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.
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**

- **Big Up-Front Design (BUFD)**
  - Assumes everything is known or foreseeable in advance
  - Can create an illusion the team knows more than they actually do
  - Poorly adapted to change
  - Danger of analysis paralysis

- **Rough Up-Front Design (RUFD)**

- **No Up-Front Design (NUFD)**
  - Good when requirements must be discovered
  - Open to continuous change
  - Assumes change is cheap
  - Failure mode degenerates to ad hoc tactical decisions
Internet of Things Architecture in the Age of Digitalization

DevOps
Microservices
Scaled Agility
Self-Learning Systems
Self-organizing Systems
Ecosystems
Cyber-Physical Systems
Autonomous Systems
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 know everything up front.

### CHAOS Resolution by Agile versus Waterfall

<table>
<thead>
<tr>
<th>SIZE</th>
<th>METHOD</th>
<th>SUCCESSFUL</th>
<th>CHALLENGED</th>
<th>FAILED</th>
</tr>
</thead>
<tbody>
<tr>
<td>All Size Projects</td>
<td>Agile</td>
<td>39%</td>
<td>52%</td>
<td>9%</td>
</tr>
<tr>
<td></td>
<td>Waterfall</td>
<td>11%</td>
<td>60%</td>
<td>29%</td>
</tr>
</tbody>
</table>

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 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
<table>
<thead>
<tr>
<th>SIZE</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>100</td>
<td>250</td>
<td>400</td>
<td>550</td>
<td>700</td>
</tr>
<tr>
<td>S2</td>
<td>175</td>
<td>325</td>
<td>475</td>
<td>625</td>
<td>775</td>
</tr>
<tr>
<td>S3</td>
<td>250</td>
<td>400</td>
<td>550</td>
<td>700</td>
<td>850</td>
</tr>
<tr>
<td>S4</td>
<td>325</td>
<td>475</td>
<td>625</td>
<td>775</td>
<td>625</td>
</tr>
<tr>
<td>S5</td>
<td>400</td>
<td>550</td>
<td>700</td>
<td>850</td>
<td>1000</td>
</tr>
</tbody>
</table>

- 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 a disaster happens here?
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
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

Application Partners

Application Marketplace
- Vertical applications
- Vertical services

Technology Partners

Application Runtime Environment
- (OSS) infrastructure services
- Standards
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.
DevOps is expanding agile principles beyond the code.
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

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

**Decentralized Operations**

- Heterogeneous, **Inconsistent, and Changing Elements**
- Erosion of User / System Boundary

**Continuous Evolution and Deployment**

- 25+ Billion connected people
- Inherently **Conflicting, Unknowable, and Diverse Requirements**
- 50+ Trillion Gigabyte of data

**Failure is the norm**

2020

Data Source: International Data Corporation (IDC)
Failures must result in systems with degraded functionality, not in dysfunctional systems.
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.

Connectivity

Resilience

Real-Time

Autonomy (Smart / Intelligent)

Human in the Loop

Security

Safety

Resilience

Human in the Loop
User experience?

An error occurred.

NO

OK
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**

versus

**End-To-End Quality and Operational Resilience**

From monolithic architectures:

- UI
- ESP
- DBMS

To microservice architectures:

- UI
- UI
- UI
- UI
- DBMS
- DBMS
- DBMS
- DBMS

From designing within systems:

- UI
- UI
- DBMS

To designing between microservices:

- UI
- DBMS
- DBMS
- DBMS
- DBMS
- DBMS
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.
Architect?

Coach

Observer

Communicator

Connector

Mentor

Experimenter

Negotiator

Developer

Advocate

Tour guide

Listener
Software Architects Are Dead!
Long Live Software Architects!