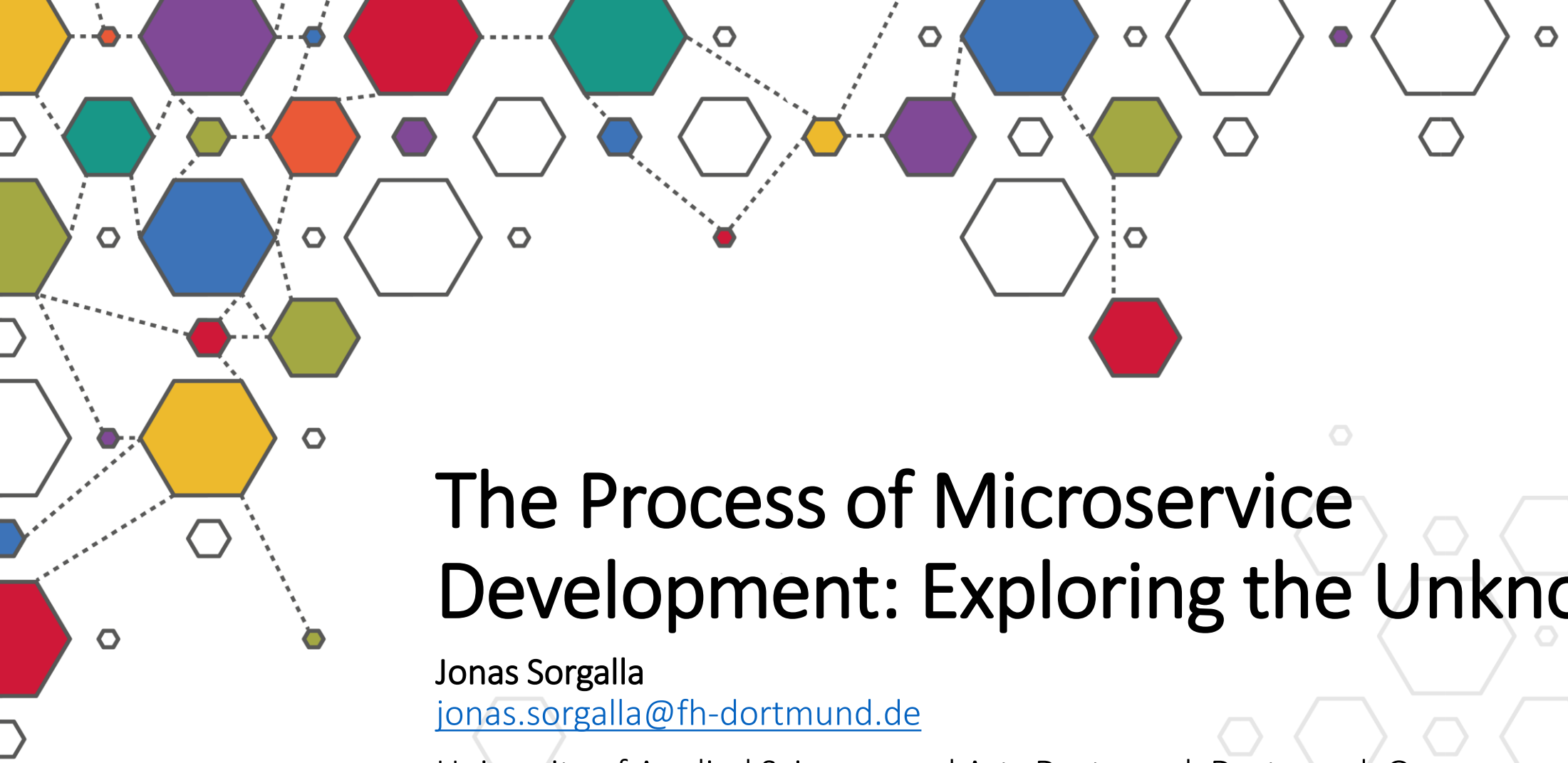


we focus on students

Fachhochschule Dortmund
University of Applied Sciences and Arts

The Process of Microservice Development: Exploring the Unknown

Jonas Sorgalla
jonas.sorgalla@fh-dortmund.de

University of Applied Sciences and Arts Dortmund, Dortmund, Germany

September 9, 2020

Microservices in a Nutshell

Microservice [1]

- is tailored around a single business capability
- is an autonomous process
- relies on stateless communication methods (e.g. RESTful HTTP [2])

Microservice Architecture Style (MSA)

- Novel architectural style for service-based software systems
- Building systems comprising multiple microservices

Microservices in a Nutshell

Pros [3]

- Maintainability
- Scalability
- Resilience
- Composability
- Technology Heterogeneity

Cons [6, 4]

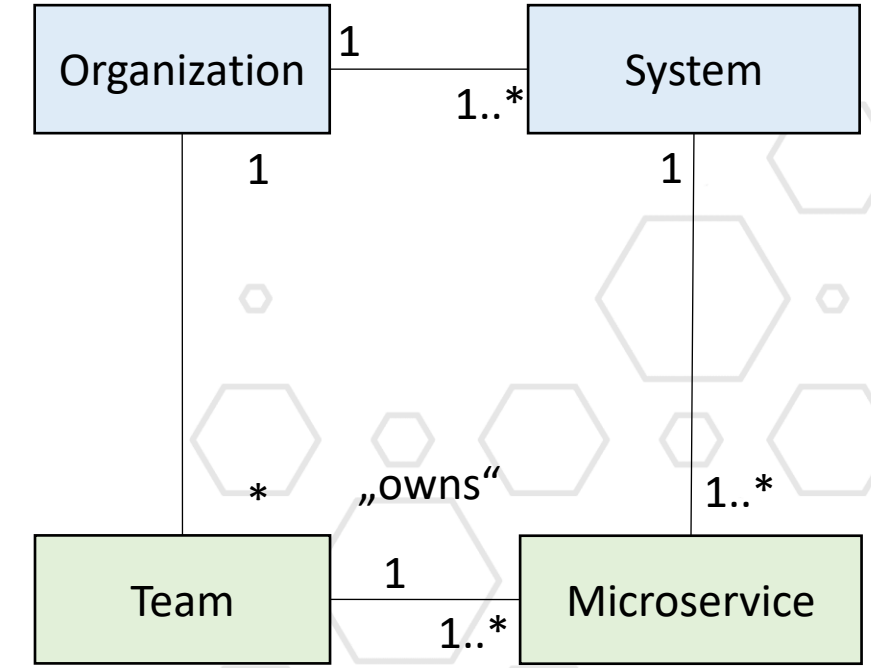
- Much to learn…
  - Organizational complexity
  - Technical complexity
- Boilerplate code
- Additional infrastructure

> See Simon‘s talk from the previous day ☺!

Microservice Development Process (MDP)

MDP's characteristics according to literature [2, 3]

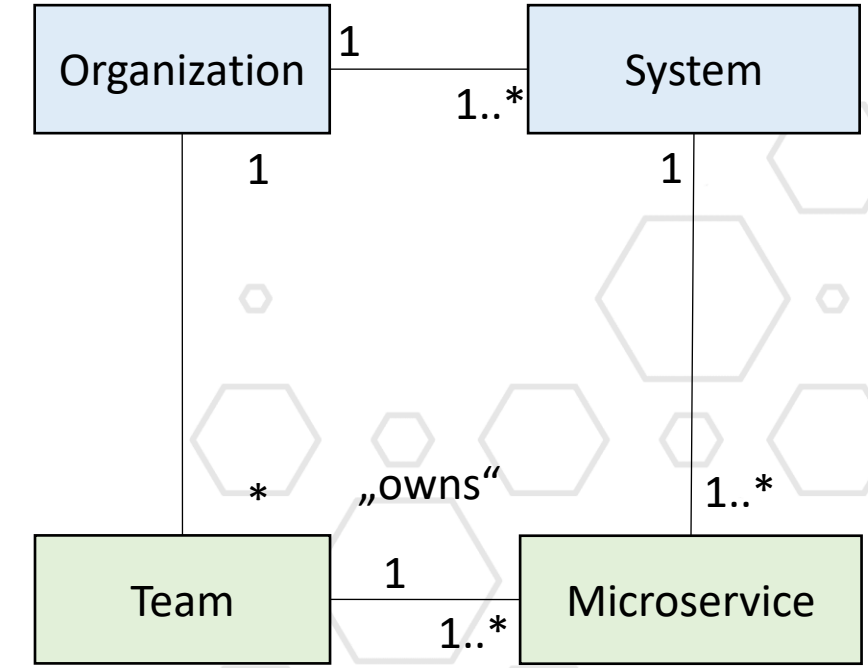
- Comprises multiple teams
- Teams collaborate
- Each team is responsible for one or more services
- DevOps paradigm
- Cross-functional teams



Microservice Development Process (MDP)

MDP's characteristics according to literature [2, 3]

- Comprises multiple teams
- Teams collaborate
- Each team is responsible for one or more services
- DevOps paradigm
- Cross-functional teams



How to organize this?

How to organize an MDP?

Conway's Law [5]: **"Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations."**

How to organize an MDP?

Formal Large-Scale Models

- Scaled Agile Framework (SAFe) [7]
- Scrum at Scale [9]
- or the Spotify Model [10]

Conway's Law [5]: **"Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations."**

How to organize an MDP?

Formal Large-Scale Models

- Scaled Agile Framework (SAFe) [7]
- Scrum at Scale [9]
- or the Spotify Model [10]

They inherently follow Conway's Law, i.e.,

- ensure a decoupled organizational structure
- foster capability-oriented service slicing

Conway's Law [5]: **"Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations."**

How to organize an MDP?

Formal Large-Scale Models

- Scaled Agile Framework (SAFe) [7]
- Scrum at Scale [9]
- or the Spotify Model [10]

Conway's Law [5]: **"Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations."**

They inherently follow Conway's Law, i.e.,

- ensure a decoupled organizational structure
- foster capability-oriented service slicing

BUT they require a certain amount of people to be feasible (50+) [11]!

How to organize an MDP?

Formal Large-Scale Models

- Scaled Agile Framework (SAFe) [7]
- Scrum at Scale [9]
- or the Spotify Model [10]

They inherently follow Conway's Law, i.e.,

- ensure a decoupled organizational structure
- foster capability-oriented service slicing

BUT they require a certain amount of people to be feasible (50+) [11]!

Conway's Law [5]: **"Organizations which design systems […] are constrained to produce designs which are copies of the communication structures of these organizations."**

? How do Small and Medium-sized Development Organizations (SMDOs) with fewer people tackle the MDP

Methodology

…how do Small and Medium-sized Development Organizations (SMDOs) tackle the MDP?

Qualitative Research

- **Exploratory** case study [8]
- Comparative multi-case design
- Interviews
- Grounded Theory
- Interview candidates must have less than 100 ppl and apply microservices

Explore this area and generate hypotheses for future research

Case Description

Case	Interviewee	Type	Domain	#Services	#People	#Teams
CS1	IP1	Templated Greenfield	Public Administration	60	~30	5
CS2	IP2	Migration	B2B E-Commerce	8	10	3
CS3	IP3	Greenfield	Internet of Things	18	28	2
CS4	IP4	Migration	B2B E-Commerce	34	~10	2
CS5	IP4	Migration	B2C E-Commerce	8	~10	2
CS6	IP5	Templated Greenfield	Logistics	15-20	75	~10

Interview execution facts

- Conducted in summer to autumn 2019
- German companies
- Small sample size
- Duration of 1 to 1.5 hours
- Done in person on site
- Audio recording
- Transcribed

Results

During analysis we identified six hypotheses in four main areas from the inductively derived coding system

- **Technology**
- **Collaboration**
- **Microservice Ownership**
- **Development Infrastructure**

Results | Technology

- All cases rely on Java whereby Spring is the prevailing microservice foundation (5 out of 6 use Spring)
- RESTful HTTP is the dominant communication mechanism (6 out of 6)
- Availability of skilled developers and good tutorials are main drivers
- Operation aspects are the main area of concern
- Criteria like performance or maintainability seem secondary

„You simply don’t find any Ruby developers on the market [...]. We had to switch to Java.“ – IP2

(H1) We suspect that this focus leads to a tendency for SMDOs to develop higher technical debt in the long run.

Results | Collaboration

- Team-internal Scrum is the prevailing process framework
- Cross-team collaboration follows no framework and is mostly customized
- Formats for knowledge sharing are common
- Formats for informing each other are not

„We got rid of these lengthy meetings which just burn time and money.“ – IP5

(H2) We suspect that teams make decisions not based on what is the best for the overall system but for their respective services.

Results | Collaboration

- Team-internal Scrum is the prevailing process framework
- Cross-team collaboration follows no framework and is mostly customized
- Formats for knowledge sharing are common
- Formats for informing each other are not

„We got rid of these lengthy meetings which just burn time and money.“ – IP5

(H2) We suspect that teams make decisions not based on what is the best for the overall system but for their respective services.

- In every case we encountered specialized units (e.g. Kubernetes, interface definition etc.)

(H3) SMDOs tend to rely more on specialized units which waters down the accountability of a team for a service (contradicts ownership principle).

Results | Microservice Ownership

- With specialized units the ownership of services change during a microservice lifecycle
- IPs are not concerned

(**H4**) SMDOs tend to shift the responsibility of a microservice. We assume this as a major factor that gives rise to an excessive coupling of services.

Results | Microservice Ownership

- With specialized units the ownership of services change during a microservice lifecycle
- IPs are not concerned

(**H4**) SMDOs tend to shift the responsibility of a microservice. We assume this as a major factor that gives rise to an excessive coupling of services.

- Migration cases are short-staffed
- Migration cases implement one service after another (“Fire-and-forget” mentality) and tend to neglect maintenance

(**H5**) We expect that in the long run such migration MDPs with few involved people suffer a growing decrease in development velocity due to people being bound by the required maintenance of previous services.

Results | Development Infrastructure

- Software tools such as GitLab, Jira or Jenkins are common
- But no special computer-aided software engineering tools (5 out of 6)
- No UML diagrams
- Heavy use of Swagger/OpenAPI

> “It shows the reality and not how it was planned.“ – IP2

(H6) We assume that these textual approaches are particularly popular because of the perceived quick returns and the derivation from source code. However, only relying on generated documentation of own code could hurt a common understanding across teams and thus impair the collaboration process.

Conclusion

- (*Disclaimer*) Most interview partners are still building and improving their MDP
- Small sample size but high saturation
- Deployment seems to be a major area of concern (especially skill-wise)
- Specialized units seem to be a common strategy for SMDOs
- Teams live in their own microcosms
- Swagger/OpenAPI are favored
- UML disliked
- Starting point for more refined studies

„For our company it is a tough journey. […] We challenge us everyday.“ – IP1

Literature

[1] Newman, S.: Building Microservices. O'Reilly Media (2015)

[2] Fielding, R.: Representational state transfer. Architectural Styles and the Design of Network-based Software Architecture pp. 76–85 (2000)

[3] Balalaie, A., Heydarnoori, A., Jamshidi, P.: Microservices architecture enables devops: Migration to a cloud-native architecture. IEEE Software 33(3), 42–52 (2016)

[4] Bogner, J., Fritzsch, J., Wagner, S., Zimmermann, A.: Microservices in industry: Insights into technologies, characteristics, and software quality. In: 2019 IEEE Int. Conf. on Software Architecture Companion (ICSA-C). pp. 187–195 (March 2019)

[5] Conway, M.E.: How do committees invent. Datamation 14(4), 28–31 (1968)

[6] Singleton, A.: The economics of microservices. IEEE Cloud Computing 3(5), 16–20 (Sep 2016)

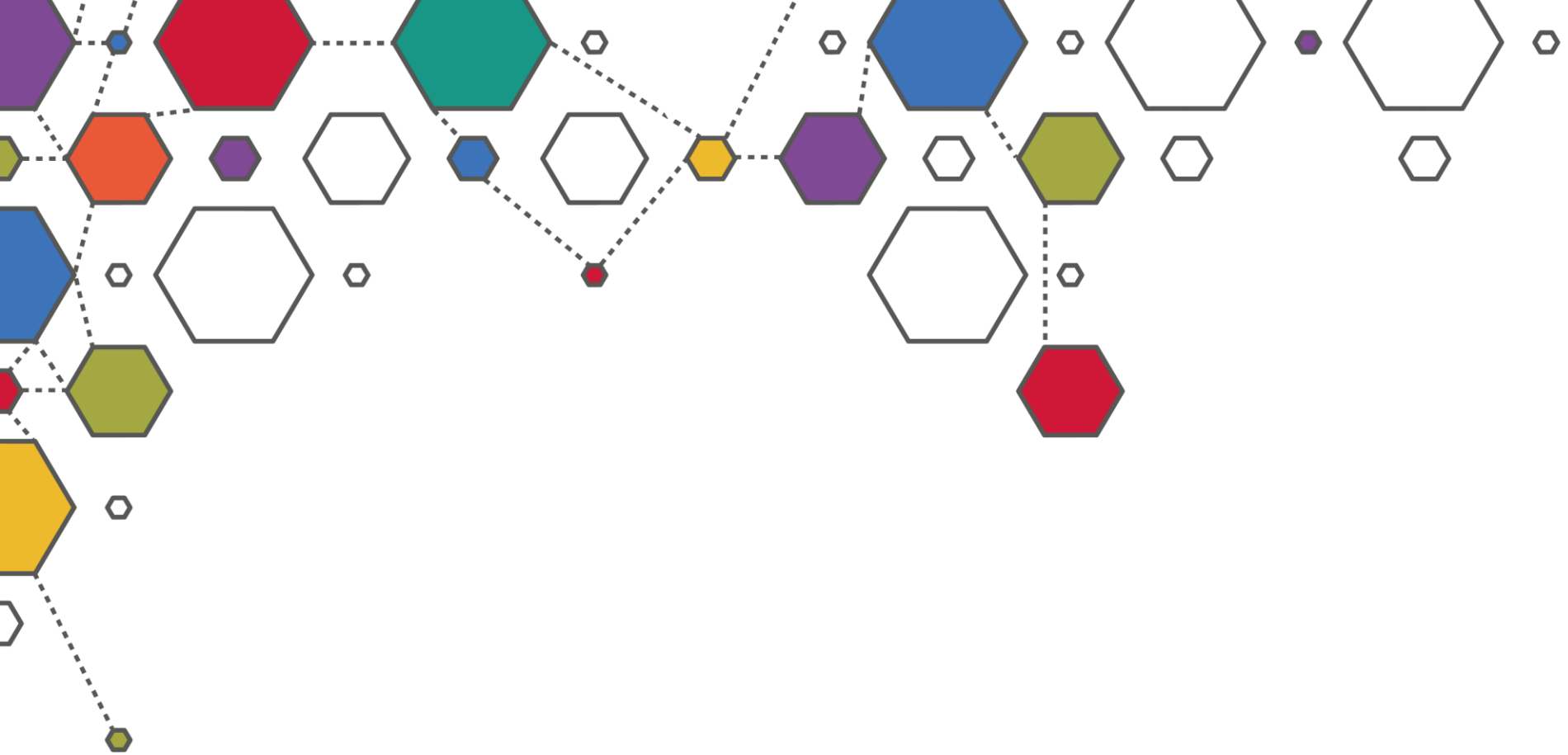
[7] Achieving business agility with safe R 5.0. Tech. rep., Scaled Agile, Inc.

[8] Yin, R.K.: Case Study Research and Applications: Design and Methods. SAGE Publications, 6 edn. (2017)

[9] Sutherland, J.: The scrum@scale guide version 2.0. Tech. rep., Scrum Inc.

[10] Smite, D., Moe, N.B., Levinta, G., Floryan, M.: Spotify guilds: How to succeed with knowledge sharing in large-scale agile organizations. IEEE Software 36(2), 51–57 (2019)

[11] Dikert, K., Paasivaara, M., Lassenius, C.: Challenges and success factors for large-scale agile transformations: A systematic literature review. Journal of Systems and Software 119, 87 – 108 (2016)



Thanks for listening!

Discussion time 😃