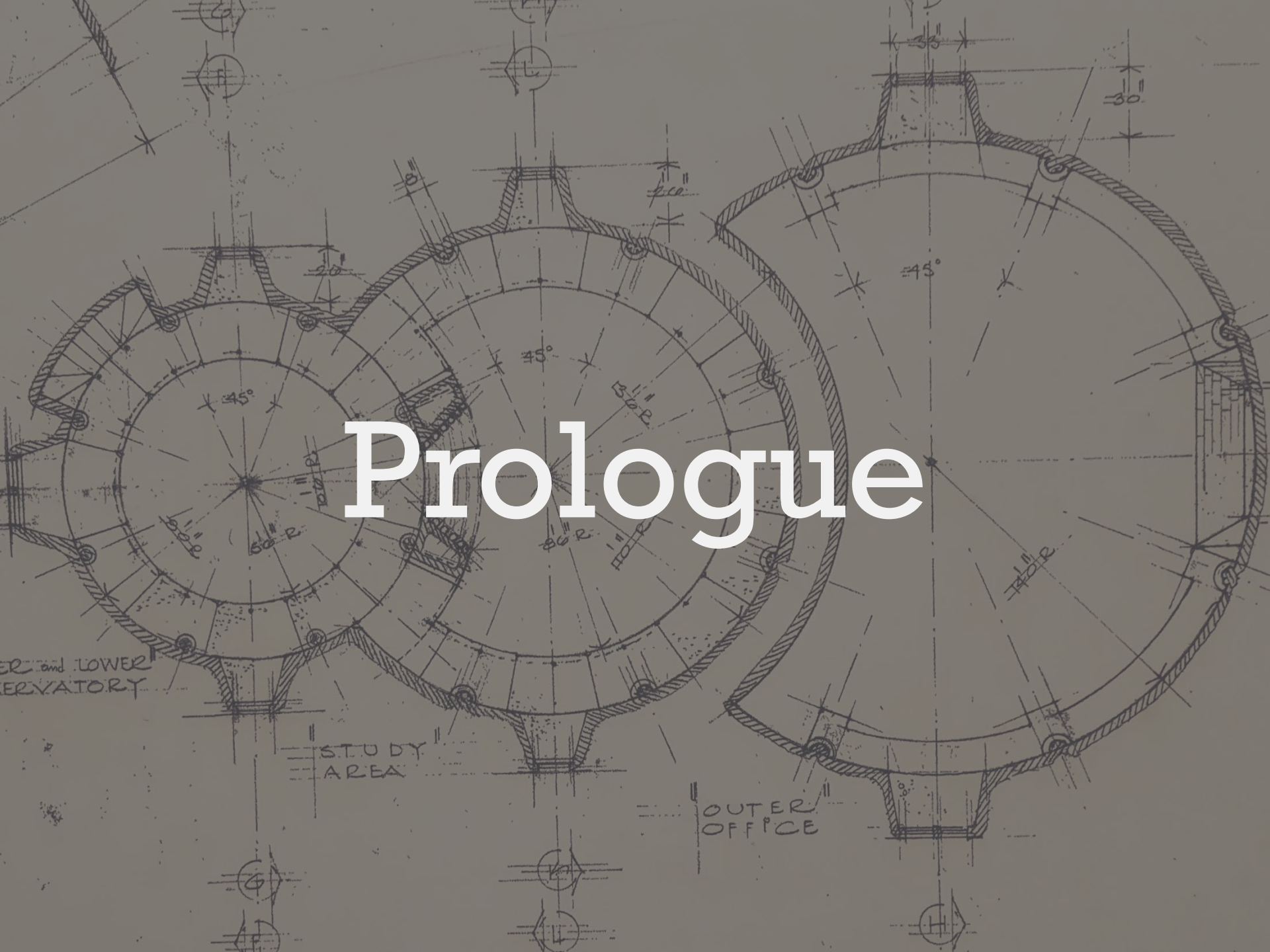
Software Architects Are Dead!
Long Live Software Architects!

Frank Buschmann
Kevlin Henney



Prologue

Software architecture is…

- **The highest level concept of a system in its environment**
- **The decisions that you wish you could get right early on**
- **The things that are hard and costly to change**
- **The important stuff… whatever that is**

If you think good architecture is expensive, try bad architecture.

Brian Foote & Joseph Yoder

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

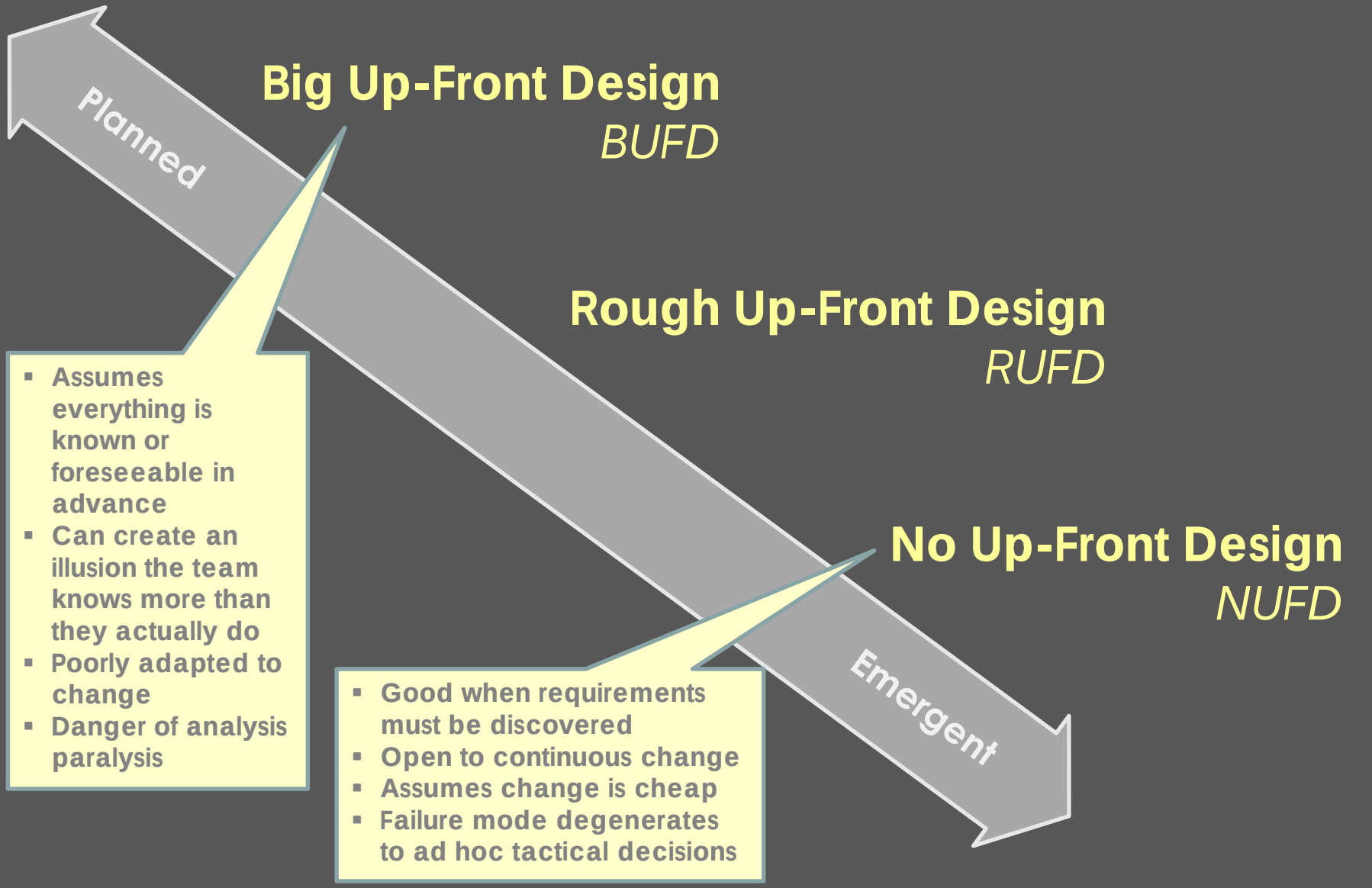
Manifesto for Agile Software Development

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

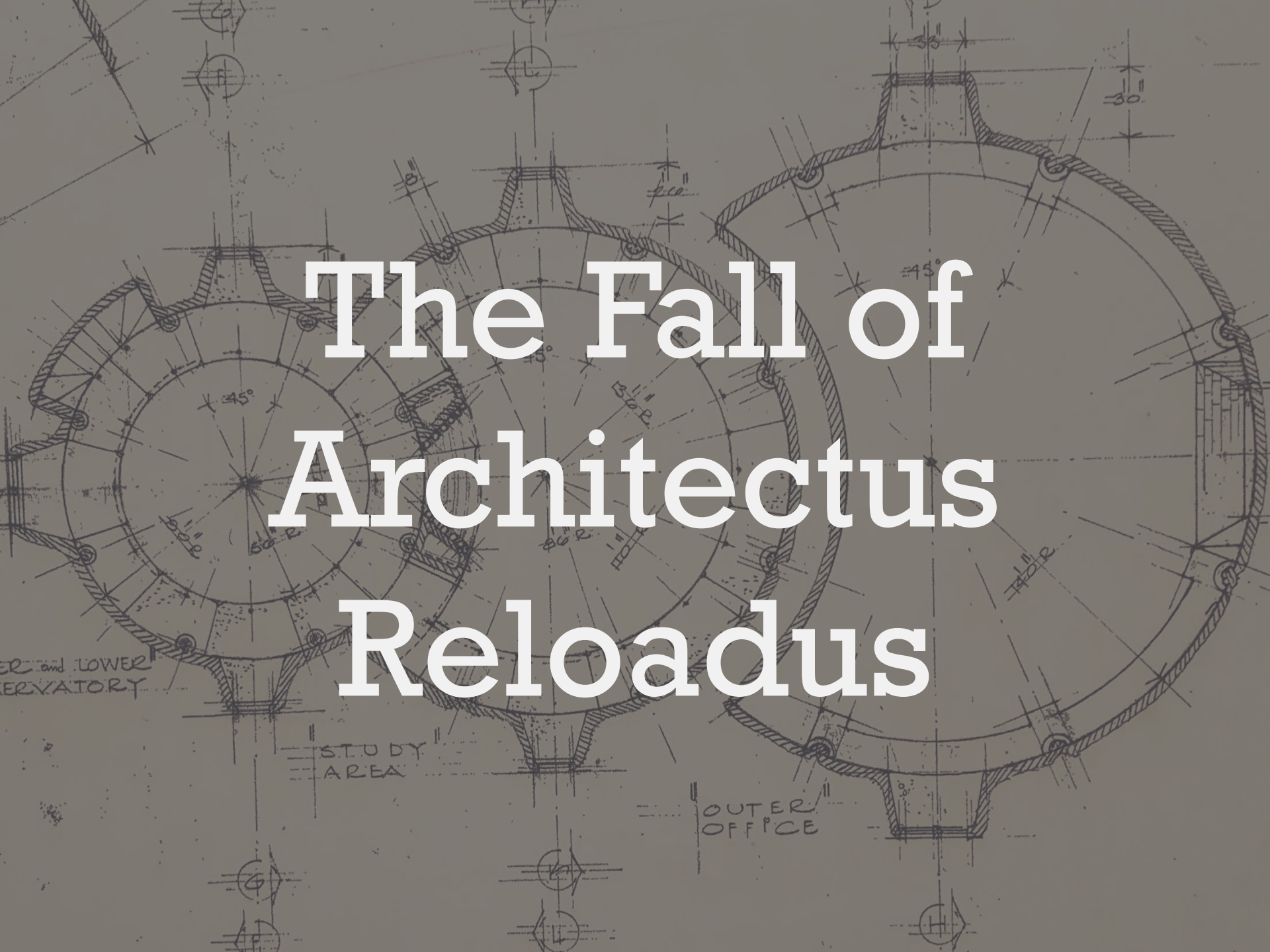
The best architectures, requirements, and designs emerge from self-organizing teams.

Approaches to Architecture Design

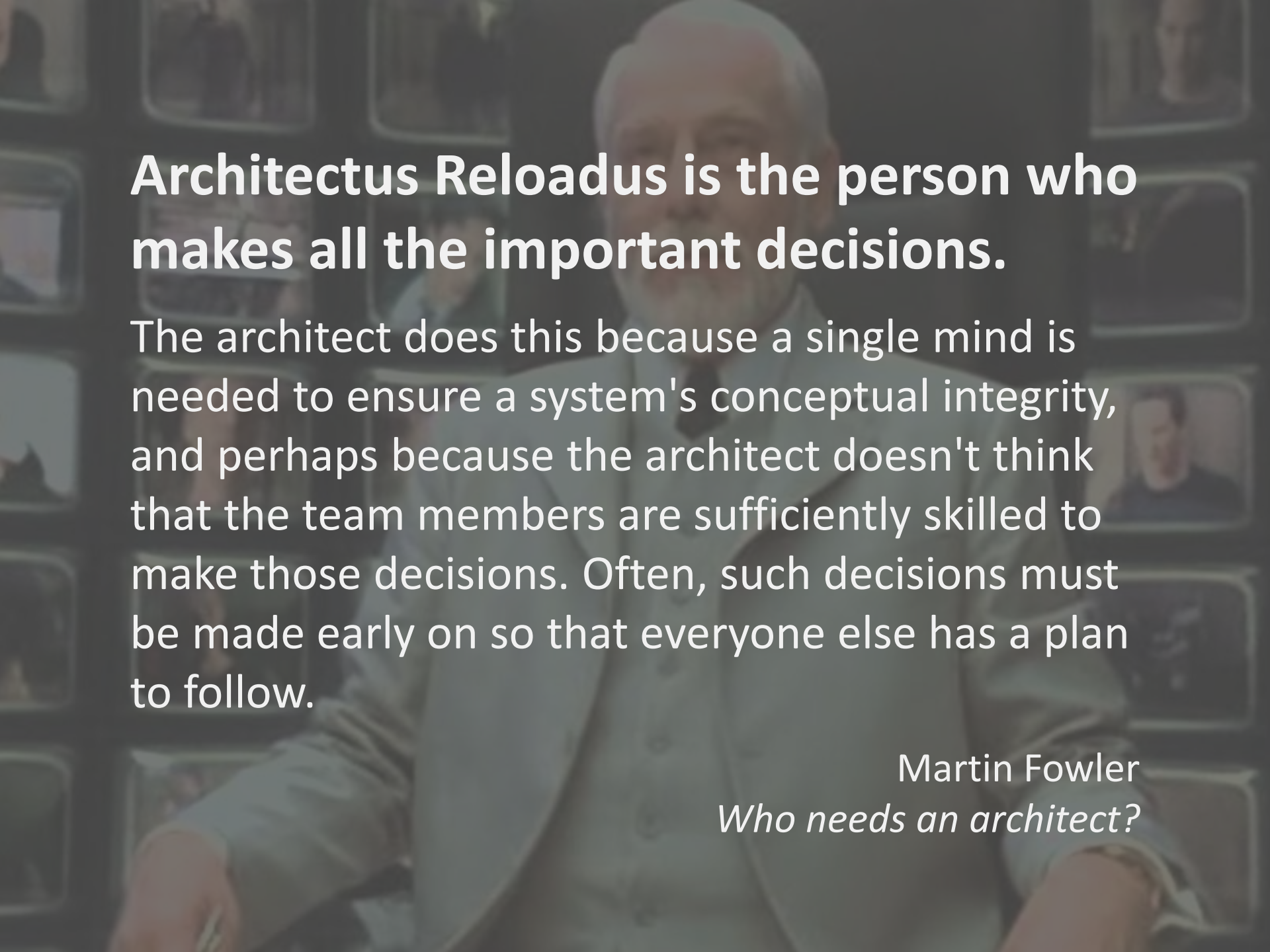



DevOps
Microservices
Scaled Agility
Internet of Things
Architecture in the Age of Digitalization
Self-Learning Systems
Cyber-Physical Systems
Self-organizing Systems
Autonomous Systems
Ecosystems


Quo Vadis
Software
Architects?



The Fall of Architectus Reloadus



Architectus Reloadus is the person who makes all the important decisions.

The architect does this because a single mind is needed to ensure a system's conceptual integrity, and perhaps because the architect doesn't think that the team members are sufficiently skilled to make those decisions. Often, such decisions must be made early on so that everyone else has a plan to follow.

Martin Fowler

Who needs an architect?

It's expensive to know everything up front.

Kolton Andrus

It's expensive to

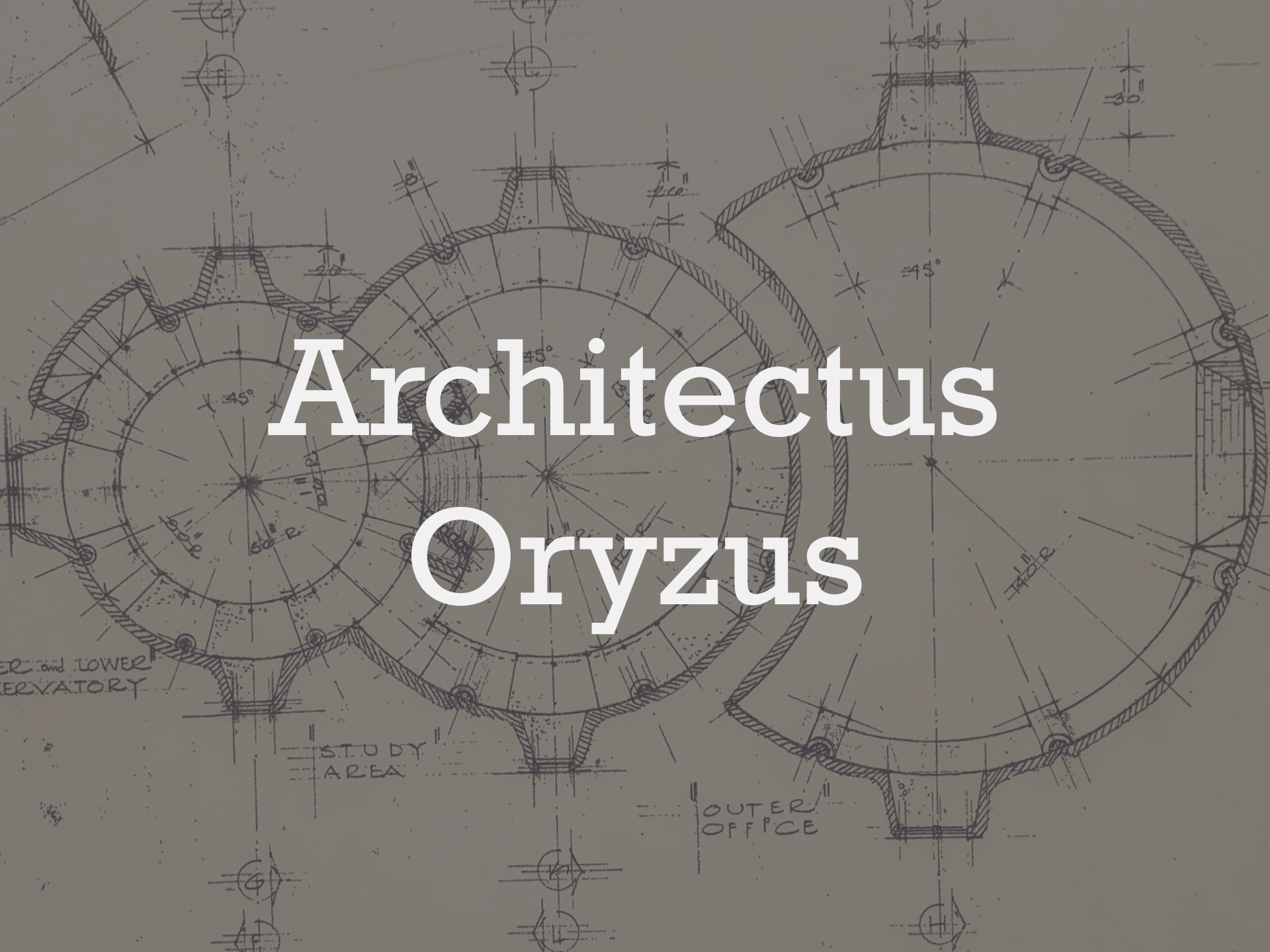
CHAOS RESOLUTION BY AGILE VERSUS WATERFALL

SIZE	METHOD	SUCCESSFUL	CHALLENGED	FAILED
All Size Projects	Agile	39%	52%	9%
	Waterfall	11%	60%	29%

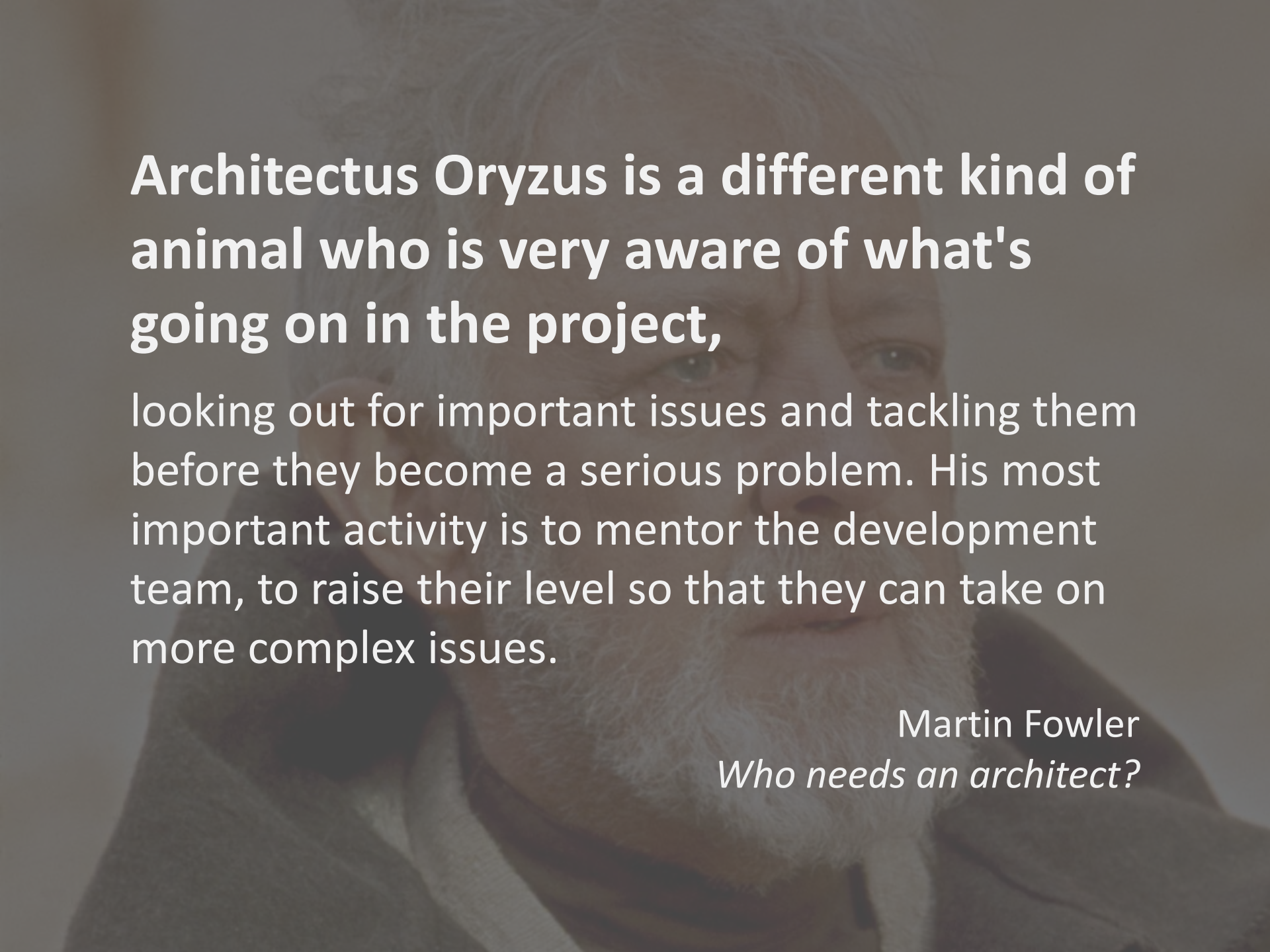
The resolution of all software projects from FY 2011 — 2015 within the new CHAOS database segmented by agile processes and the waterfall approach. The total number of software projects is over 10,000.

Source: Standish Group 2015 Chaos Report — Q&A with Jennifer Lynch (infoQ)

Kolton Andrus



Architectus Oryzus



Architectus Oryzus is a different kind of animal who is very aware of what's going on in the project,

looking out for important issues and tackling them before they become a serious problem. His most important activity is to mentor the development team, to raise their level so that they can take on more complex issues.

Martin Fowler

Who needs an architect?

Architecture is a hypothesis, that needs to be proven by implementation and measurement.

Tom Gilb

		COMPLEXITY				
		C1	C2	C3	C4	C5
SIZE	S1	100	250	400	550	700
	S2	175	325	475	625	775
	S3	250	400	550	700	850
	S4	325	475	625	775	625
	S5	400	550	700	850	1000

Low risk of failure

Medium risk of failure

High risk of failure

Source: Standish Group 2015 Chaos Report - Q&A with Jennifer Lynch (infoQ)

Fit for the digital future?

- Architectus Oryzus is well adapted for projects of small to medium complexity, and those with a developmental focus
- Yet even small projects can be complex, especially in an IoT or digitalization context
- Confounding complexities exist outside of development
- Complex projects more likely to fail
- Early failure and fast restart is not always an option
- Emergent design is not always fast enough, changes are not always cheap

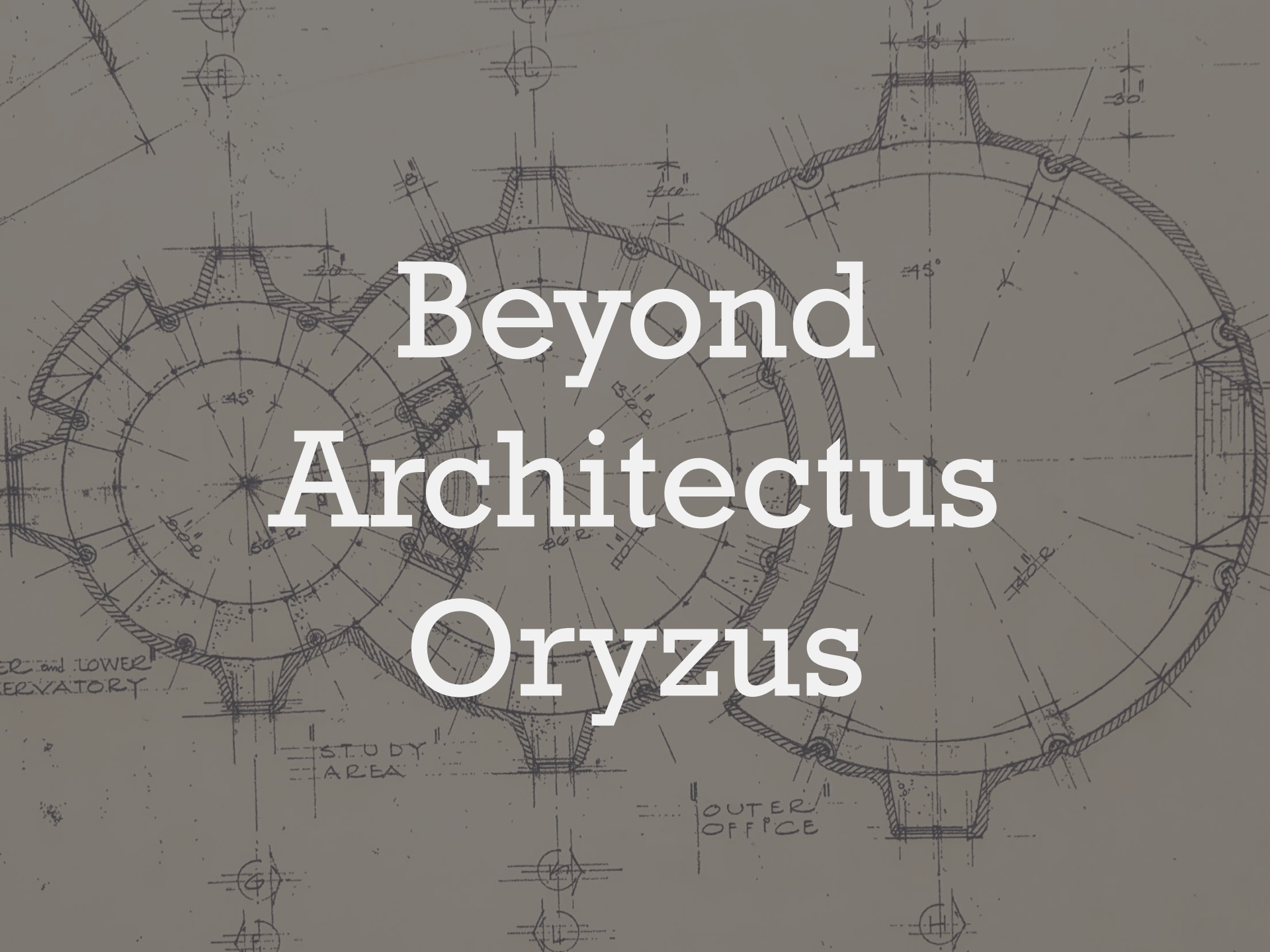
Architecture or not architecture?

- A financial services firm updated their high-speed, algorithmic router that sends orders into the market
- The software was manually deployed on 8 servers — deployment failed on one server
- Result was a system capable of sending automated, high-speed orders into the market without tracking to see if enough orders had been executed
- When going live, the software sent orders into the market resulting in 4 million transactions against 154 stocks for more than 397 million shares
- The firm realized a $460 million loss in 45 minutes... and went bankrupt

Source: *https://dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/*

What if such disaster happens here?

Source: Google



Beyond Architectus Oryzus

Microservices are becoming the predominant architectural style

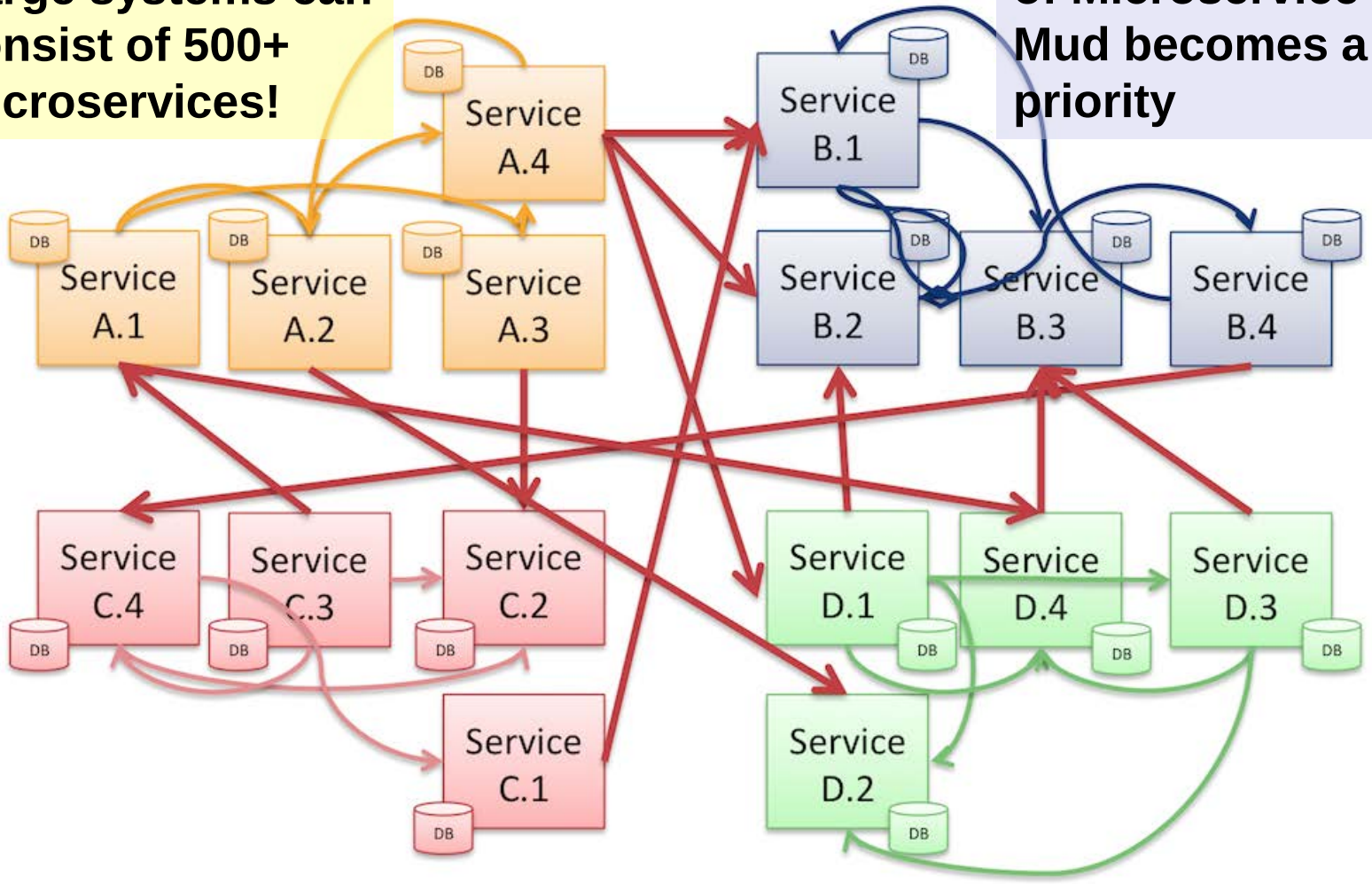
The Microservice architectural style is an approach to developing a single application as a suite of small, independently deployable services, each running in its own process and communicating with lightweight mechanisms.

While there is no precise definition of this architectural style, there are certain common characteristics around organization, business capability, automated deployments, intelligence in the endpoints, and decentralized control of languages and data

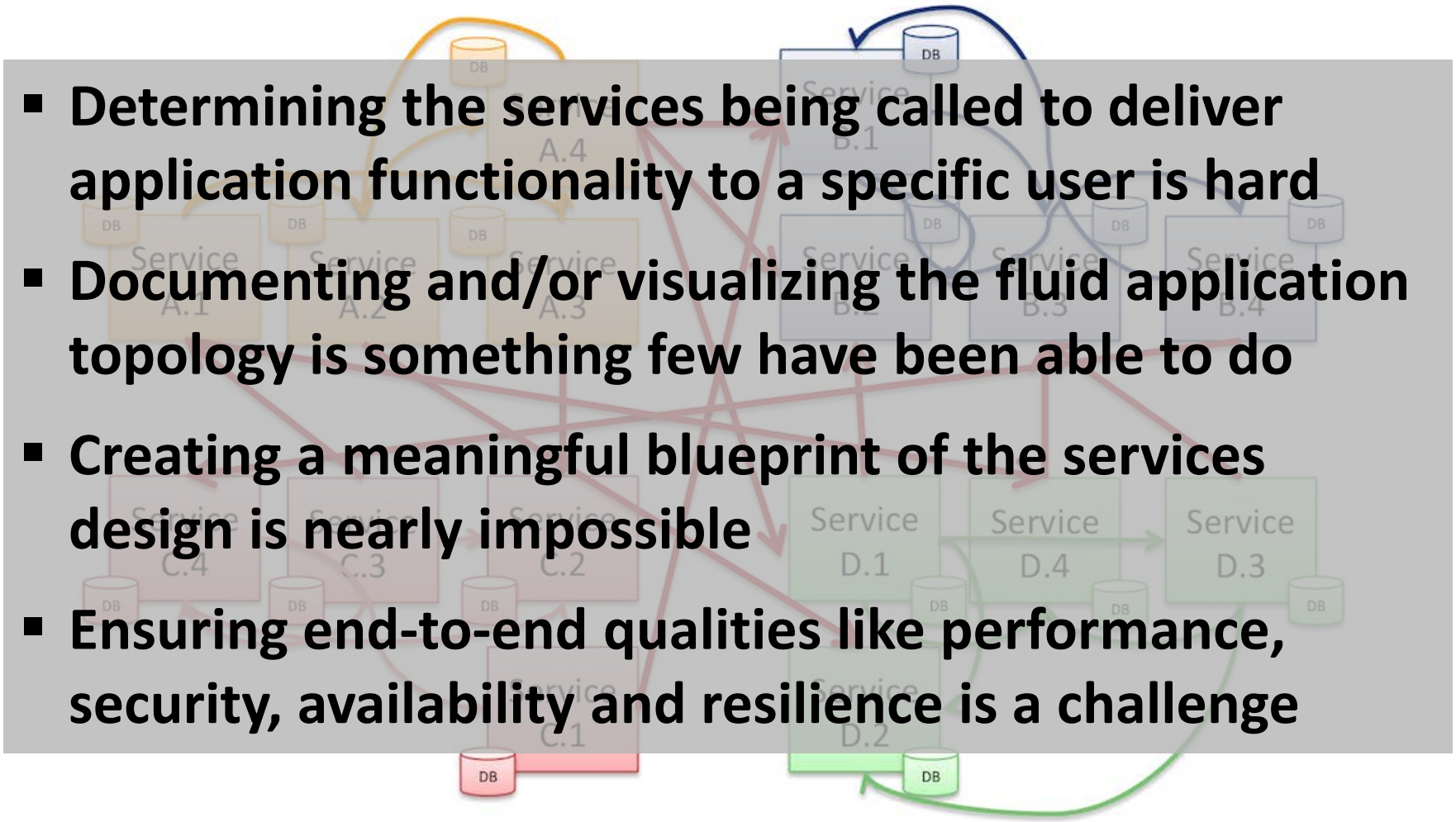
Martin Fowler

Large systems can consist of 500+ microservices!

Avoiding a Big Ball of Microservice Mud becomes a priority



From: https://blog.appdynamics.com/news/visualizing-and-tracking-your-microservices/



- **Determining the services being called to deliver application functionality to a specific user is hard**
- **Documenting and/or visualizing the fluid application topology is something few have been able to do**
- **Creating a meaningful blueprint of the services design is nearly impossible**
- **Ensuring end-to-end qualities like performance, security, availability and resilience is a challenge**

From: https://blog.appdynamics.com/news/visualizing-and-tracking-your-microservices/

Be Where Microservices Connect

The architect's main territory is between the services, where they meet, connect and hurt: Interfaces, Interactions, Integration

Interfaces	▪ Complete, meaningful, role-specific, usable ▪ Defined contract, managed evolution
Interaction	▪ End-to-end quality (reliable, fast, scalable, secure, …) ▪ Task-oriented
Integration	▪ UI integration, data management ▪ Versioning and release management

Deficiencies in interfaces, interactions and integration tend to show up late: during system test, roll out and operations – thus their resolution is costly!

You'll never walk alone

… in the end, the maximum customer value is going to be in the ecosystem.

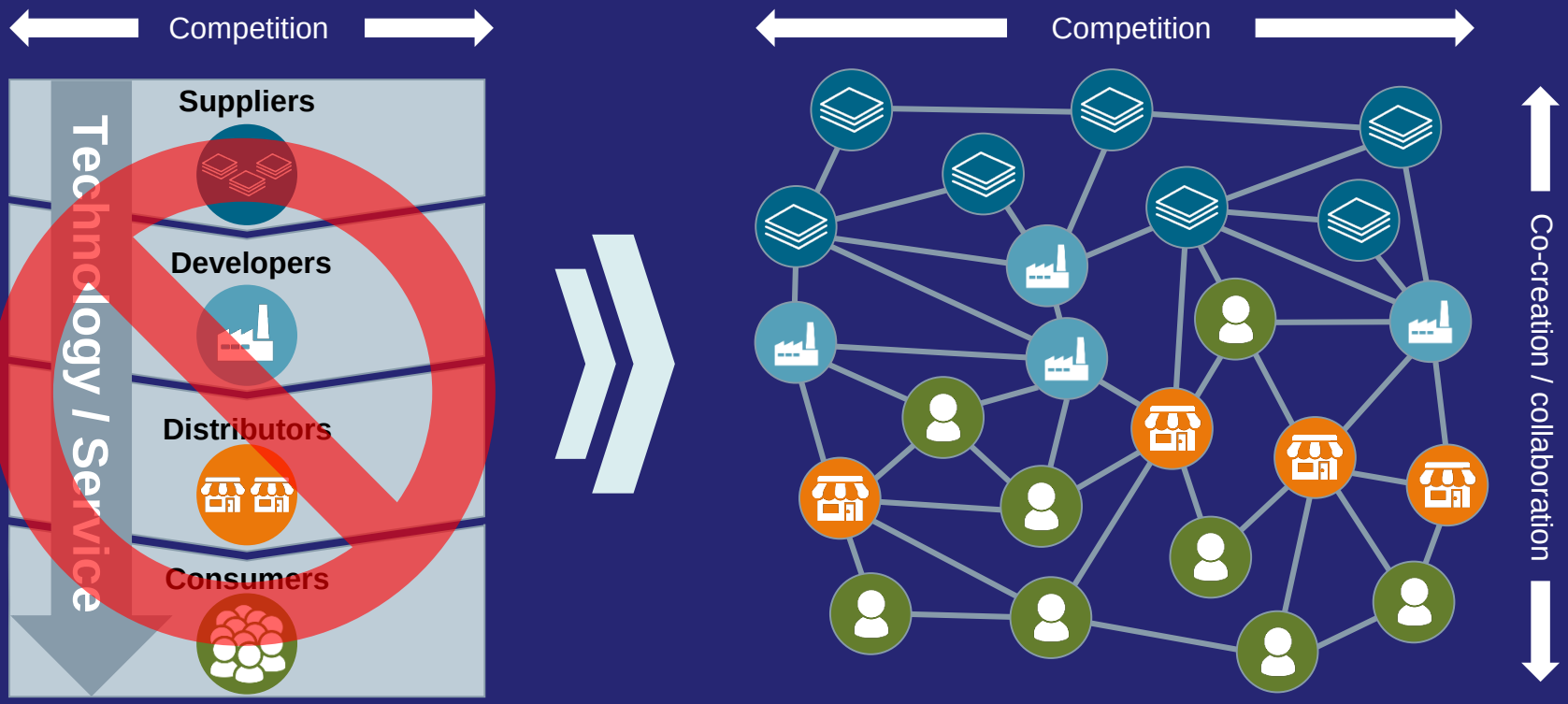
How open can we be? How open do we want to be? How far are we willing to go? …

There's no way to do this, no way to make this valuable going halfway!

Jeffrey Immelt, General Electric Chairman and CEO

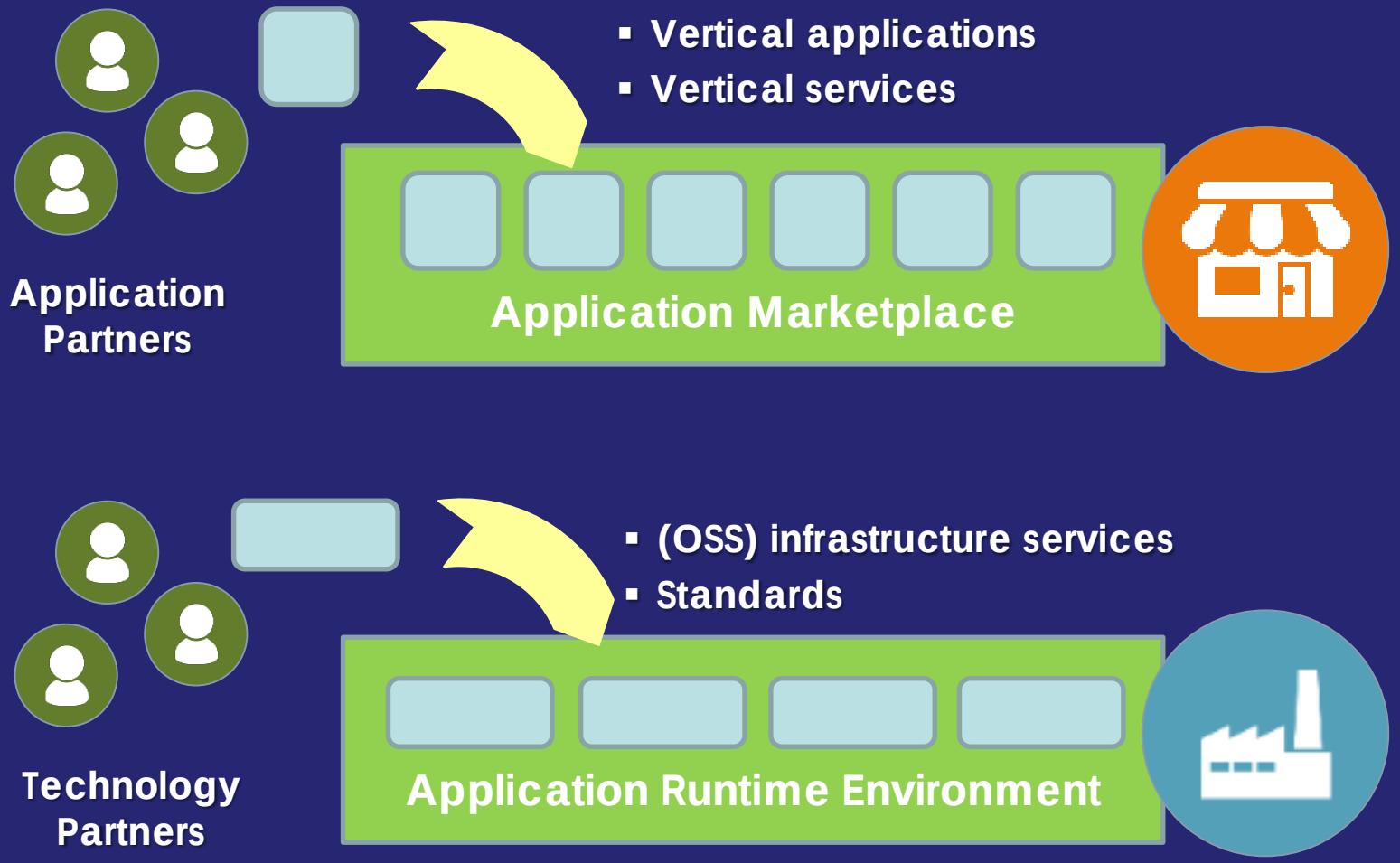
From linear supply chains to connected, complex and dynamic value networks

Competition in balance with co-creation and partner collaboration



Source: Deloitte analysis

Connect to the development community

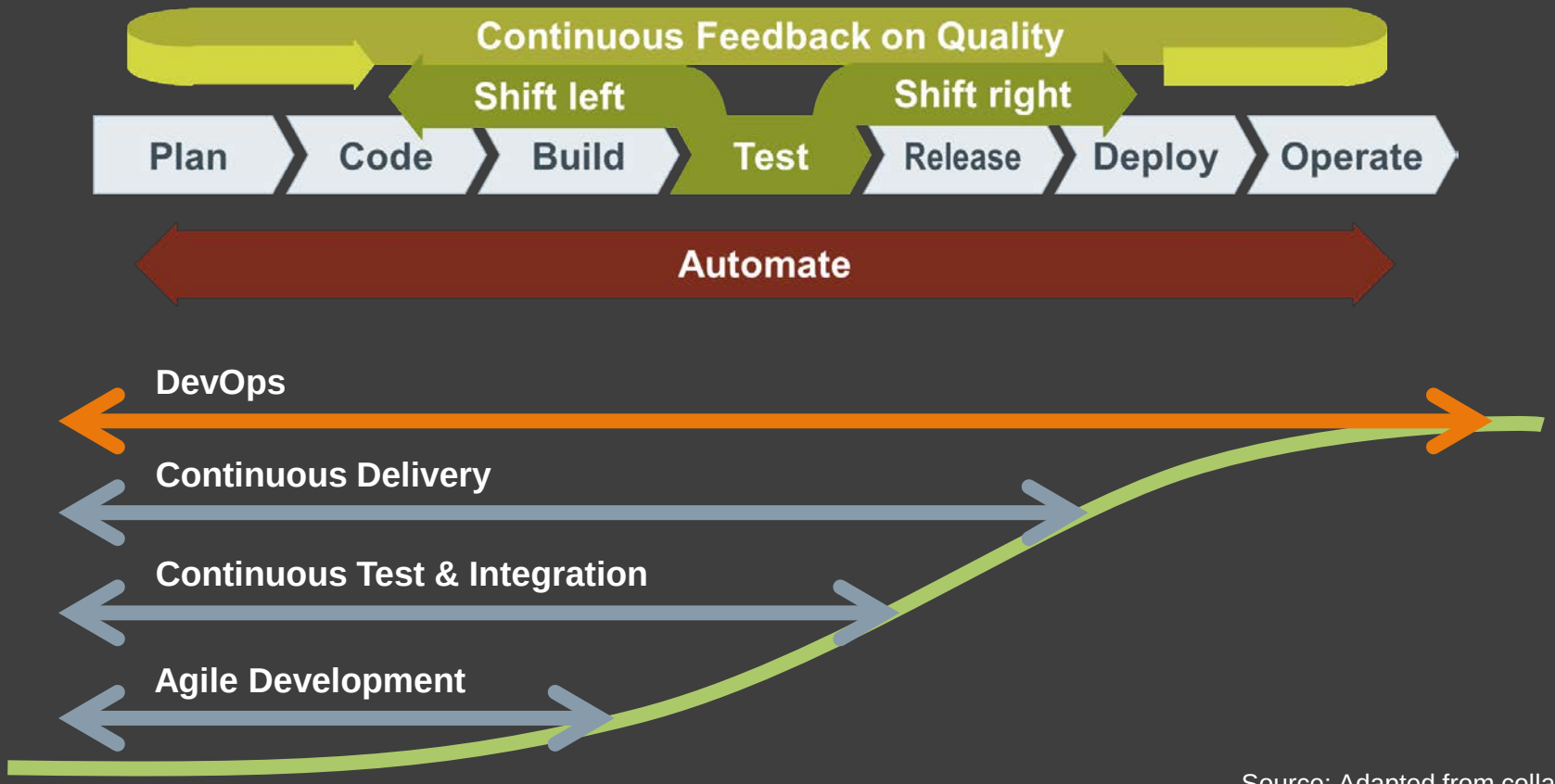


You build it, you run it

- DevOps is a culture, movement or practice that emphasizes the collaboration and communication of both software developers and IT professionals while automating the process of software delivery and infrastructure changes
- DevOps aims at establishing a culture and environment where building, testing, and releasing software, can happen rapidly, frequently, and more reliably

Wikipedia

DevOps is expanding agile principles beyond the code

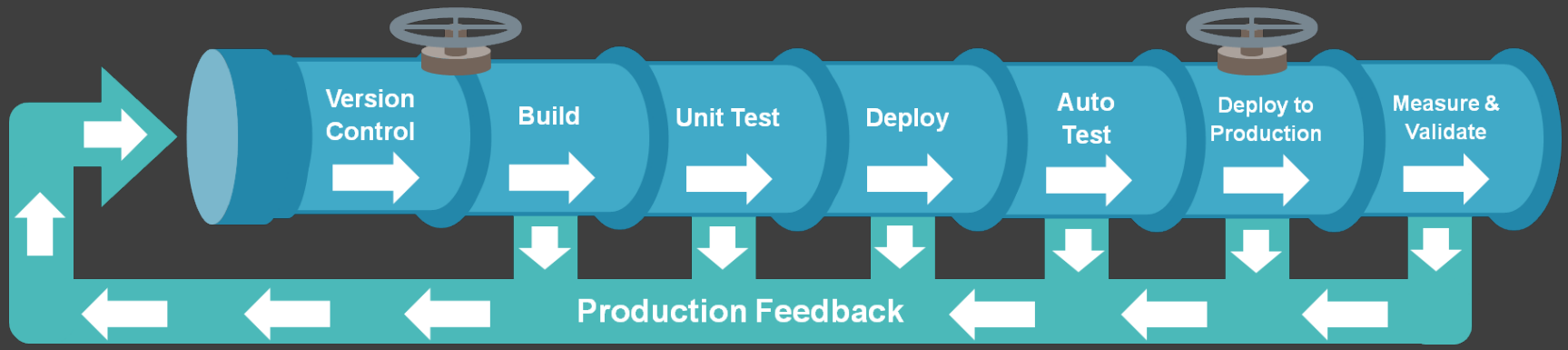


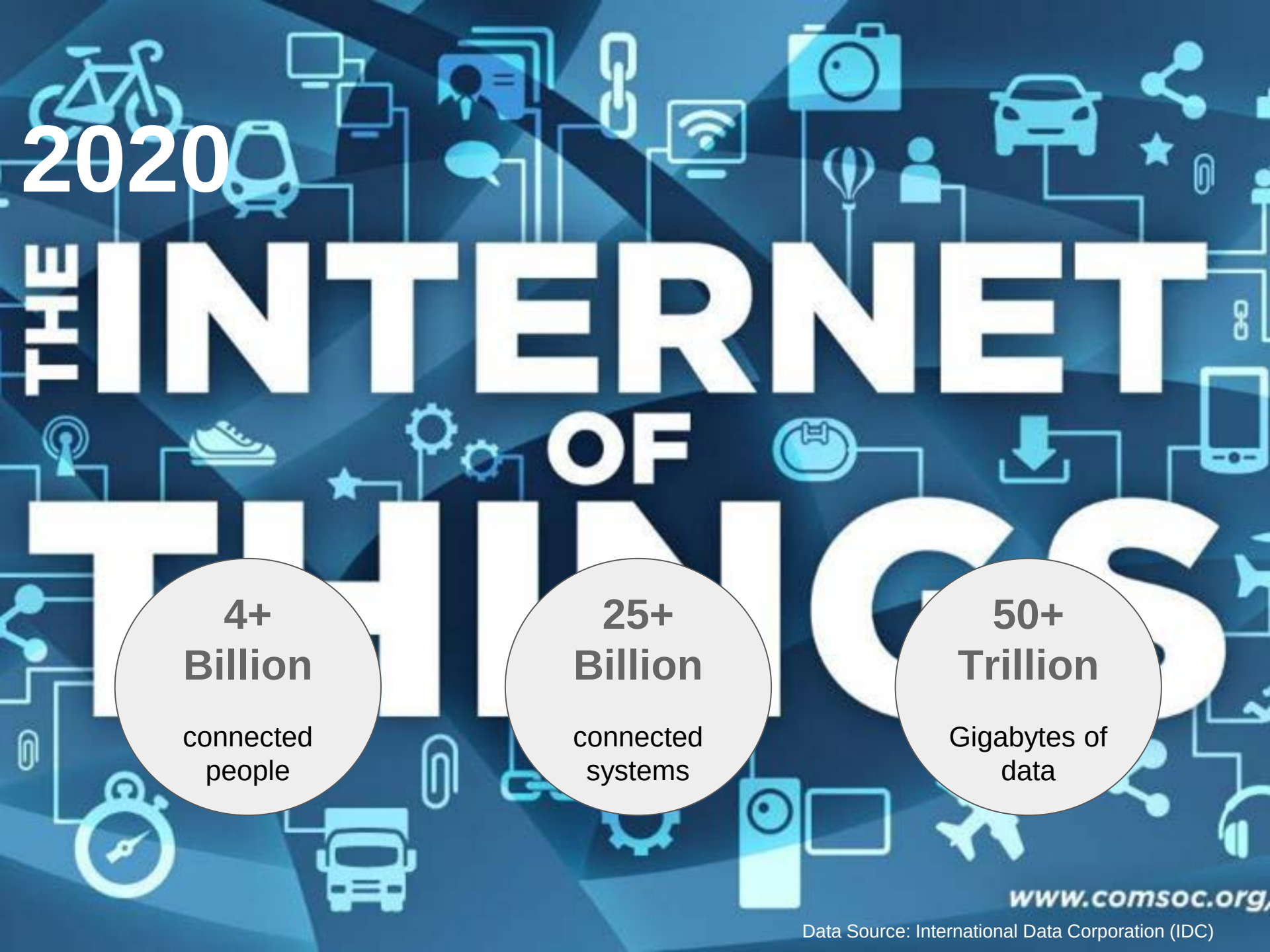
Source: Adapted from collab.net

Connect and integrate Dev and Ops

Consider the design of Dev and Ops environments as important as the design of the product itself – to balance speed and quality:

- **Safety net for developers to run code they built without compromising system quality and integrity**
- **Feedback loop for developers and operators to monitor, assess, and improve system quality for continuous system evolution**




2020
THE INTERNET OF THINGS
4+ Billion
connected people
25+ Billion
connected systems
50+ Trillion
Gigabytes of data
www.comsoc.org/
Data Source: International Data Corporation (IDC)

Do you think you can control the Internet of Things?

Scale: code; users; data managed; connections among software components; hardware elements

Failure is the norm

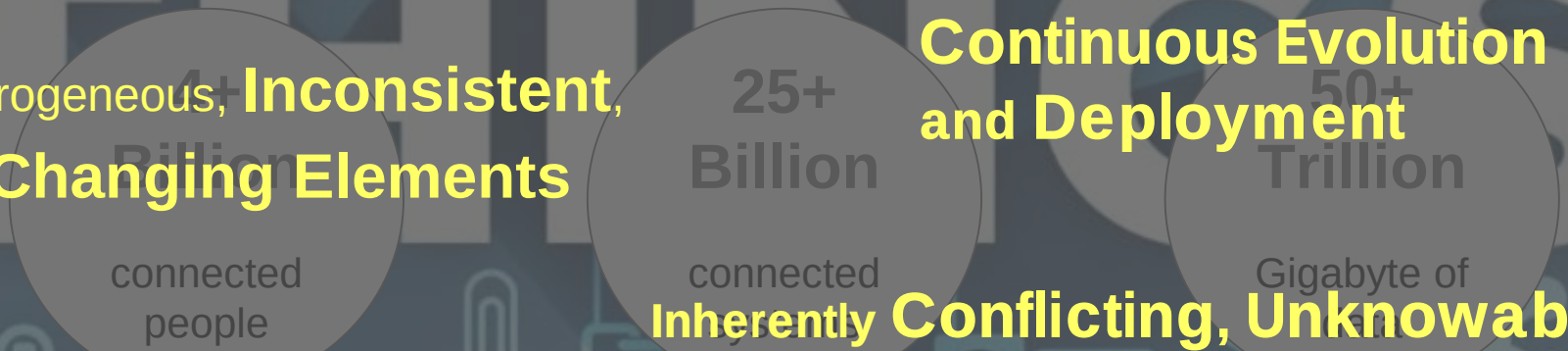
Decentralized Operations

Heterogeneous, **Inconsistent**, and **Changing Elements**

Continuous Evolution and Deployment

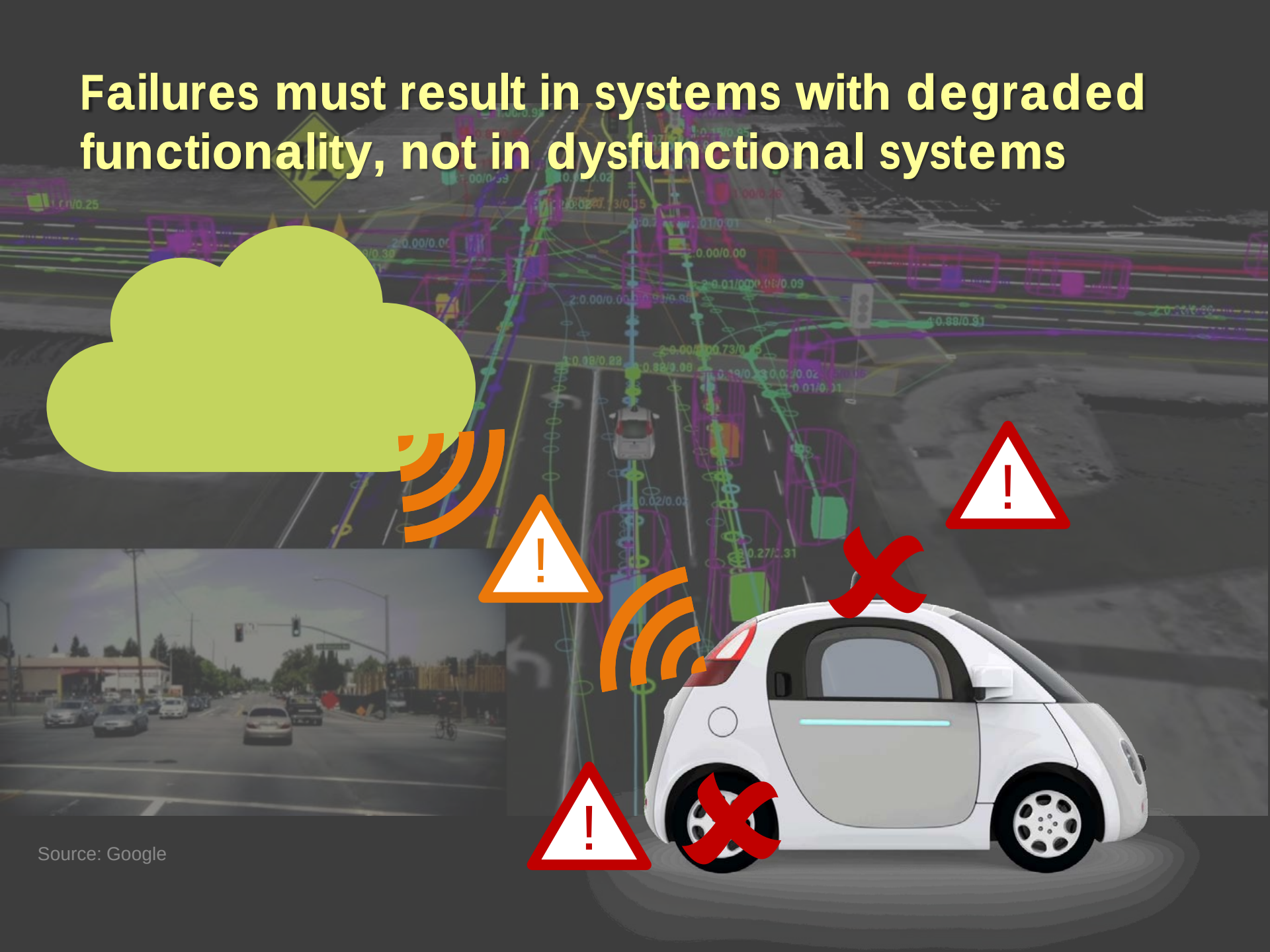
Erosion of User / System Boundary

Inherently Conflicting, Unknowable, and Diverse Requirements



Data Source: International Data Corporation (IDC)

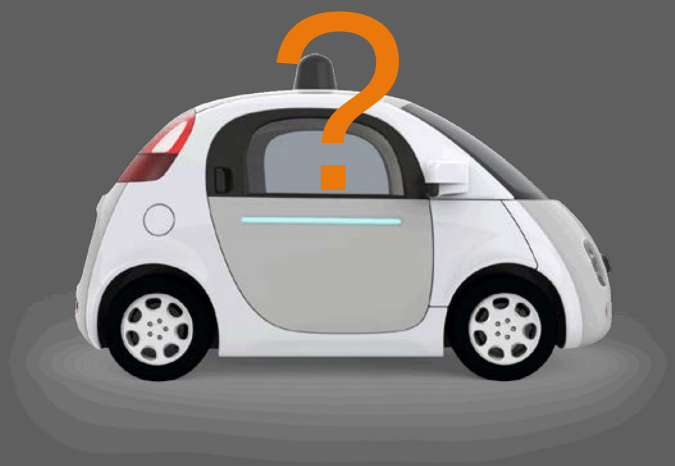
Failures must result in systems with degraded functionality, not in dysfunctional systems



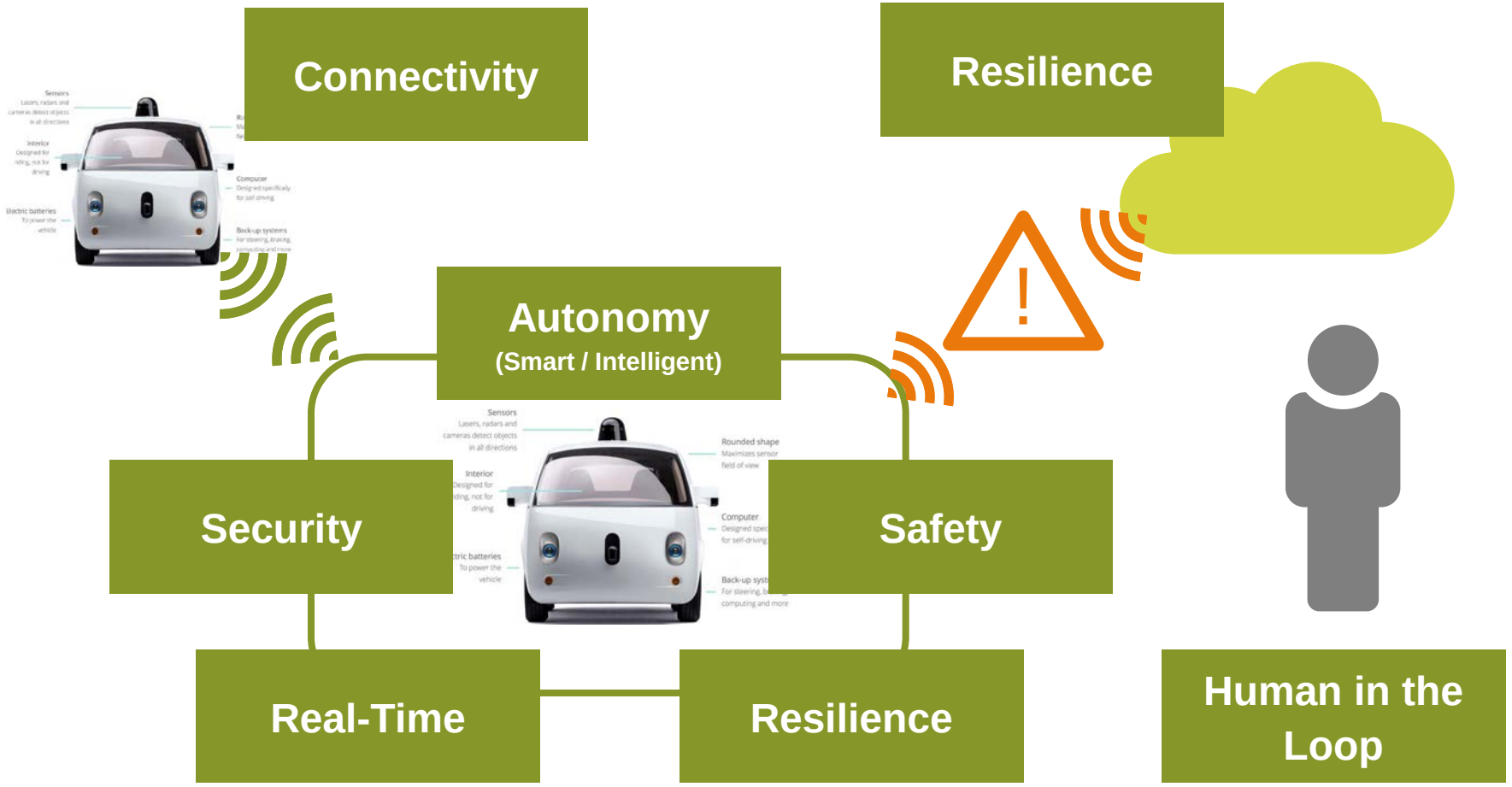
Source: Google

Correct behavior in unforeseen and emergent situations must be guaranteed

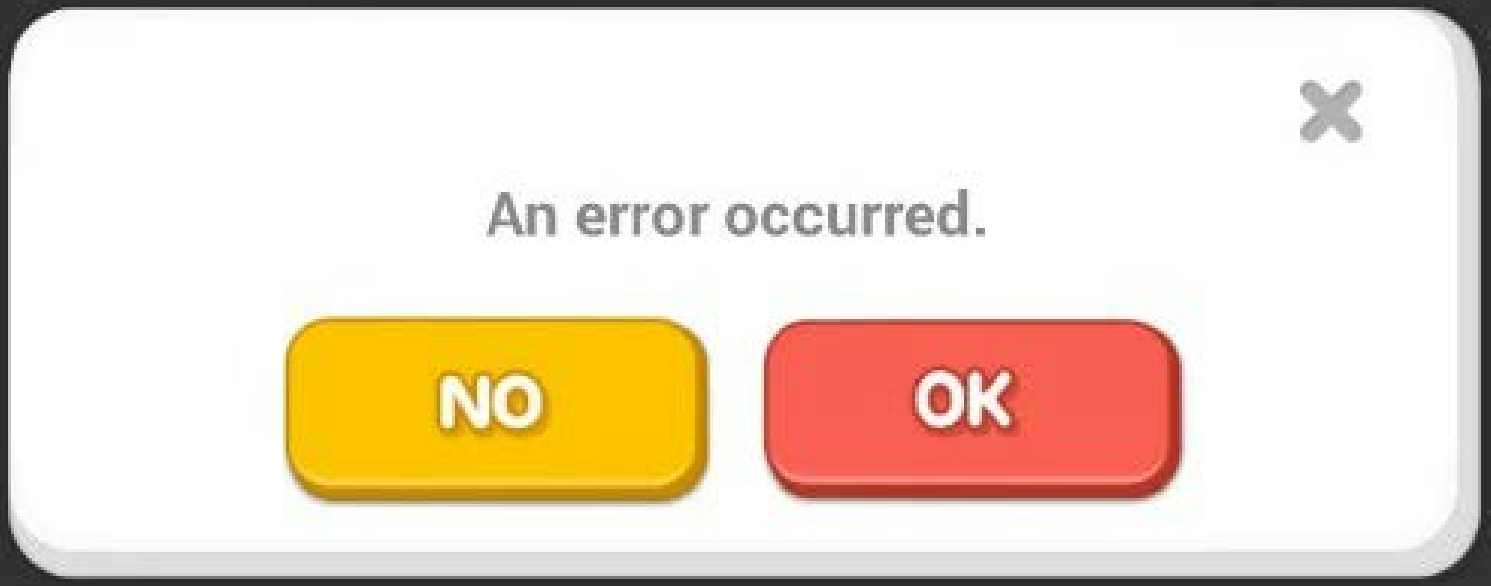
- Unlimited sensor input space
- Unlimited actor output space
- Unexpected environmental conditions
- Software updates
- Emergent behavior
- Humans in the loop
- “Gaming” systems



Connect systems to the Internet of Things but design for resilience in an uncontrollable world



User experience?



Connect to users by providing responsive user experience

The user is always right

User flow is important

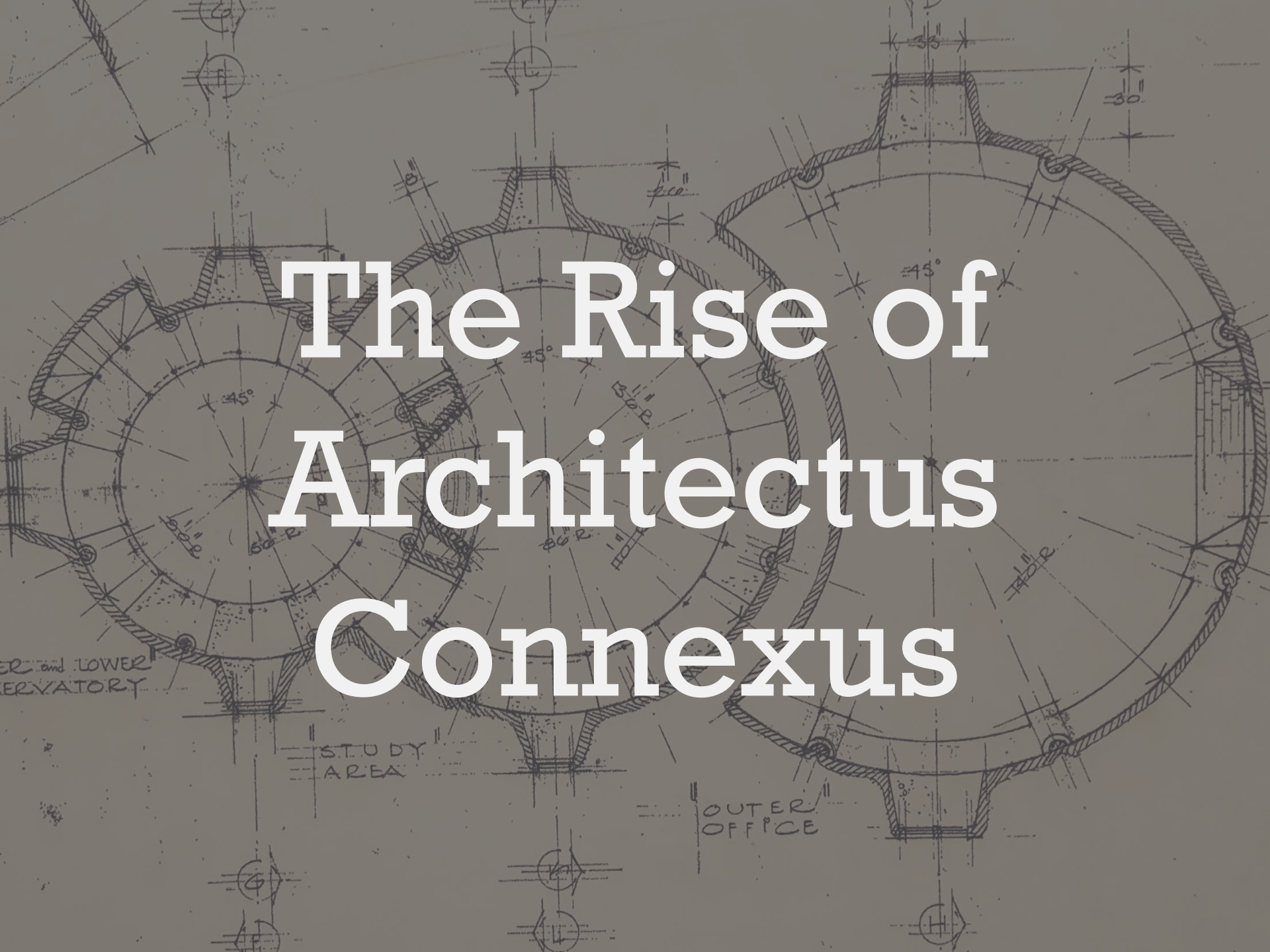
Form follows function

Content is king

Innovate, not imitate

Access is for everyone

Speed matters



The Rise of Architectus Connexus

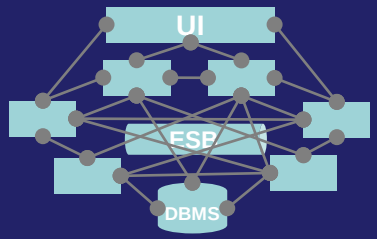
Digitalization demands a shift in focus and perspective from software architects

- From solving challenges within system services to full trust in dev teams that *do the right thing*
- From mentoring developers to integrating independent teams from multiple organizations
- From a development-centric view to a lifecycle view that explicitly includes deployment and operations
- From emergent architecture to a clear architecture vision and a design for continuous system evolution
- From designing for full control in protected environments to designing for resilience in an uncontrollable, ever-changing world

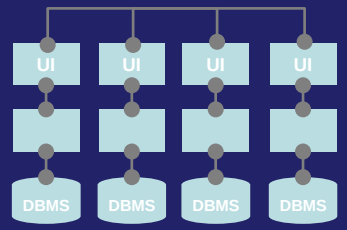
Software architects must balance inherently conflicting design forces

Scaled Agility and Continuous Evolution

From monolithic architectures



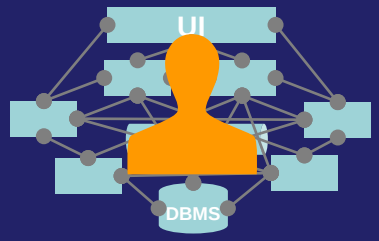
To microservice architectures



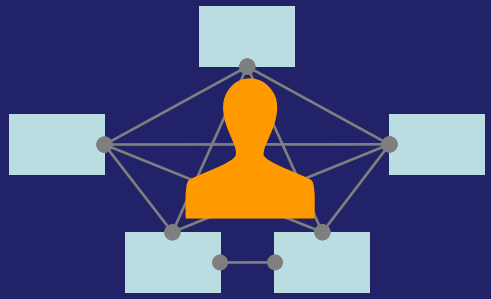
versus

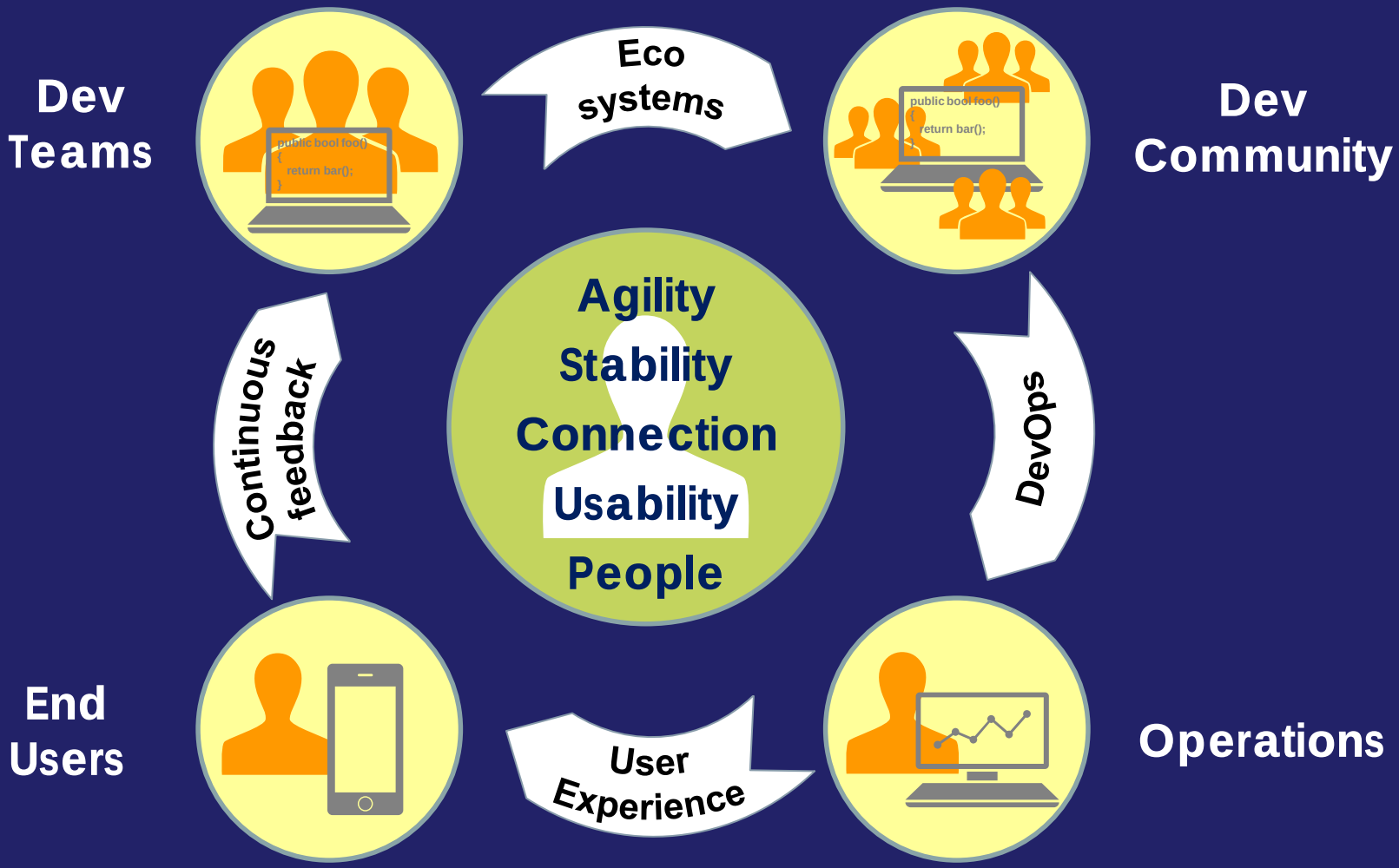
End-To-End Quality and Operational Resilience

From designing within systems



To designing between microservices




Dev Teams
Eco systems
Dev Community
public bool foo()
{
return bar();
}
Agility
Stability
Connection
Usability
People
Continuous feedback
DevOps
End Users
User Experience
Operations

UNITED FEDERATION OF PLANETS

Type P

Issuing state AURORA

Passport number CAN4185455074

Surname
OLIVAW

Name
R. DANEEL

Travel Bug number
BH1ZMJ

Mission/Destination
LE GARS DU NORD

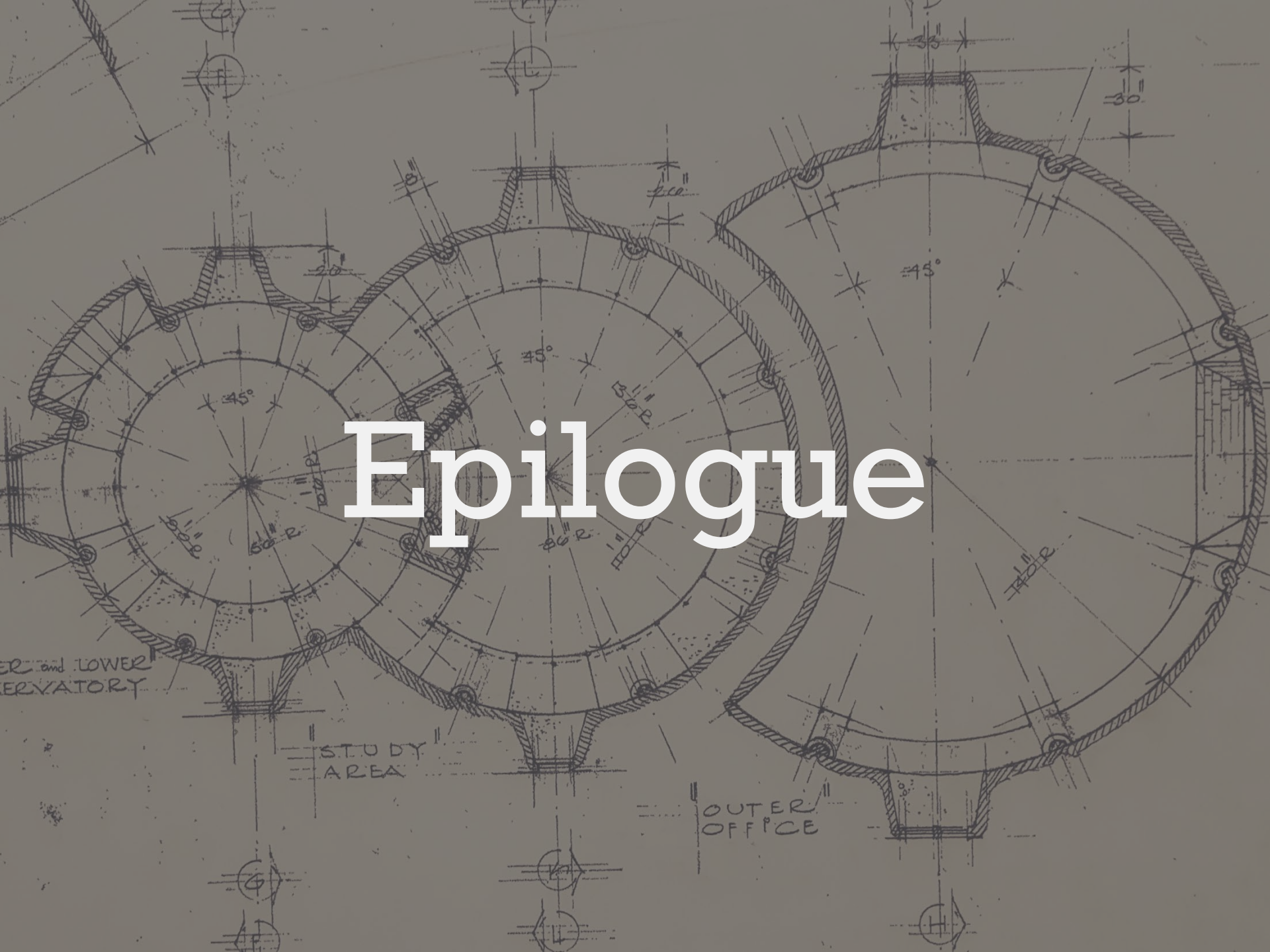
Cache identification
GC190TK

CITIZEN

Expiry date
F-24920

Architectus Connexus is a child of the digitalization age.

- Develops an architectural vision of the system that balances the need for continuous innovation with the need for operational quality and usability.
- Gives up developmental control by empowering an ecosystem of development teams to decide on the system's realization without corrupting its resilience in an IoT environment.
- Connects the people and organizations who develop, operate and use the system.



Epilogue

Mentor

Coach

Communicator

Observer

Connector

Architect?

Negotiator

Developer

Experimenter

Advocate

Tour guide

Listener


Software Architects Are Dead!
Long Live Software Architects!