSS improvements evolving through the standards process, many of which may be used today with PostCSS or CSS Houdini.

Ionic 4 is currently in beta and has undergone a substantial change to decouple itself from

specific frameworks like Angular and focuses on aligning with modern standards.

NativeScript provides Angular and Vue.js users with native compilation options for deploying mobile applications authored with web technologies.

Finally, Nest.js is a promising server-side framework authored in TypeScript that runs on top of Node.js

Early Majority

In our early majority category, Vue.js continues to evolve and gain support, and we feel that it has crossed the chasm at this point to "early majority" status. We see particularly strong interest in Vue.js in China, and the newly re-designed InfoQ, also makes use of the framework. Vue.js 3 is under active development, including a rewrite using TypeScript.

Version 7 of Angular was recently released and continues to improve.

Build and bundling tools, in general, are highly useful for optimizing applications for production. Nearly every framework leverages some combination of tools to optimize performance.

We have promoted Electron and TypeScript to early majority status. Electron is a widely adopted desktop application shell combining Node.js and Chromium to provide infrastructure for applications with web technologies.

TypeScript is the most widely adopted JavaScript variant, has made substantial progress over the past few years, and a majority of JavaScript frameworks now leverage its tooling and infrastructure. TypeScript on its own is one of the top 10 languages according to a recent GitHub Octoverse report, and the State of JS survey lists TypeScript as by far the most widely used JavaScript variant.

We have added three items directly to the early majority category:

React Native is a framework for building native mobile apps using JavaScript and React, and has rapidly become a popular way of building cross-platform mobile apps.

Likewise, functional and reactive programming patterns dominate the discussion around how to build JavaScript applications most efficiently. Libraries like lodash have helped popularize these patterns.

Late Majority

As the JavaScript ecosystem matures, there are now foundational technologies in the late majority category that show no sign of being replaced, but are used by nearly every JavaScript developer.

We have promoted several items to our late majority status.

IDEs and editors, in general, are widely used. On the desktop, VS Code appears to have taken a substantial lead over Atom, Sublime Text, WebStorm, and even vim or emacs. Browser-based IDEs like CodeSandbox bring much of the VS Code experience to a web browser for rapid development, experimentation, and sharing of development efforts.

Babel version 7 was recently released and provides the default transpiler between emerging versions of JavaScript and the version developers need to support based on their needs.

Babel, the JavaScript transpiler for converting source code from emerging versions of JavaScript to sets of language features supported in production environments.

React has emerged as the most widely adopted framework since jQuery, and continues to evolve; new projects get announced almost daily with support for React.

Node.js is a widely adopted server-side and command-line environment for JavaScript, and NPM is the most commonly used Node.js package manager. Nearly every JavaScript project leverages Node.js for its command-line and build tooling, and Node.js has received significant adoption

for server-side development as well as within embedded systems.

Express is also on our list, as the most widely adopted Node.js-based server-side framework. Webpack makes our list too as the defacto build and bundling tool, also based on Node.js.

We have added HTML5 and ES6/2015 as late majority, as nearly every web application today uses these features as a minimum baseline.

Laggards

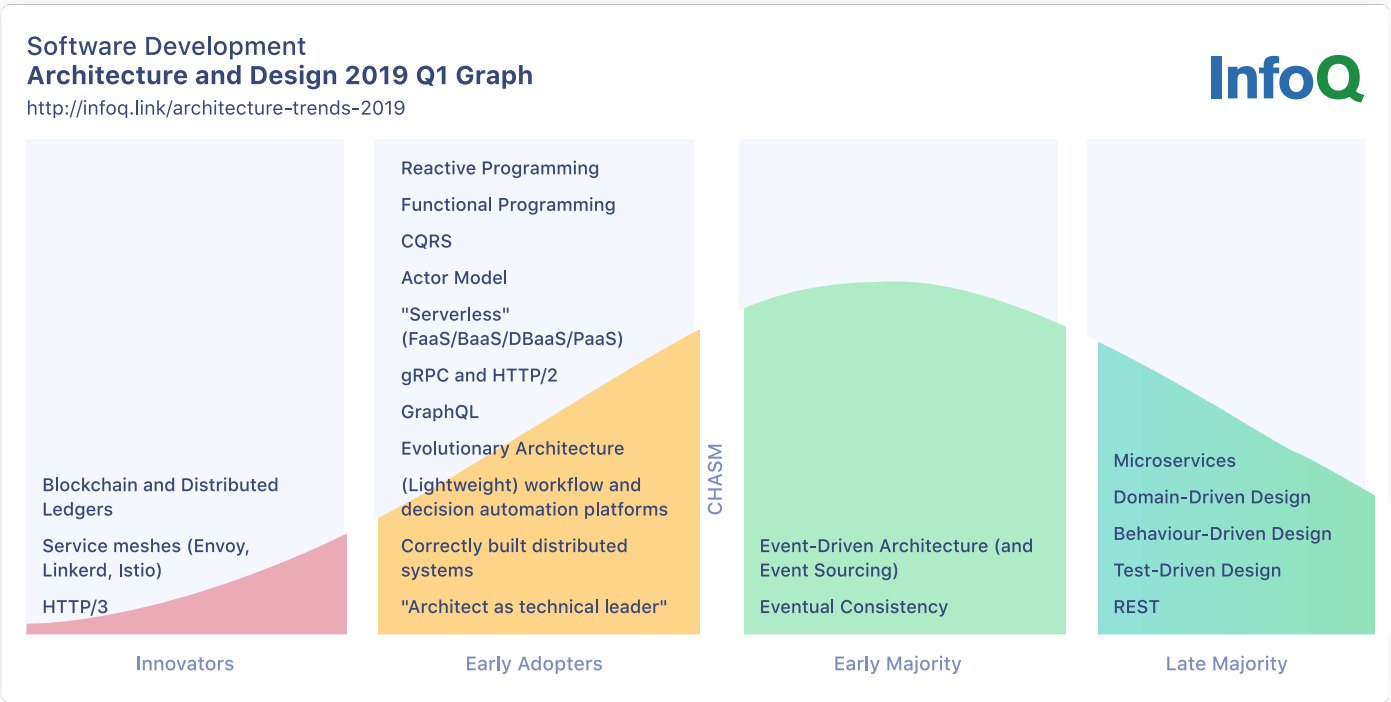
We conclude our report with laggards, which while still used in many applications, do not receive significant interest for new development efforts. Newer versions have supplanted ES5 and HTML4. Dojo 1.x and Angular 1.x, while still widely deployed, are in maintenance mode. jQuery, while receiving some improvements, has mostly been replaced by new HTML and JavaScript features or more full-fledged frameworks.

Conclusions

The JavaScript ecosystem is thriving, and while it may be challenging to keep up with the rate of change, we believe that many foundational pieces within this domain have become stable, which is creating additional areas for innovation. The team at InfoQ is here to provide you with expert coverage of the JavaScript landscape, and we welcome feedback, comments, and external article contributions.

TL;DR

- The rate of new JavaScript, CSS, HTML, and WebAssembly standards is accelerating to cover all facets of modern application development.
- TypeScript has had a dramatic rise in popularity, now in the top 10 most popular programming languages on GitHub, with many frameworks and libraries covered by InfoQ making the switch this year.
- WebVR/WebXR and machine learning and pushing the drive towards better 3D rendering and data visualizations
- React adoption has greatly increased, but a new wave of frameworks are gaining significant usage including Vue.js
- A wide range of options for applications, from Progressive Web Apps, to hybrids like Electron, Ionic, and Cordova, to full native compilers like React Native and NativeScript and providing increasing flexibility for creating competitive applications using web technologies.



InfoQ Trends Report - January 2019

Architecture and Design

Both InfoQ and QCon focus on topics that we believe fall into the “innovator, early adopter, and early majority stages”. What we try to do is identify ideas that fit into what Geoffrey Moore referred to as the early market, where “the customer base is made up of technology enthusiasts and visionaries who are looking to get ahead of either an opportunity or a looming problem”. We are also on the lookout for ideas that are likely to “cross the chasm” to broader adoption. It is perhaps worth saying, in this context, that a technology’s exact position on the adoption curve can vary. As an example, microservices is widely adopted at this point amongst Bay Area companies, but may be less widely adopted, and perhaps less appropriate, elsewhere.

This article provides a summary of how we currently see the “architecture and design” (A&D) space, which focuses on fundamental architectural patterns,

realisation of patterns in technology frameworks, and the design processes and skills that a software architect must cultivate.

Notable changes since we last reviewed this topic include “microservices” moving into the late majority, but our internal discussions also highlighted that the closely related themes of “correctly designing distributed systems”, and designing for reactivity and fault tolerance are not so far along the adoption curve. We may well be approaching the bottom of the microservice “trough of disillusionment” in respect to the Gartner Hype Cycle.

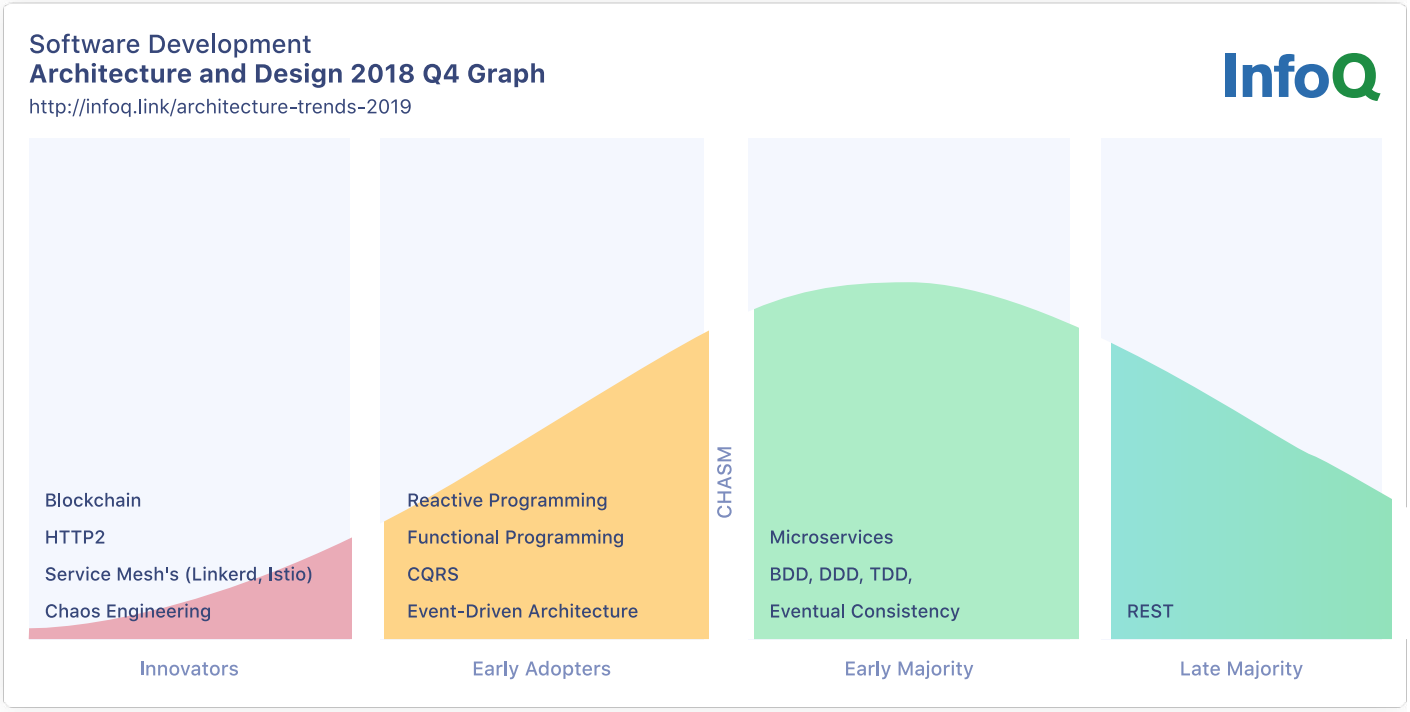
We have also speculated that there are areas of architecture that will never move along the adoption curve to early majority or late majority, and unfortunately, they include several highly-effective patterns -- such as event-sourcing/CQRS-based or Actor model-based systems -- which can provide highly effective solutions for certain organisations and business problems.

Although we feel the term “serverless” is slightly vague, we appreciate the potential to drive focus on designing modular, event-driven systems, and the possibility of automating several underlying operational platform concerns. On a related theme, we’re also seeing more discussion around the need for evolutionary architectures that support future changes in requirements and constraints.

We are increasingly seeing that the role of “architect” is becoming focused on soft skills such as technical leadership, in additional to modern technical skills like architectural pattern recognition and framework awareness, and dealing with the design of cross-cutting concerns.

For context you can see what the topic graph looked like for the second half of 2018 below. The 2019 version is at the top of the article.

The following is a lightly-edited copy of the corresponding internal chat-log between three of the InfoQ architecture and design topic editors, which provides for more context for our recommended positioning on the adoption graph.



Daniel Bryant, Independent Tech Consultant, Product Architect at Datawire, and InfoQ News Manager:

As a starter-for-ten, I'm thinking that HTTP2 moves to Early Adopter (EA), and HTTP3 comes in at innovator. GraphQL (and potentially gRPC) could be EA (or innovator?). I also think Chaos Engineer belongs in the DevOps queue. Microservices could be Late Majority (LM), as could BDD, DDD and TDD.

I would be keen to see "evolutionary architecture" appear somewhere -- potentially EA? And what about "Architect as technical leader" (emphasising the non-technical evolution of the role).

I'm interested in hearing what your thoughts are here, and also to ask if we need to move things around, or add or remove topics?

Jan Stenberg, IT consultant with more than 25 years in building systems on both .Net/C# and JVM/Java platforms:

I think that Architecture & Design (AD) to some extent differs from the other topics we cover at InfoQ.

In AD we don't have a regular base of new or updated versions of architectures coming up. Instead existing ideas get popular again, and perhaps packaged and branded, due to new tooling, frameworks or smart architects that make them feasible.

We also have areas that may fit in two queues. At a higher level they may be suitable for AD and the more technical parts to another queue. I think that Serverless is an example of that, at a higher level it's an important area for AD, the more technical parts may belong to the Cloud queue. Micro frontends and similar techniques are another example, is it AD or HTML5 and JavaScript?

There are also areas or architecture that I think will never end up in EM or LM, and unfortunately, they include several of my favourite architectures like event-sourced/CQRS-based or Actor model-based systems. I think they for the foreseeable future will be niche architectures used by a few. I'm not sure how we should deal with those, maybe they just fade away when architects and developers stop talking about them?

So, my take on the AD future (or my hopes maybe):

- Serverless. From the presentations I've listened to the last year, my impression is that this will be more and more automatic, with less work on the underlying infrastructure.
- Workflow platforms like Camunda. I think they are very important in microservices or distributed systems with more complex business logic.
- Event-sourcing/CQRS. I hope it will become more mainstream. Probably EA or EM.
- Event-driven architectures. EA or maybe EM.
- Actor Model / Reactive. Last year I talked to Vaughn Vernon about this and he believed it will become mainstream someday but I'm sceptical.
- Evolutionary architecture is interesting, and I think EA is correct.
- Chaos Engineering. Yes, normally it's DevOps, a presentation discussing the subject from an AD perspective might be an exception.
- GraphQL and similar tooling is I or EA I think, replacing REST (hopefully also properly implemented).
- Architect as technical leader. I've been in meetings with various architects at home and for many of them their main work is getting the business/government domain experts to understand their own domain. But maybe it's then more of an Agile queue story?
- Microservices is LM. (I think that Microservices soon will

be “SOA of today”. Many are using them in a good way. Too many have put the label on a distributed monolith).

- DDD is late majority, but I hope it still is an interesting subject for InfoQ.
- BDD is late majority or rather “late minority”
- TDD still has some more or less interesting discussions. Too little or too much, unit tests or black box tests, ice-cream cone or what, but LM, at least.

When I meet architects, developers, domain experts and others in daily life, not at conferences and similar events, I too often realize that many of the concepts we are talking about here are unknown or very diffuse for them, which also make it hard for them to see the benefits of InfoQ. I remember a presentation I listened to about two years ago from a developer conference (I think it was in Canada) where Vaughn Vernon asked how many that knew about DDD and about half of the audience raised their hand.

When I started to write for InfoQ I wrote a fair bit about updates of frameworks and libraries that I thought added functionality that could influence architecture, but with time my writing has more and more focused on interesting blog posts and presentations, with just a few items about frameworks like Axon, Akka, and others that I think are closely related to a specific architecture.

This kind of discussion would be great to have during a QCon conference.

Charles Humble, Editor in Chief at InfoQ:

I’m with Vaughn Vernon with regards the actor model - and think it is likely to become mainstream - either directly or through something like messaging. In the JVM space Akka is doing a good job of popularising the approach, and messaging-based systems have been a popular way of doing something like actor-like in financial systems for a long time.

Actors seem to be easy for people to grasp and reason about, and a good way of dealing with large-scale parallel work. I’d love to see momentum build up around Pony as an example of a modern, actor-based system, but I have to say I think that is unlikely.

With regards evolutionary architecture I was interested to hear Martin Fowler talk about this on the podcast last year and the paralel he drew to extreme programming. I’m looking forward to reading the Thoughtworks’ Book on this.

Thomas Betts, Principal Software Engineer at IHS Markit and InfoQ Architecture Queue Lead:

At a very high level, I agree with most of what Daniel proposed. And Jan is correct that some of the architectural patterns fit well into the natural progression of the topic graph, while others will probably never move beyond the early adopter phase because they aren’t widely applicable.

I do sometimes struggle with the overlap between A&D and the other topics on InfoQ, especially Culture & Methods. Blame Conway’s Law, I suppose. So much about an architecture comes down to communication -- What are the external communication points entering and leaving my system? How will my internal services communicate with each other? How will my data be persisted and accessed?

In many ways, a company’s way of answering these questions, and the options they are able to choose from, will be based on their place on the adoption life cycle curve for both A&D and C&M. I’d like to say A&D is the technical side of the equation, and C&M is the non-technical, but that seems almost too simplistic. Also, the technical implementationwould probably fall into the development and/or language queues. A&D is

in the squishy place in between, dealing with cross-cutting concerns, hopefully providing direction on how to implement the system.

I'll stop the philosophical rant, and just add a few specific discussion points.

Serverless - While I personally dislike the term, because it doesn't seem to mean anything specific, serverless needs to be represented, probably in EA.

Reactive - Probably EA. I think reactive architectureswill become more common because developers are getting familiar with reactive programming, especially in JavaScript. That might be the tail wagging the dog.

DDD - While DDD itself can probably move to LM, there are a lot of spin-off ideas that get closely associated with DDD, and are in I or EA. For example, Event Sourcing could merit mentioning as EA/LM. However, I don't think many those sub-topics warrant inclusion on the AD topic graph.

Microservices - Right up there with "serverless" as a commonly misused or misunderstood term. I see this as moving into late majority in terms of widespread adoption, but might only be EA for a solid distributed architecture.

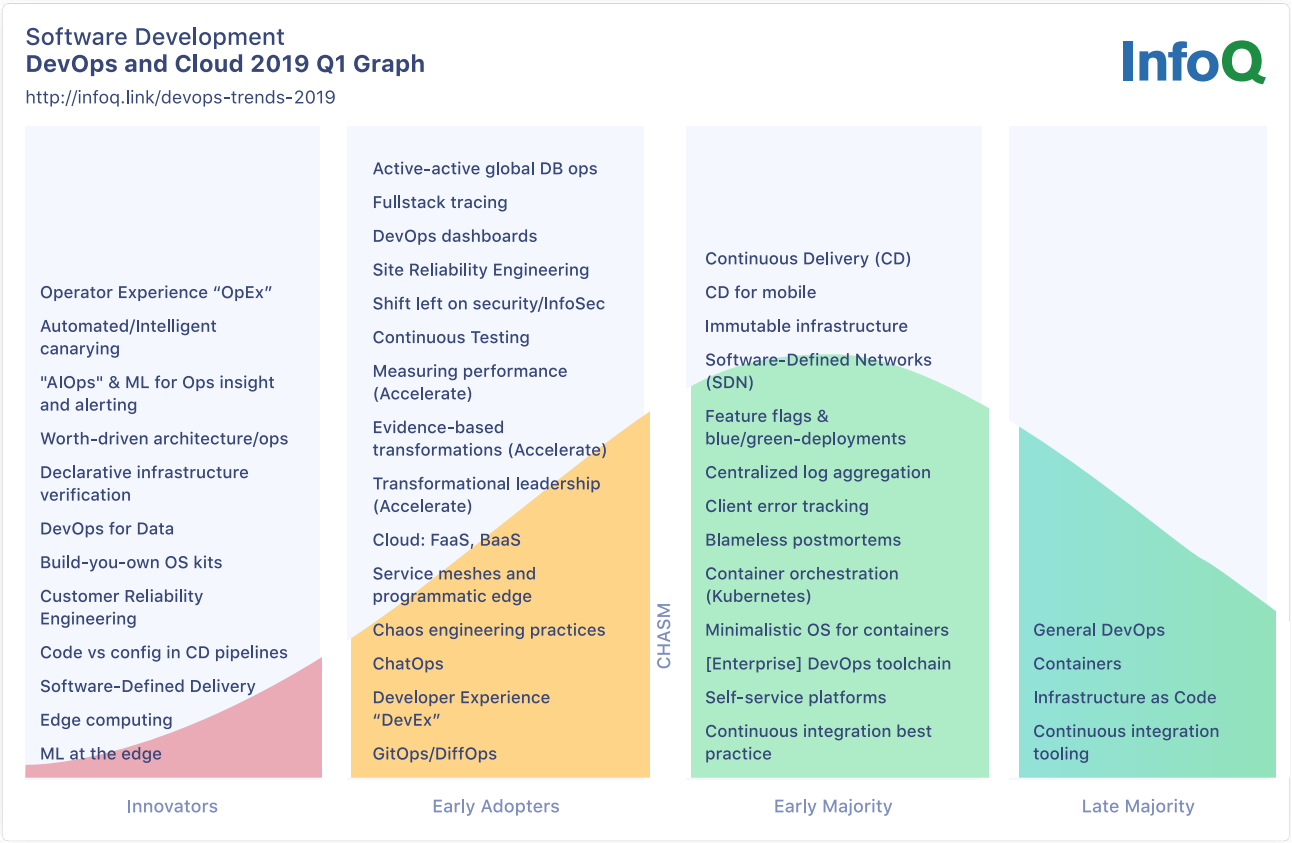
Distributed Systems - I don't think it would work to have this as an item on the topic graph, because it's too broad. But I'd like to see us talking about designing with distribution in mind. Ideas like reactive and failure tolerance are critical to a robust distributed system in ways that weren't important in a monolith. That would be the argument for leaving chaos on the A&D topic graph.

I fully support having this discussion at a QCon!

The InfoQ editorial team is built by recruiting and training expert practitioners to write news items and articles and to analyse current and future trends. Apply to become an editor via the editor page, and get involved with the conversation.

TL;DR

- We're seeing more need for the design of "evolutionary architectures", which builds on previous discussions around designing for replaceability and the need of focusing on the "glue" components within an architecture. An evolutionary architecture supports future changes in both functional and cross-functional requirements and constraints.
- Awareness of the "microservices" architectural style may be moving into the late majority, but the related themes of "correctly designing distributed systems", and designing for reactivity and fault tolerance are not so far along the adoption curve
- We speculate that there are architectural topics that will never move along the adoption curve to early majority or late majority, and unfortunately, they include several highly-effective patterns for specific use cases, like event-sourced/CQRS-based or Actor model-based systems.
- We increasingly see the role of "architect" as becoming more focused on technical leadership, architectural pattern recognition and framework awareness, and dealing with the design of cross-cutting concerns.



InfoQ Trends Report - February 2019

DevOps and Cloud

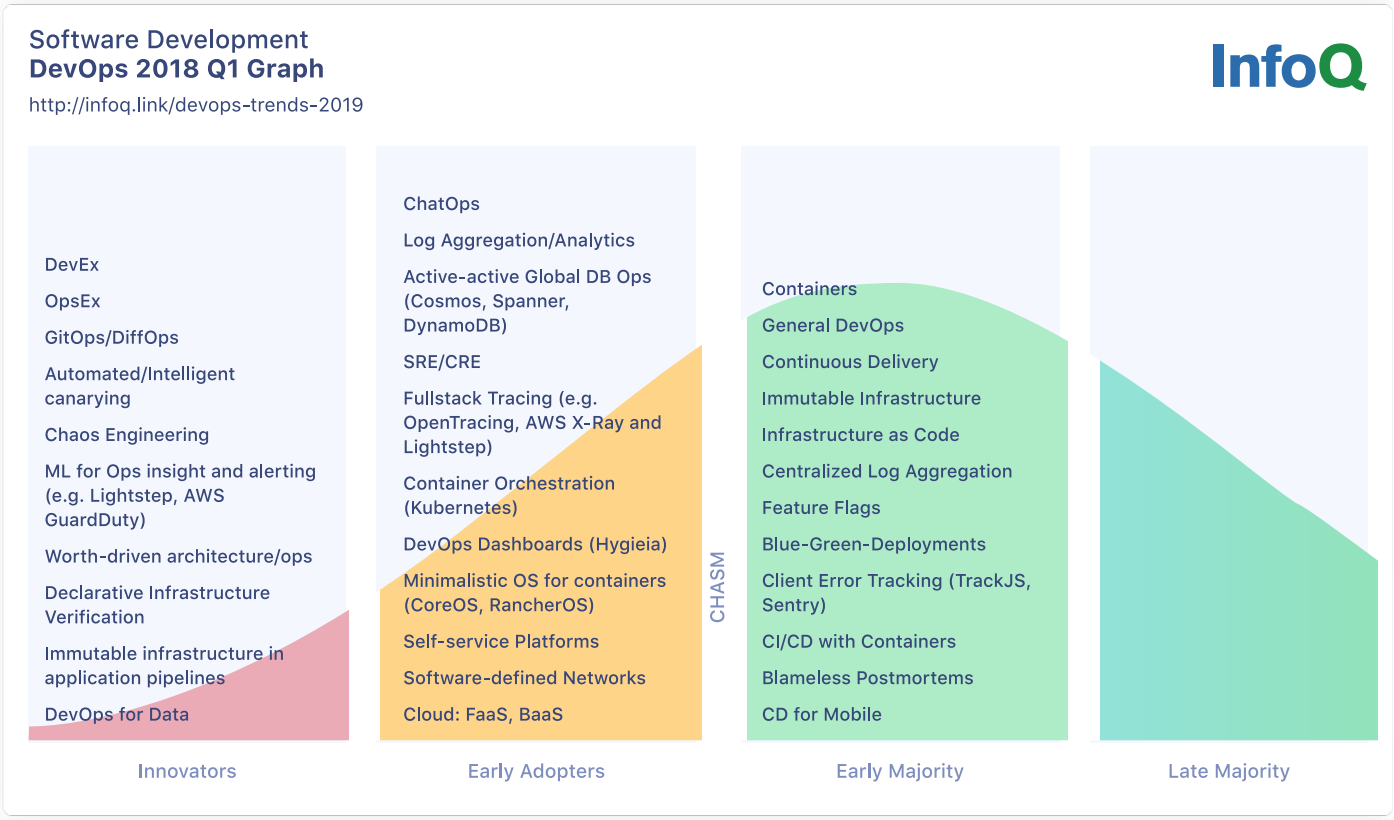
Both InfoQ and QCon focus on topics that we believe fall into the "innovator, early adopter, and early majority stages". What we try to do is identify ideas that fit into what Geoffrey Moore referred to as the early market, where "the customer base is made up of technology enthusiasts and visionaries who are looking to get ahead of either an opportunity or a looming problem". We are also on the lookout for ideas that are likely to "cross the chasm" to broader adoption. It is perhaps worth saying, in this context, that a technology's exact position on the adoption curve can vary. As an example, microservices is widely adopted at this point amongst Bay Area companies, but may be less widely adopted, and perhaps less appropriate, elsewhere.

This article provides a summary of how we currently see the "cloud computing and DevOps" space, which focuses on fundamental architectural patterns,

realisation of patterns in technology frameworks, and the design processes and skills that a software engineer must cultivate.

In this edition of the cloud computing and DevOps trend report we believe that Kubernetes has well and truly cornered the domain of container orchestration, and is arguably becoming the default cloud-agnostic compute abstraction. However, Kubernetes is not a complete Platform-as-a-Service (PaaS), which is arguably what most organisations require for effective deployment and operation of software, and so the next "hot topics" in this space appear to be "service meshes" for managing interservice communication and release control, and developer experience and workflow tooling to allow engineers to implement effective dev-test-deploy-observe cycles.

We believe that the topic of chaos engineering has moved into early adopter, largely due to the increased promotion by the Netflix team and the O'Reilly Chaos Engineering book authors, and tooling such as the Chaos Toolkit and Gremlin's as-a-service offerings. Based on discussions with John Allspaw, Casey Rosenthal, Nora Jones and other members of the community, we have separated out the topic of "resilience engineering", which we have previously conflated with chaos engineering, and placed this within the innovator category.



We are following the developments with edge computing, ML inference at the edge, and "originless" computing with keen interest. Work here is mostly within the innovator space, but the public cloud vendors and innovators like Cloudflare are clearly investing large amounts of resources here.

The book "Accelerate" by Nicole Forsgren, Jez Humble and Gene Kim, has helped to define several topics within the domains of digital transformation and high-performing teams. Based on discussions with external contributors we are now tracking with interest topics such as "measuring [for high] performance", evidence-based transformations, and transformational leadership. We see these as early adopter, and are also keen to publish more articles within this space on InfoQ.

Although a potentially overloaded buzzword, we are tracking "AIOps" within the innovator space, and more specifically the potential value offered by ML for operational insight and alerting. As the software systems being built become more complex and distributed, this type of technology could complement distributed (fullstack) tracing for problem detection and help reduce the search space during fault isolation.

We see that there is a danger with conflating continuous integration (CI) and continuous delivery (CD), and also lumping practice and tooling together in the same category -- indeed we acknowledge that our own published articles and associated article topic tagging is far from perfect within these domains. We see applications/services that are being marketed as CI tooling as within the late majority adoption phase, but CI best practices are still within the early majority phase. We also believe that CD is within the early majority category.

We are keenly following recent work around the software-defined delivery manifesto and "code vs config" for delivery and pipeline automation closely. The Atomist and Pivotal teams are currently driving a lot of the conversation (and technology) within this space.

For context you can see what our graph looked like for the first half of 2018 on the previous page. The 2019 version is at the top of the article.

The following is a lightly-edited copy of the corresponding internal chat-log between several of the InfoQ cloud computing and DevOps topic editors, which provides for more context for our recommended positioning on the adoption graph.

Daniel Bryant, Independent Tech Consultant, Product Architect at Datawire, and InfoQ News Manager:

As a starter-for-ten, I'm thinking that Chaos Engineering is moving to Early Adopter (EA), what with the Gremlin and AWS teams' talking a lot about this as a re-imagining/extension to traditional DR/BC; I also think that the minimalist container images move to Early Majority (EM) -- what with build-your-own kits like LinuxKit entering in Innovator -- distributed tracing potentially to EM, general DevOps to Late Majority (LM), and potentially containers, IaC and continuous delivery to LM?

We need to put service meshes in here somewhere, as they are a super-hot topic, and I'm thinking EA?

I'm interested in hearing what your thoughts are please, and also to ask if we need to move things around, or add or remove topics?

Helen Beal, DevOps consultant, coach, trainer, speaker and writer.

My thoughts are:

- CI/CD to late majority as you say
- Add AIOps to Innovator along with ML for Ops?
- Should 'DevOps Toolchain' be there? In early majority?
- Add Electric Cloud and Xebia Labs to DevOps Dashboard
- add AI to dashboards?
- ChatOps to early majority

Steffen Opel, is managing partner at Utoolity

Agreed on moving DevOps, containers, IaC, and CD to LM; everyone not adopting this will be left behind quickly. Adding service meshes to EA sounds good as well.

I'd also join the proposal to add AIOps as a topic to innovator, though I see it more as the generalized superset of and thus replacement for the current "ML for Ops insight and alerting topic" (which also covers other areas like predictive scaling for example, which has been recently released for AWS EC2).

However, while I see further increasing interest in both ChatOps and fullstack tracing, I don't think their adoption qualifies for EM already (at least in terms of "established way of handling most/all of your operations"), and tooling maturity around ChatOps leaves a lot to be desired still. On the other hand, "Log aggregation/analytics" seems to have surpassed that threshold (complementing "Centralized log aggregation", maybe these topics can be merged now).

Finally, I'm still missing 'edge computing' as an innovator/early adopter topic (e.g. services like Lambda@Edge, AWS Snowball Edge, AWS Greengrass, Cloudflare Workers) - these capabilities are quickly becoming accessible once provided by already widely adopted cloud services like CloudFront, Lambda, or just recently Cloudflare.

Chris Swan, Fellow, VP, CTO for the Global Delivery at DXC Technology

I'd go along with what Daniel, Helen and Steffen have already said.

What I'm not seeing here is the swing to code vs config for CD per Atomist etc. and the Software Defined Delivery (SDD) manifesto.

On the edge topic that Steffen highlighted I also wonder if 'originless' will turn into a thing? It's at that awkward stage right now where it's hard to tell whether the early demos are pointless parlour tricks or the beginning of a new trend (but when combined with IoT and the quest for lower UI latency I'm inclined to give it some credence).

One nitpick - I know people lump together CI/CD because they're often done with some of the same tools, but as we all know too well they're not remotely the same thing. CI is definitely in the late majority now, because it's (relatively) easy to do in a traditional organisation. CD is I think struggling to get to late majority because it doesn't happen without that Dev/Ops to DevOps reorganization, and many companies simply haven't got there yet. The innovation gap that we're used to early on in the adoption curve is actually a giant canyon now later along that curve because adopting CD isn't just a technology pick.

Helen Beal

I think that's a really important clarification Chris et al.

Manuel Pais, independent DevOps and Continuous Delivery Consultant

On the CI/CD discussion, my concern is that we're conflating tooling and practices which are at different stages. CI/CD tooling I agree is moving to late majority, but only some of the practices (as defined in the CI and CD books) are commonly applied. Many organizations are not practicing CI as such (a recent Jenkins survey clearly shows a lot less builds are performed than you'd expected when practicing CI), they are simply using a CI server. That's illustrated also by Dave Farley's rant earlier this year on (not) branching.

(note: I'm guilty of conflating them as well, but it wasn't such a clear issue when the topic was in early majority)

If we split between tools and practices, I'd put tools in late majority and practices (still) in early majority. I also suggest mapping some of the practices in the "Accelerate" book on the topics page, that makes for a very good reference/framework on important practices for high performing teams.

As for other changes here are my thoughts:

- DevEx -> move to EA - being pushed by Atomist as a common concern
- GitOps -> move to EA - being pushed forward by Weave
- Chaos Eng -> move to EA - being pushed forward by Gremlin
- AIOps (what we used to refer as "ML for Ops insight and alerting") -> still in Innovator
- Immutable infrastructure in application pipelines -> move to EA
- Log aggregation/analytics -> move to EM
- SRE/CRE -> this needs to be split up, Google is still figuring out CRE (should be moved to Innovator) while SRE should move to EM
- K8s -> move to EM
- Self-service platforms -> definitely move to EM
- Software-defined networks -> I'm not an expert here but it seems to me to be in EM
- Containers/General DevOps/ infrastructure as code -> move to LM
- centralized log aggregation -> duplicate of "Log aggregation" (see above, should be single entry in EM)

I would also add this new topic in Innovator:

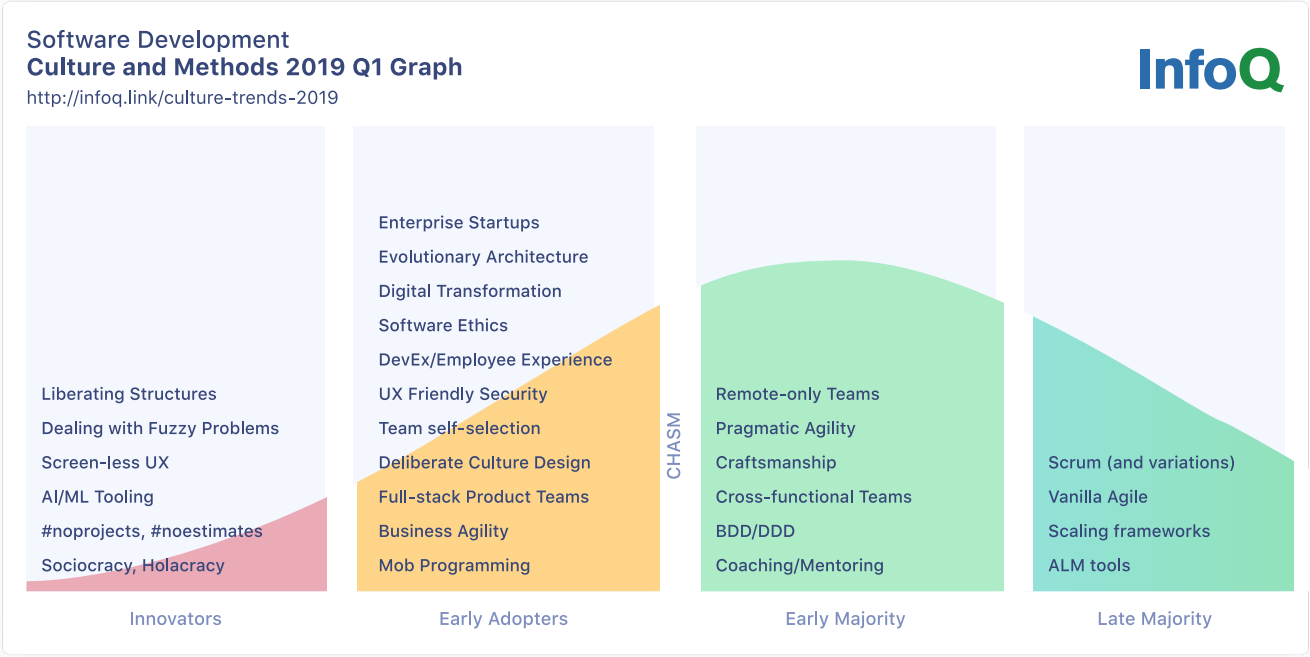
software defined delivery -> being pushed forward by Atomist and the SDD manifesto

Daniel Bryant

Thanks all for the excellent discussion -- I'll be sure to summarise this for the topic report. The split between the maturity of CI and CD (and the associated next level split between practices and tooling) is especially interesting to me, as is something we should probably focus on in regard to our own tagging of topics.

I also like Manuel's suggestion of incorporating key ideas that are gaining adoption from the great work of the "Accelerate" book from Forsgren, Humble and Kim into this edition of the topic graph. This book is rapidly becoming a go-to reference within the industry, as are several other books by the same authors that Manurel recently recommended in the InfoQ editors list of recommended reading article series.

The InfoQ editorial team is built by recruiting and training expert practitioners to write news items and articles and to analyse current and future trends. Apply to become an editor via the editor page, and get involved with the conversation.



The State of Practice in 2019

Culture & Methods

Every year at InfoQ we update our topic graph to show where we think different topics are in the technology adoption curve. When we do so we look at the state of practice, emerging ideas and information we hear on the various grapevines the editorial team is tapped into.

If a topic is on the right-hand part of the graph, you will probably find lots of existing content on InfoQ about it – we covered it when it was new, and the lessons learned by the innovators and early adopters are available to help guide individuals, teams and organizations as they adopt these ideas and practices.

The techniques and practices on the left-hand side are the ones we see as emerging now, being used by innovators and early adopters. We focus our reporting and content on bringing these ideas to our readers' attention so that they can decide if they want to explore (some of) these now or wait to see how they unfold.

There has been some movement of ideas from Innovator to Early Adopters, but none of the Early Adopter ideas from 2018

have jumped the chasm to Early Majority. Many of the organizational challenges which existed at the beginning of 2018 are still present, and they are slowing the adoption of innovator practices in the majority of organizations. We still see too much emphasis on changing the name of practices rather than genuinely changing organizational behavior and culture. For example, a daily status meeting where everyone accounts for every minute of their day to a project manager is not the Daily Standup prescribed in the Scrum Guide.

It is only with cultural change that the promise of the new ways of working will be achieved. Many organizations are embarking on "Digital Transformation", and it is often the same organization which has undergone two or three "Agile Transformations" in the past without seeing the promised benefits. We believe that this is because the adoption is often implemented to "pay lip-service" to the idea, and is shallow rather than truly transformational.

According to the State of Agile report, agile software development has become a late majority approach; almost all software is now built using iterative and incremental approaches, and mainly using some derivation of Scrum or Kanban. The strong technical practices from eXtreme Programming are still the exception rather than the norm in early and late majority firms. As an industry, we may know how to build software better, but there isn't the appetite to truly empower the teams and make the organisational changes needed to actually achieve the outcomes that Innovators have shown is possible.

There are organizations who have achieved significant productivity, employee engagement, customer satisfaction and profitability improvements from adopting ideas, such as #NoProjects and DevEx, yet they still remain in the minority. Transformational change is hard.

So what are the practices which we (the Culture and Methods editorial team) feel are gaining traction, albeit slowly, and that our readers should be considering as 2019 starts:

Innovators

Acknowledging and accepting that there are fuzzy problems, which are hard to define and require completely different approaches to solving them is a really important starting point for changing the way organizations operate. The complexity and ambiguity in today's marketplaces make it crucial to adopt new approaches, building on agile, DevOps practices and technical craftsmanship to reduce cycle times and feedback loops to minutes rather than hours or days.

Liberating Structures provide a framework for working together in teams and organizations, they are explicitly designed to create safety and encourage innovation. Similar in concept to the Core Protocols, they are explicitly intended to grow humanistic cultures that enable high performance.

UX + SecOps = UX Friendly Security: "Shift Left Security" and SecOps are coming together with user experience (UX) to enable product-accessible user experiences that make security less jarring. Some organizations are getting benefit from dual-track practices.

Shift Left Security is becoming the norm – DevSecOps and variations are bringing compliance and security into the product team rather than approaching these as independent specializations.

Artificial Intelligence (AI) and Machine Learning (ML) will start to impact development teams and their development practices; teams are using insights from data to drive estimates and select fit for purpose features to work on more than they have in the past. There are a number of tools starting to make this easier. However, AI and ML will also force teams to look at different ways to leverage Agile techniques as it can break many existing practices.

The shift from project-based work to product lines (#noprojects) is starting to make inroads and is delivering value for innovators.

Early Adopters

DevEx and employee experience, in general, continues to be something that Innovators are exploring and applying, creating environments which reduce friction and enable people to be their most effective and most honest selves. Practices such as Mob Programming and Mob Testing are becoming more accepted, and the social aspects of technical work are being recognized as important for technical success. In addition to the ethical considerations, diversity is a source of competitive advantage, and diverse teams achieve better outcomes.

Organisation and culture design continue to be two important pillars for achieving more effective outcomes. The new models such as Teal Organisations, Sociocracy (and S3), Holacracy are starting to gain more traction, but still, have a long way to go. Along with this, there are moves towards shorter, more effective working hours in a few organizations.

Some organizations are enabling Enterprise Innovation through spinning-off new disruptive Start-Ups as a survival strategy. These 'lean'-startups have the advantage of a large parent to draw skills from and easy funding, and allow a safe ground for disruption.

With the rapid pace of change, Evolutionary Architecture is absolutely necessary for success, and implementing such an architecture often necessitates a cultural as well as a technical change.

We hope that ethics is something that has always mattered, and 2018 certainly saw questions about software ethics be much more frequently asked.

Business Agility is a topic which has gained traction as more and more organizations realize that they are actually in the software industry, irrespective of what products and services they produce.

Fully Fullstack Product Teams: T/π-shaped product creators have all the skills needed to go from product inception to deployment and support. Teams transition from technical teams to complete end-to-end product teams which have business knowledge, marketing expertise, technical development, UX, design, support competencies and any other skills needed to bring a product to market, respond to customer feedback and maintain it.

TL;DR

- Several new ideas came into view in 2018, but few from the previous year have managed to jump the chasm to Early Majority adoption.
- Many organizations are embarking on "Digital Transformations", often the same ones who have undergone two or three "Agile Transformations" in the past without seeing the promised benefits. We believe that this is because the adoption is often implemented "to pay lip-service" to the idea, and is shallow rather than truly transformational.
- Pragmatic approaches that mesh ideas from agile, DevOps, Machine Learning and Artificial Intelligence are appearing, and the methodology brand wars are diminishing.
- "Shift Left Security" is slowly becoming the norm – DevSecOps and variations are bringing compliance and security into the product team rather than seeing these as independent specializations.
- Organization and culture design continue to be two important pillars for achieving more effective outcomes. The new models such as Teal Organisations, Sociocracy (and S3), Holacracy are starting to gain more traction but still have a long way to go.

Curious about previous issues?

Working remotely is becoming routine. Our goal in this eMag is to help you do things better. We'll show how people all over the world are successfully facilitating complex conversations, remotely. We'll also share practical steps you can take right now, to upgrade the remote conversations that fill your working days.

To tame complexity and its effects, organizations need a structured, multi-pronged, human-focused approach, that: makes operations work sustainable, centers decisions around customer experience, uses continuous testing, and includes chaos engineering and system observability. In this eMag, we cover all of these topics to help you tame the complexity in your system.

"Before you are a leader, success is all about growing yourself. When you become a leader, success is all about growing others." – Jack Welch. For this eMag we've pulled together six articles from the InfoQ content that explore different aspects of what it takes to lead effectively in the technical world.

InfoQ