SummerSOC 2018

Using Architectural Modifiability Tactics to Examine Evolution Qualities of Service- and Microservice-Based Systems

An Approach based on Principles and Patterns

Justus Bogner[1,2], Stefan Wagner[2], Alfred Zimmermann[1]
[1] University of Applied Sciences Reutlingen, [2] University of Stuttgart

Motivation: Software Evolution

"Highly business critical *modifiability* tends to be detrimental to project success, even when the architect is aware of it. [...] Modifiability gets too little attention." (Poort et al., 2012)

"40% of participants were not satisfied with the degree of *maintainability* of their software, while 35% were somewhat satisfied. Only one fourth of participants (15 of 60) were actually content with the degree of maintainability." (Bogner et al., 2018)

What about Service Orientation?

"Literature reports that SOA directly supports *changeability* by promoting loose coupling between service consumers and providers. Issues, however, are only seen on second sight." (Voelz and Goeb, 2010)

"Lastly, ~67% reported to not treat SBSs and µSBSs with special *maintainability* controls. [...] 6 participants mention relaxed maintainability controls because of trust in the high base level of maintainability gained through service orientation." (Bogner et al., 2018)

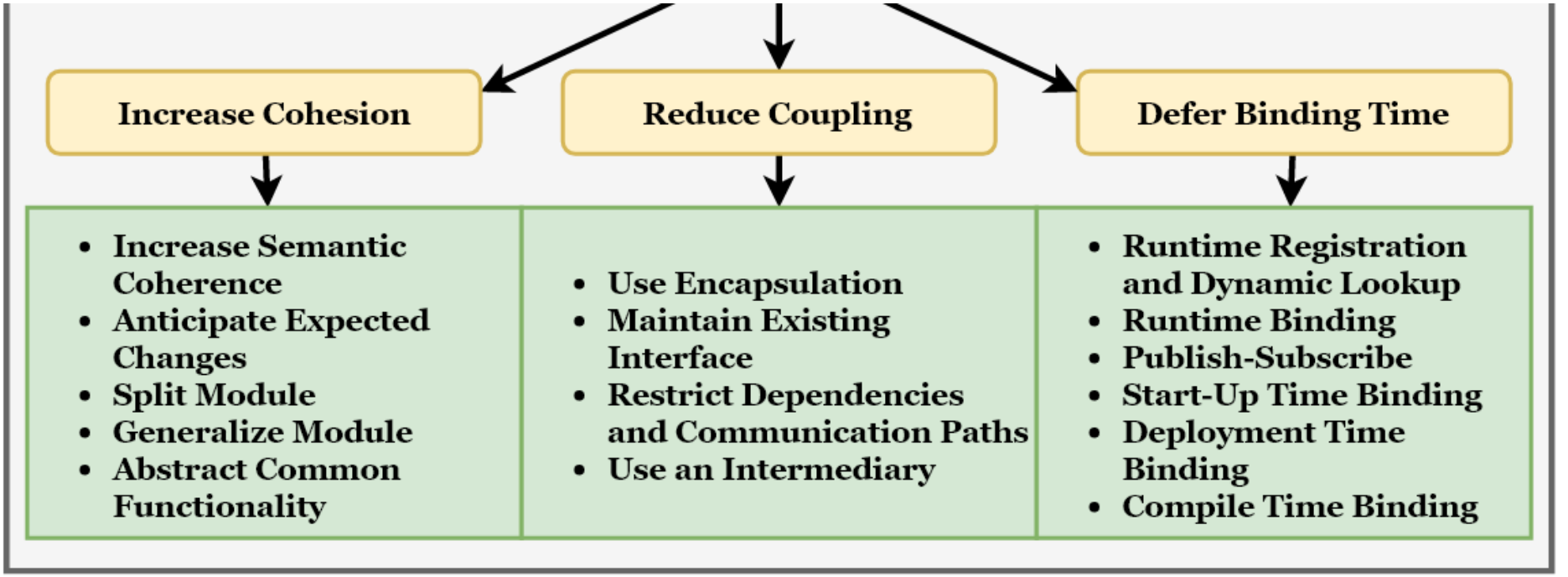
→Analyze the evolution qualities of service orientation by using architectural modifiability tactics

Architectural Tactics

- High-level techniques to achieve quality attribute goals
- Conceptualized by the Software Engineering Institute (CMU SEI, Pittsburgh, PA)
- Exist for various quality attributes
- **Modifiability tactic**: a design decision or an architectural transformation that positively affects system properties related to modifiability → reduce time and effort necessary to introduce future changes

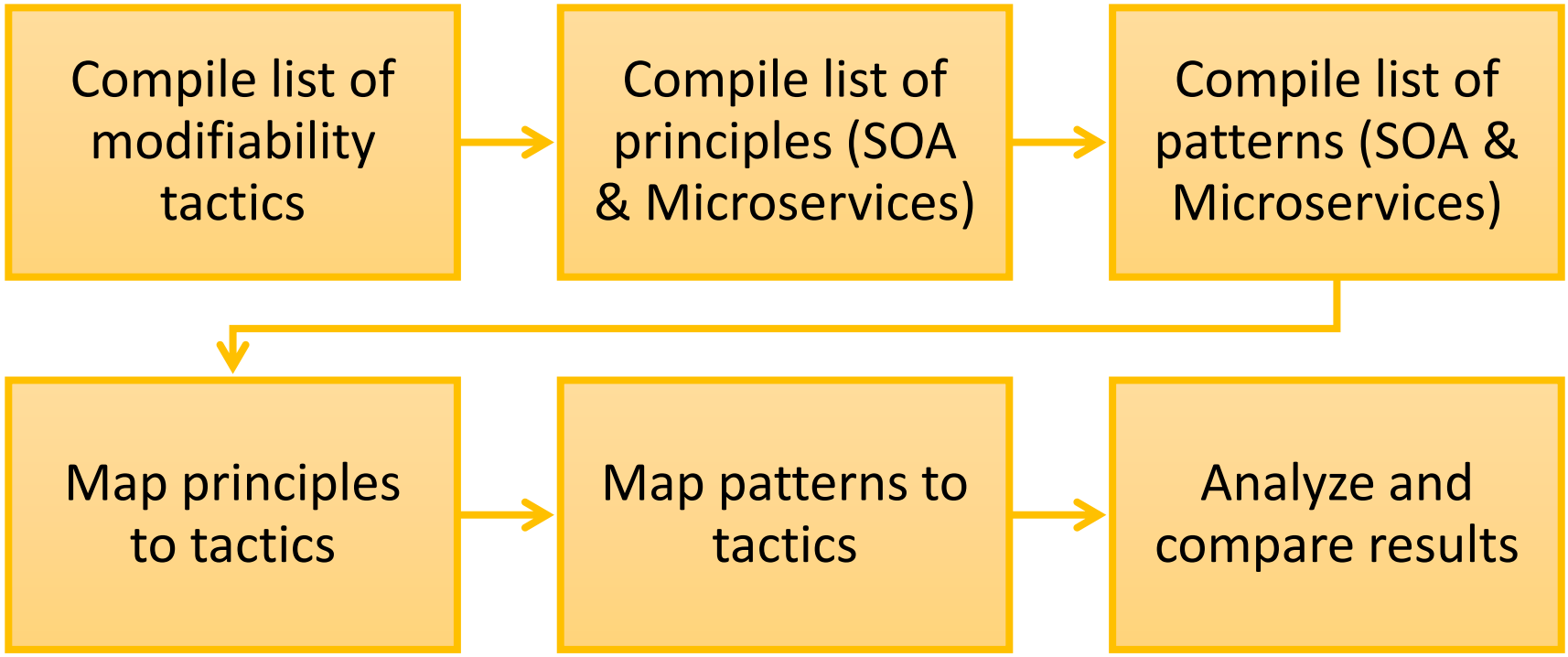
Modifiability Tactics

15 tactics in 3 categories, compiled from (Bass et al., 2003), (Bachmann et al., 2007), and (Bass et al., 2012)



Scope and Method

Goal: Systematic understanding of service-oriented evolution qualities via modifiability tactics



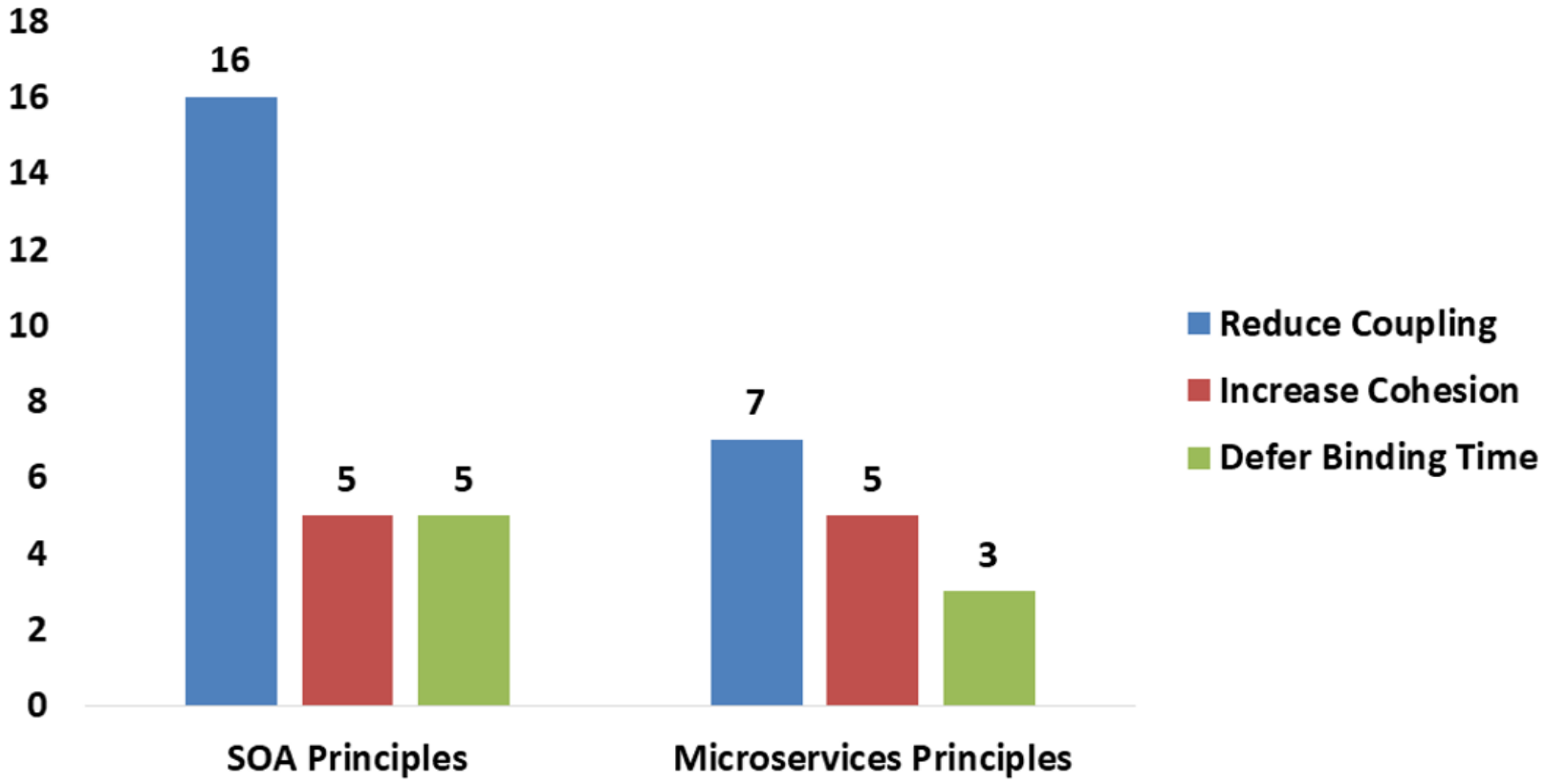
Used Principles

SOA (Erl, 2005)	Microservices (Fowler, 2015)
Standardized Service Contract	Componentization via Services
Service Loose Coupling	Organized around Business Capabilities
Service Abstraction	Products, not Projects
Service Reusability	Smart Endpoints, Dumb Pipes
Service Autonomy	Decentralization
Service Statelessness	Infrastructure Automation
Service Discoverability	Design for Failure
Service Composability	Evolutionary Design

Results: Principles

SOA: 26 mappings

Microservices: 15 mappings



Results: Principles

	SOA (8)	Microservices (8)
General:	- 26 mappings (~22%) - Focused on the "Reduce Coupling" category (~60%)	- 15 mappings (~13%) - More evenly distributed over the categories
Most mapped principle:	"Service Loose Coupling" (7)	"Evolutionary Design" (5)
Most mapped tactic:	"Maintain Existing Interface" (5)	"Restrict Dependencies and Communication Paths" (3)
Not mapped principles:	- "Service Statelessness"	- "Products, not Projects" - "Design for Failure"

→SOA principles of a more "technical" nature

→But: Microservices principles often very fitting
(e.g. "Evolutionary Design", "Organized around Business Capabilities")

Used Patterns

SOA (118 patterns):

- "SOA Design Patterns" (Erl, 2009)
- "SOA with REST" (Erl et al., 2012)
- "SOA Patterns" (Rotem-Gal-Oz, 2012)

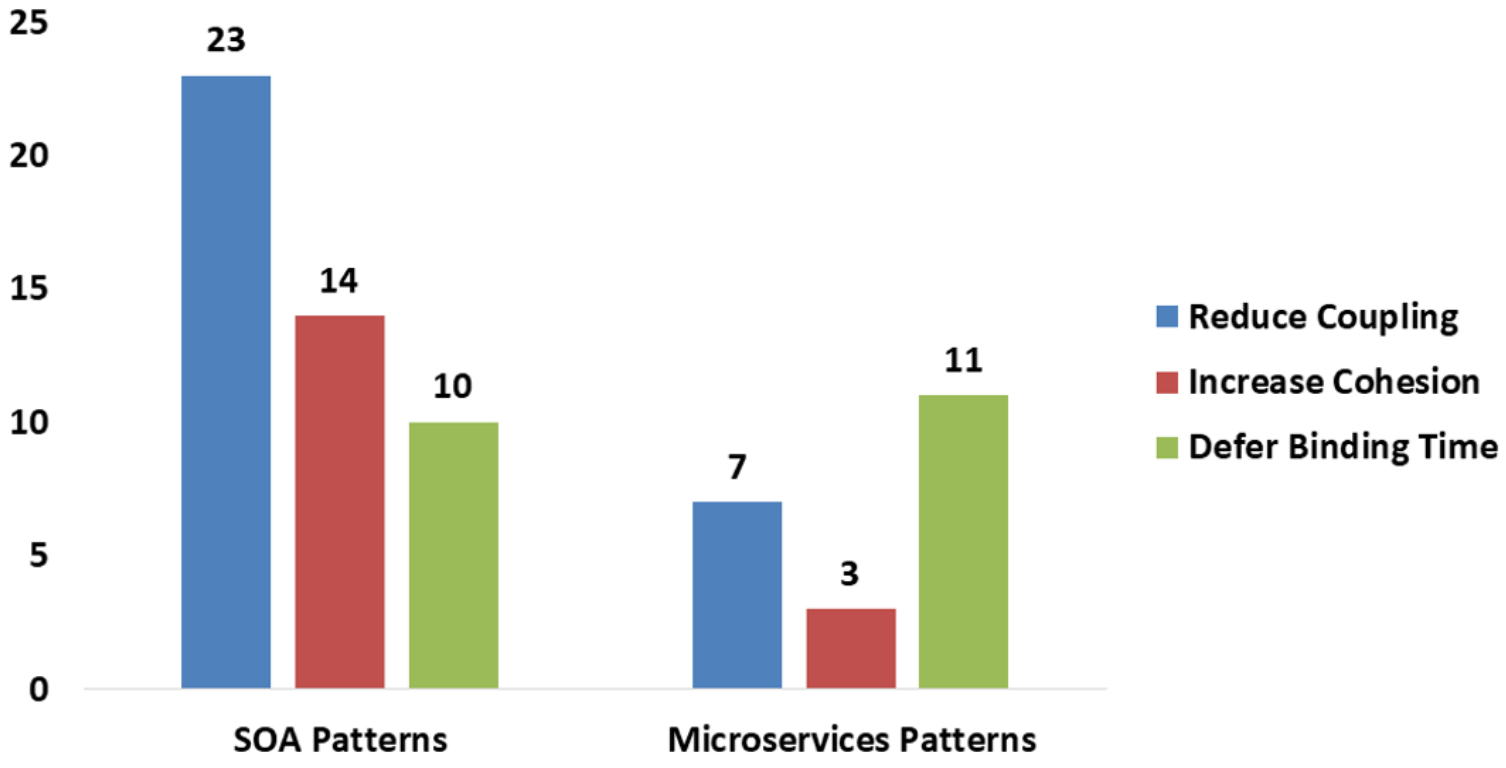
Microservices (42 patterns)

- "Microservices Patterns" (Richardson, 2018)
- ...

Results: Patterns

SOA: 47 mappings (~40%)

Microservices: 21 mappings (50%)



Results: Patterns

	SOA (118)	Microservices (42)
General:	- 47 mappings (~40%) - Focused on the "Reduce Coupling" category (~49%)	- 21 mappings (50%) - Focused on the "Defer Binding Time" category (~52%)
Frequently mapped tactics:	- "Use an Intermediary" (9) - "Restrict dependencies" (8) - "Generalize Module" (8)	- "Runtime Registration and Dynamic Lookup" (5) - "Use an Intermediary" (3) - "Publish-Subscribe" (3)
Not mapped tactics:	- "Start-Up Time Binding" - "Deployment Time Binding" - "Compile Time Binding"	- "Maintain Existing Interface" - "Runtime Binding" - "Compile Time Binding"

→~10% more patterns mapped for Microservices than SOA

→Microservices patterns slightly more coined at modifiability?

→Only 3 Microservices patterns for "Increase Cohesion" category

Threats to Validity

- Qualitative nature of the approach
  - Possibility of subjective bias
  - Revalidate with professionals / pattern experts?
- Microservices are much younger than SOA
  - Principle & pattern quantity/quality is different
  - Overlapping of principles/patterns
- Not all allegedly beneficial principles/patterns could be included
- Focus on “theoretical” modifiability

Conclusion

- We mapped 15 modifiability tactics to
  - Principles of SOA (8) and Microservices (8)
  - Patterns of SOA (118) and Microservices (42)

→Analysis: Evolution qualities seem to be roughly equal

- **But:** partly different strategies to achieve modifiability
  - SOA: governance, restrictions, interoperability, reuse
  - Microservices: evolutionary design, basic restrictions (DB access, protocols, …), heterogeneity, runtime bindings decentralization
- **Vision:** Incorporate this knowledge into the SDLC

Thank you!

Q & A