Tampere University

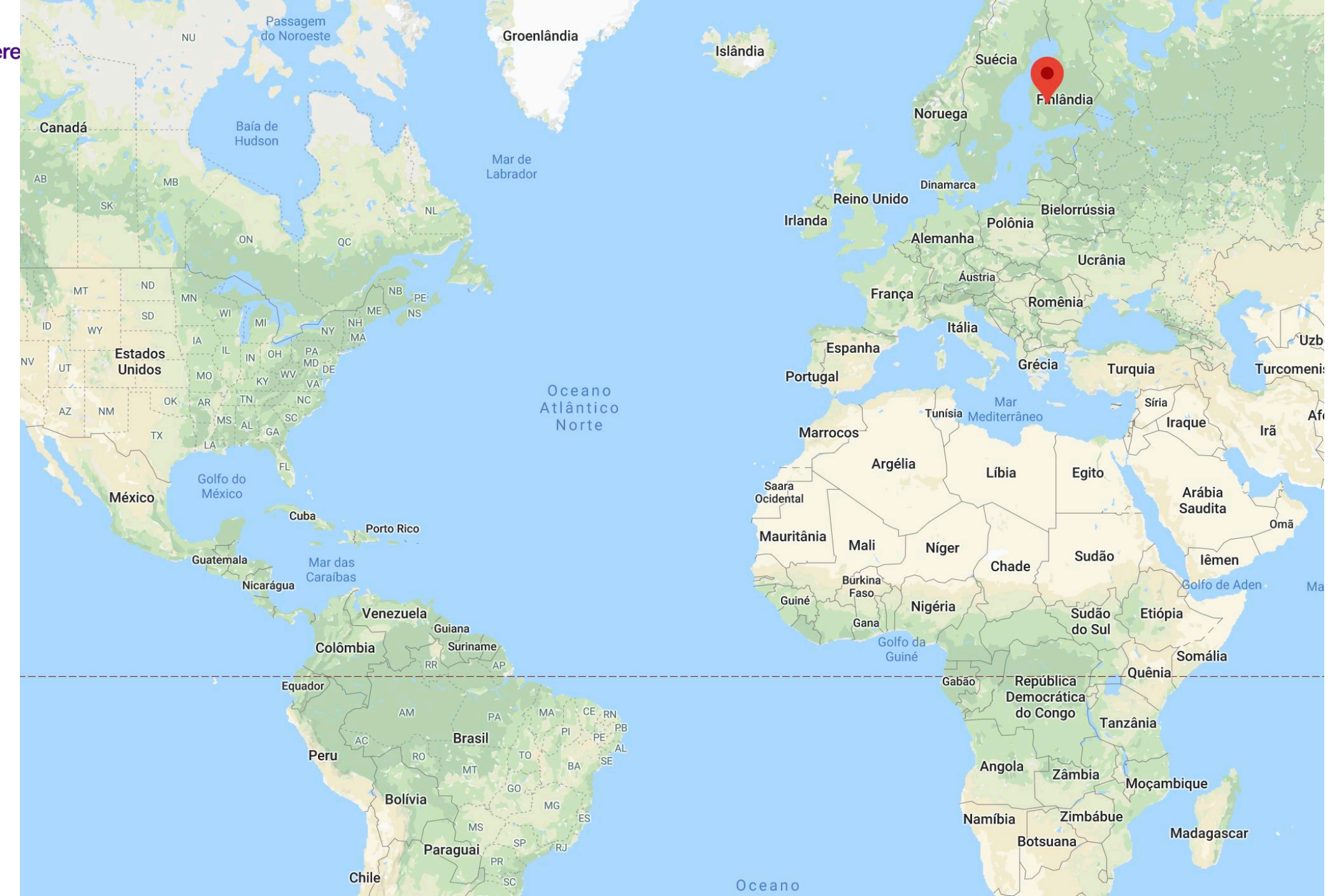
From Monolithic to Microservices to Serverless

Davide Taibi

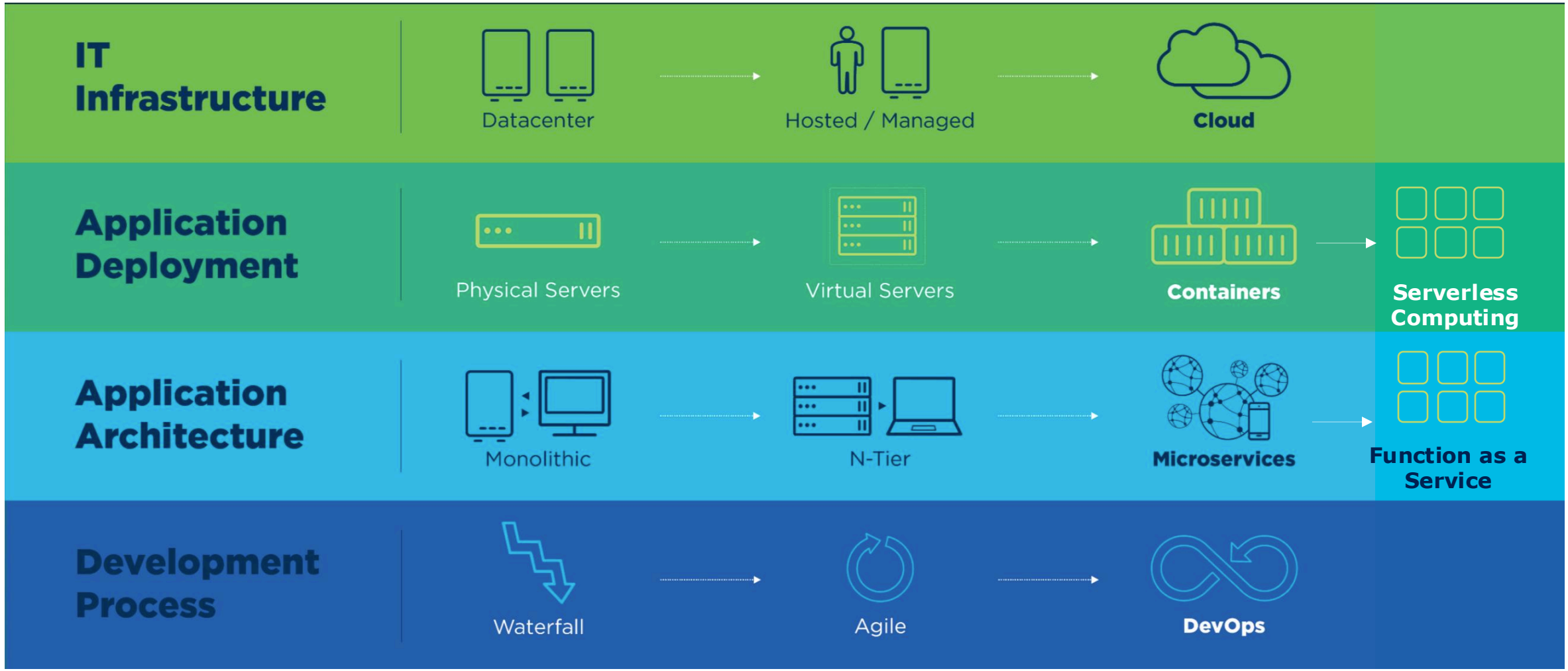
CloWEE – Cloud and Web Engineering
http://research.tuni.fi/clowee

17.05.2020

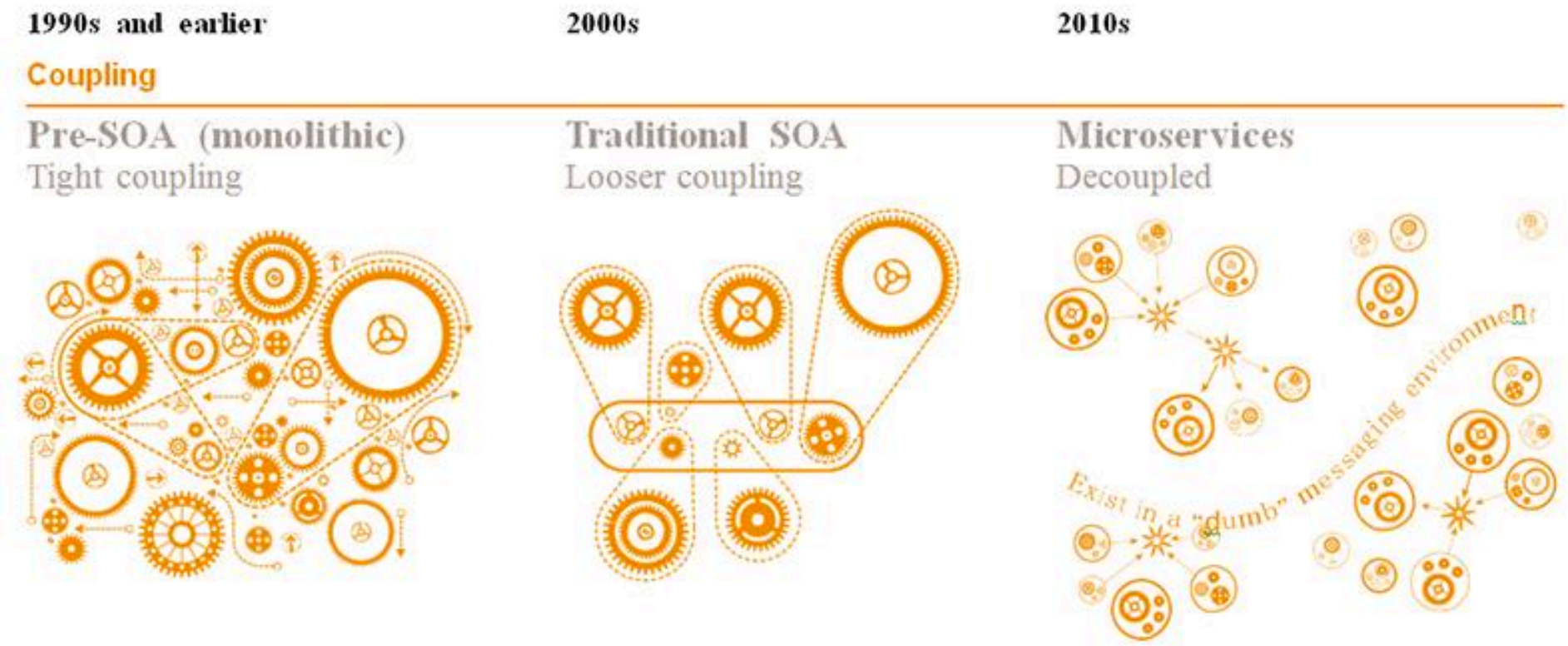
Tampere


Passagem do Noroeste
Groenlândia
Islândia
Suécia
Finlândia
Noruega
Canadá
Baía de Hudson
Mar de Labrador
Dinamarca
Reino Unido
Irlanda
Bielorrússia
Polônia
Alemanha
Ucrânia
Áustria
França
Romênia
Itália
Espanha
Portugal
Grécia
Turquia
Síria
Iraque
Irã
Tunísia
Mar Mediterrâneo
Marrocos
Argélia
Líbia
Egito
Saara Ocidental
Arábia Saudita
Omã
Mauritânia
Mali
Níger
Chade
Sudão
Iêmen
Golfo de Aden
Burkina Faso
Guiné
Gana
Nigéria
Sudão do Sul
Etiópia
Golfo da Guiné
Somália
Quênia
Gabão
República Democrática do Congo
Tanzânia
Angola
Zâmbia
Moçambique
Namíbia
Zimbábue
Botsuana
Madagascar
Oceano Atlântico Norte
Estados Unidos
México
Golfo do México
Cuba
Porto Rico
Guatemala
Nicarágua
Mar das Caraíbas
Venezuela
Guiana
Suriname
Colômbia
Equador
Peru
Brasil
Bolívia
Paraguai
Chile
Oceano

Software Systems Evolution



Software Architecture Evolution



D.Taibi, V.Lenarduzzi, C.Pahl, A.Janes. Microservices Architectural Styles: Agile or not Agile? XP 2017

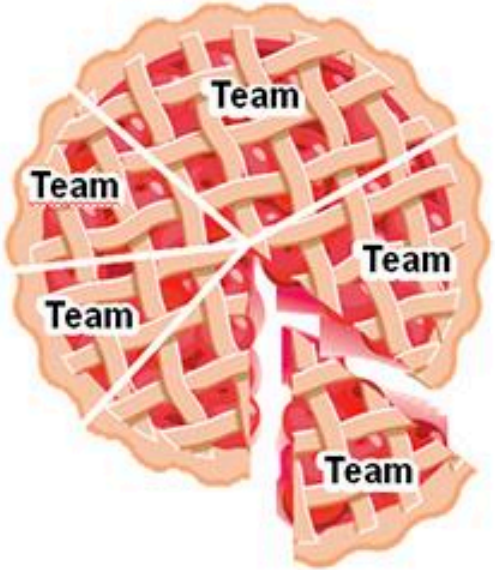
Software Architecture Evolution

1990s and earlier

Pre-SOA (monolithic)
Tight coupling

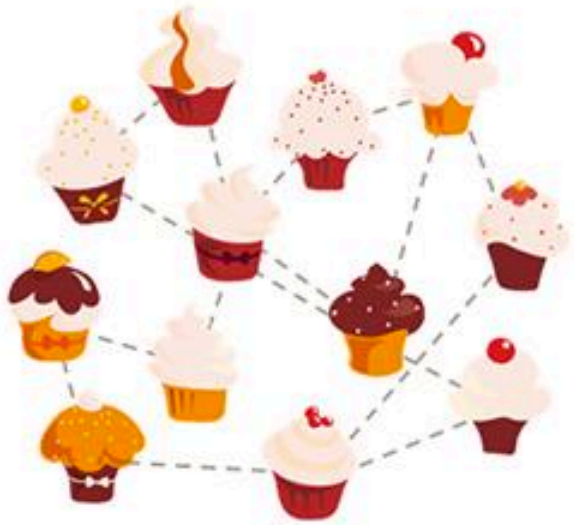
2000s

Traditional SOA
Looser coupling



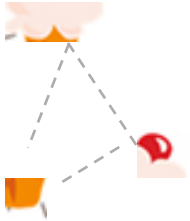
2010s

Microservices
Decoupled

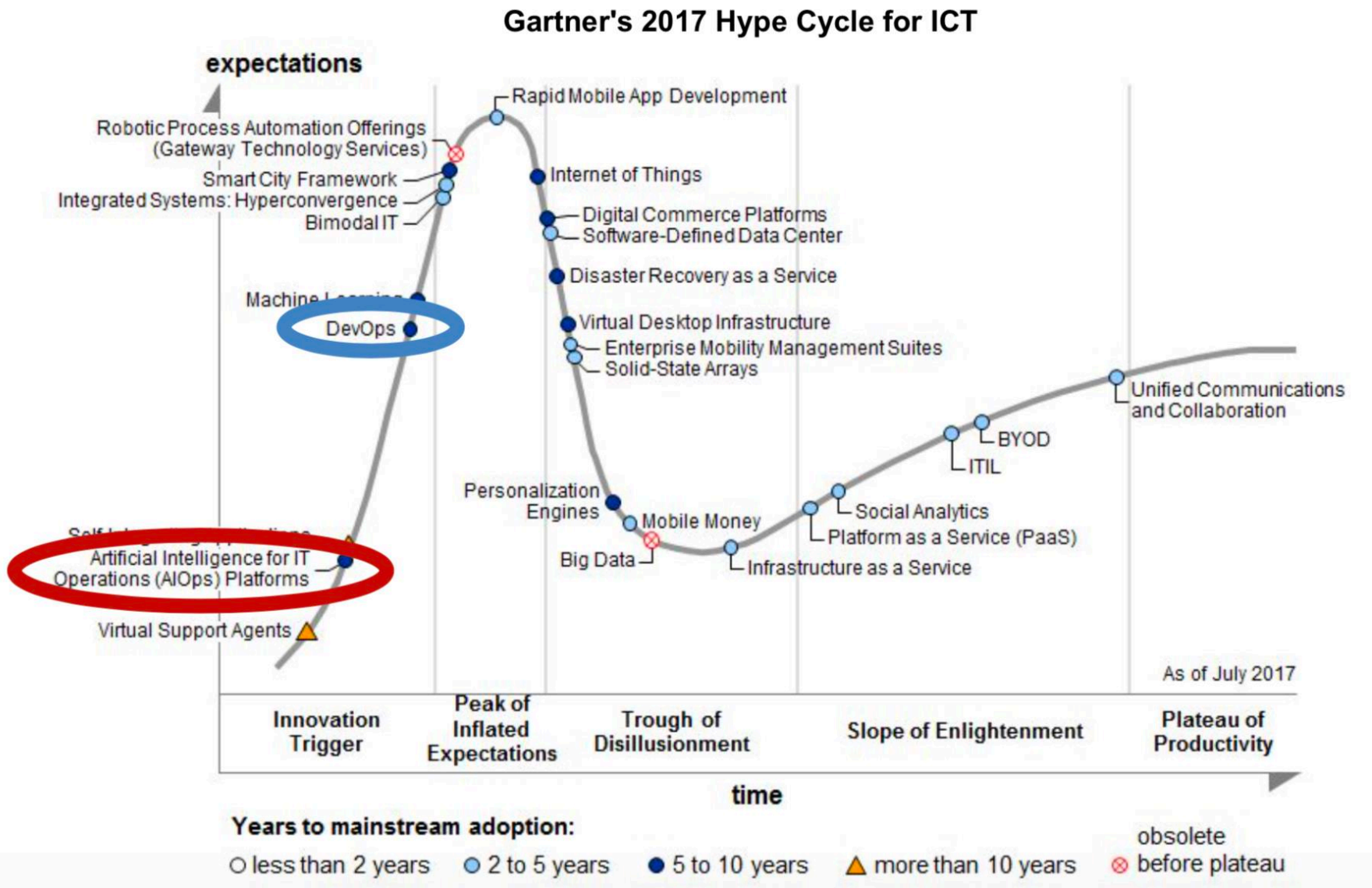


2015s

Function as a Service
Highly Decoupled



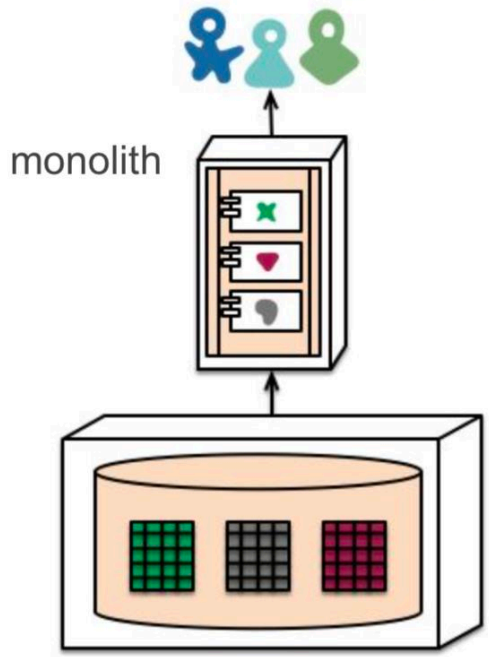
Why?

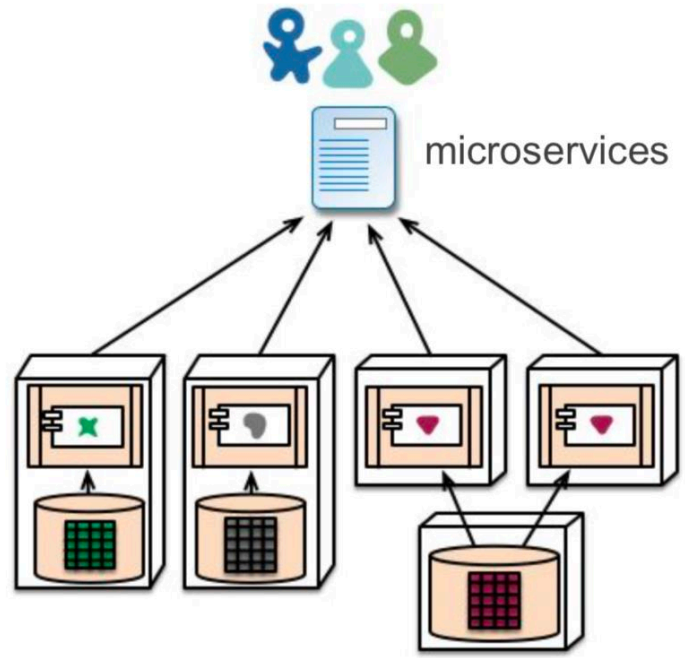

Gartner's 2017 Hype Cycle for ICT
expectations
Rapid Mobile App Development
Robotic Process Automation Offerings (Gateway Technology Services)
Smart City Framework
Integrated Systems: Hyperconvergence
Bimodal IT
Internet of Things
Digital Commerce Platforms
Software-Defined Data Center
Disaster Recovery as a Service
Machine Learning
DevOps
Virtual Desktop Infrastructure
Enterprise Mobility Management Suites
Solid-State Arrays
Unified Communications and Collaboration
BYOD
ITIL
Personalization Engines
Mobile Money
Social Analytics
Platform as a Service (PaaS)
Big Data
Infrastructure as a Service
Artificial Intelligence for IT Operations (AIOps) Platforms
Virtual Support Agents
As of July 2017
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time
Years to mainstream adoption:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau

"A **suite of small services**, each **running in its own process** and **communicating with lightweight mechanisms**, often an HTTP resource API."

"These services are **built around business capabilities** and **independently deployable** by **fully automated** deployment machinery."

"There is a **bare minimum of centralized management** of these services, which may be written in **different programming languages** and use **different data storage technologies**."
J. Lewis & M. Fowler, ThoughtWorks





source: https://martinfowler.com/articles/microservices.html

"Small **autonomous** services that **work together**, modelled around a **business domain**."

S. Newman, ThoughtWorks, author of "Building Microservices"

"**Loosely coupled** service-oriented architecture with **bounded contexts**."

"Monolithic apps have **invisible internal complexity**. Microservices expose that [complexity] as **explicit micro service dependencies**."

Adrian Cockcroft, AWS (formerly at Netflix)

"We need to move to **managed complexity**. Microservices are about **negotiated interfaces**, **strict boundaries**, **shared nothing**!"

J. Higginbotham, LaunchAny

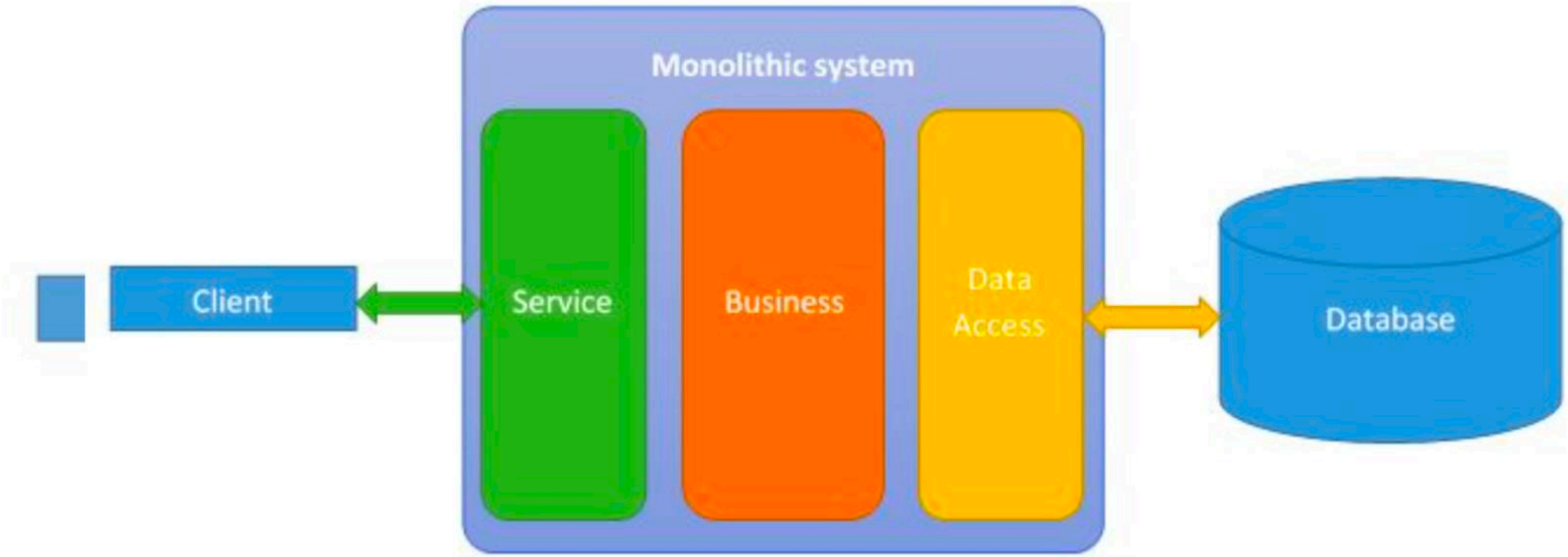
Amazon's now famous migration from the Obidos monolithic application to a service-oriented architecture with encapsulated databases and small, "two-pizza" teams

Amazon's design principles:

- Design for flexibility
- Design for on demand
- Design for automation
- Design for failure
- Be elastic
- Design for utility pricing
- Break transparency
- Decompose to its simplest form
- Design with security in mind
- Don't do It alone
- Focus on what doesn't change
- Let your customers benefit
- Continuously innovate

"For us service orientation means encapsulating the data with the business logic that operates on the data, with the only access through a published service interface. No direct database access is allowed from outside the service, and there's no data sharing among the services."

-- Werner Vogel, Amazon's CTO, 2006



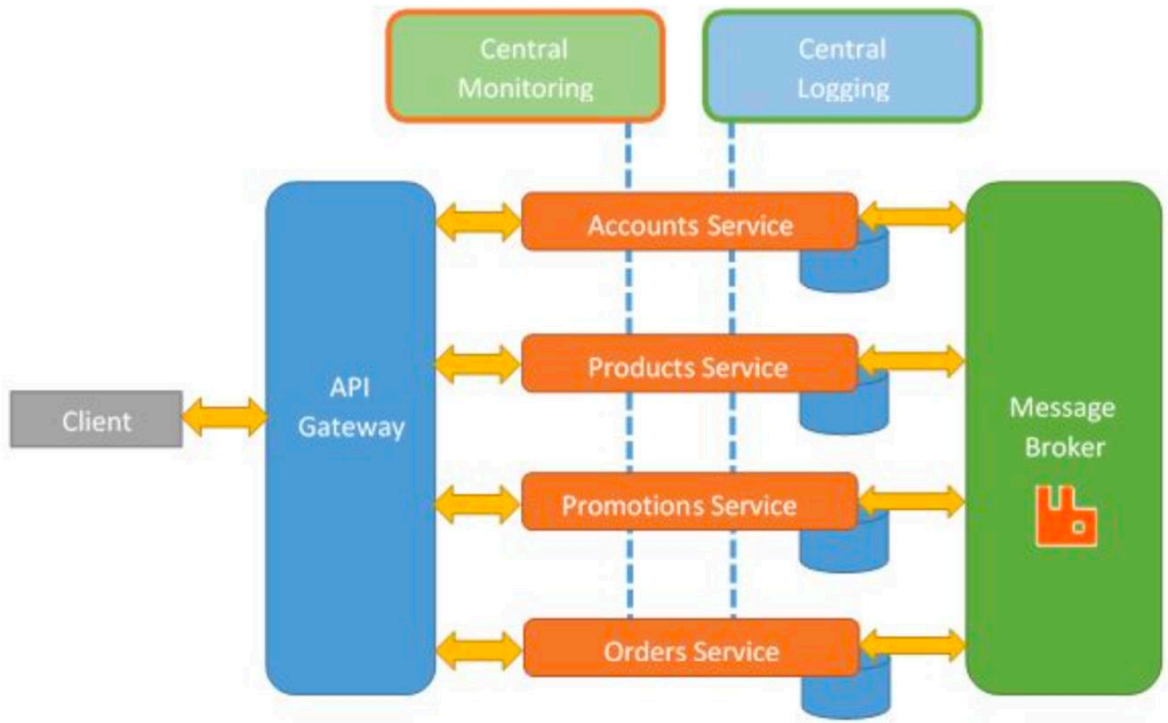
source: http://www.acarlstein.com/

"There's no reason why you can't make a single monolith with well defined module boundaries. At least there's no reason *in theory*. In practice, it seems too easy for module boundaries to be breached and monoliths to get tangled as well as large."

-- Martin Fowler

The microservices style is an approach to design systems whose parts are easy to change and replace *at runtime*

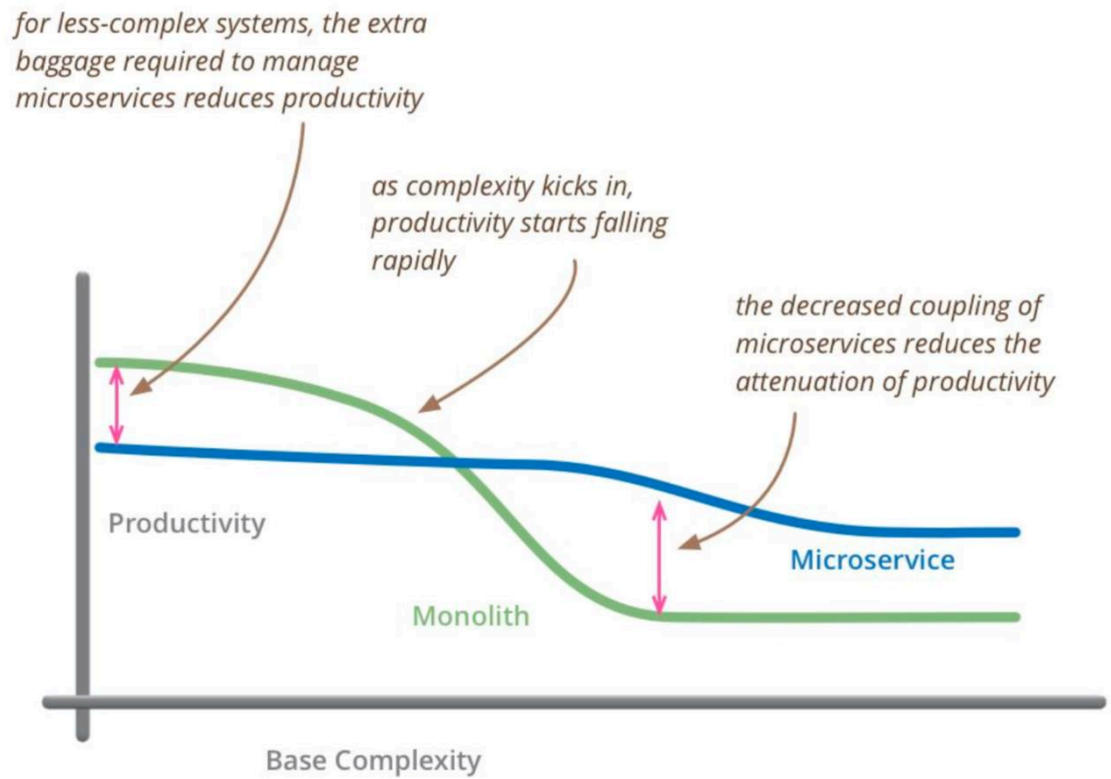
It pushes information hiding to new heights by enforcing strict module boundaries and by promoting information isolation



source: http://www.acarlstein.com/

The microservices style introduces its own set of (distributed systems related) complexities:

- automated deployment
- continuously monitoring
- dealing with failure
- eventual consistency
- security



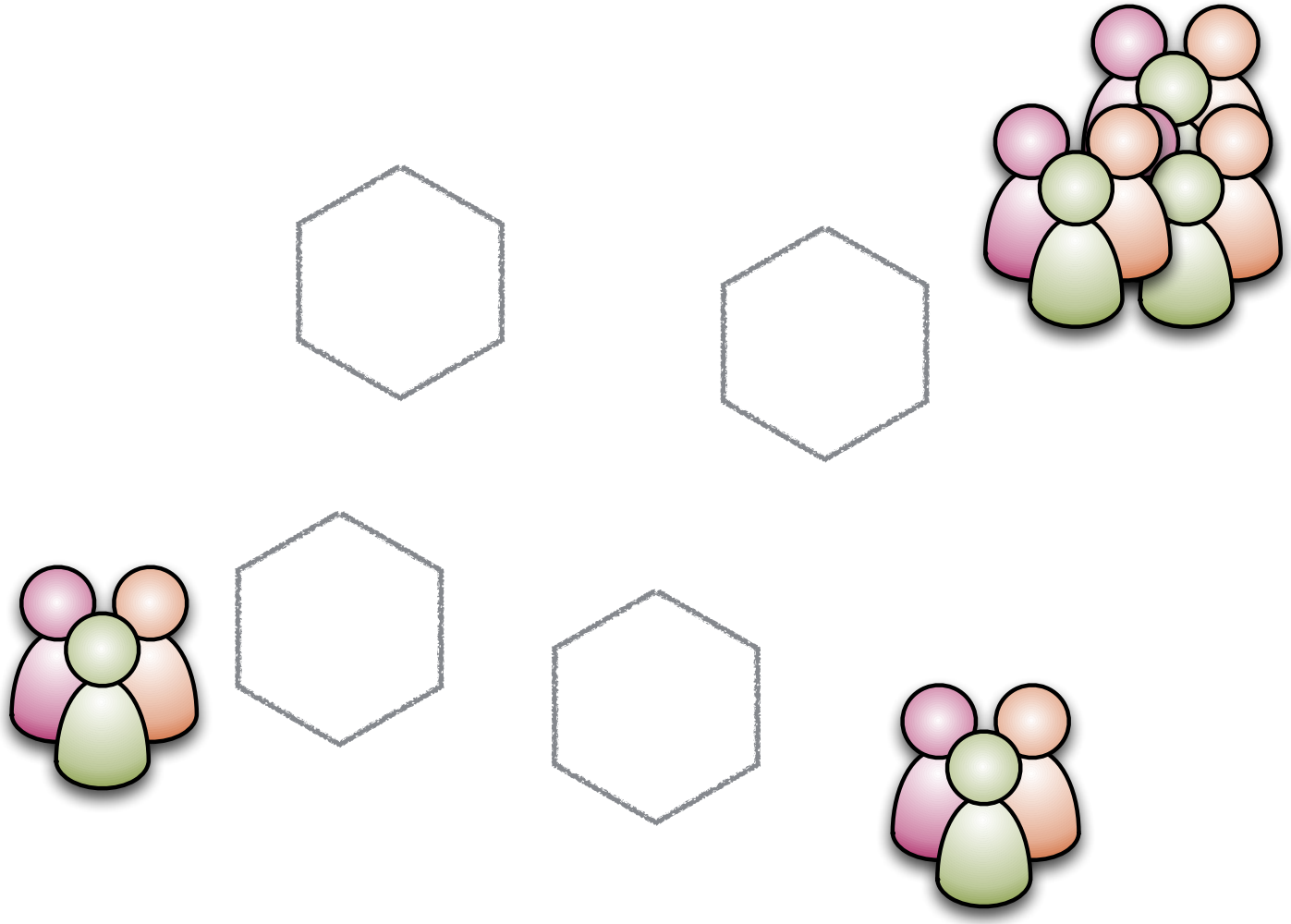
source: https://martinfowler.com/bliki/MicroservicePremium.html

"The majority of software systems should be built as a single monolithic application. Do pay attention to good modularity within that monolith. Don't even consider microservices unless you have a system that's too complex to manage as a monolith."

-- Martin Fowler

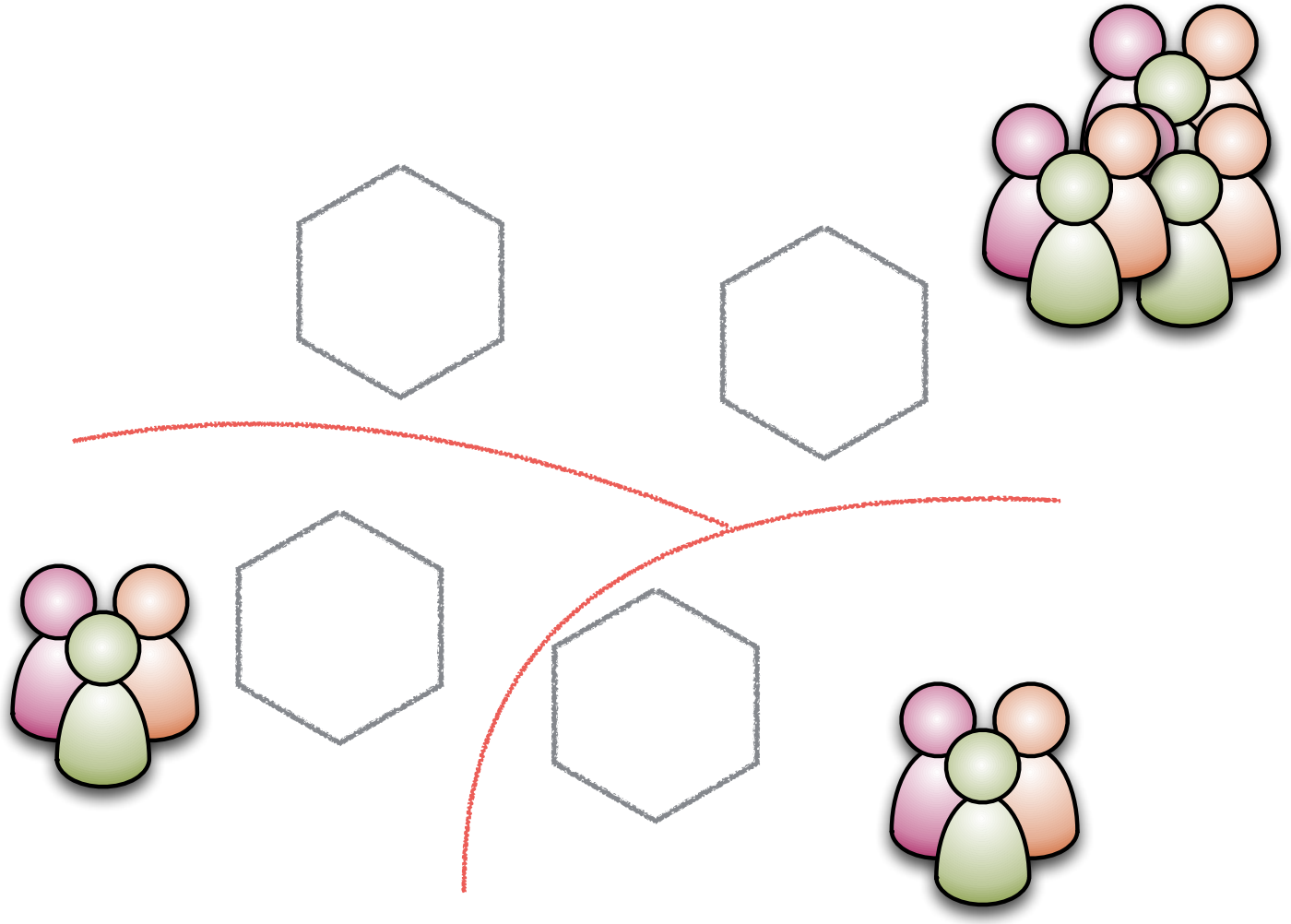
Tampere University

We can organise services along organisational boundaries



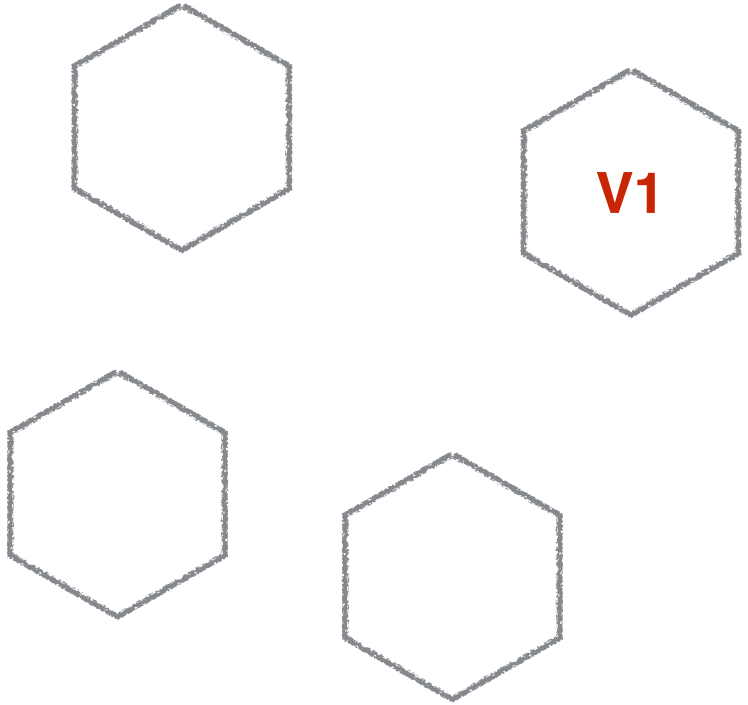
Tampere University

We can organise services along organisational boundaries



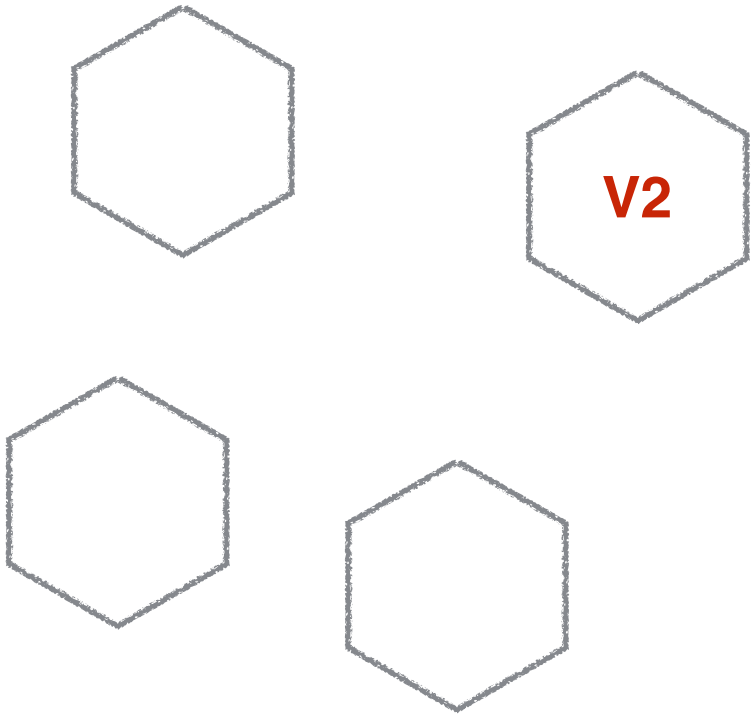
Tampere University

By sub-divinding our systems, we can speed the release of new features



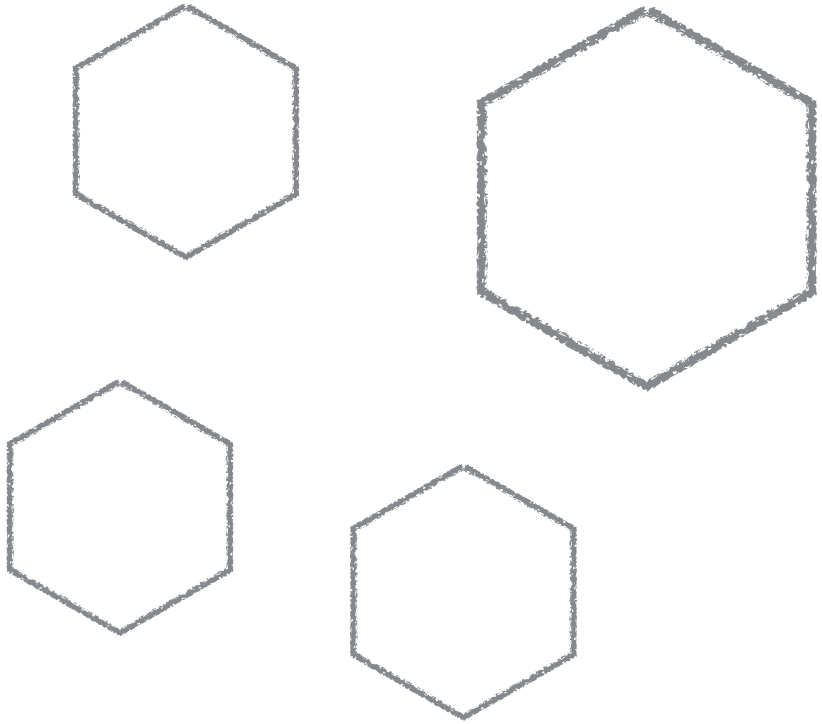
Tampere University

By sub-divinding our systems, we can speed the release of new features

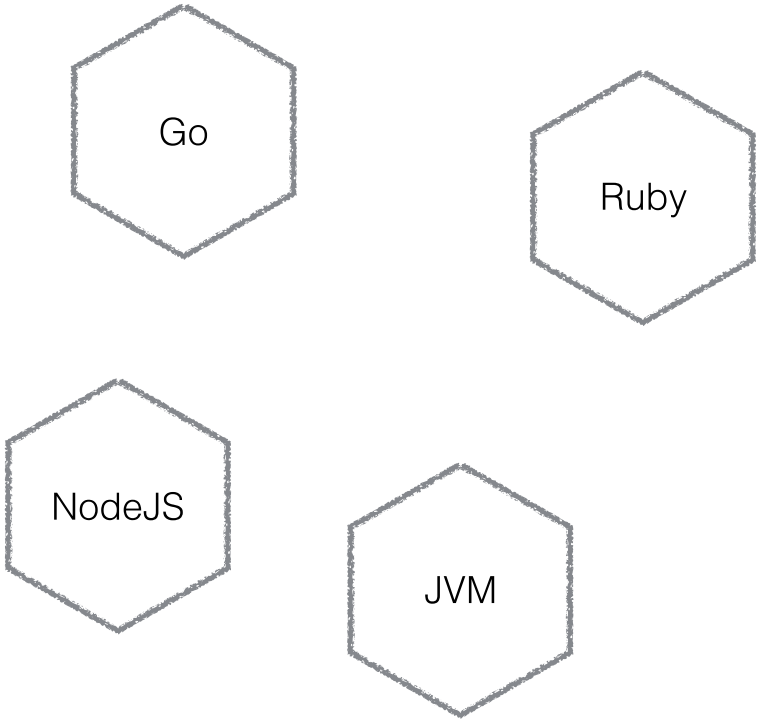


Tampere University

It allows us different options in terms of scaling



and we can use different tools and tech



Summary

We understand more about building reliable distributed systems

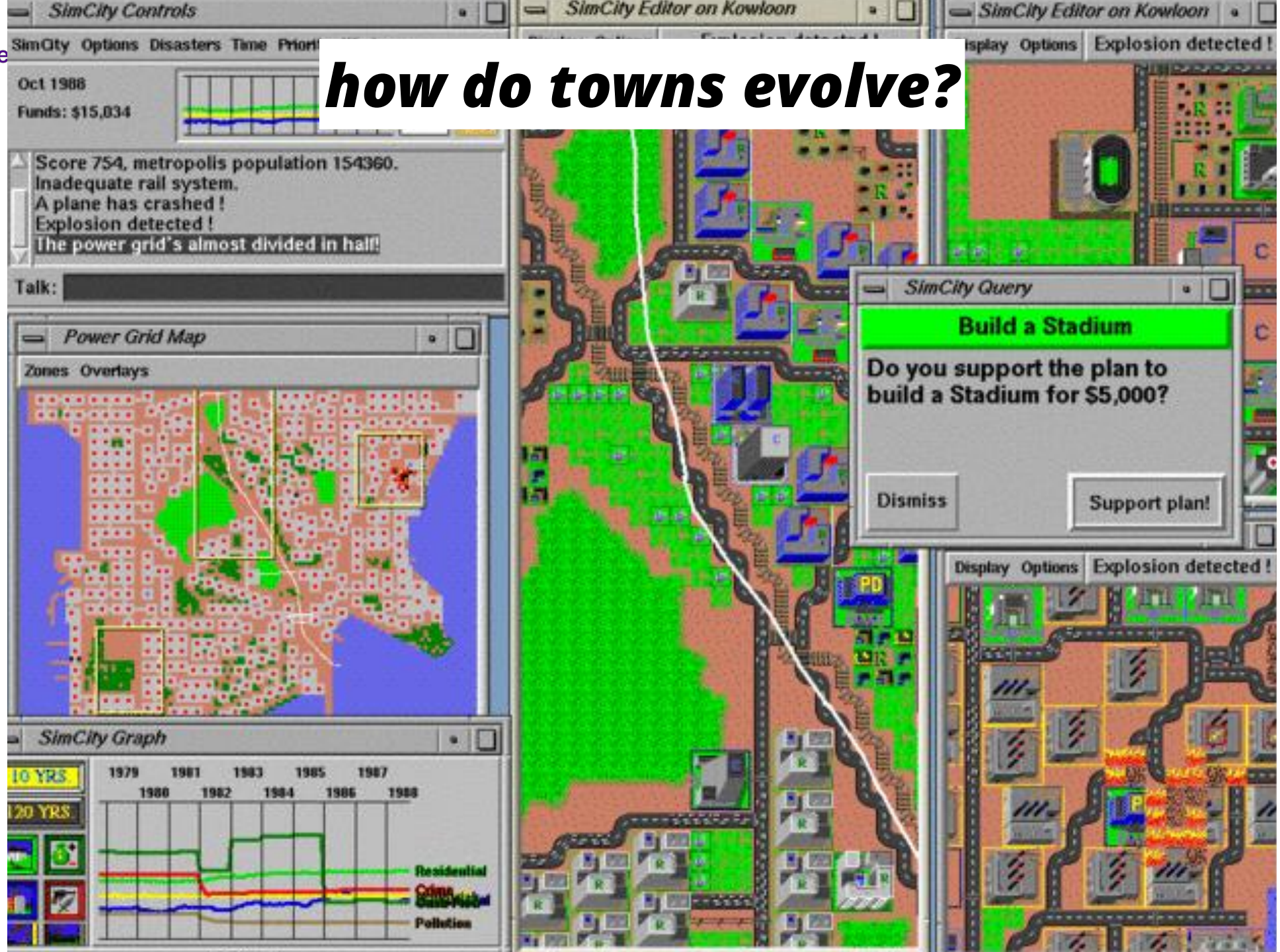
cloud compute and programmable infrastructure has matured

organisations need to adapt and change quickly to survive

we spend **too much money** on building monoliths

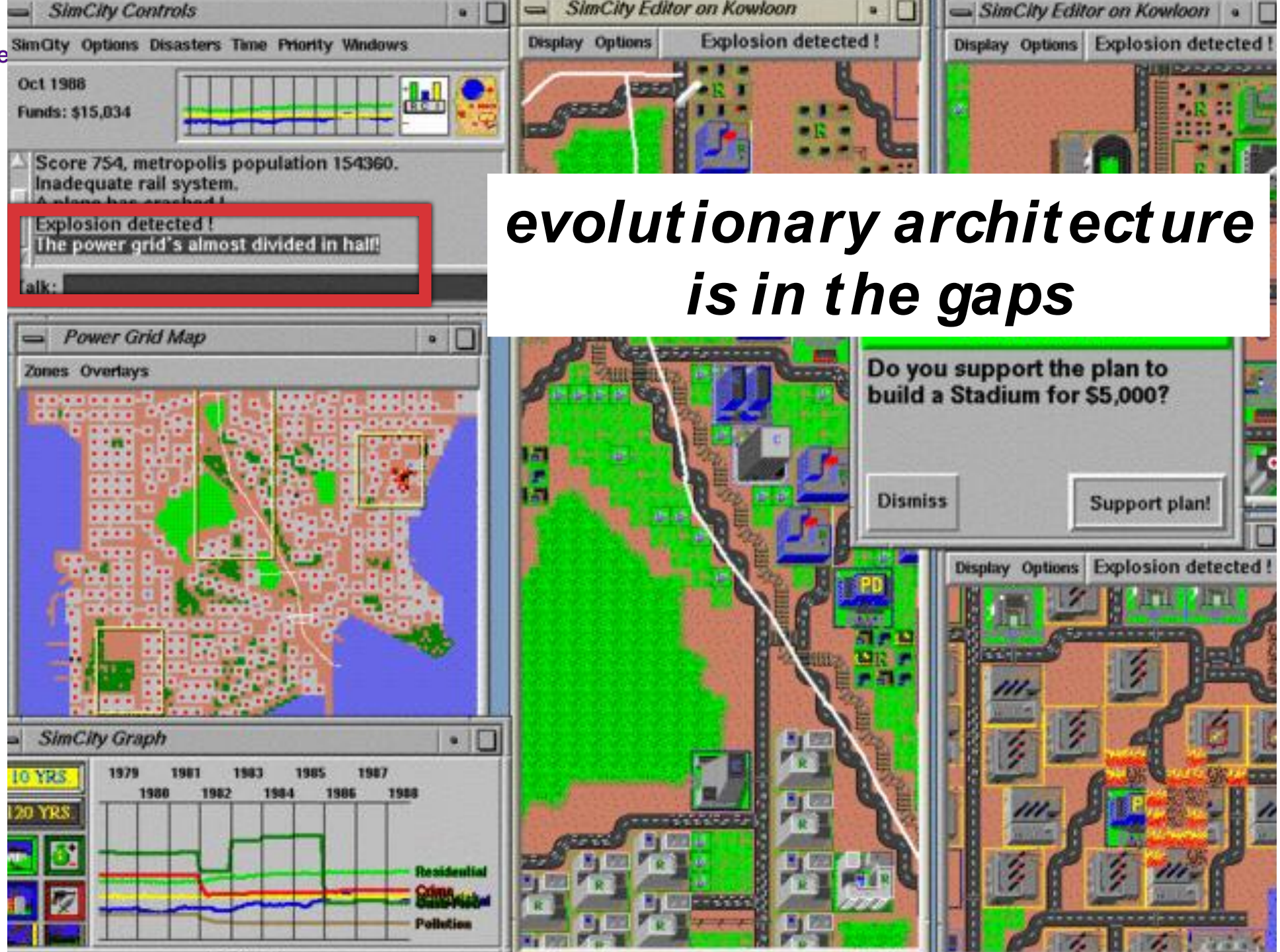
Tampere Unive

how do towns evolve?

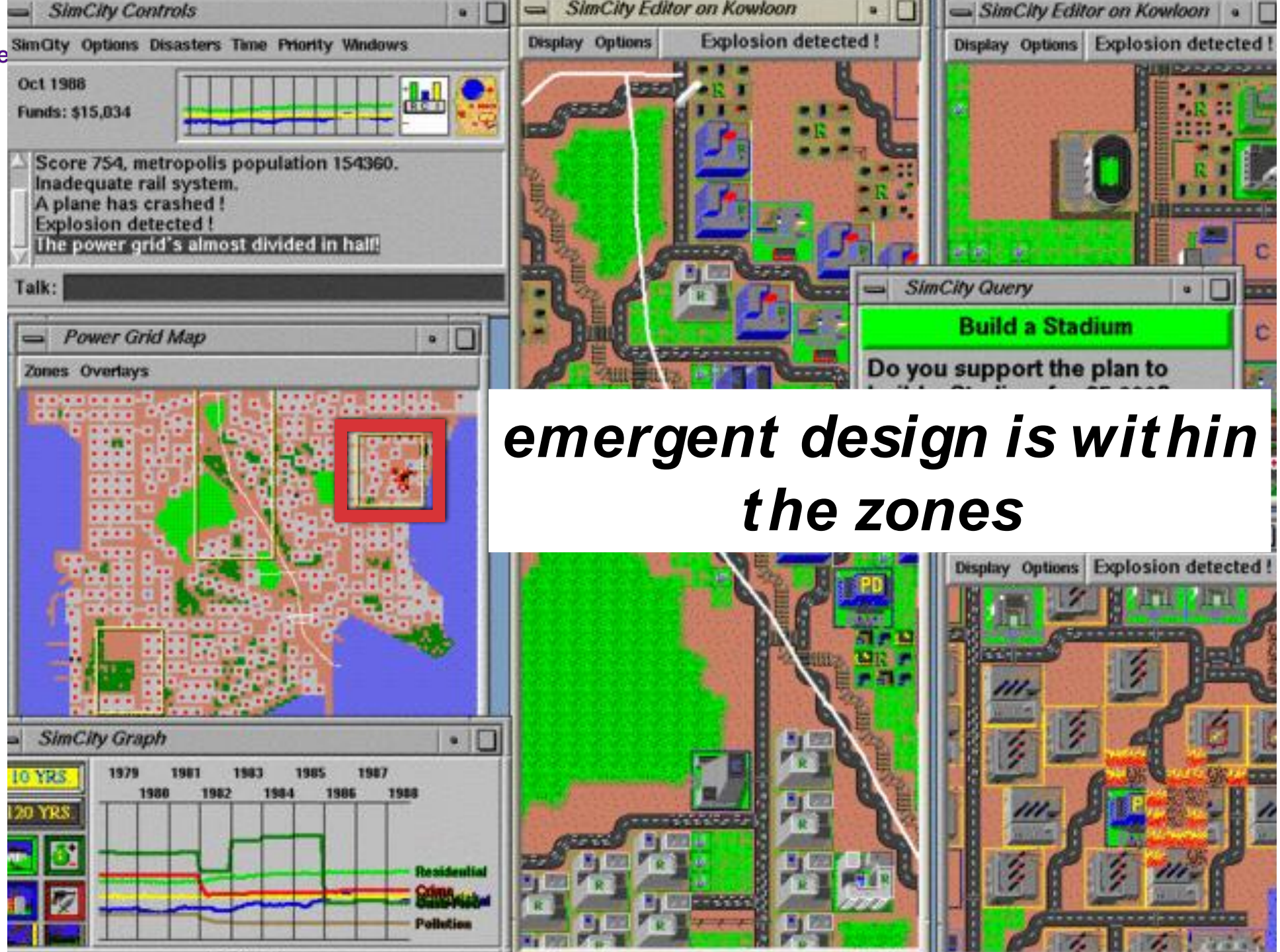


Tampere Unive

evolutionary architecture is in the gaps

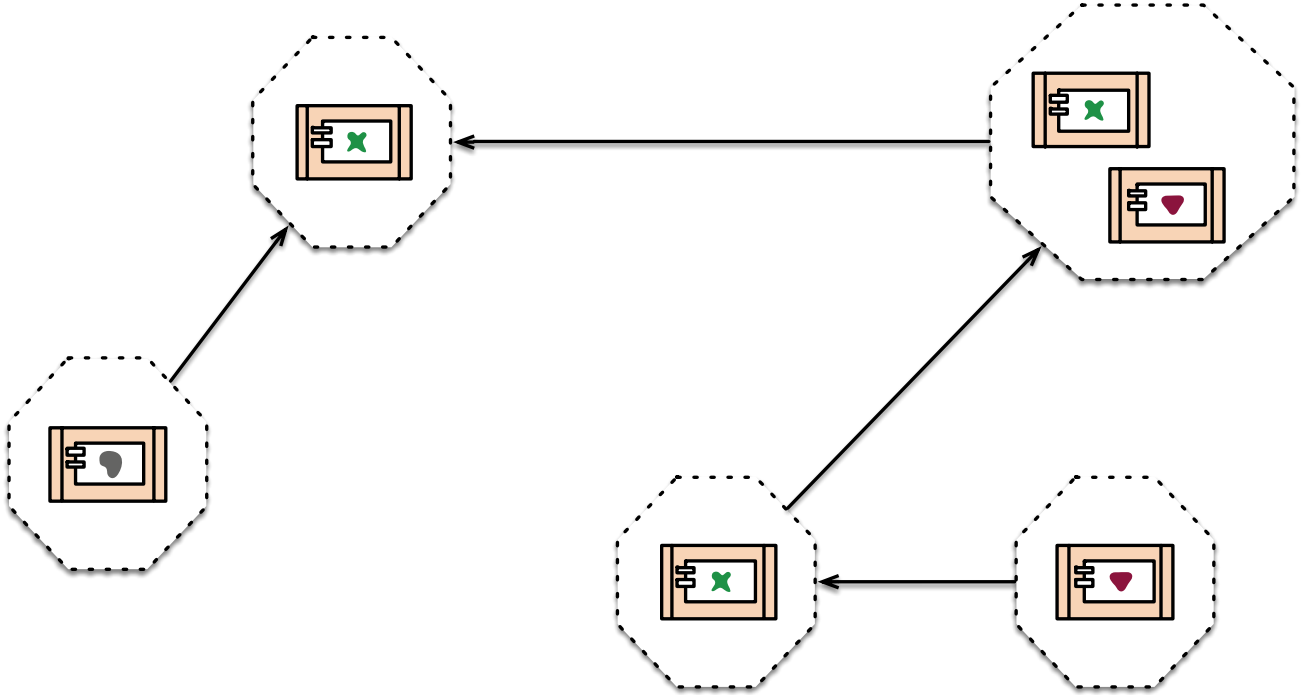


Tampere Unive



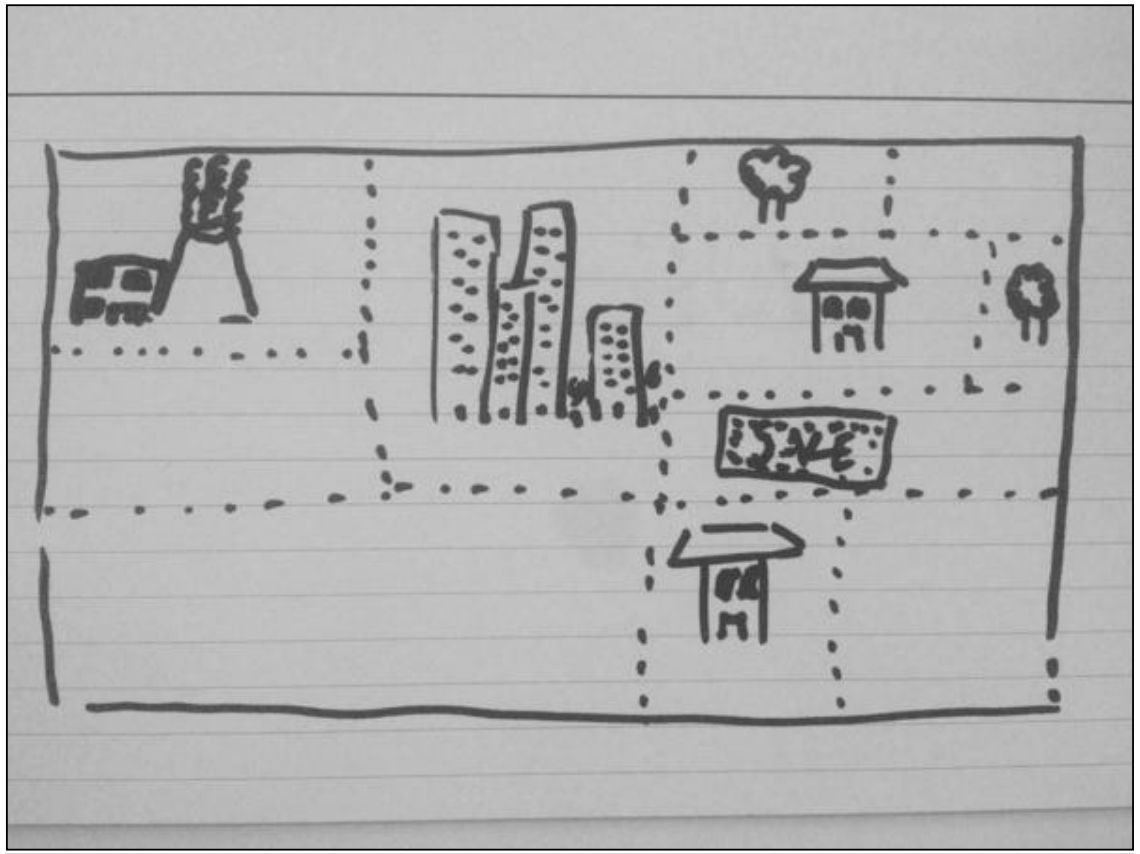
emergent design is within the zones

ALLOW BUSINESS CAPABILITIES TO EVOLVE



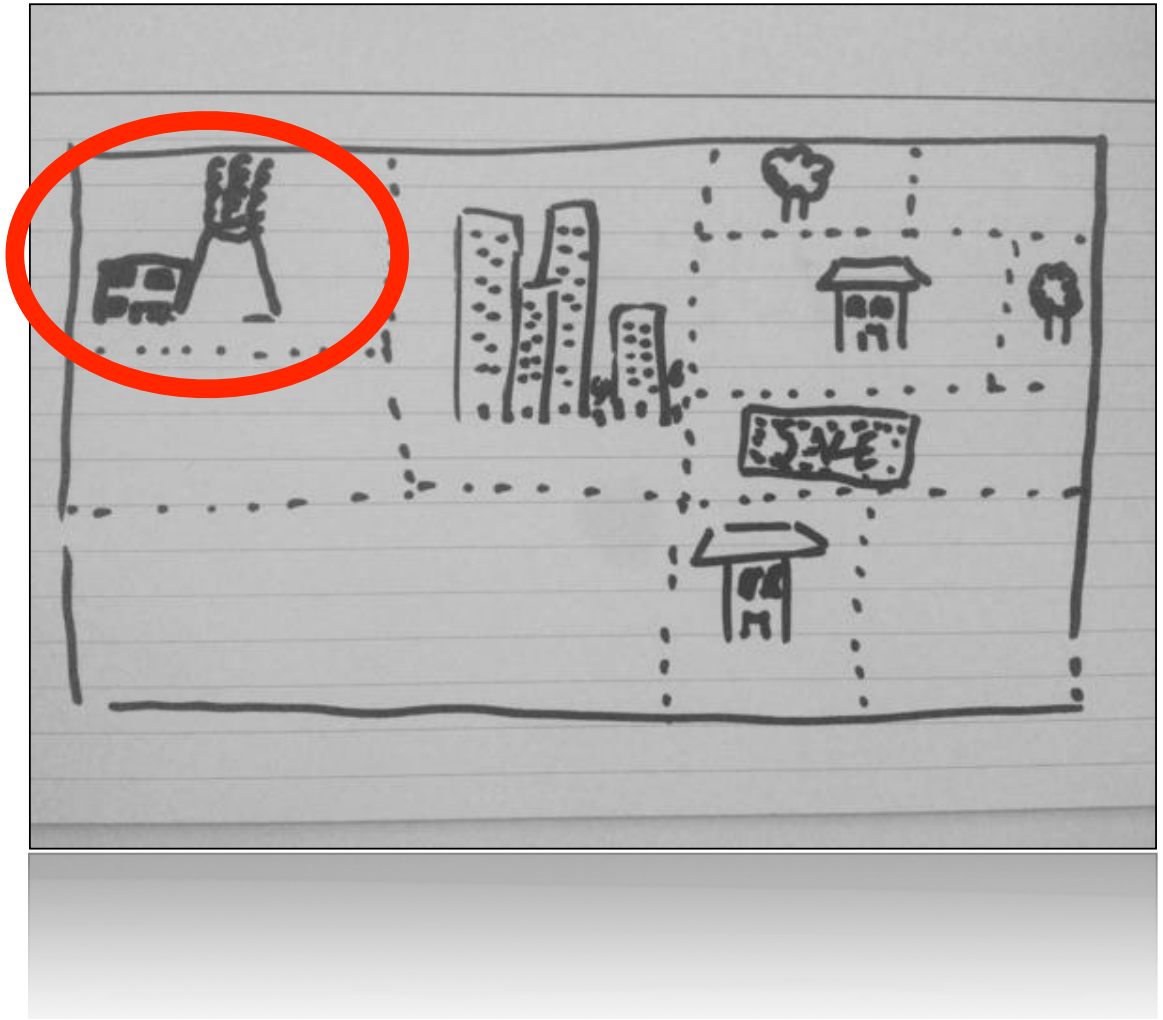
HAVE A ROUGH IDEA ABOUT WHAT YOU WANT TO BUILD, AND DEFER DECISIONS UNTIL YOU KNOW MORE


TOWN
PLAN

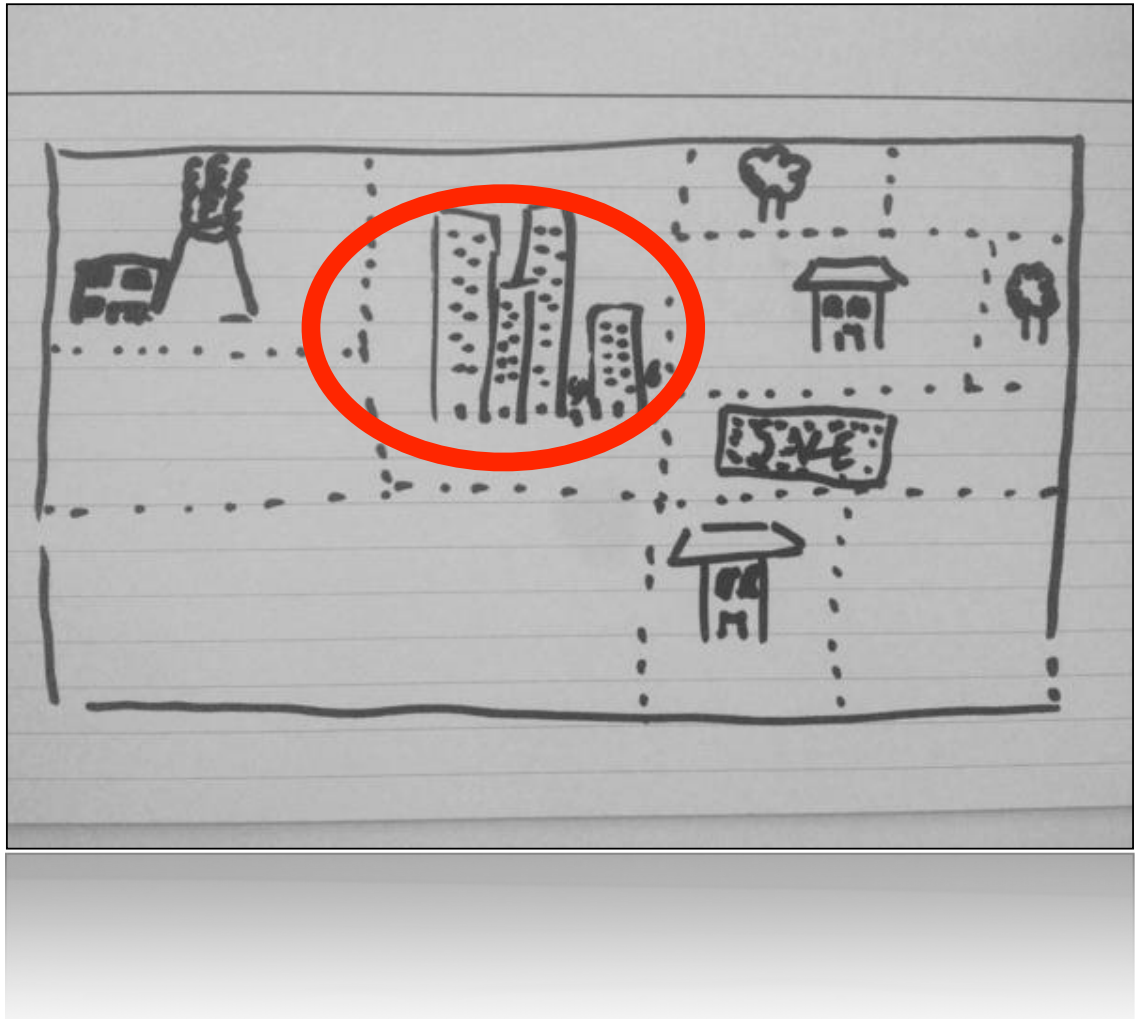

SALE

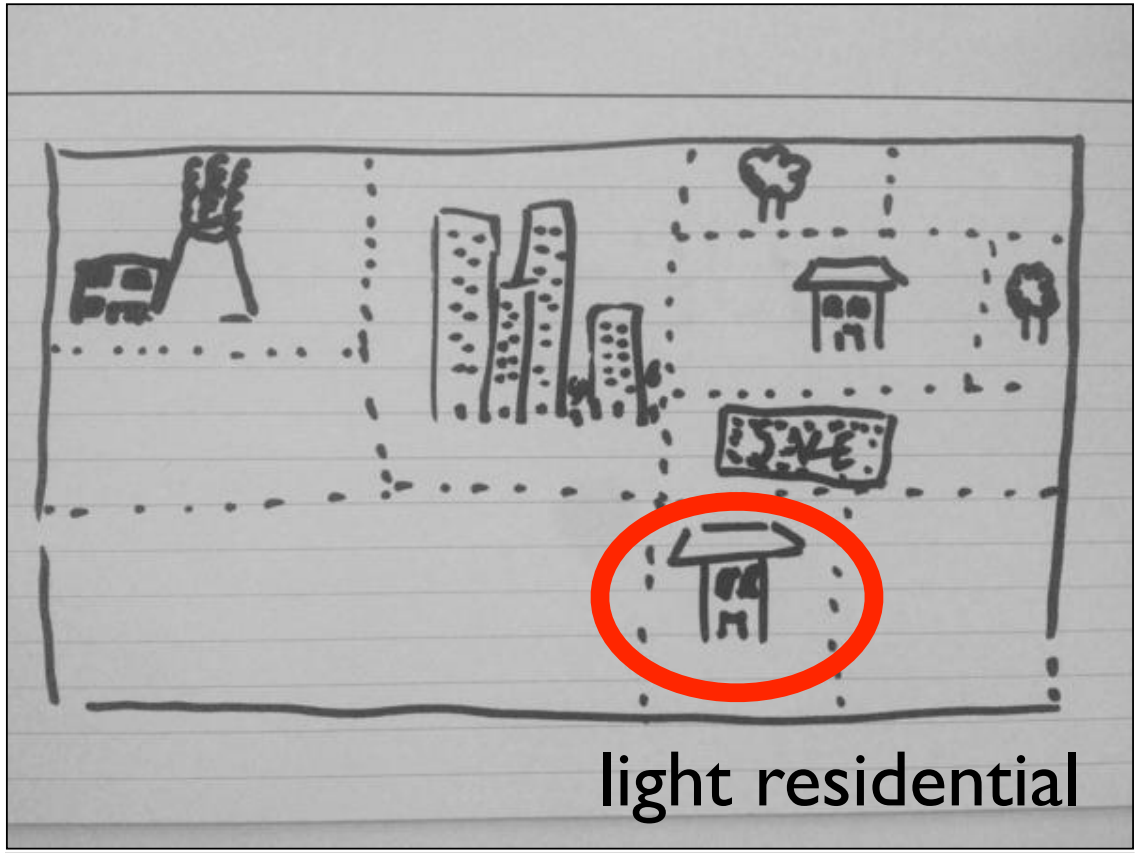
- Towns are Zoned

heavy industrial



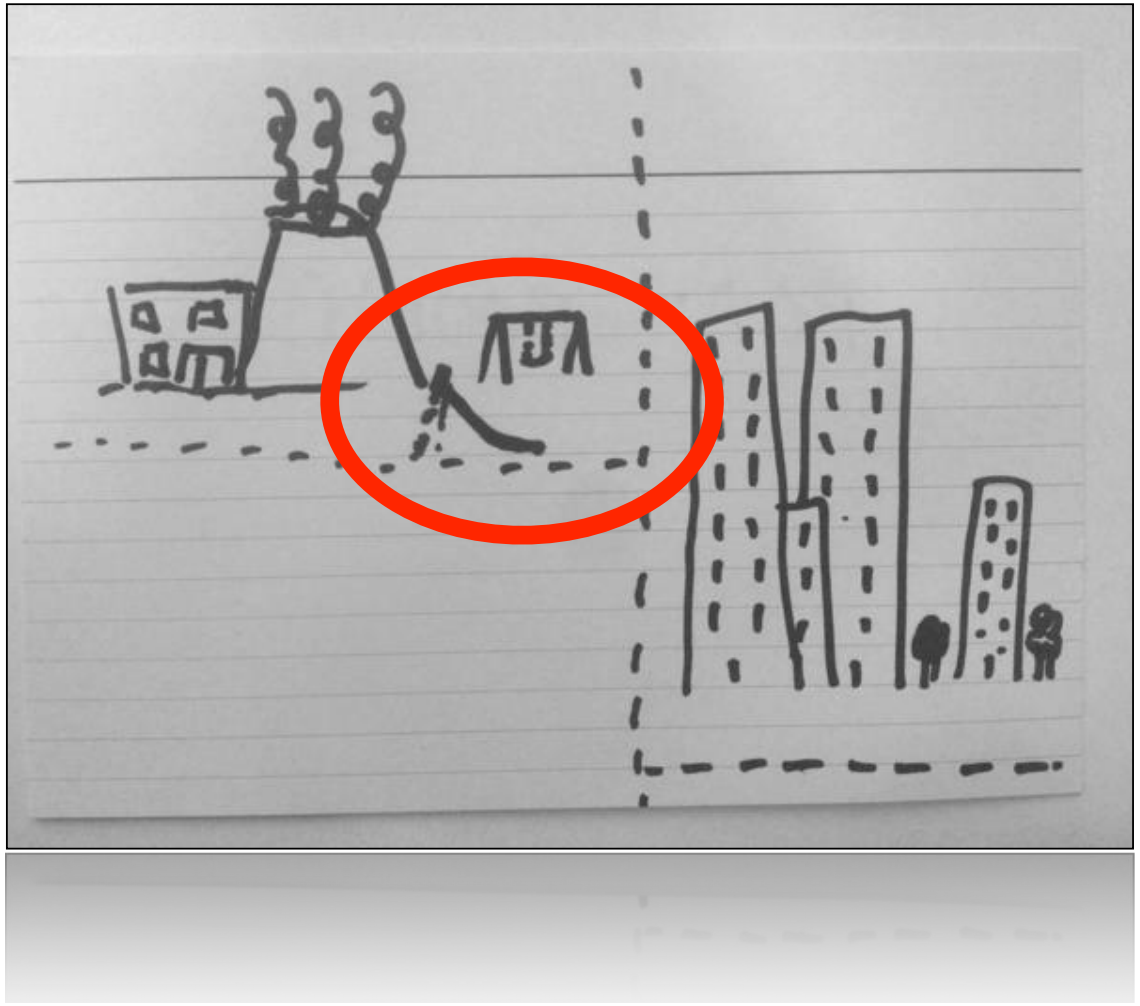
commercial




SALE
light residential

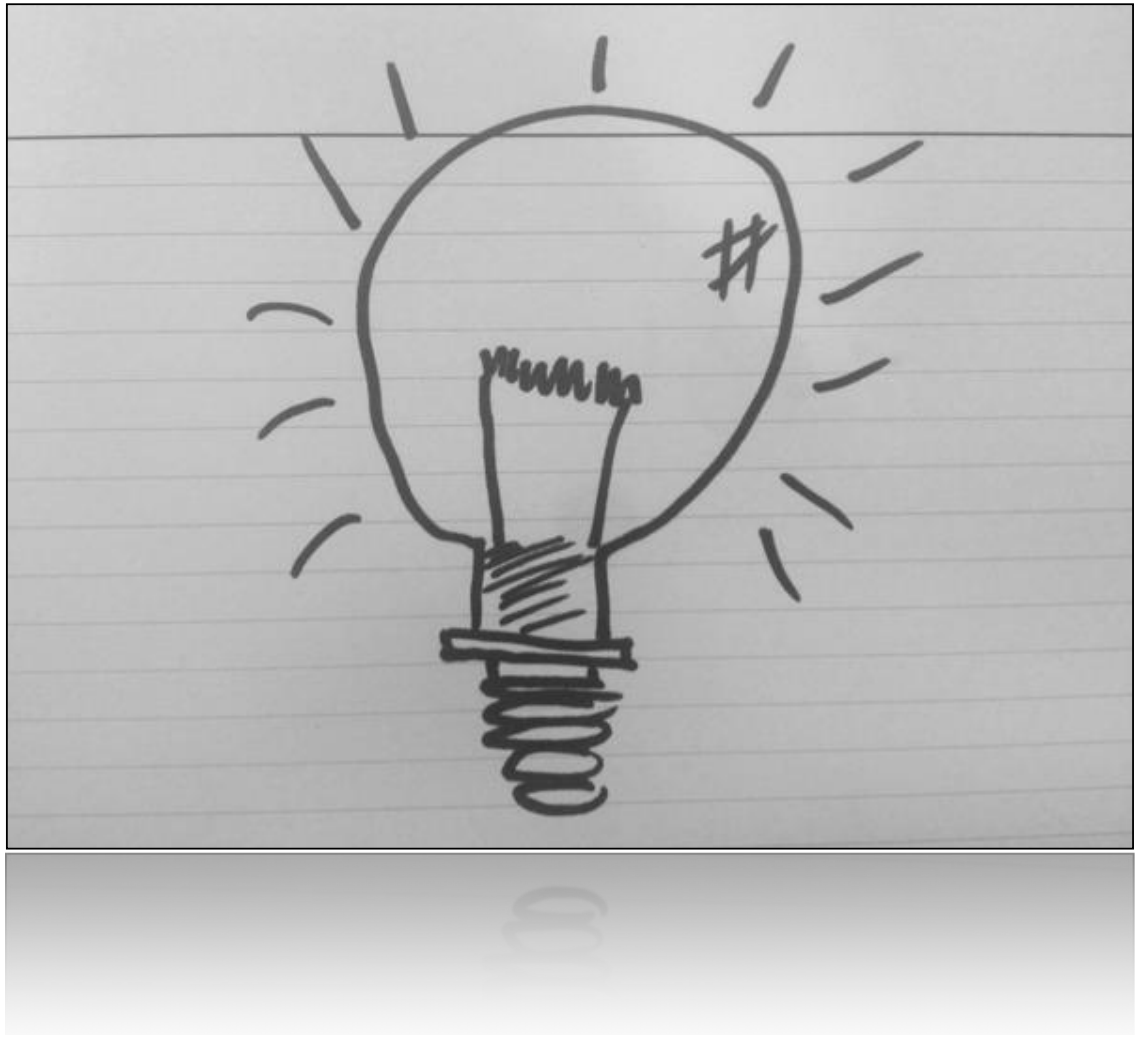
Tampere Unive

Would you build a playground next to a power station?



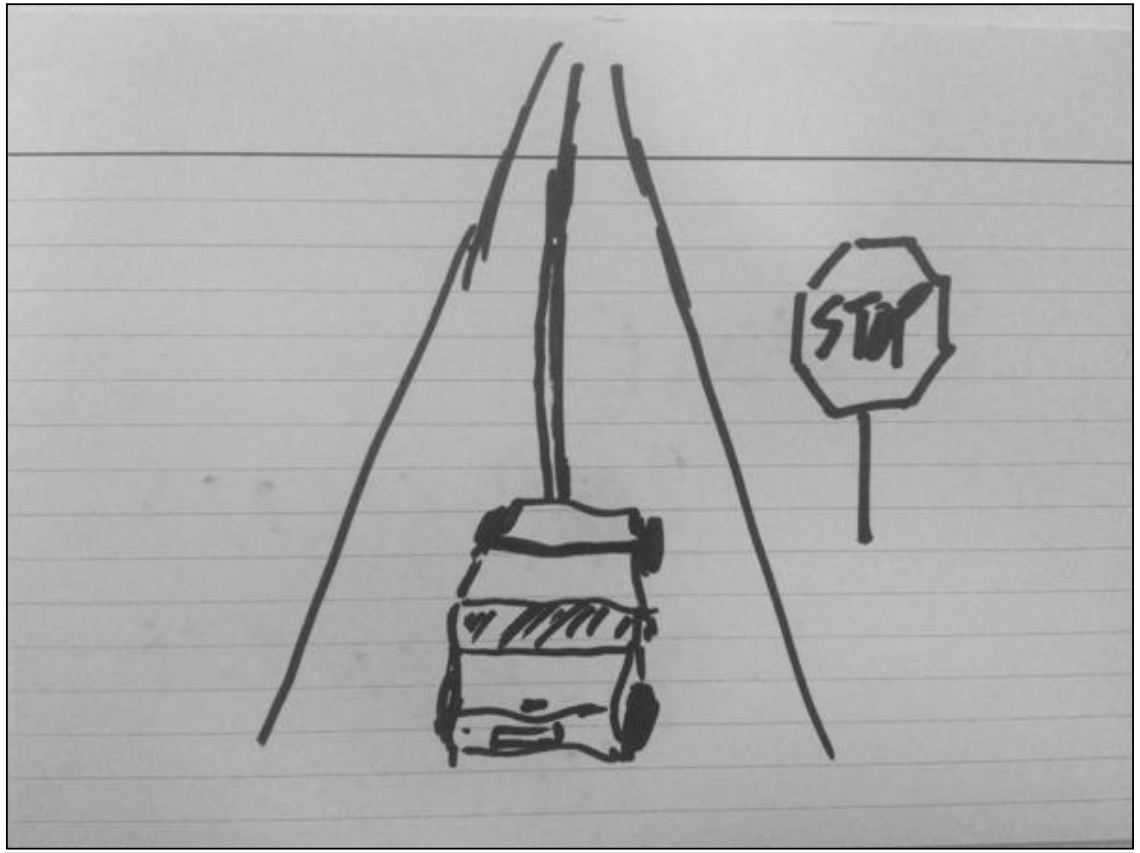
- Town share utilities

Everyone uses 240V DC right?



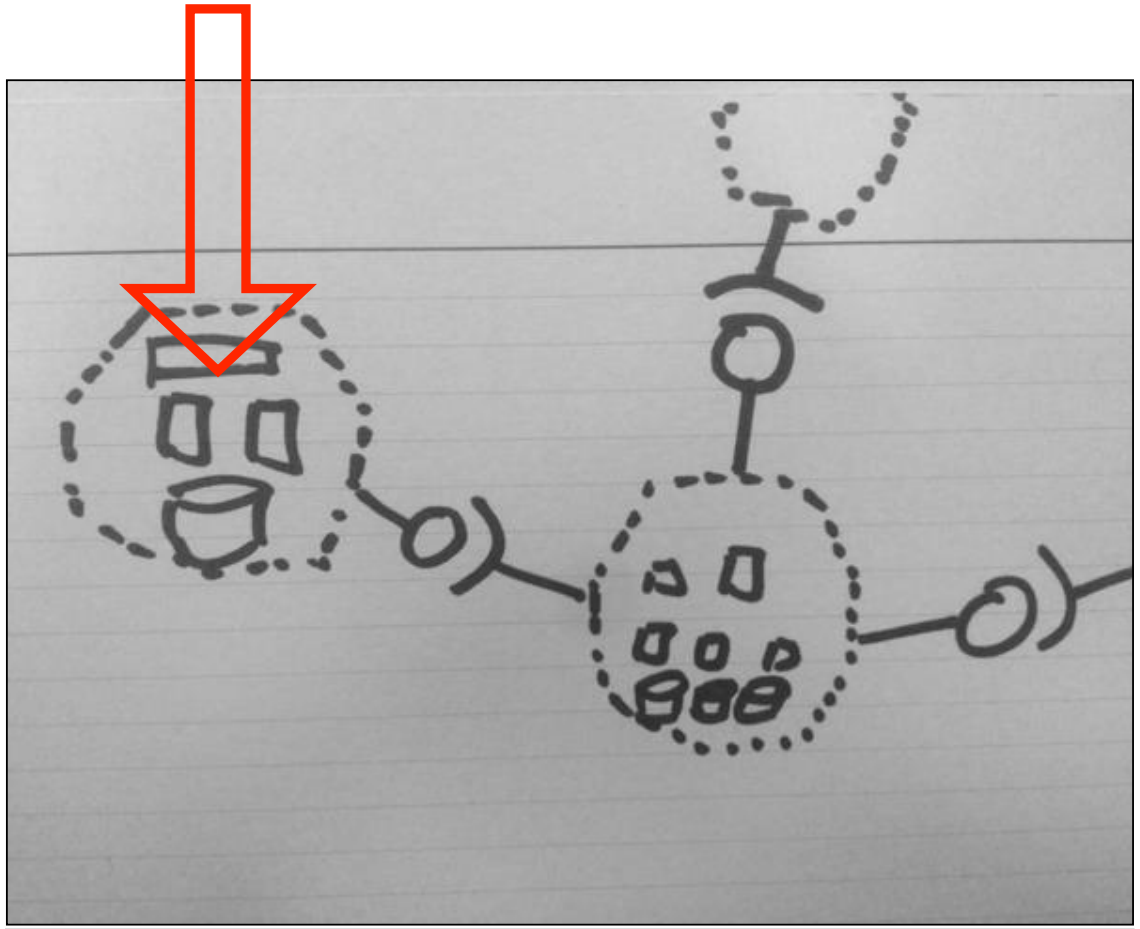
Tampere Unive

and it would be a bad idea not to use the same language for stop signs...



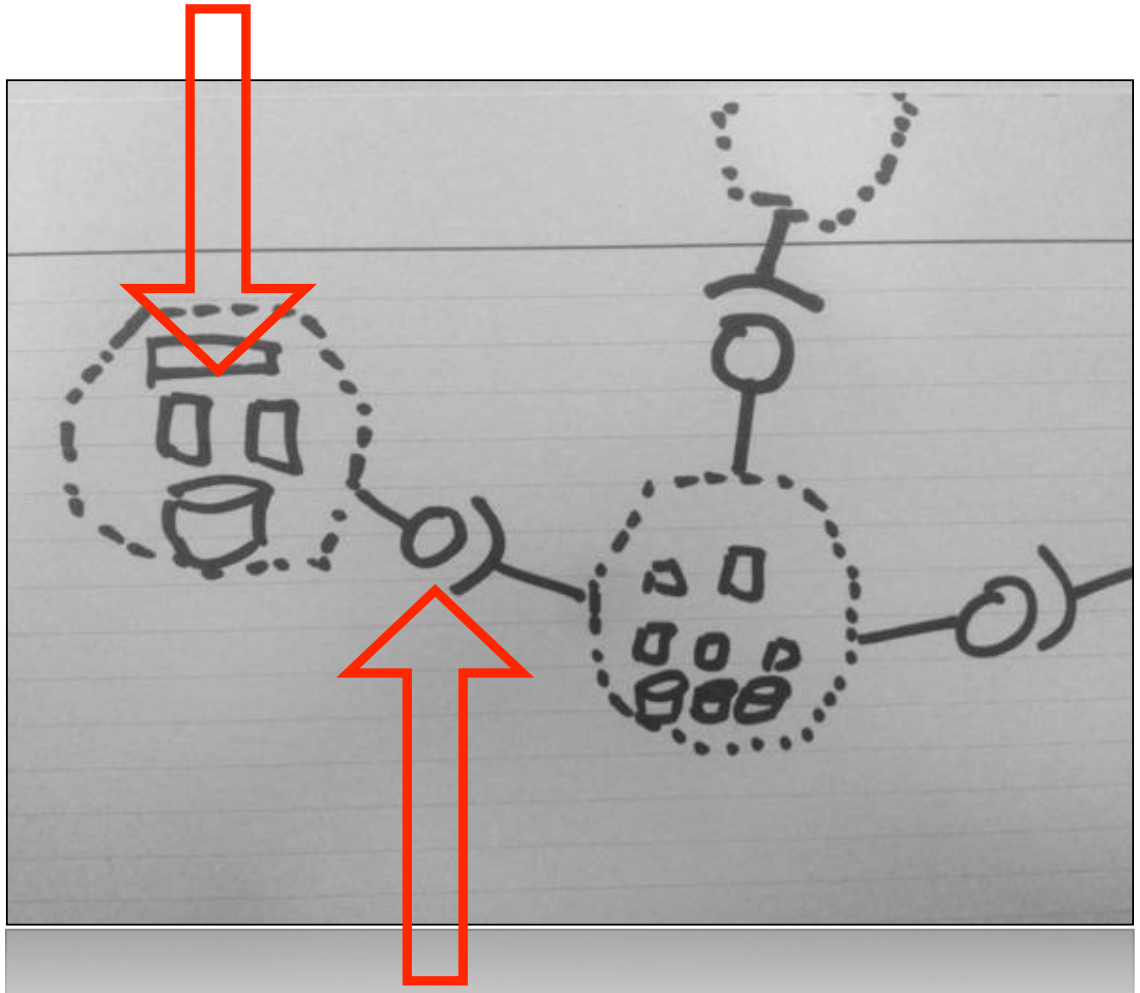

CAPABICITY
MAP

emergent design is within the **zones**



evolutionary architecture is in the **gaps**

emergent design is within the **zones**



evolutionary architecture is in the **gaps**

Think about

- Concentrate on the business capabilities
  - technical acronyms make us think the wrong way
- What are the common features? Integration methods?
- What different types of data live where?

Evaluating the Architectural Quality in the Cloud Era

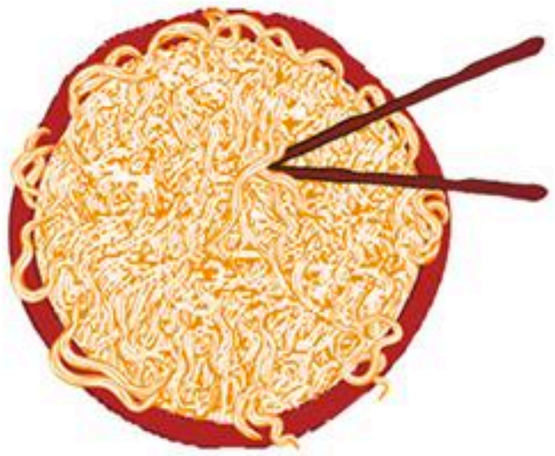
Davide Taibi

Professor. University of Oulu

Software Evolution: Organization

1990s and earlier

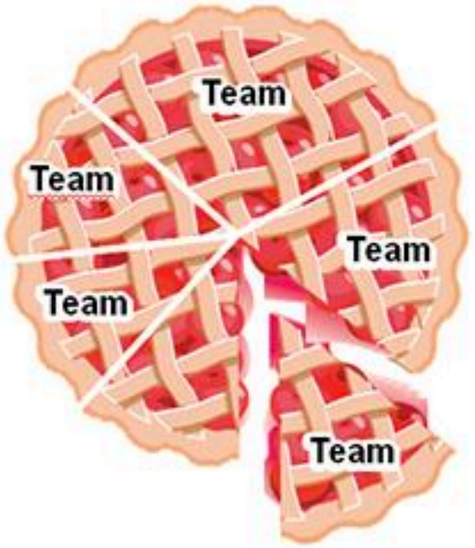
Pre-SOA (monolithic)
Tight coupling



Traditional Metrics
Monolithic-Specific

2000s

Traditional SOA
Looser coupling

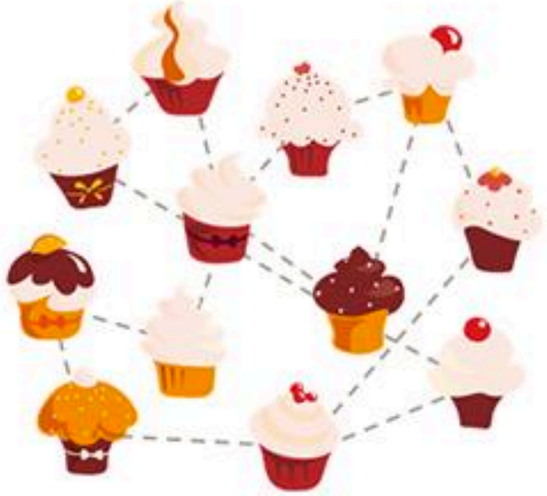


Traditional Metrics
Monolithic-Specific

Distributed-System Metrics

2010s

Microservices
Decoupled



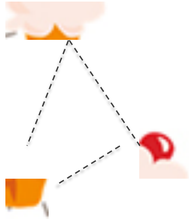
Traditional Metrics
Monolithic-Specific

Distributed-System Metrics

Microservice-Specific Metrics

2015s

Function as a Service
Highly Decoupled



Service Oriented Architectural Quality: Metrics

- *Size*
- *Complexity*
- *Coupling*
- *Cohesion*

Justus Bogner, Stefan Wagner, and Alfred Zimmermann. 2017.
Automatically measuring the maintainability of service- and microservice-based systems: a literature review.
IWSM Mensura '17

How to evaluate Architectural Quality

Software Architecture:

Set of structures needed to reason about a software system and the discipline of creating such structures and systems.

How to evaluate Architectural Quality

Software Architecture:

Set of structures needed to reason about a software system and the discipline of creating such structures and systems.

How to evaluate Architectural Quality

Software Architecture:

Set of structures needed to reason about a software system and the discipline of creating such structures and systems.

How to measure the software structures

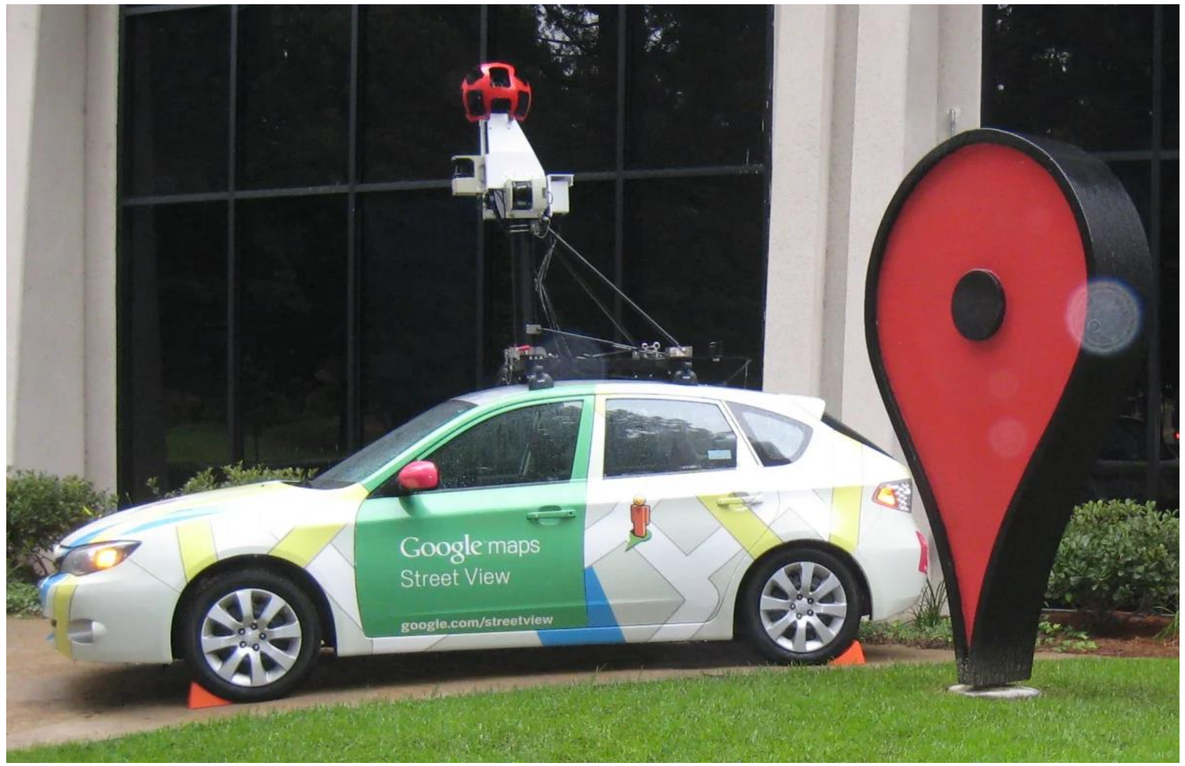
Reverse Engineering

- Architectural reconstruction
- Business process mining
- Organizational structure reconstruction

Architectural Reconstruction

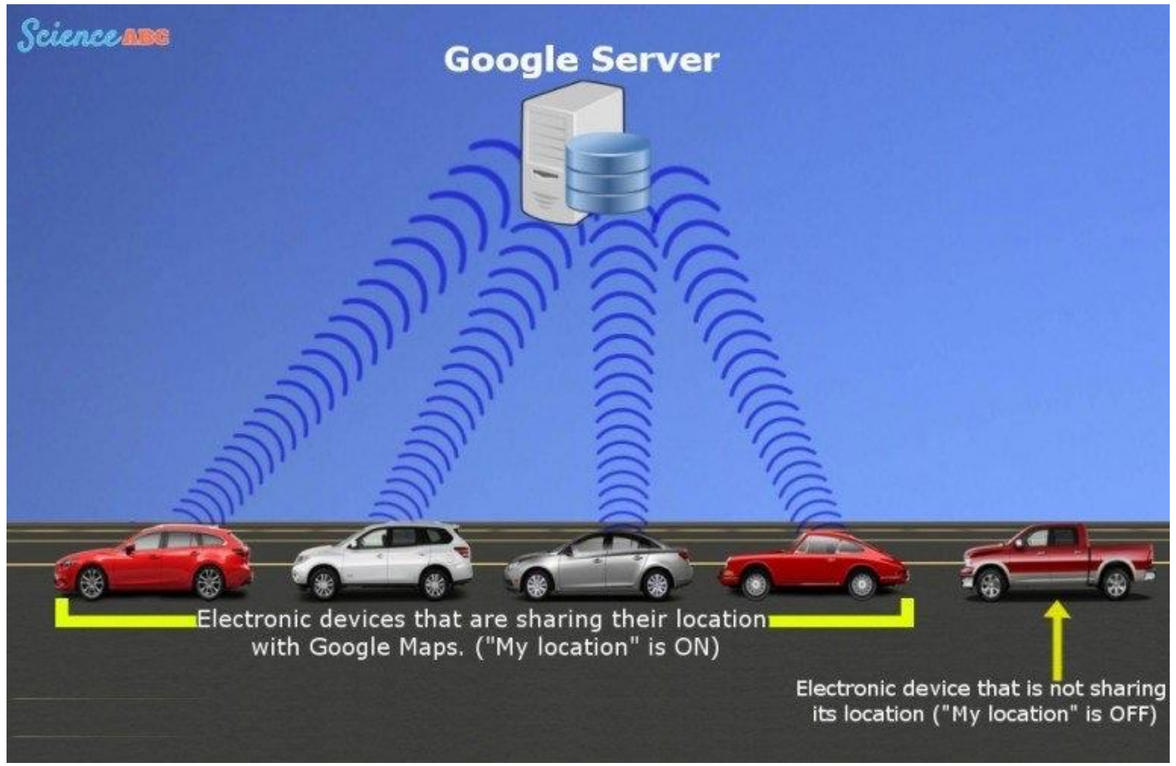
- Static
  - From source code
- Dynamic
  - From the execution of the system (e.g. log traces)

Static Analysis

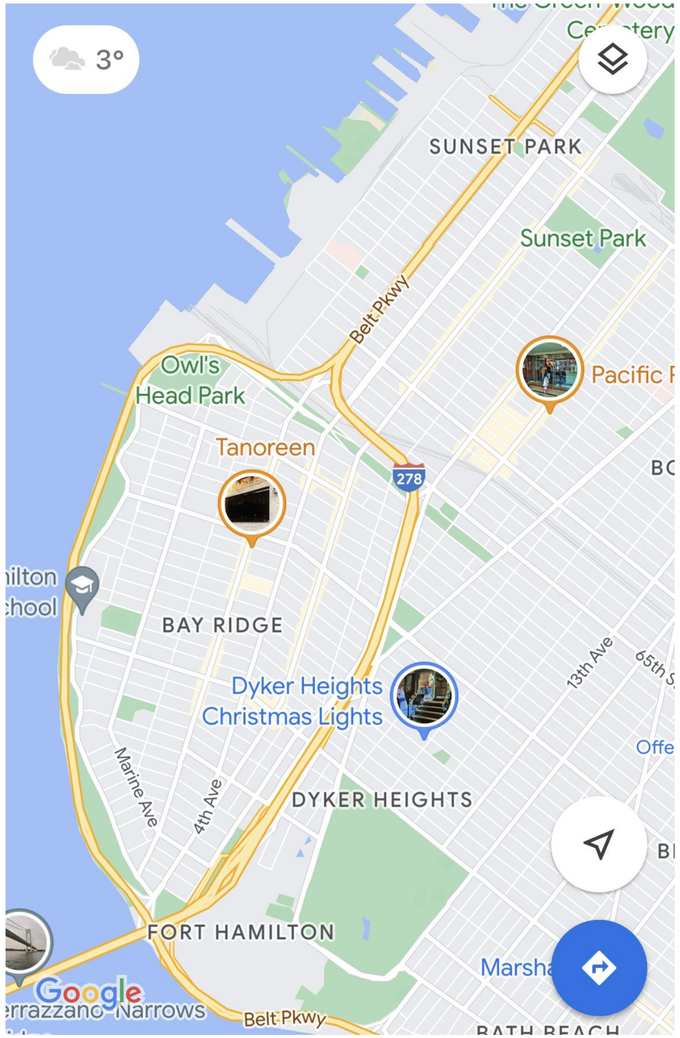


Mapping all the possible “paths” in your system

Dynamic Analysis



Static Analysis

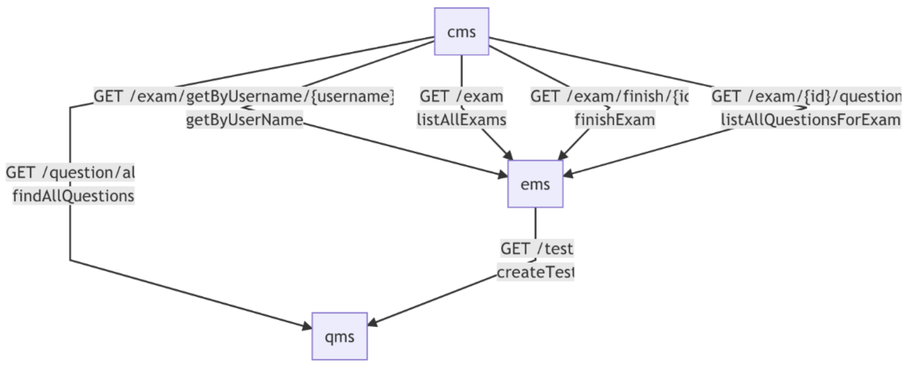


Credits: Google Maps

Static Analysis

- Source Code
- IaC
- Git Log
  - Commit message
  - PR
  - Comments
- Issue Trackers.
- …

Service Call Graph

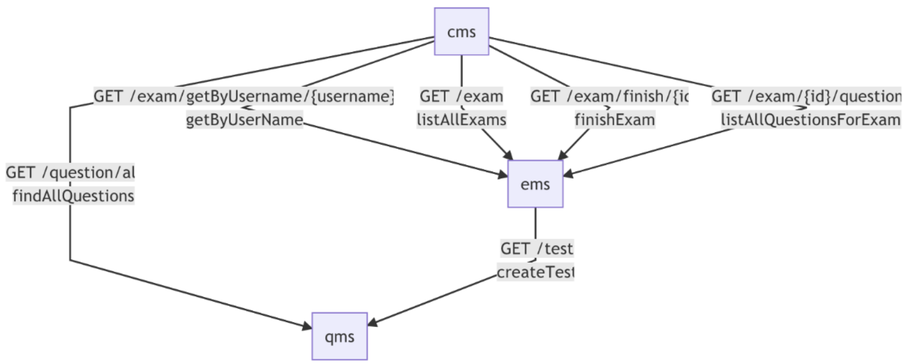


Static Analysis for Microservice-Based Applications
https://github.com/cloudhubs/prophet-web

Alexander Bakhtin, Xiaozhou Li, Jacopo Soldani, Antonio Brogi, Tomas Cerny, and Davide Taibi. 2023. Tools Reconstructing Microservice Architecture: A Systematic Mapping Study. AMC@ECSA Workshop

Static Analysis: Tools

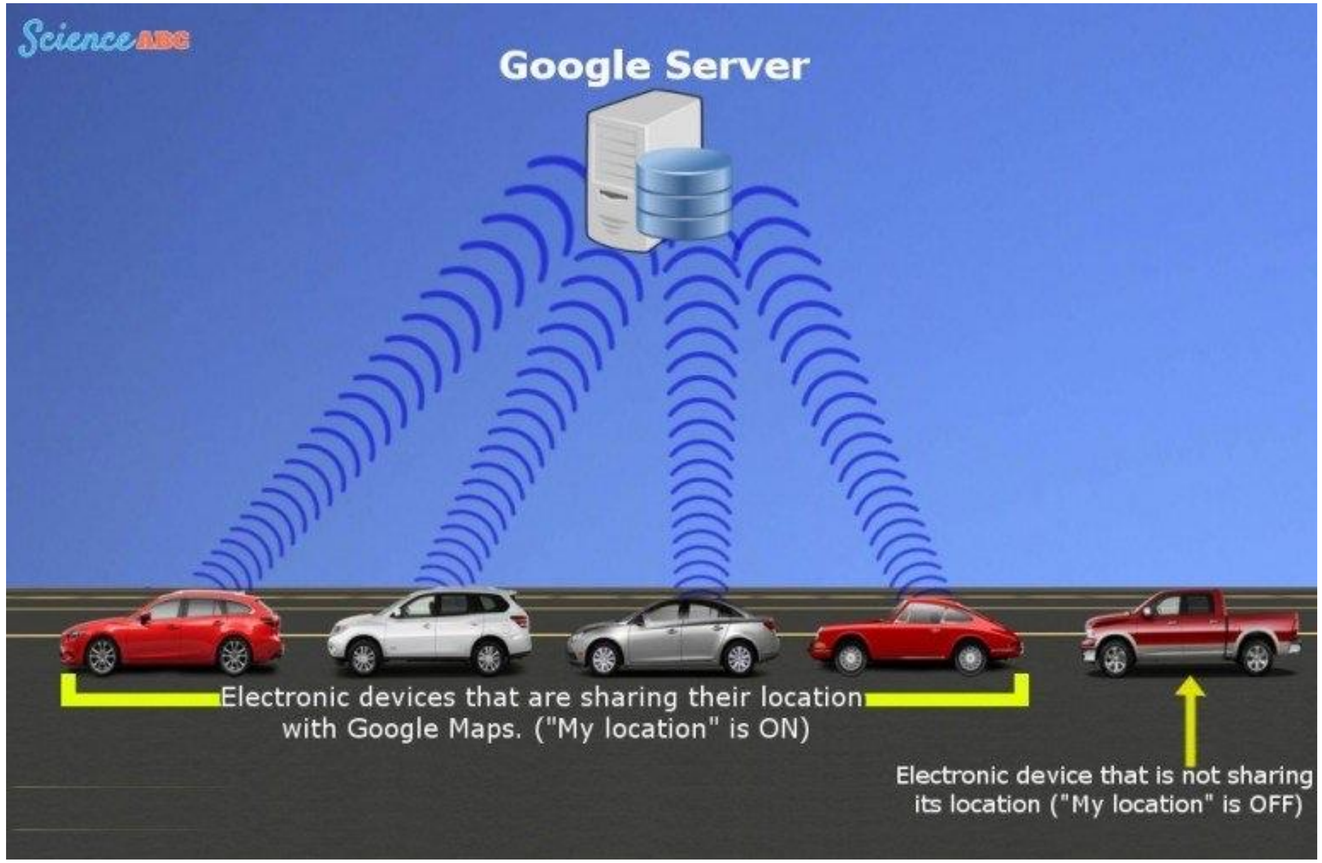
- 19 Research-based tools for Architectural reconstruction [Bakhtin et al]
  - Arcan (Univ. Milano Bicocca)
  - Prophet (Oulu and Baylor University)
  - …



Static Analysis for Microservice-Based Applications
https://github.com/cloudhubs/prophet-web

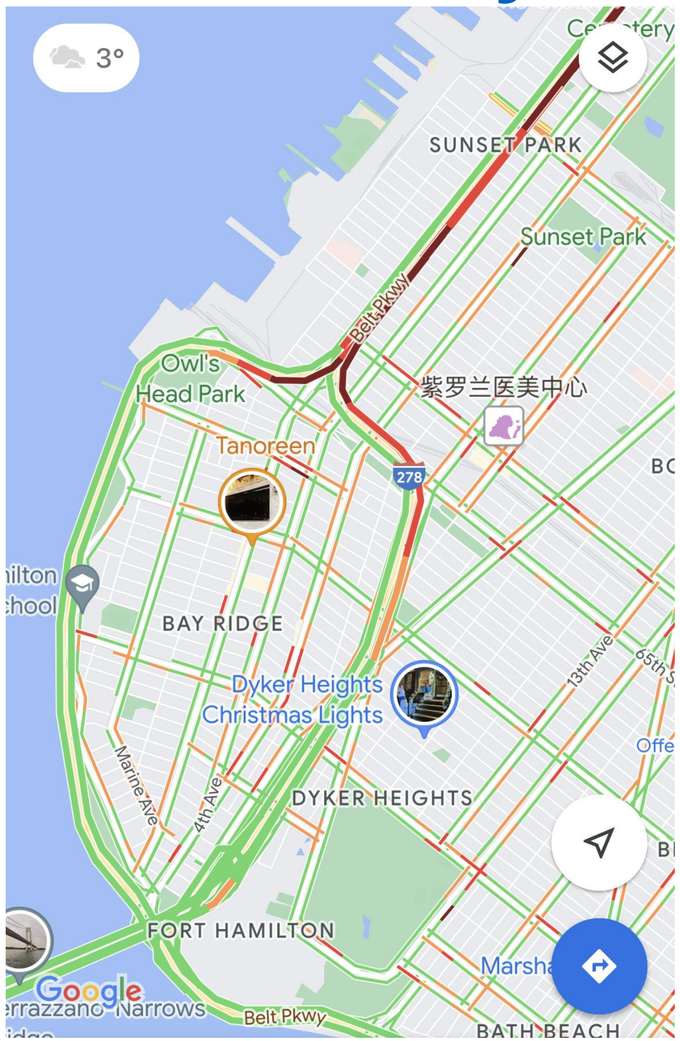
Alexander Bakhtin, Xiaozhou Li, Jacopo Soldani, Antonio Brogi, Tomas Cerny, and Davide Taibi. 2023. Tools Reconstructing Microservice Architecture: A Systematic Mapping Study. AMC@ECSA Workshop

Dynamic Analysis

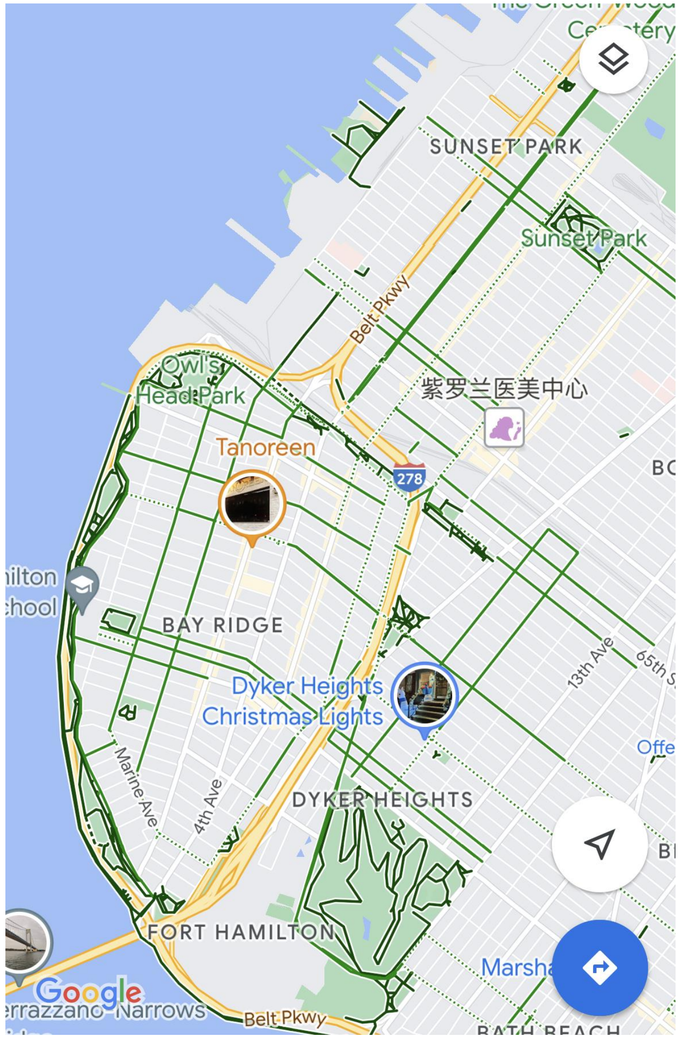


Credits: Science ABC

Dynamic Analysis



Street Traffic View



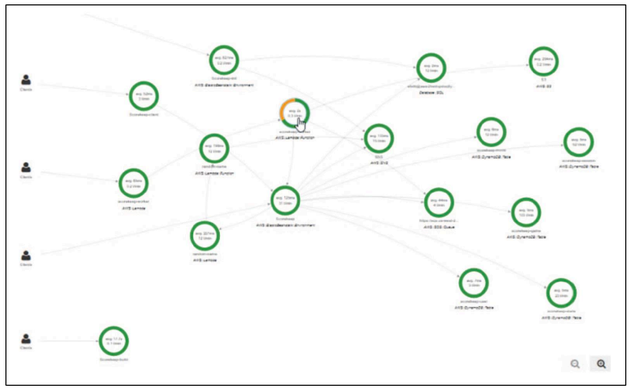
Bicycle Traffic View

Dynamic Analysis: Tools

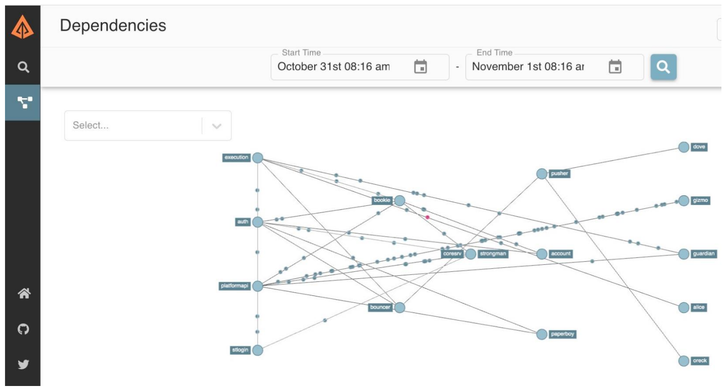
18 tools

(10 commercial, 8 research-based)

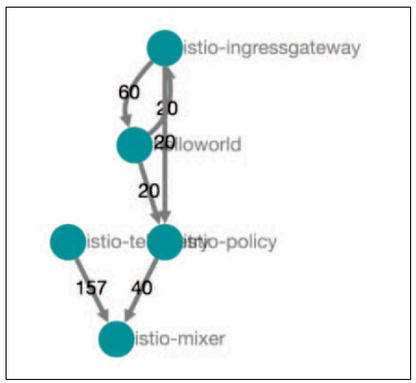
[Bakhtin et al]



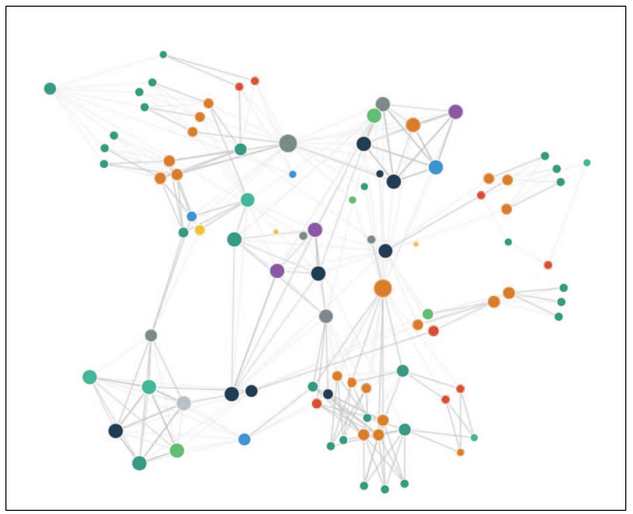
AWS X-Ray



Zipkin



Jaeger



Netflix Interactive Visualization

Alexander Bakhtin, Xiaozhou Li, Jacopo Soldani, Antonio Brogi, Tomas Cerny, and Davide Taibi. 2023.
Tools Reconstructing Microservice Architecture: A Systematic Mapping Study. AMC@ECSA Workshop

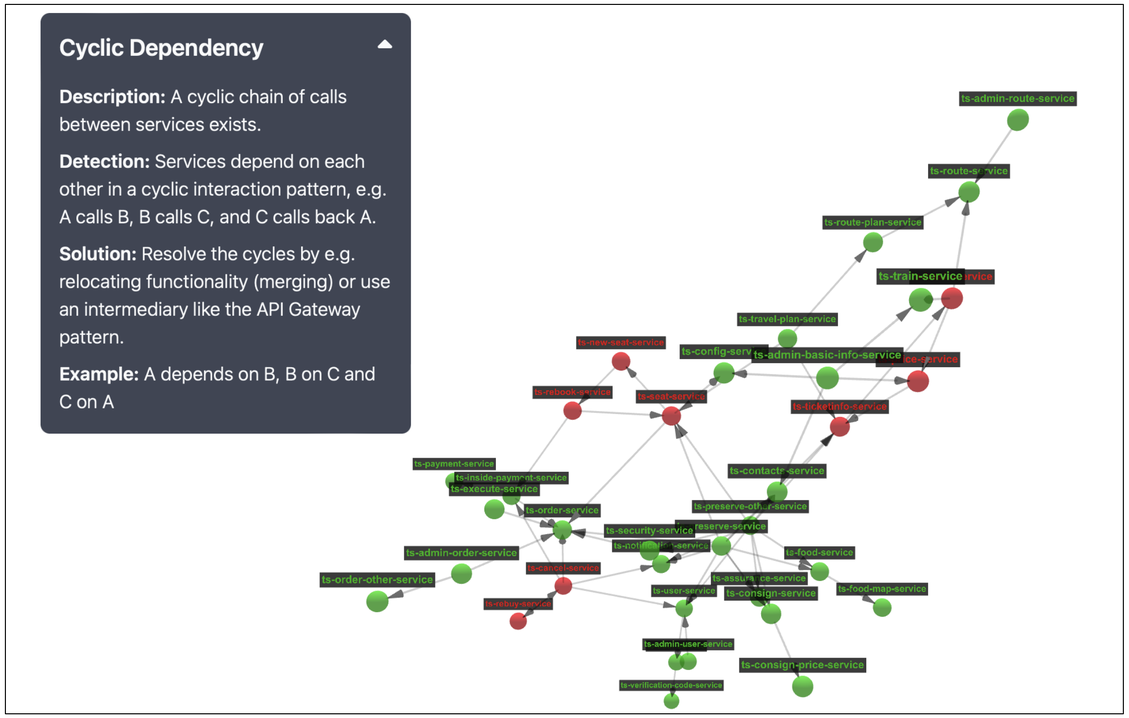
Existing tools shortfalls

- Reconstruction only
- Mainly for visualization
- High potential for SQA

What can we measure from Call Graphs?

Architectural Quality

- Microservice Patterns and Anti-Patterns
  - Identification
  - Detection tools and methods



Anti-Patterns Detection and Visualization

T Cerny, AS Abdelfattah, A Al Maruf, A Janes, D Taibi. Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study. Journal of Systems and Software. 2023

Architectural Quality: Coupling

- Structural Coupling (calls between services)

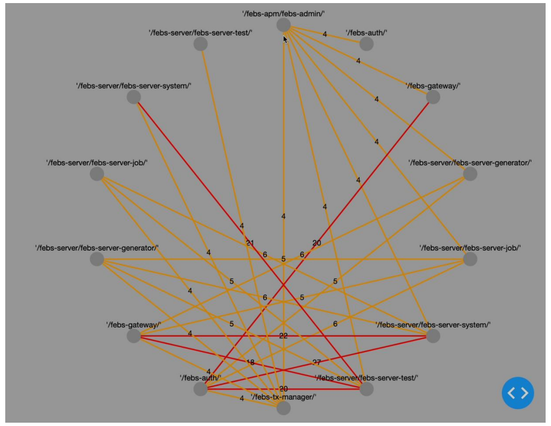
  [1] S Panichella, M. Rahman, D Taibi. Structural Coupling for Microservices. 11th International Conference on Cloud Computing and Services Science 2021

- Cognitive Coupling (co-changes between services)

  [2] D. A. d'Aragona, L. Pascarella, A. Janes, V. Lenarduzzi and D. Taibi, "Microservice Logical Coupling: A Preliminary Validation," IEEE 20th International Conference on Software Architecture ICSA 20223

- Organizational Structure coupling (organizational structure vs sw. Arch)

  [3] Li, X., d'Aragona, D.A., Taibi, D. (2024). Evaluating Microservice Organizational Coupling Based on Cross-Service Contribution. Product-Focused Software Process Improvement. PROFES 2023



Microservice Coupling Visualization [1]

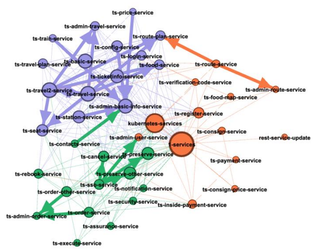
Microservice Coupling and Anti-Pattern Visualization

Software Degradation - Architecture

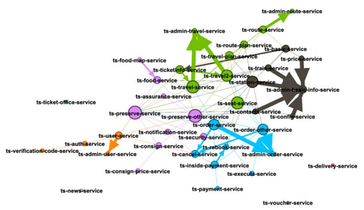
- Temporal Graph Analysis
  - Identification of degradation of metrics, anti-patterns

- CI/CD Integration

Structural Coupling Evolution



Structural Coupling Trainticket V1



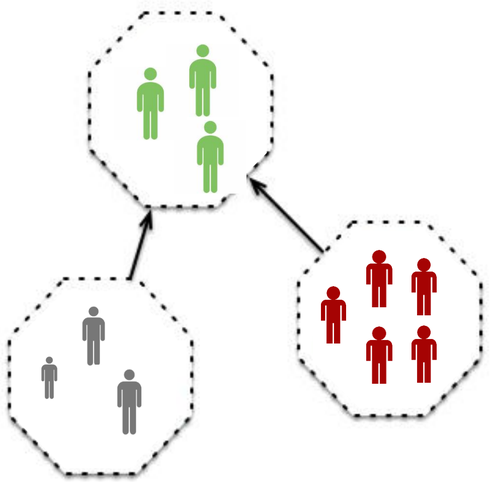
Structural Coupling Trainticket V2

Structural Coupling Trainticket V3

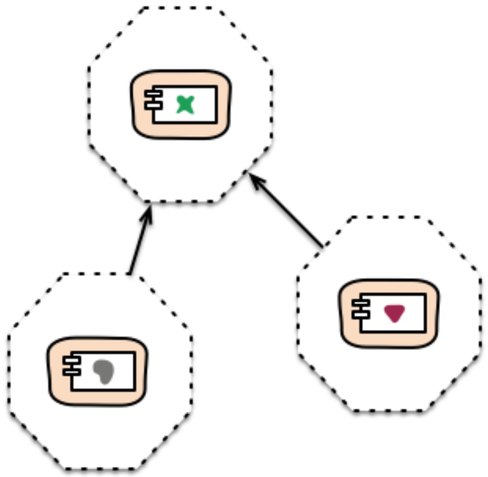
Elsayed A., Li, X., Cerny, T., and Taibi, D. (2024) “Reconstruction of Microservice Domain and Service Views to Reason about System Evolution” 2024 IEEE International Conference on Cloud Computing (IEEE CLOUD)

Organizational Structure vs Architecture

The Dream



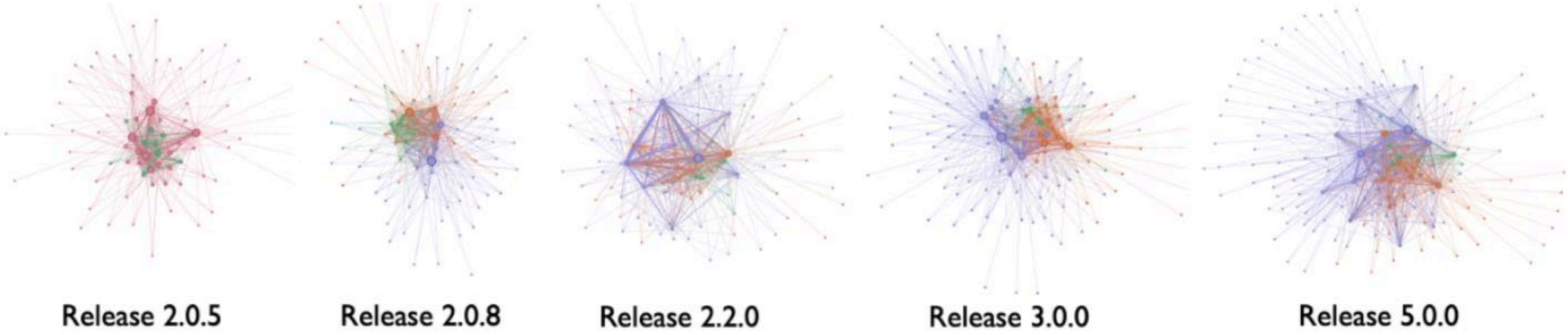
Organizational Structure



Architecture

Organizational Structure vs Architecture

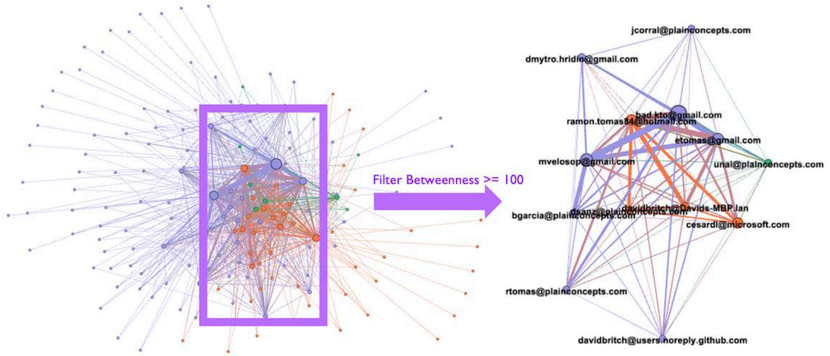
The Reality



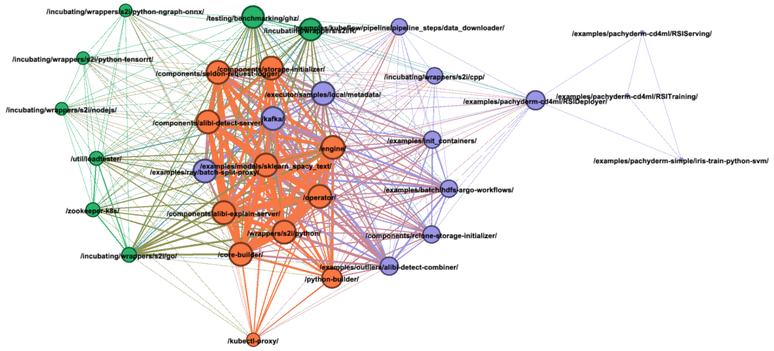
Li, X., Abdelfattah, A. S., Yero, J., d'Aragona, D. A., Cerny, T., & Taibi, D. (2023, July). Analyzing organizational structure of microservice projects based on contributor collaboration. In *2023 IEEE International Conference on Service-Oriented System Engineering (SOSE)* (pp. 1-8). IEEE.

Potential of Architectural Reconstruction

- Identification of possible mismatch
- Recommendations
  - team or SW reorganization
  - Anti-patterns and code smells prevention



Example of developer network analysis



Example of microservice network analysis

[1] A. Bakhtin, X.Li, and D.Taibi. 2024. Temporal Community Detection in Developer Collaboration Networks of Microservice Projects. In Software Architecture: 18th European Conference, ECSA 2024

Serverless Computing
Function-as-a-service

O’Reilly SW Architecture Conference 2018

Stop using microservices!

Move to serverless functions as soon as possible!

What is Serverless [3]

a cloud-native platform

for

short-running, stateless computation

and

event-driven applications

which

scales up and down instantly and automatically

and

charges for actual usage at a millisecond granularity

[3] Baldini I. et al. (2017) Serverless Computing: Current Trends and Open Problems. In: Chaudhary S., Somani G., Buyya R. (eds) Research Advances in Cloud Computing. Springer, Singapore

How does it work

Runs code only on-demand on a per-request basis

Serverless deployment & operations model

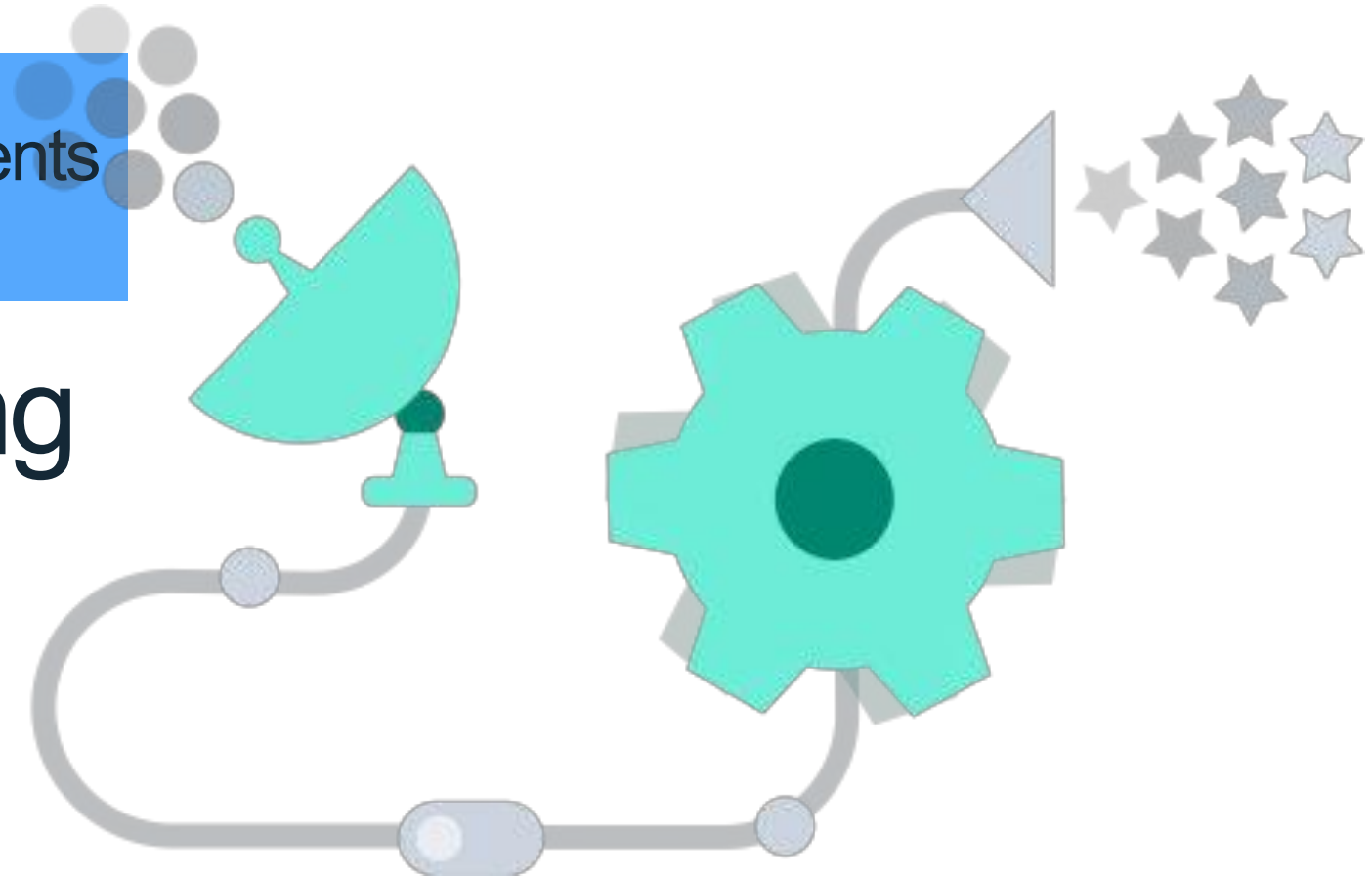
No servers

Just code

What triggers code execution?

Runs code **in response** to events

Event-programming model



Current Platforms for Serverless

AWS Lambda

OpenLambda

Azure Functions

Kubernetes

Red-Hat

Google Functions

Why practitioners are moving to serverless

Migration Motivations

Companies Already moved to Microservices

- OPS Effort for Microservices
- Get rid of Kubernetes
- No OPS

Companies Migrating from Monolithic systems

- New (hype) technology
- Promising technology
- No initial infrastructural costs (pay as you use)
- Automatic scaling
- Lack of skilled OPS personnel

Preliminary Results – Migration Issues

Developers are not used to the event-oriented programming

Very hard to test

Debug almost impossible

Unknown Patterns and antipatterns

Anomalies can generate unexpected costs!

Tampere University

Serverless Anti-Patterns

Preliminary Results presented
@ICSA 2020

Serverless Anti-Patterns Summary

#1 Async Calls

#2 Functions calling other functions

#3 Shared Code

#4 Shared Libraries as Functions

#5 Too many libraries

#6 Too Many technologies

#7 Too many functions

EXTRA: The distributed Monolith

Open Questions - Serverless

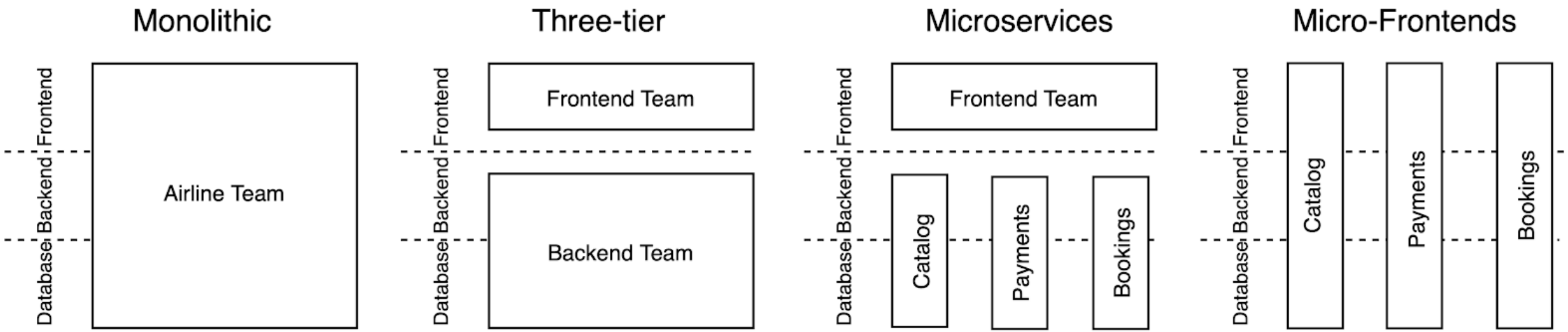
- When is better to use serverless and when microservices
- How to architect a system based on serverless functions?
  - Or to combine functions to create a microservice?
- Architectural Patterns? Anti-Patterns?
- How to prevent anomalies?

Microservices and FaaS

- Practitioners started migrate to microservices and FaaS
- Mixed Approach (microservices + Functions)
- **Open Issues**
- When and Why Extract a feature as Function or as Microservice?
- Which pattern should be adopted

Micro-Frontends

From Microservices to Micro-Frontends



Micro-Frontends

- Adopted by several large-companies
  - SAP, Zalando, Springer, NewRelic, Ikea, Starbucks, Spotify, DAZN, …
- Increase the team velocity
- But… create duplications in common parts

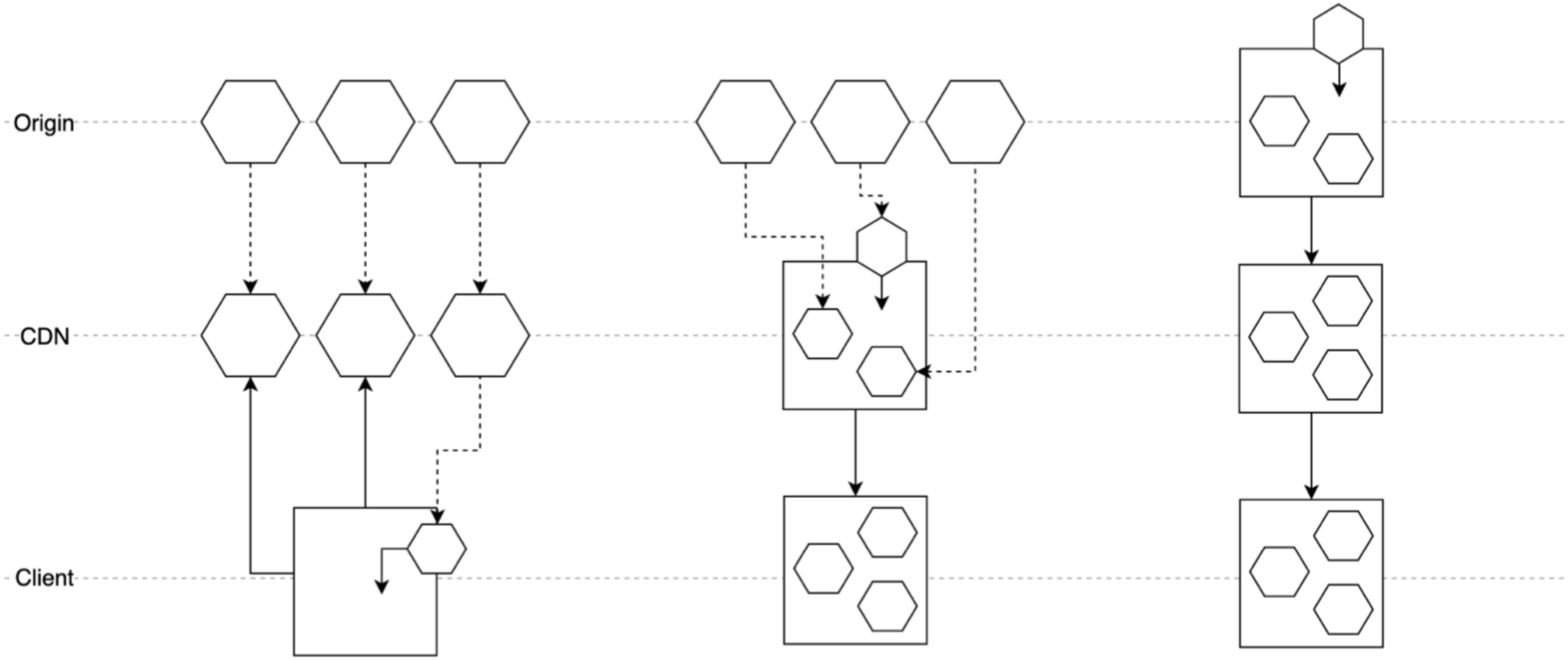
Micro-frontends - Motivations

- Decompose the front-end into individual and semi-independent micro applications.
- In microservice-based systems, the frontend is implemented as a monolithic (or fat)
- Microservice and frontend teams need to synchronize changes

Micro-Frontends – How they are implemented

- One micro-frontend per page
  - Each team develop a page completely
  - Common parts are duplicated (sidebar, header, footers, …)

- Multiple micro-frontends per page
  - Client-side composition
  - Edge-side composition
  - Server-side composition



Client-side composition | **Edge-side composition** | **Server-side composition**

Micro-Frontends – Open Issues

- When and why they should be used?
- Not for every project.
- Are duplications “beneficial”?
  - How about coupling between teams
- Increased application complexity due to page composition at run-time.
- Increased observability complexity, due to the increased number of moving parts.

New Trends

- Serverless and micro-frontends adoption is increasing

- Companies are trying to move the computation closer to the data ➔ Edge Computing
  - New on-premise solutions for edge computing
  - Hybrid cloud/on-premise solutions

Conclusion

- Serverless and Microservices are very powerful and useful technologies
- Still several open questions
- Developers should carefully consider the “old fashioned” software engineering practices
  - Properly design a modular system
  - Pay attention to coupling and cohesion
  - Think about long-term maintenance
  - Avoid the distributed monolith
- Some companies are moving back to on-premise, other are considering hybrid approaches
  - Edge computing is coming