

Microservices beyond COVID-19

Antonio Brogi

Department of Computer Science
University of Pisa, Italy

MICROSERVICES 2020 - 3rd International Conference on Microservices
Invited keynote - September 10th, 2020

Q: Beyond COVID-19?



Nick

A: Sorry, just a dirty trick to attract audience :)

15:00 Coffee Break

15:30 Keynote: Microservices beyond COVID-19

16:30 **Closing**



Microservices, microservices, microservices ...

Microservices

Main motivations



(1) Shorten lead time for new features/updates

- accelerate rebuild and redeployment
- reduce chords across functional silos



(2) Need to scale, effectively

- millions of users

amazon

Google

NETFLIX

Spotify



LinkedIn



ebay

Uber



GROUPON

...

OK but ... what are microservices?



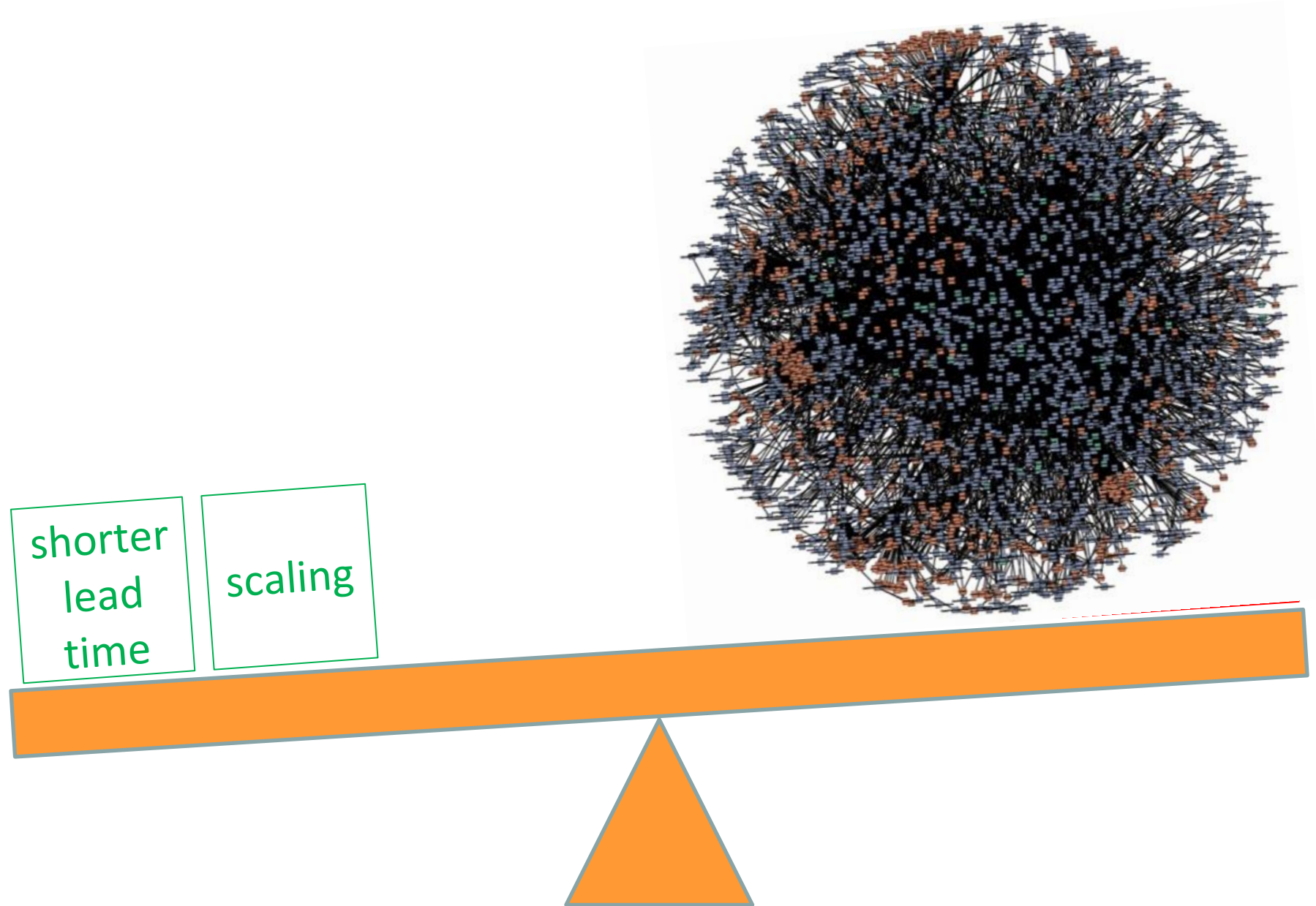
Microservices

Applications = sets of **services**

- + each running in its own ~~process~~ container
- + communicating with lightweight mechanisms
- + built around business capabilities
- + decentralizing data management
- + independently deployable
- + horizontally scalable
- + fault resilient
- + DevOps culture and tools!

(service-orientation done right?)

Microservices



Does my app respect the "microservices principles"?

If not, how can I refactor it?



Microservices, microservices, microservices ...

Design principles, architectural smells and refactoring

Question

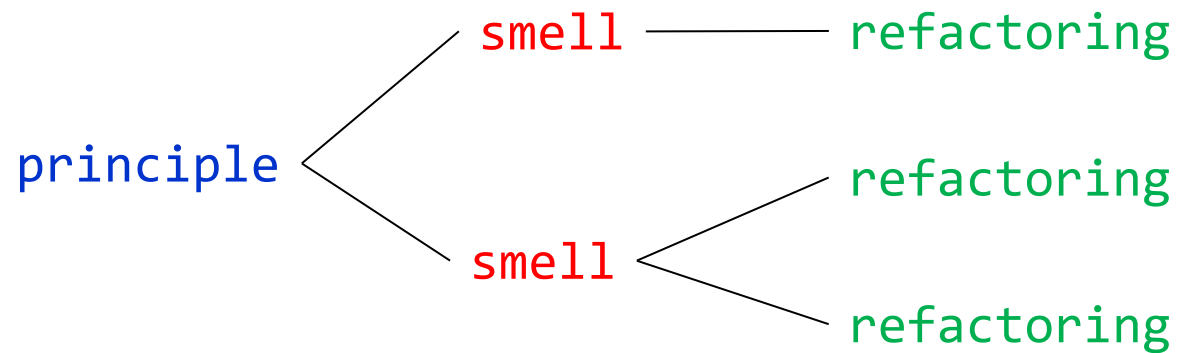
How can architectural **smells** affecting design **principles** of microservices be detected and resolved via **refactoring**?



A multivocal review

Recent review of white and grey literature aimed at identifying

- the most recognised **architectural smells** for microservices, and
- the architectural **refactorings** to resolve them



(review of 41 studies presenting architectural smells & refactorings for resolving them)

Design principles



Independent deployability

The microservices forming an application should be independently deployable

Horizontal scalability

The microservices forming an application should be horizontally scalable

[= possibility of adding/removing replicas of single microservices]

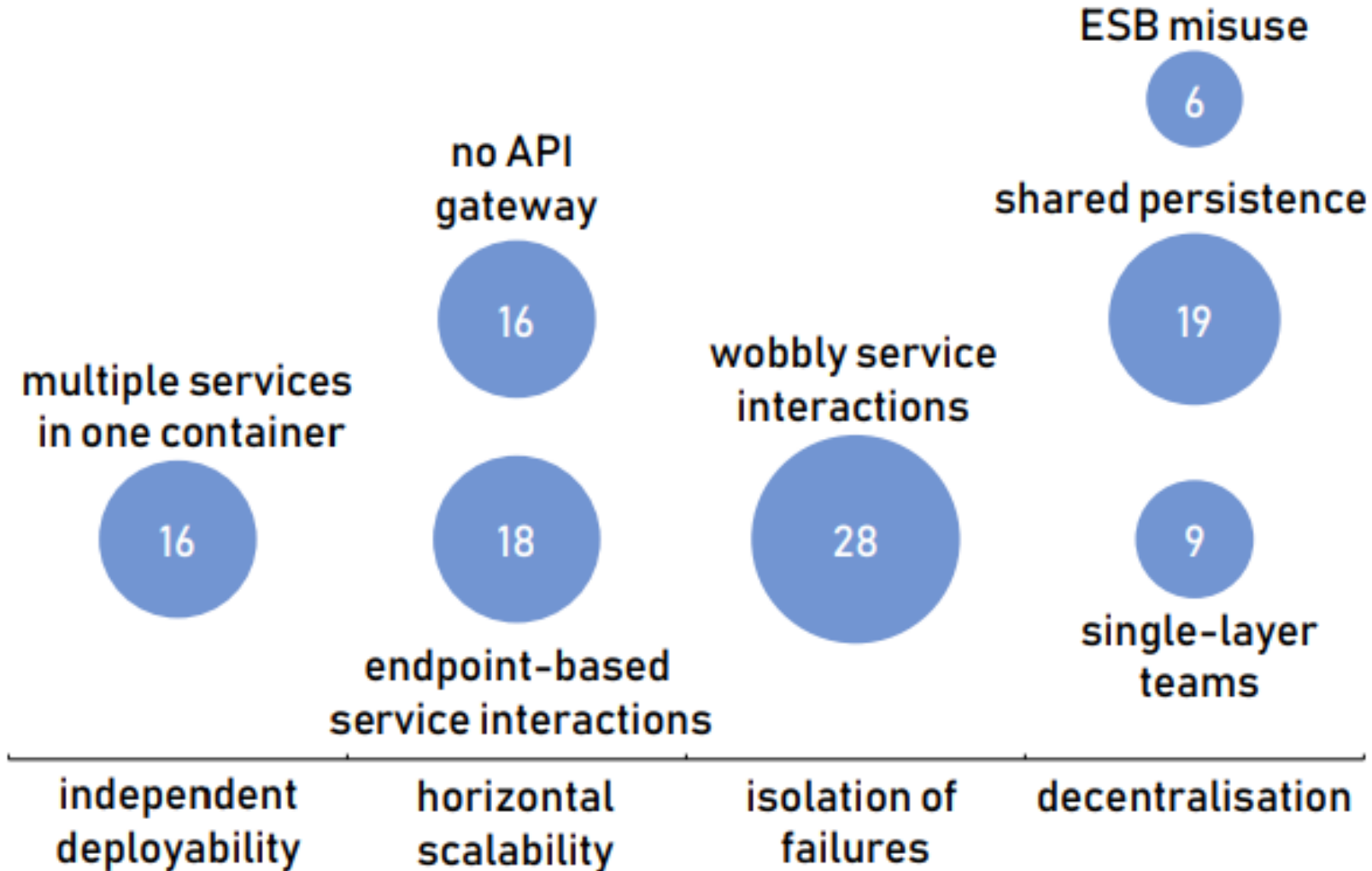
Isolation of failures

Failures should be isolated

Decentralization

Decentralisation should occur in all aspects of microservice-based applications, from data management to governance

Architectural smells



Multiple services in one container

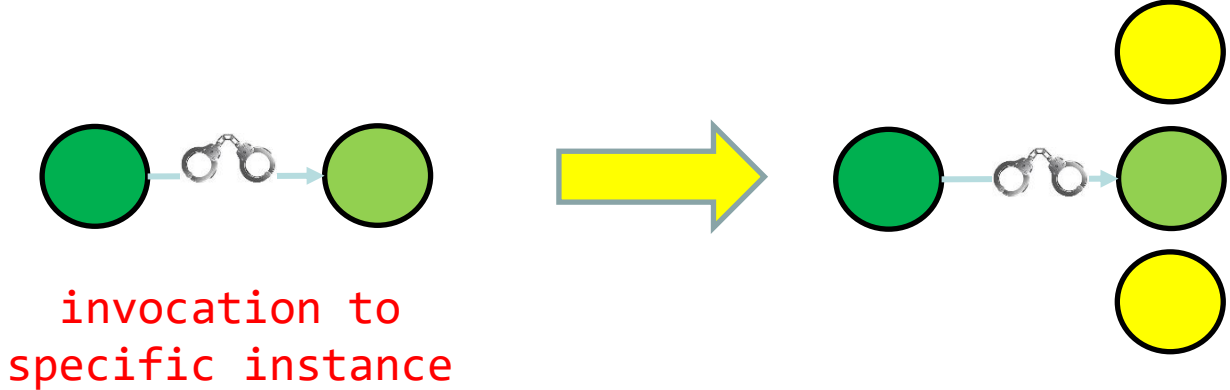
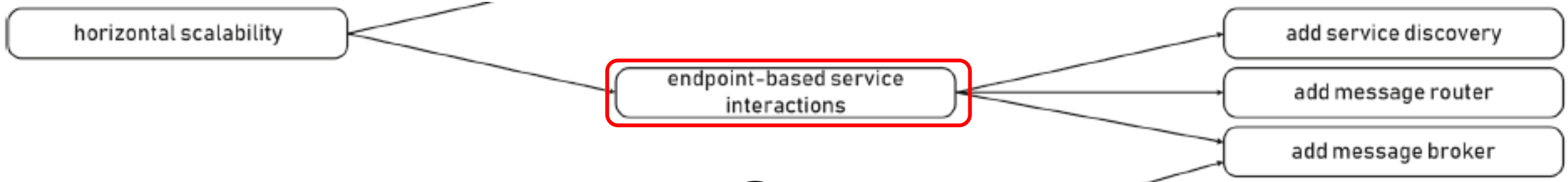
independent deployability

multiple services
in one container

package each service in a
separate container



Endpoint-based service interactions

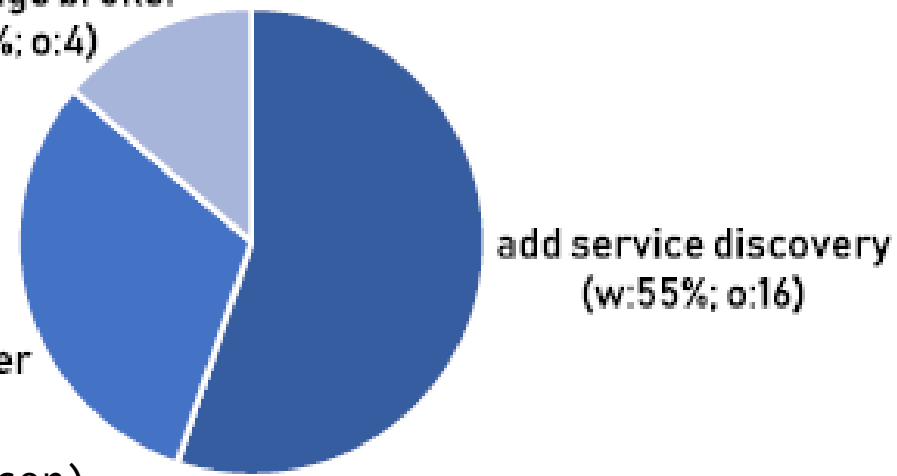


(e.g. message queue)

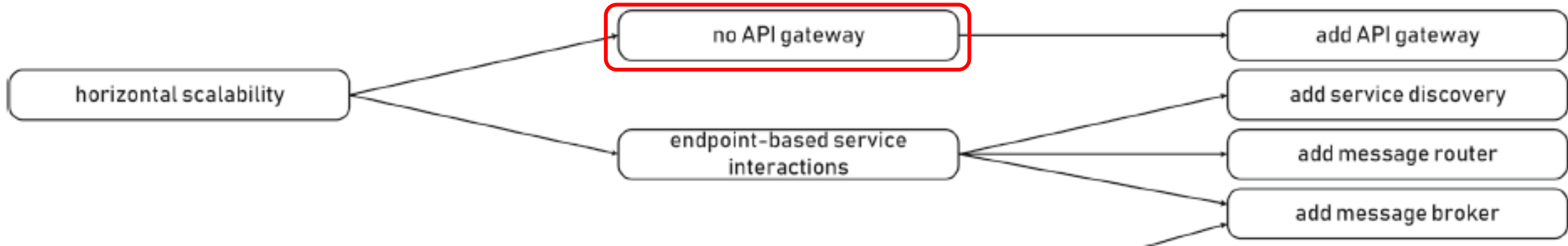
add message broker
(w:14%; o:4)

add message router
(w:31%; o:9)

(e.g. load balancer)



No API gateway



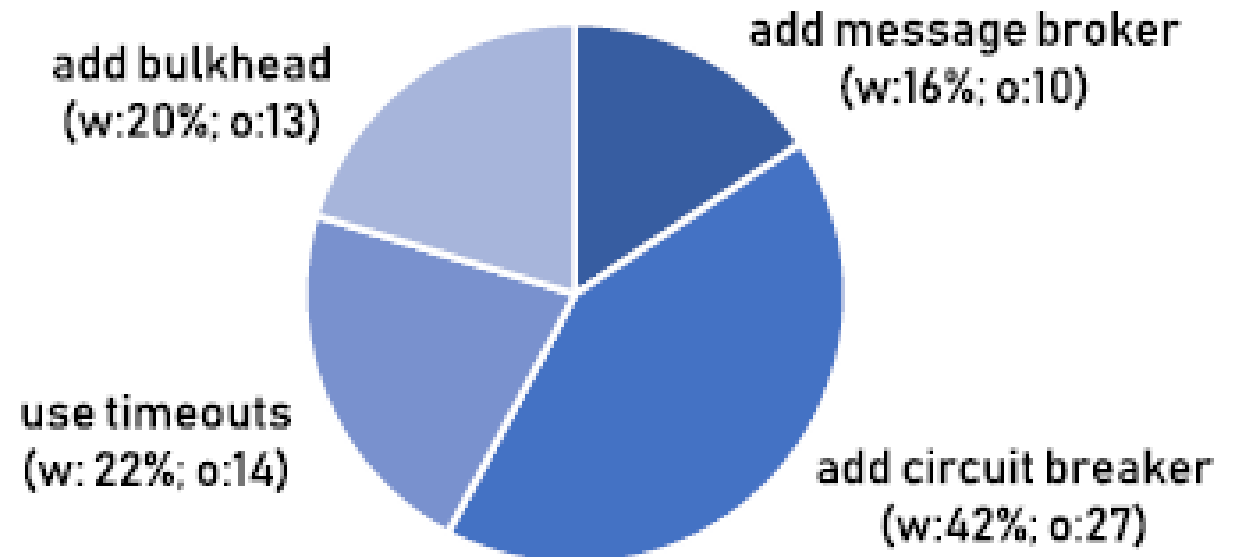
App clients must invoke directly app services
(similar to endpoint-based service interaction smell)

Refactoring: add API gateway (that can be useful also for authentication, throttling, ...)

Wobbly service interactions



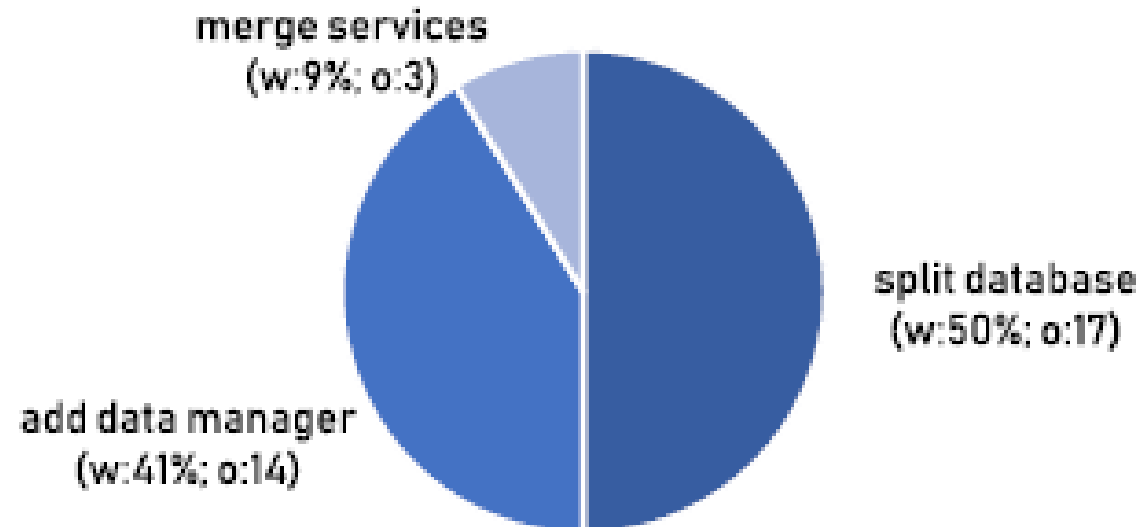
The interaction of m1 with m2 is *wobbly* when a failure of m2 can trigger a failure of m1



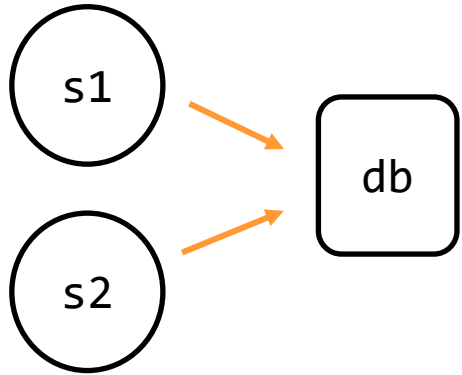
Shared persistence



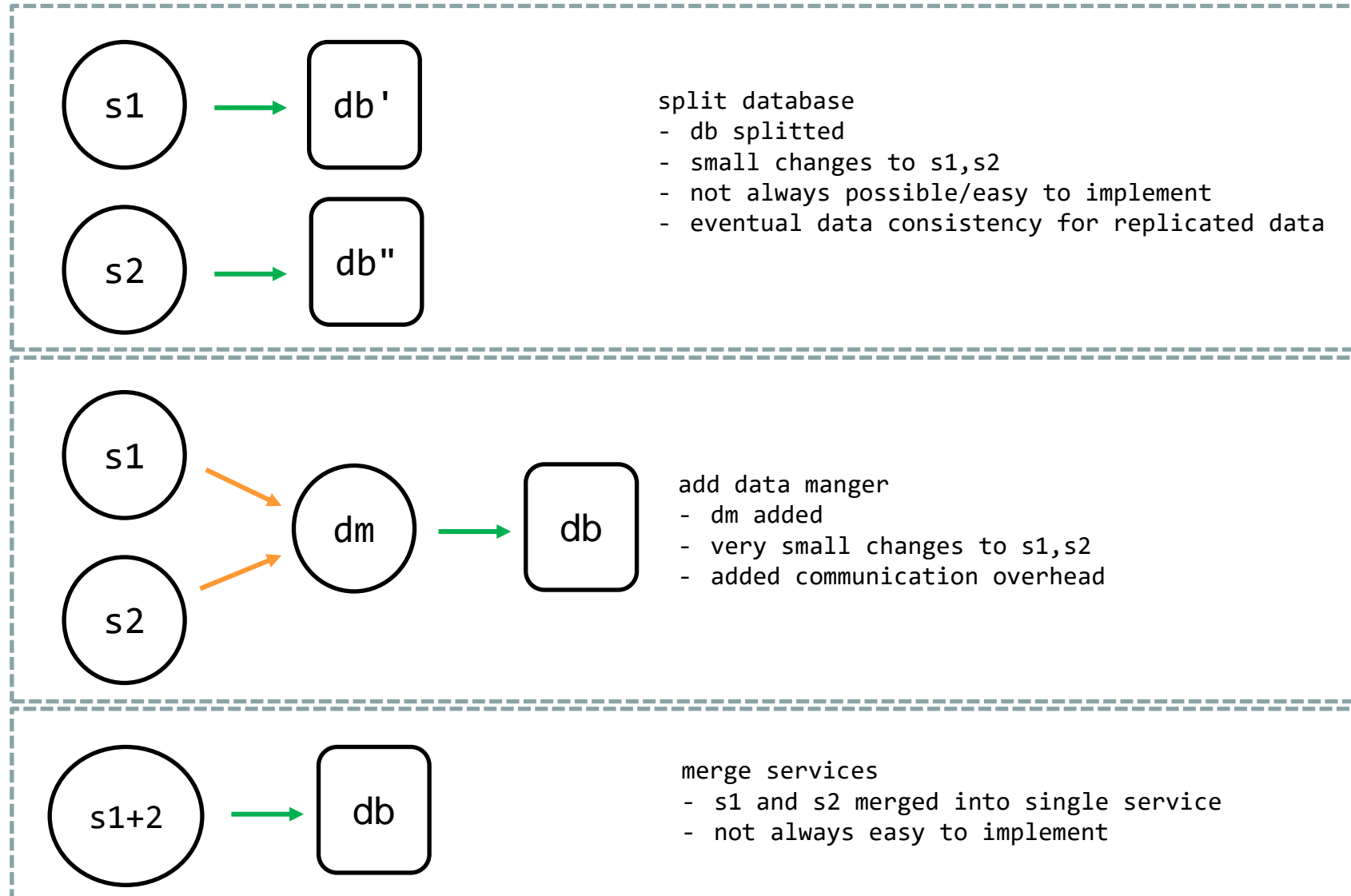
Multiple services access/manage the same DB



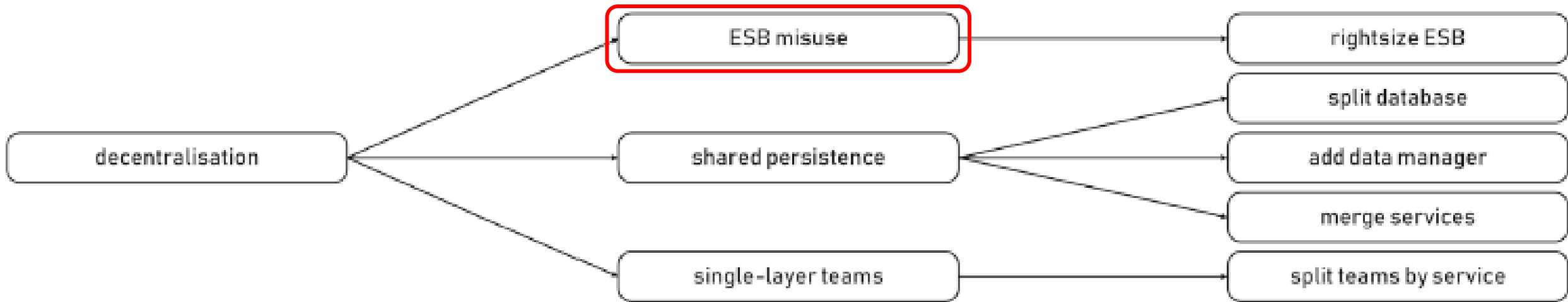
Shared persistence



db shared by
multiple services



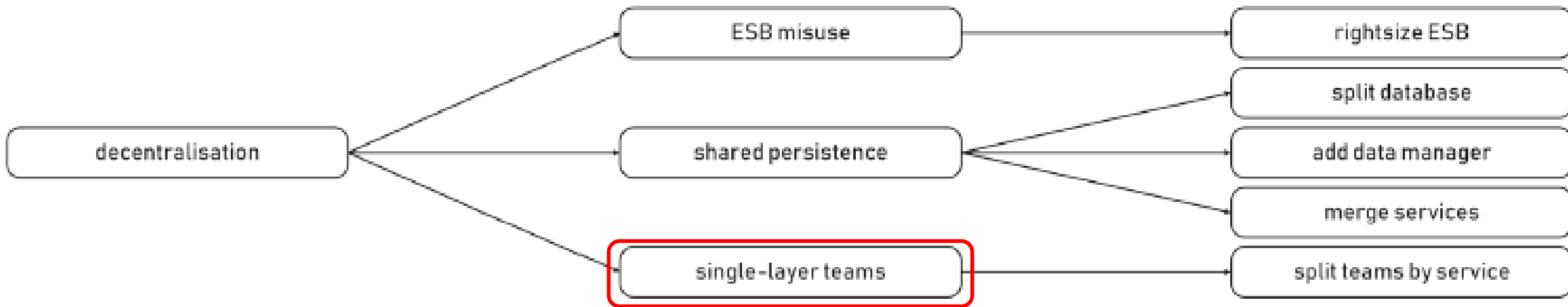
ESB misuse

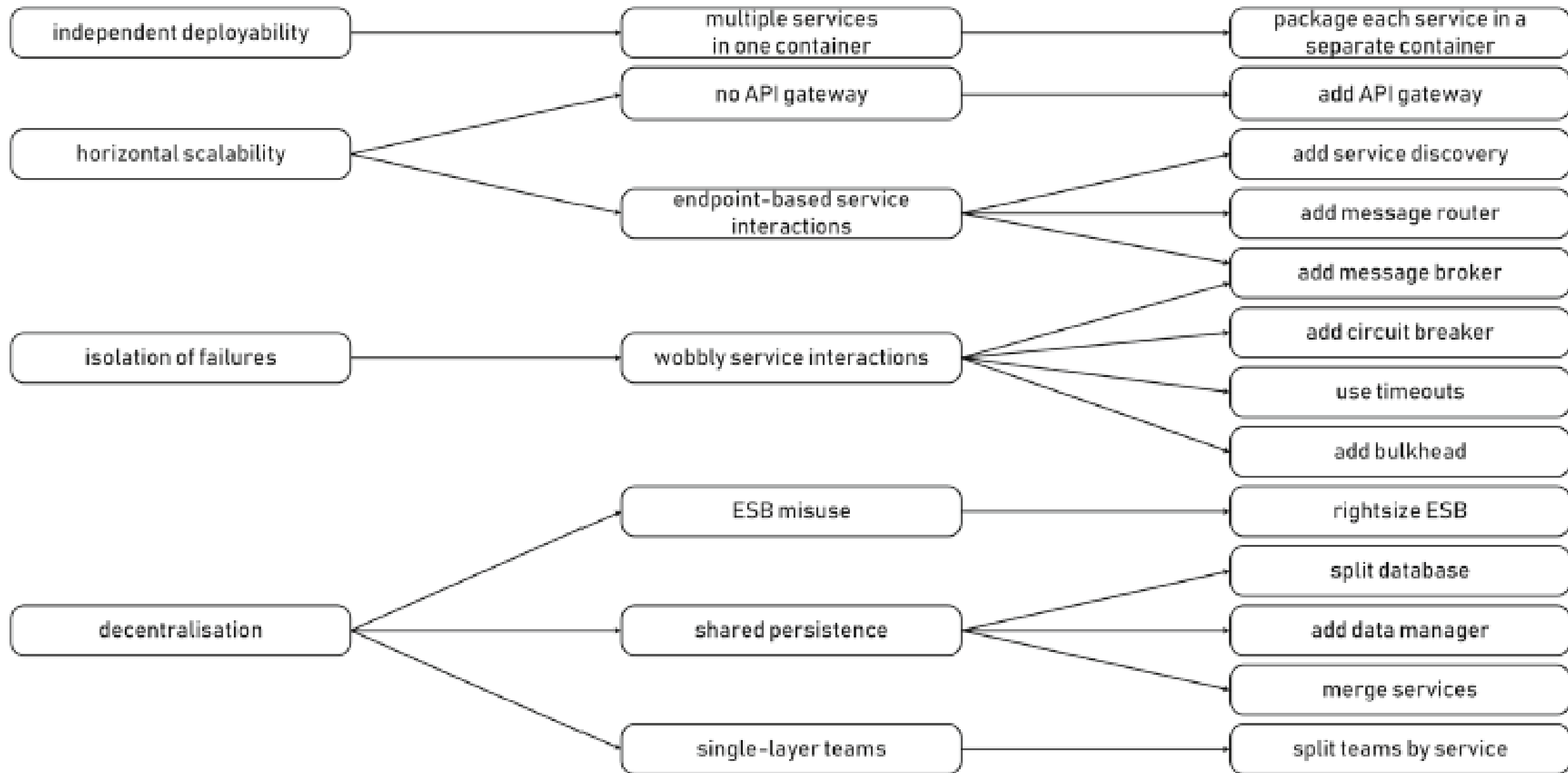


ESB misuse may lead to
undesired centralisation of
business logic and dumb services

Smart endpoints & dumb pipes !

Single-layer teams





Microservices, microservices, microservices ...

Design principles, architectural smells and refactoring

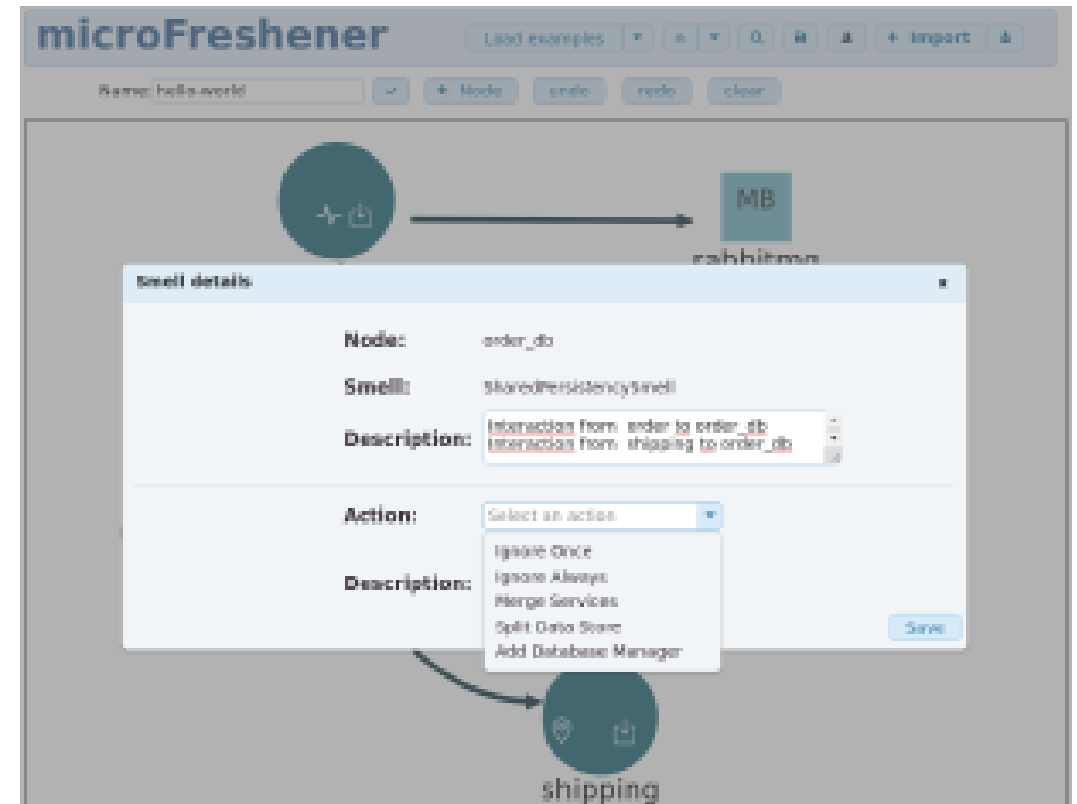
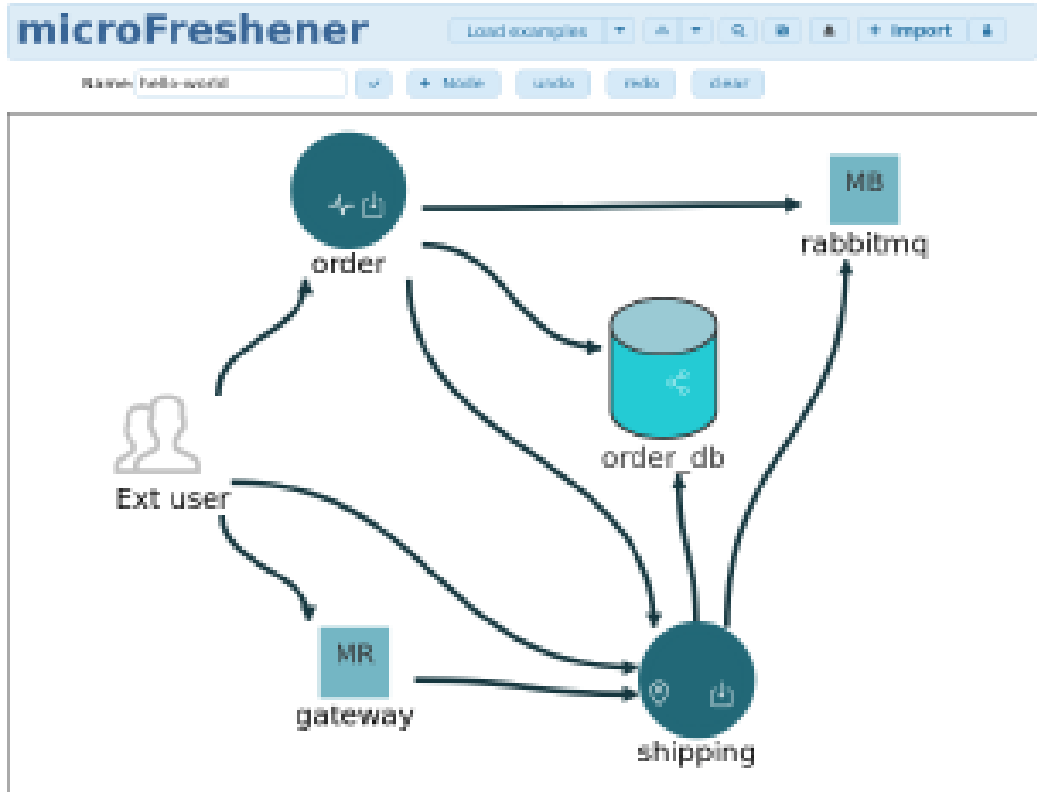
- μ Freshener

μFreshener

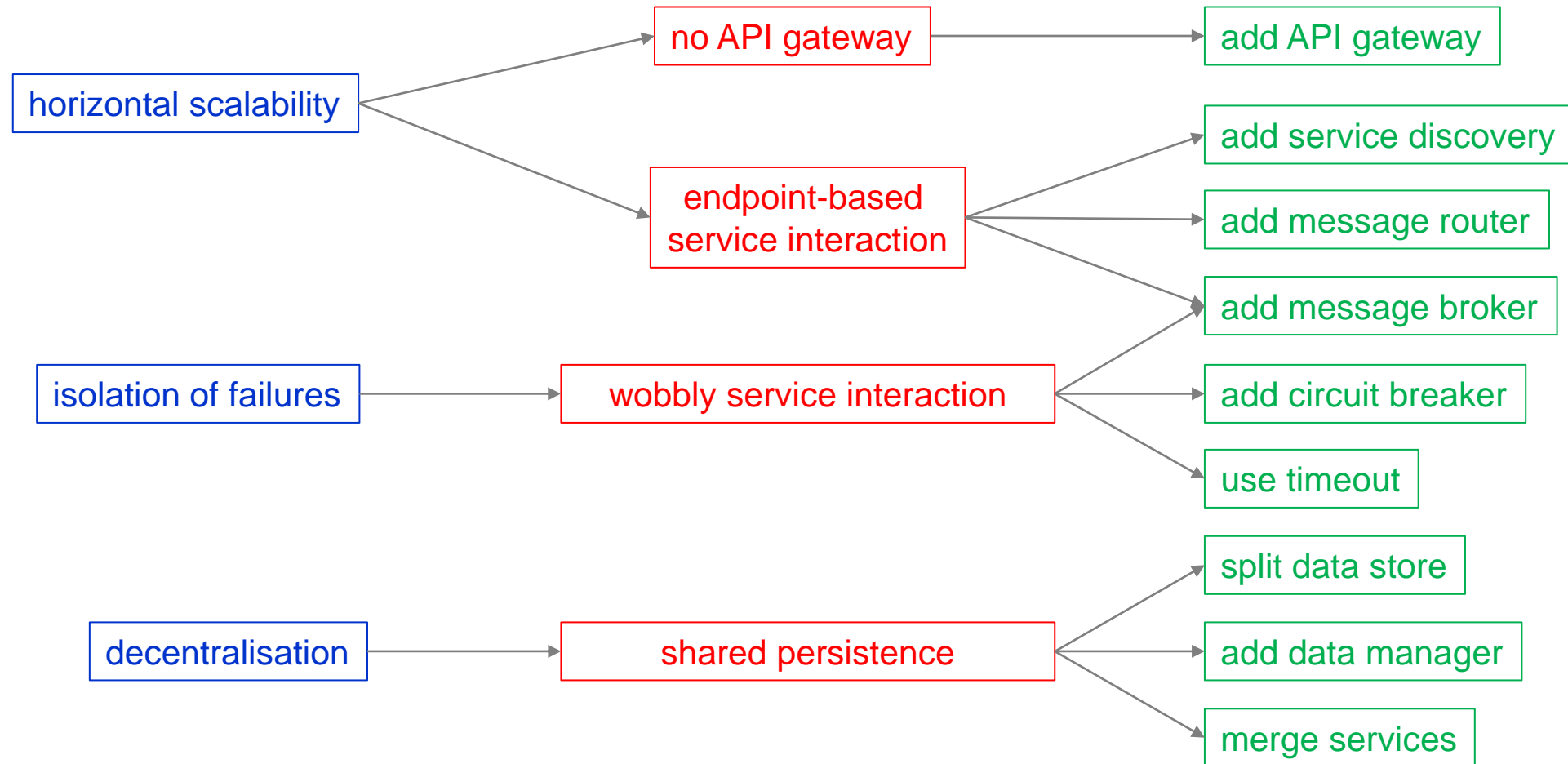
A web-based GUI for

- editing app specifications
- automatically identifying architectural smells
- applying architectural refactorings to resolve the identified smells

μFreshener

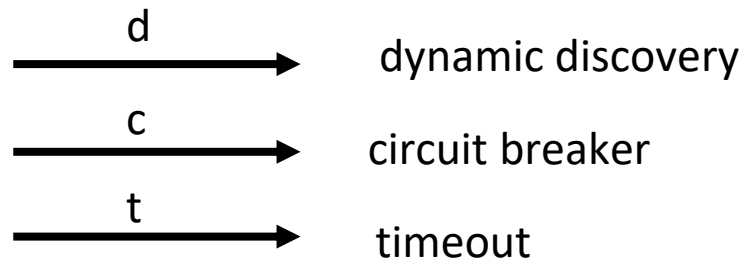
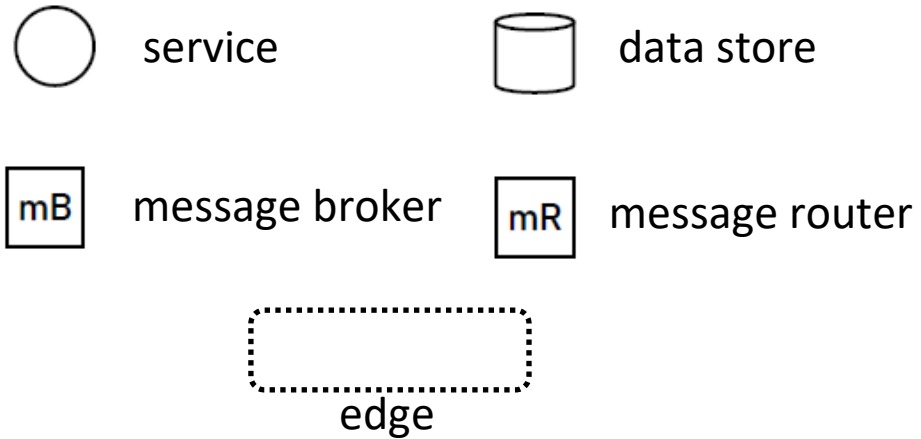


Excerpted principle-smell-refactoring taxonomy

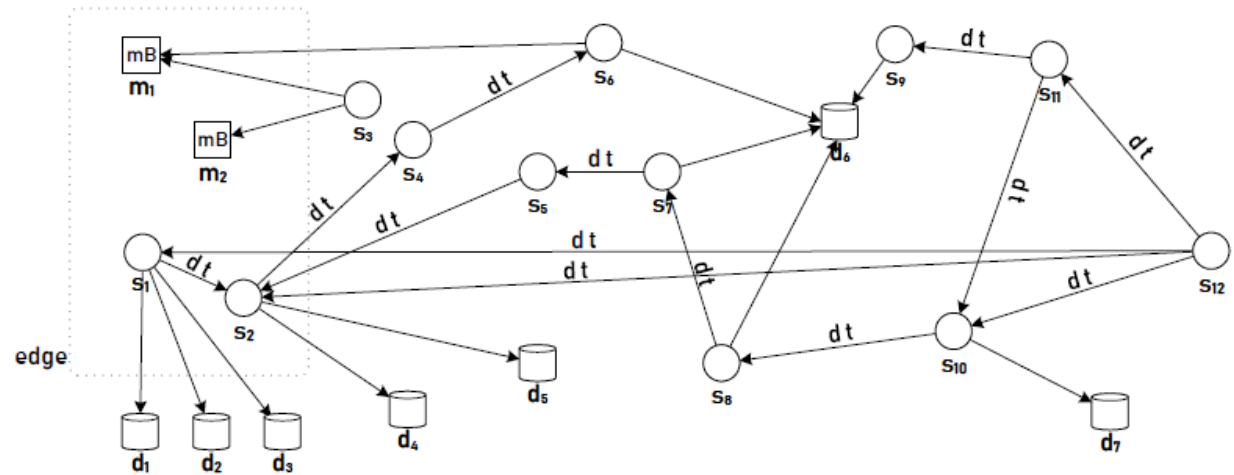


Modelling application architecture

Graphical representation (of μ TOSCA model)

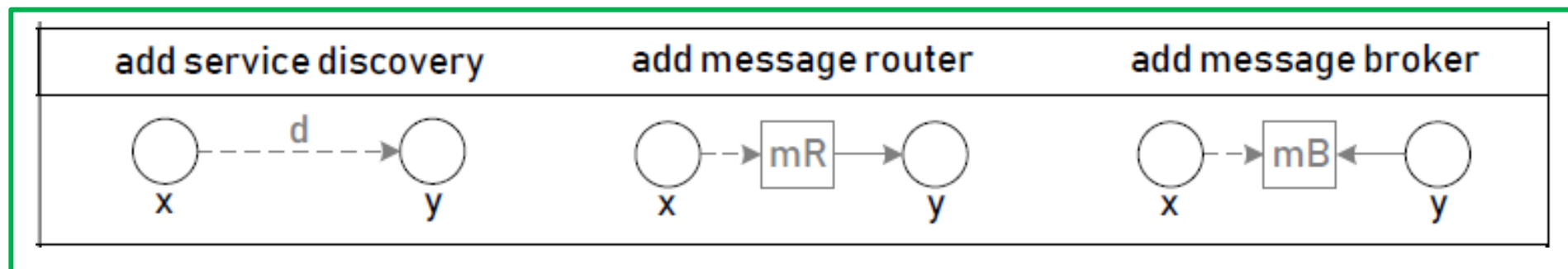
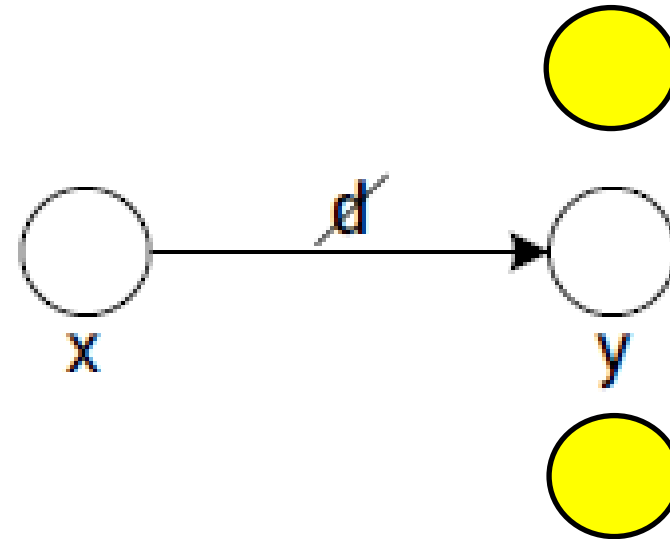


Example



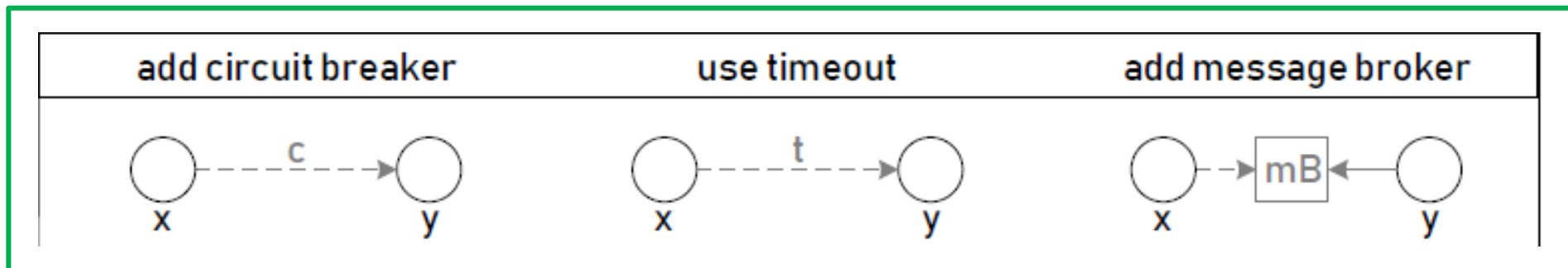
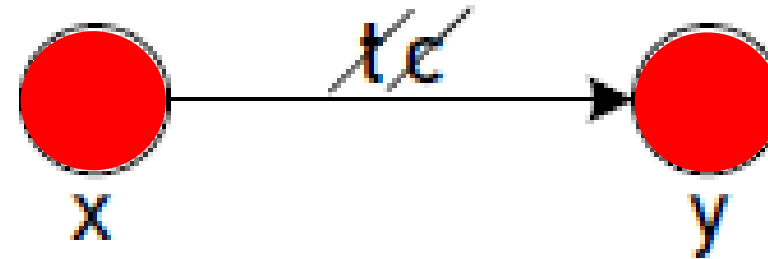
μFreshener: horizontal scalability

endpoint-based service interaction



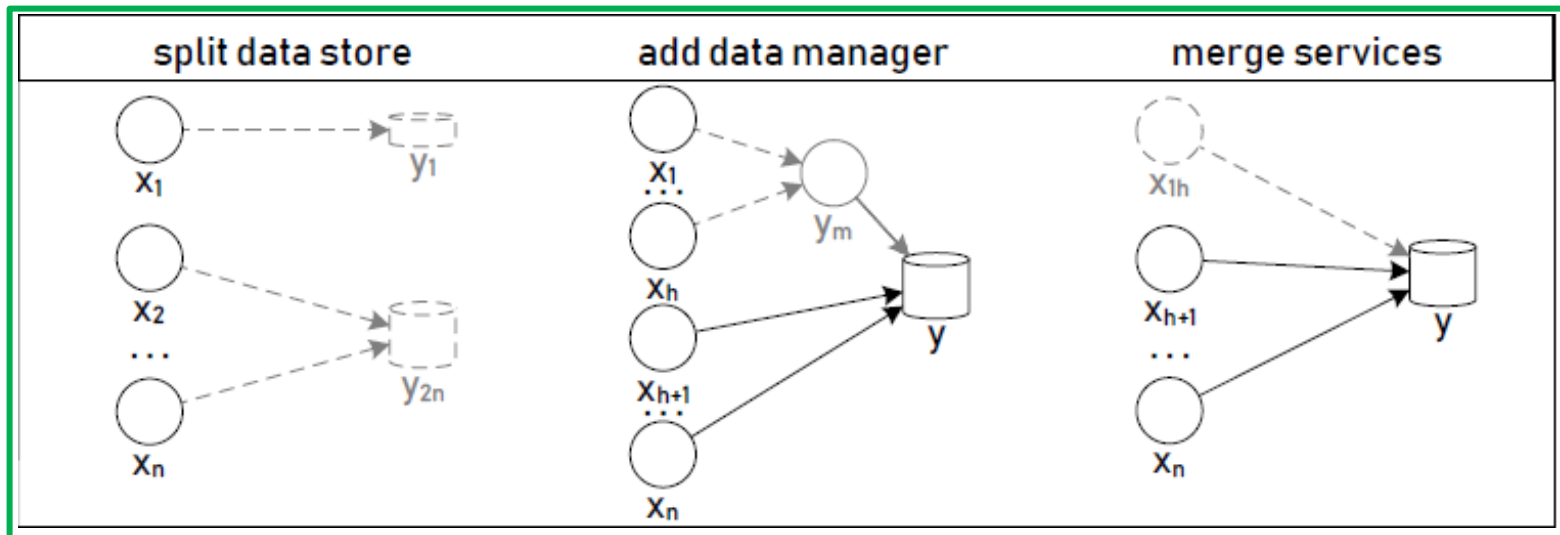
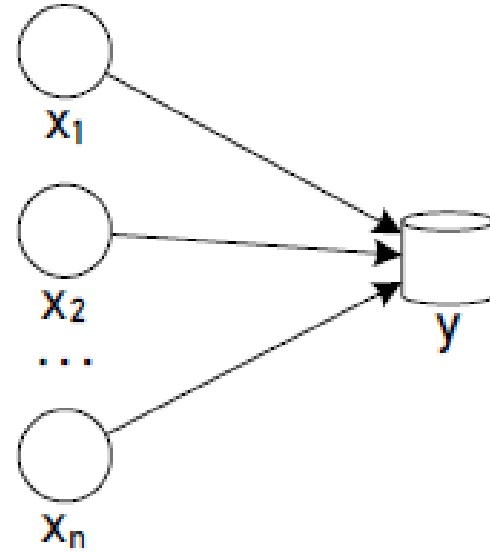
μ Freshener: isolation of failures

wobbly service interaction



μ Freshener: decentralisation

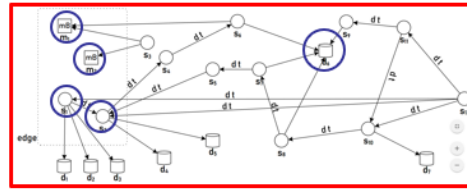
shared persistence



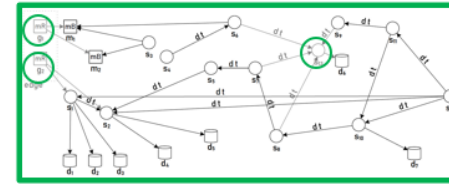
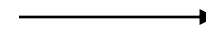
Remarks 1/2

- mFreshener (freely) usable to analyse & refactor microservice-based apps

- industrial case study



- 4 no API gateway smells
- 1 shared persistence smell



- 2 API gateways added
- 1 data manager added

- controlled experiment (100% vs. 49% smells identified, 83% vs. 1% resolved all smells)

- a smell is not necessarily a principle violation



≠



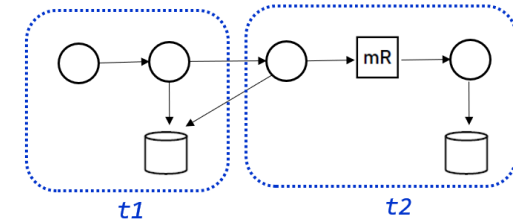
- “let it be” refactoring supported



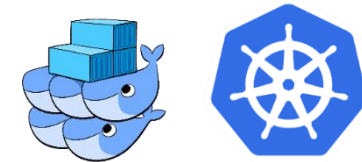
Remarks 2/2

- μ Freshener works at the architecture level
concrete implementation of refactoring left to application manager – much like in design patterns

- scalability: μ Freshener features team-based view



- ongoing work: dealing with container orchestration



Can I play with μ Freshener?



<https://github.com/di-unipi-socc/microFreshener>

Microservices, microservices, microservices ...

Design principles, architectural smells and refactoring

From incomplete specs to running apps

Motivations

- Microservice-based applications integrate many interacting services

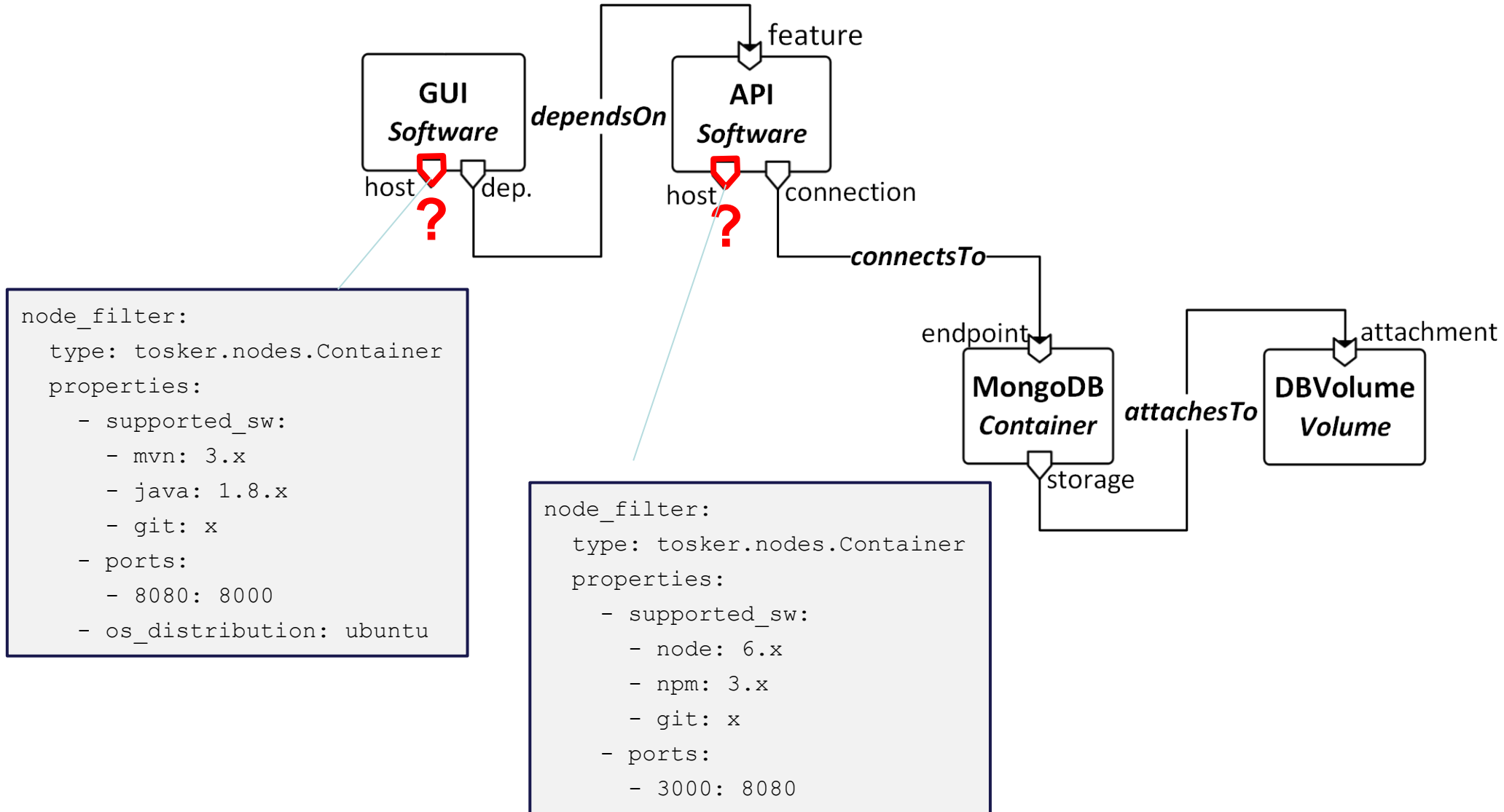
→ Need to select an appropriate runtime environment for each microservice

→ Need to package each microservice into the selected runtime environment

Idea (1/2)

Exploit the TOSCA-based representation of microservice-based applications to **specify only the application components and the software support they need**

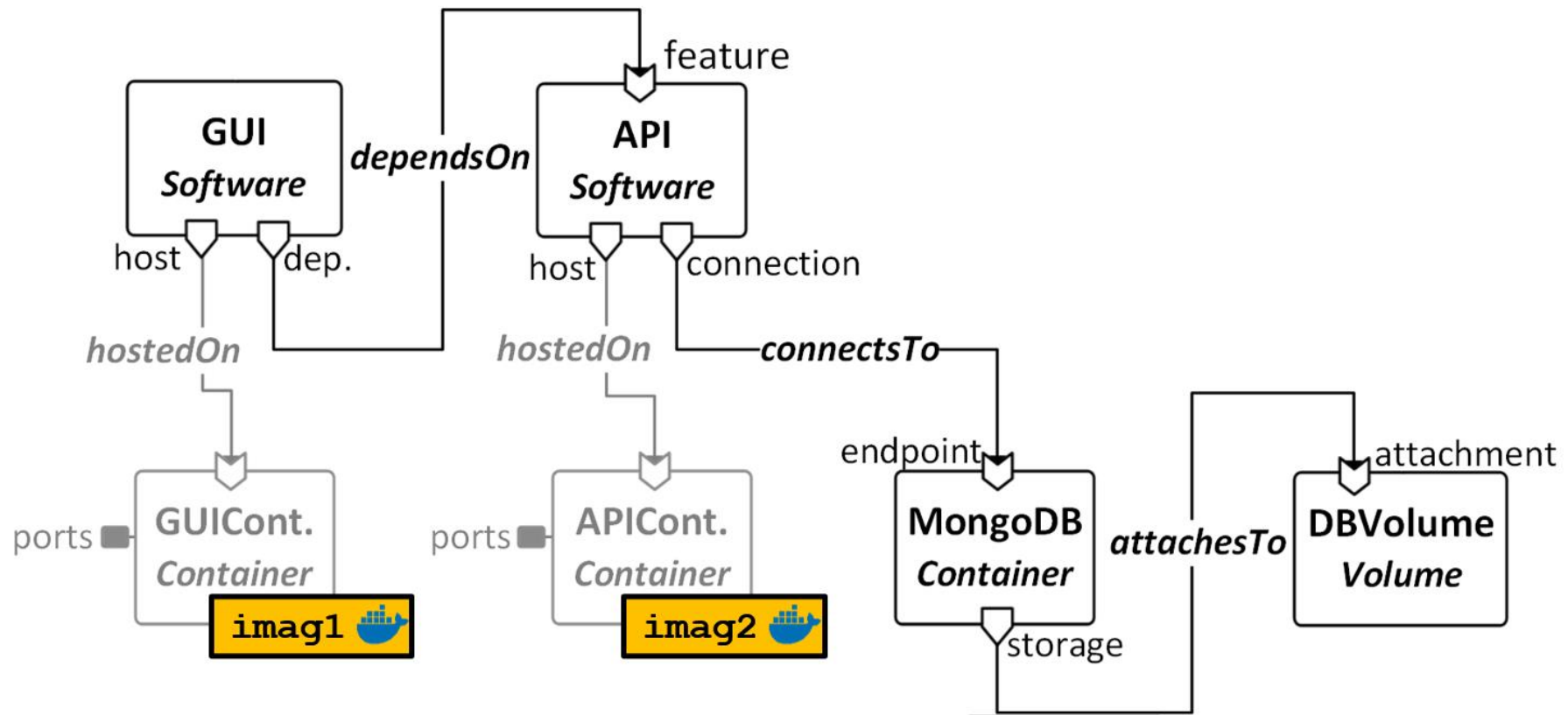
Example



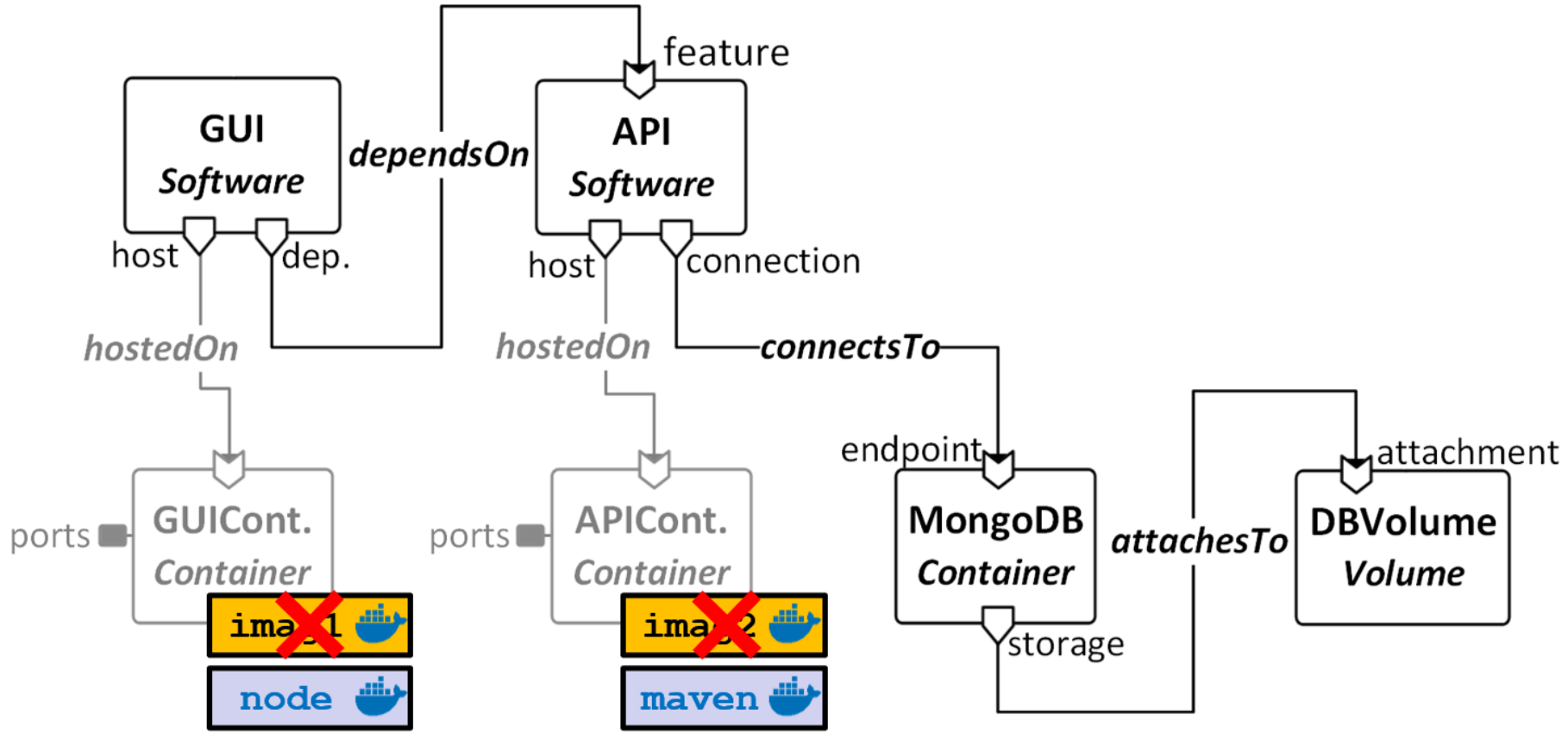
Idea (2/2)

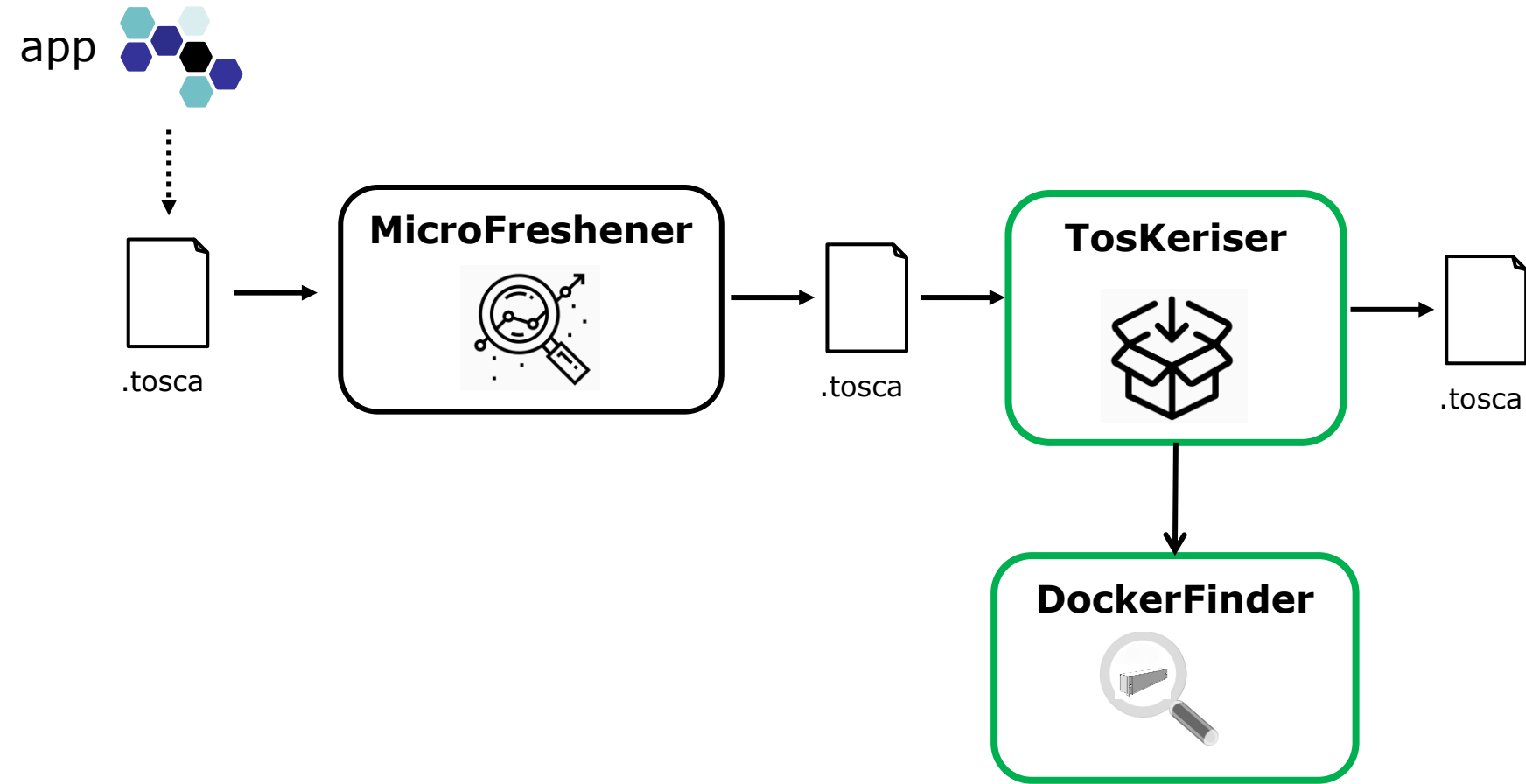
Develop a tool for **automatically completing (and updating) TOSCA application specifications by discovering and including Docker-based runtime environments** providing the software support needed by each microservice

```
$ toskerise thinking.csar --policy size
```



```
$ toskerise thinking.completed.csar -f --policy most_used
```





A. Brogi, D. Neri, L. Rinaldi, J. Soldani. *Orchestrating incomplete TOSCA applications with Docker*. *Science of Computer Programming*. 2018.

A. Brogi, D. Neri, J. Soldani. *A microservice-based architecture for (customisable) analyses of Docker images*. *Software: Practice and Experience*. 2018.

Motivations

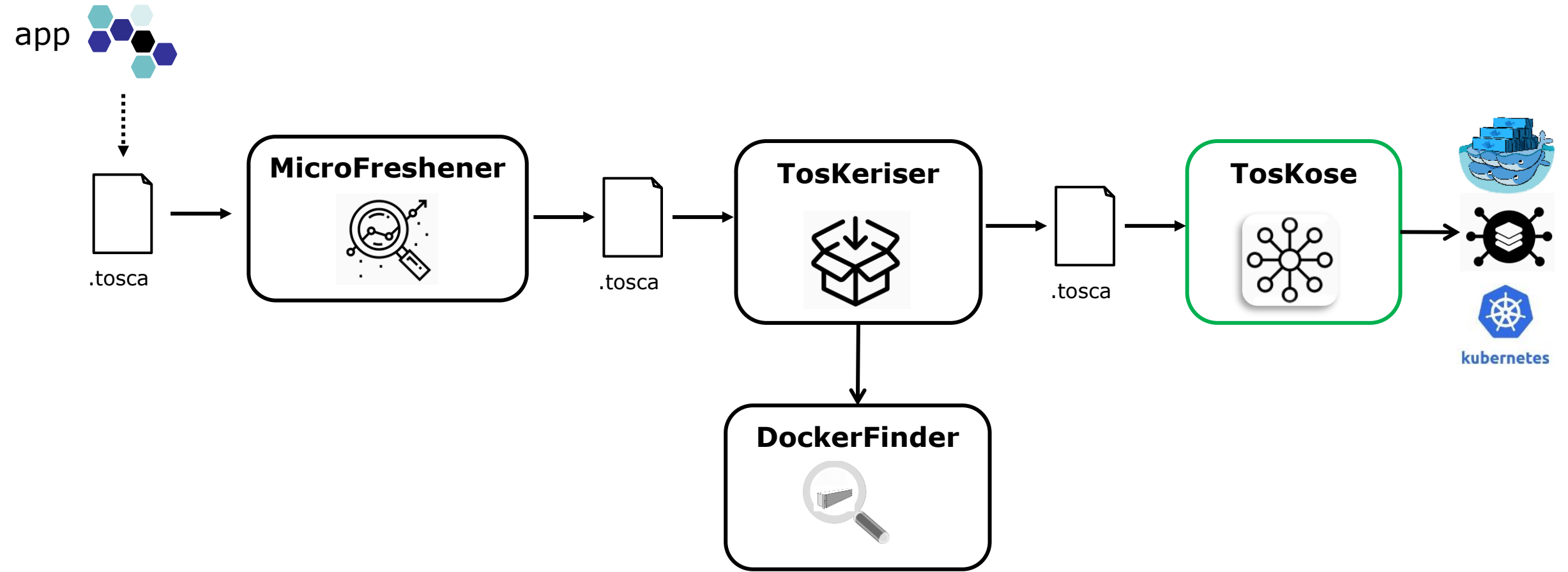
- Microservice-based applications integrate many interacting services
 - Need to select an appropriate runtime environment for each microservice
 - Need to package each microservice into the selected runtime environment

Idea

Develop a tool to **automate the deployment on top of existing container orchestrators**

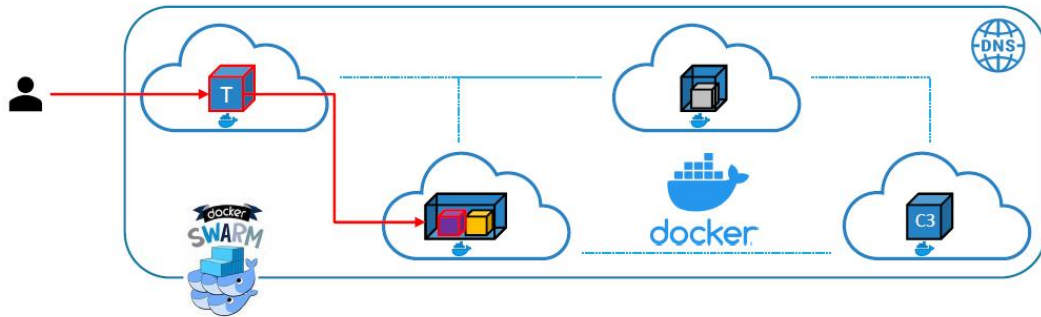
Ingredients:

- a *process management* system inside containers
- a *service* for component-aware orchestration
- a *packager* capable of deploying on existing container orchestrators

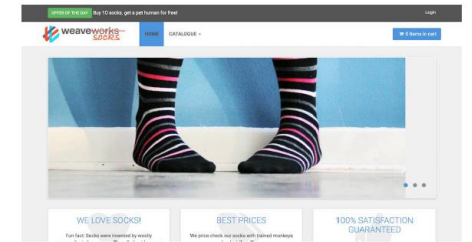


Case studies

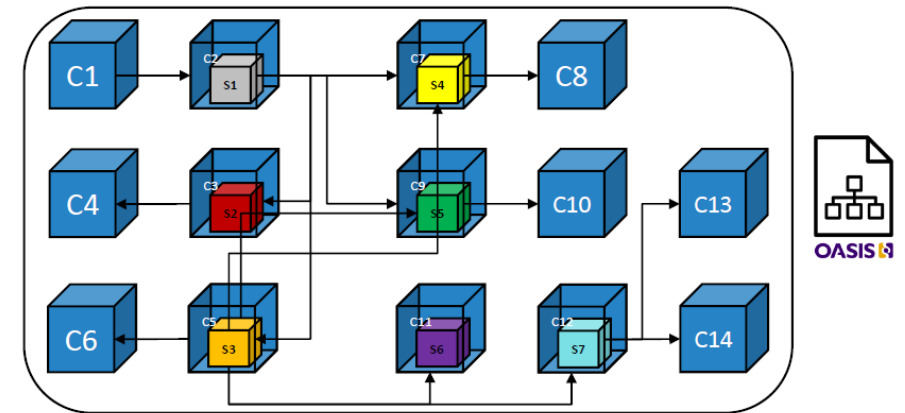
Thinking



Sock Shop



- 7 software components
- 14 containers (7 standalone)
- Deploy on Cluster of 4 VMs with Docker Swarm



Can I play with these tools too?

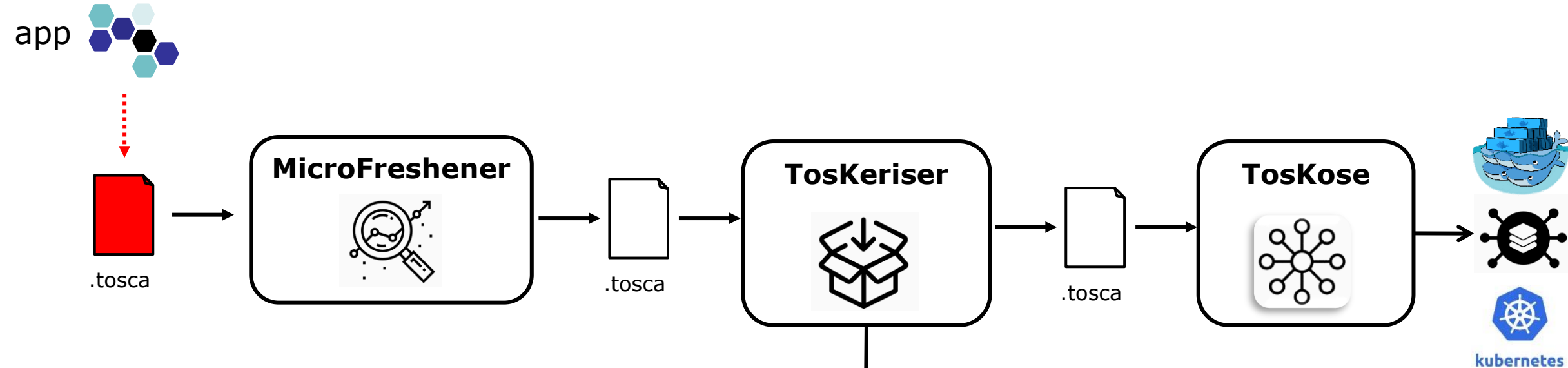


<https://github.com/di-unipi-socc/TosKeriser>



<https://github.com/di-unipi-socc/DockerFinder>

<https://github.com/di-unipi-socc/toskose>



One sec ... do I have to write myself the TOSCA spec of my app?



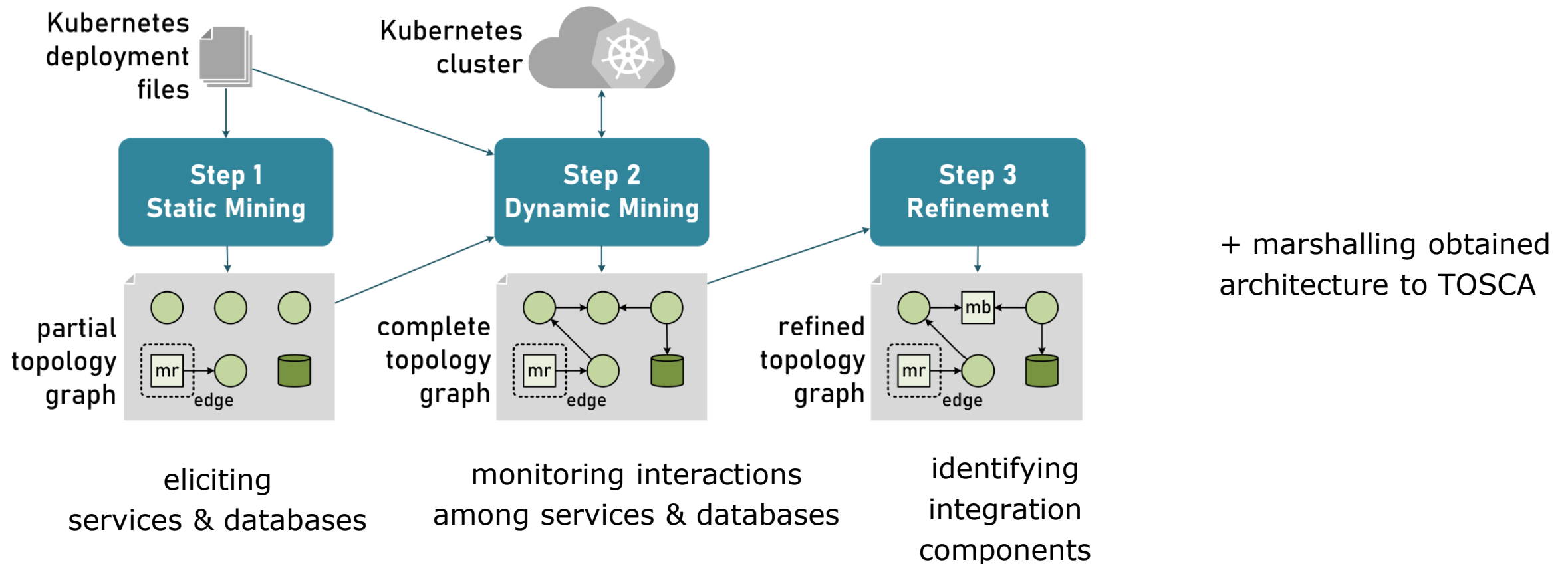
Microservices, microservices, microservices ...

Design principles, architectural smells and refactoring

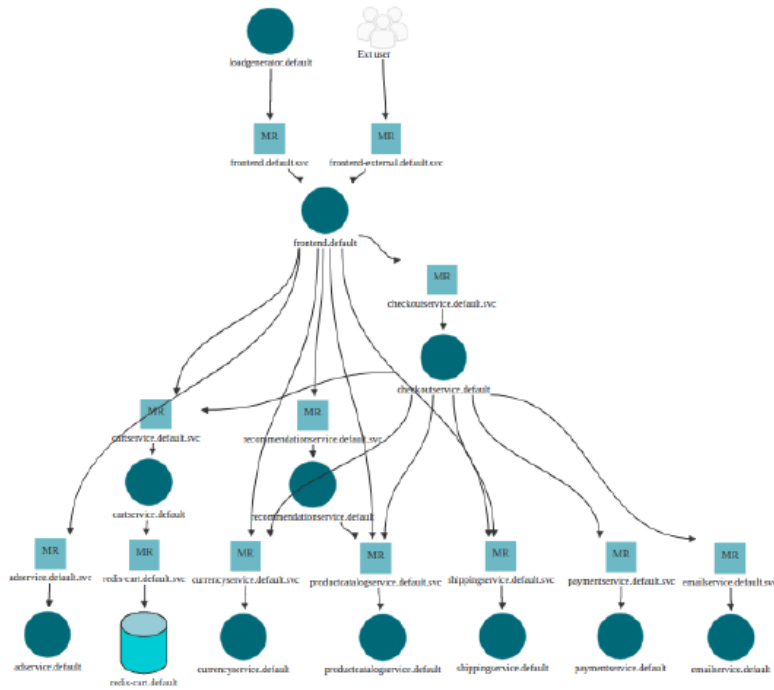
From incomplete specs to running apps

Mining the architecture of microservice-based apps

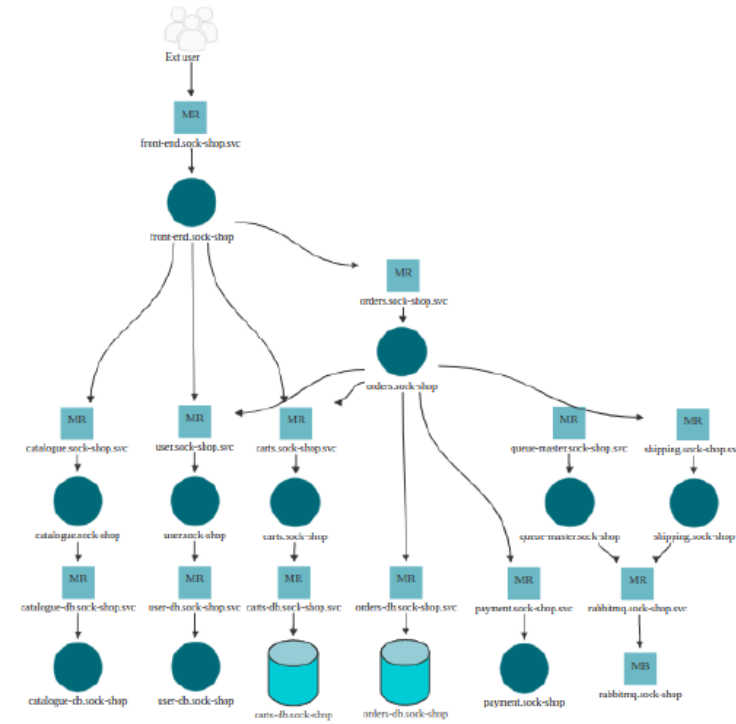
Automatically deriving the architecture of black-box applications



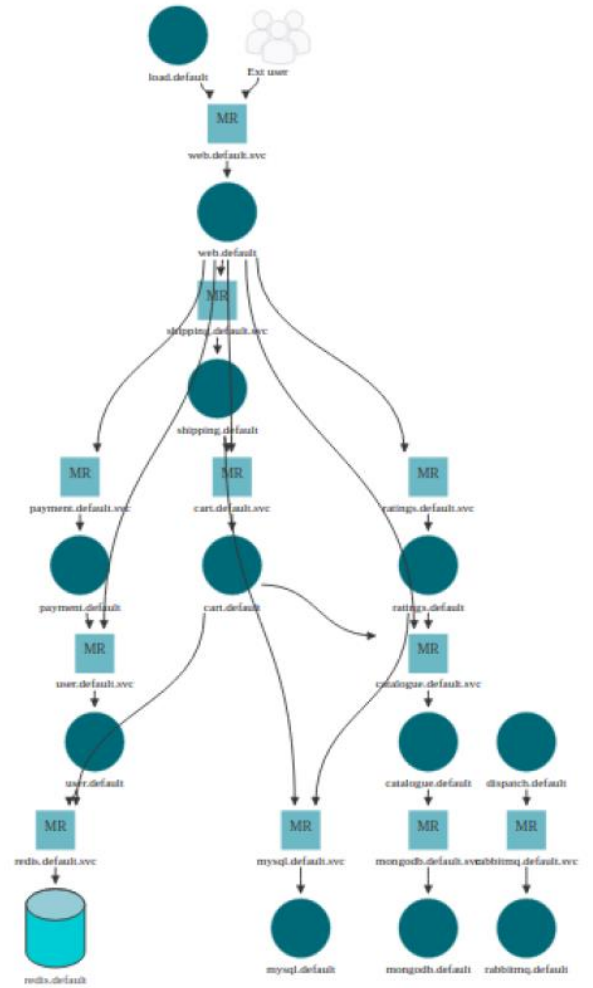
Case studies



Online boutique



Sock shop

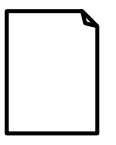


Robot shop


app 

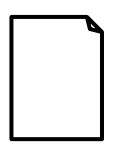
MicroMiner




.tosca

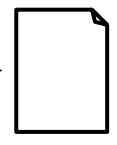
MicroFreshener



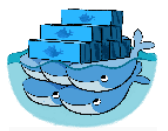



.tosca

TosKeriser





.tosca

TosKose

kubernetes

DockerFinder



Microservices, microservices, microservices ...

Design principles, architectural smells and refactoring

From incomplete specs to running apps

Mining the architecture of microservice-based apps

Concluding remarks

Take-home message: A (minimal) modelling of microservice-based applications can considerably simplify their design and analysis and allow automating their container-based completion and deployment



Many interesting research directions on microservices
(non-exhaustive, biased list):

- DSLs for microservices
- Security
- Monitoring
- Identifying failure causalities
- Continuous reasoning
- Green computing
- ...



... and thanks to



J. Soldani



D. Neri



O. Zimmermann



M. Bogo



G. Muntoni



L. Rinaldi



Microservices beyond COVID-19

Antonio Brogi

Department of Computer Science
University of Pisa, Italy

MICROSERVICES 2020 - 3rd International Conference on Microservices
Invited keynote - September 10th, 2020