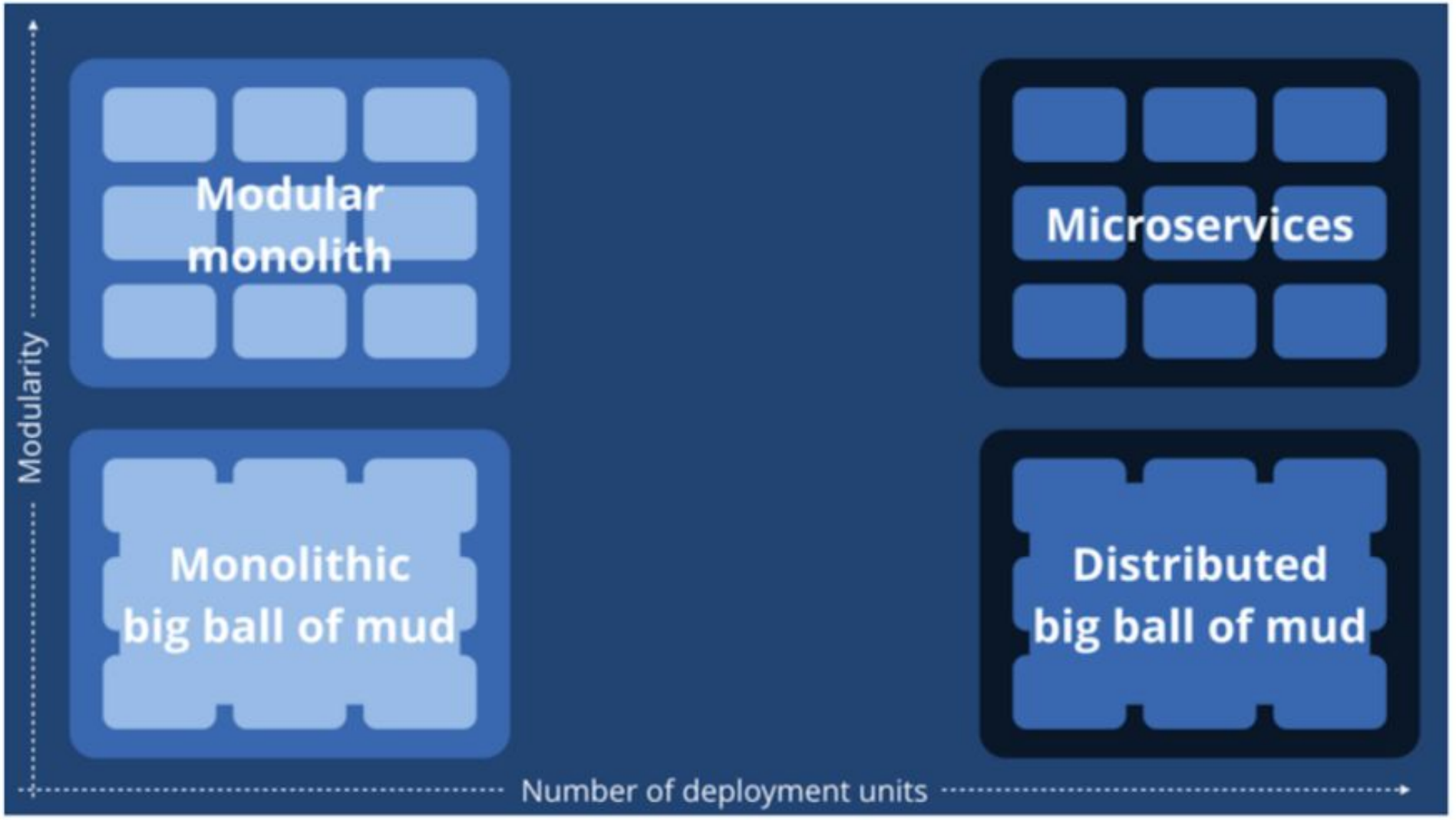
Deconstructing the Monolith

Designing Software that Maximizes Developer Productivity

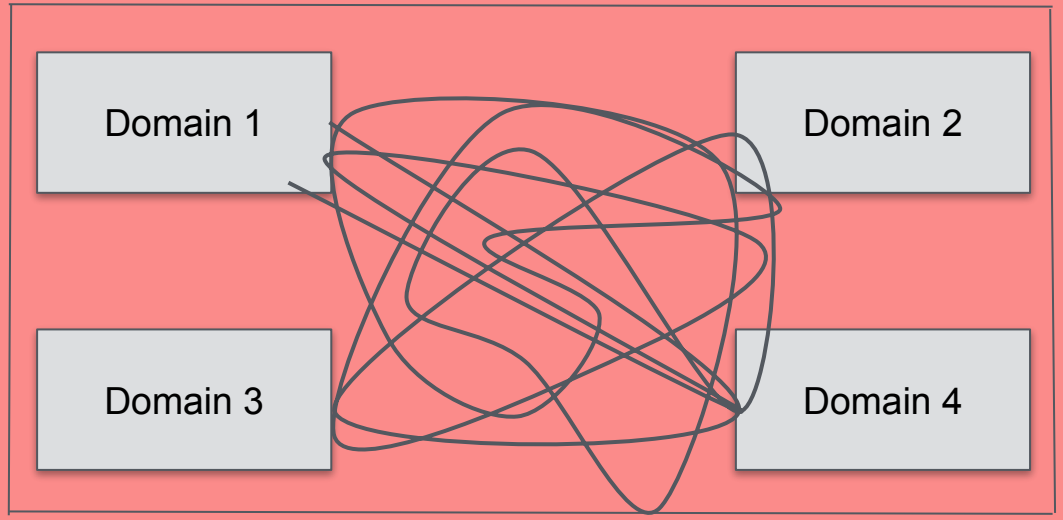
Kirsten Westeinde
Senior Software Engineer
shopify


Modularity
Modular monolith
Microservices
Monolithic big ball of mud
Distributed big ball of mud
Number of deployment units

Agenda

1. Define monoliths and microservices
2. Analyze pros & cons
3. Introduce a third option
4. Conclusion: is there a *right* choice?

Monolith

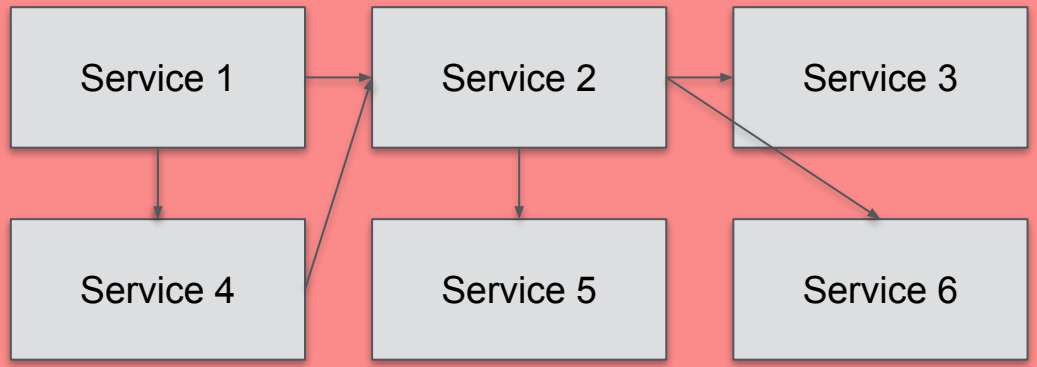


“A software system in which functionally distinguishable aspects are all interwoven, rather than containing architecturally separate components.”

Source: Wikipedia

Microservices

“Suite of independently deployable, small, modular services in which each service runs a unique process and communicates through a well-defined, lightweight mechanism to serve a business goal”



Source: Smartbear

Monoliths

Pros	Cons
• The code is all in one place • One test/deployment pipeline • All data is available everywhere • One infrastructure	• Slow tests • New code having unexpected repercussions • Need to understand the entire codebase

Microservices

Pros	Cons
• Each service is lightweight and has one clear responsibility • Individuals/teams can work on separate services • Individual systems are easily understandable • Services can scale independently	• Many different test/deployment pipelines • Infrastructural overhead for each service • Network communication needed between services • Data not always available where you need it • Large refactors can be tedious

EXPECTED BENEFITS OF MODULAR PROGRAMMING

The expected benefits of modular programming fall into three classes: (1) managerial -- development time could be shortened because separate groups would work on each module with little need for communication (and little regret afterward that there had not been more communication); (2) product flexibility -- it was hoped that it would be possible to make quite drastic changes or improvements in one module without changing others; (3) comprehensibility -- it was hoped that the system could be studied a module at a time with the result that the whole system could be better designed because it was better understood.

Department of Computer Science
Carnegie-Mellon University
Pittsburgh, Pa.

August, 1971

Source: “On the Criteria to be used in decomposing systems into modules” by D. L. Parnas

Is there a happy medium?

Modular Monolith

Monoliths

Pros	Cons
• The code is all in one place • One test/deployment pipeline • All data is available everywhere • One infrastructure	• Slow tests • New code having unexpected repercussions • Need to understand the entire codebase

Modular Monoliths

Pros	Cons
• The code is all in one place • One test/deployment pipeline • ~~All data is available everywhere~~ • One infrastructure	• ~~Slow tests~~ • ~~New code having unexpected repercussions~~ • ~~Need to understand the entire codebase~~

Microservices

Pros	Cons
• Each service is lightweight and has one clear responsibility • Individuals/teams can work on separate services • Individual systems are easily understandable • Services can scale independently	• Many different test/deployment pipelines • Data not always available where you need it • Infrastructural overhead for each service • Network communication needed between services • Large refactors can be tedious

Modular Monoliths

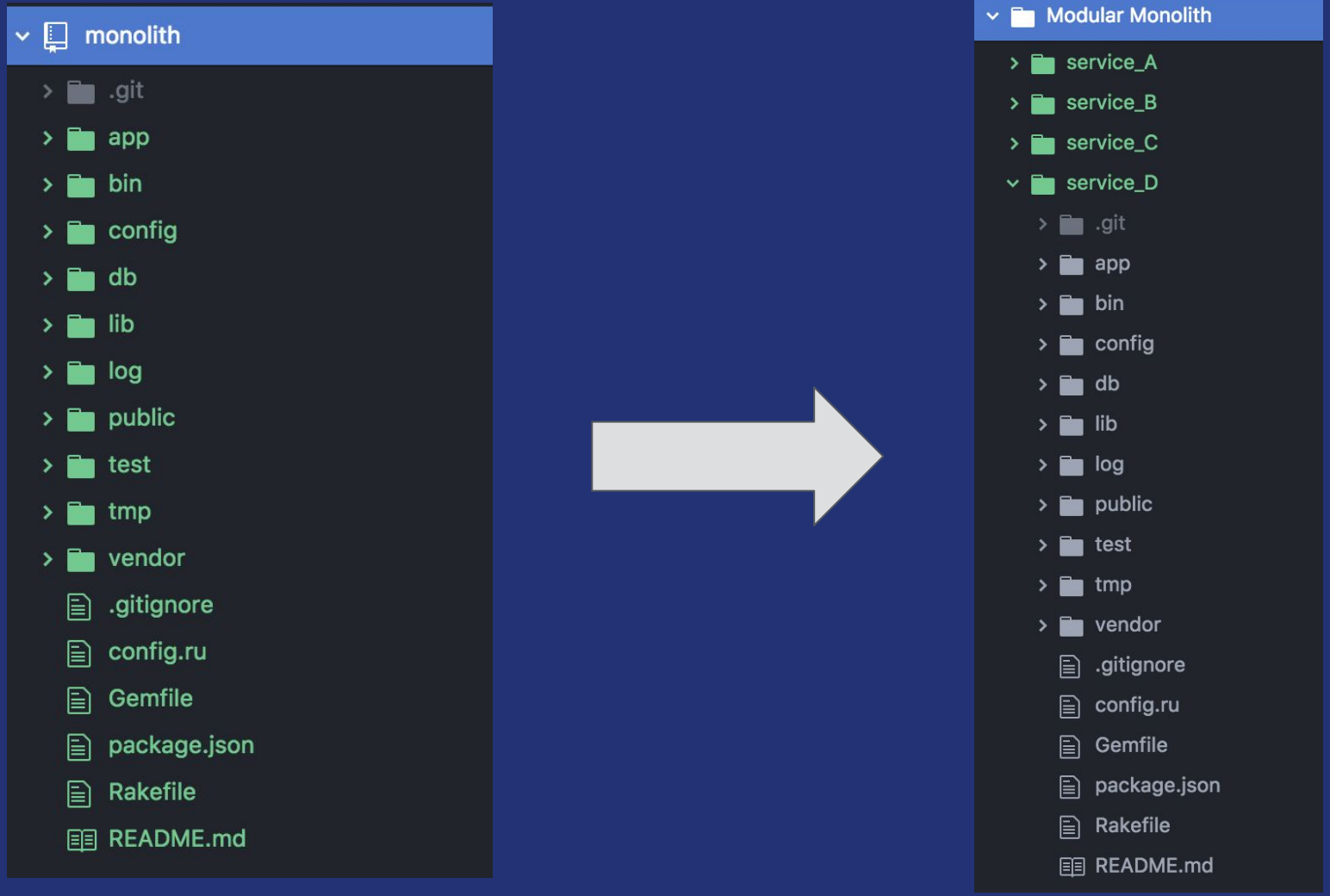
Pros	Cons
• Each ~~service~~ (module) is lightweight and has one clear responsibility • Individuals/teams can work on separate ~~services~~ (modules) • Individual systems are easily understandable • ~~Services can scale independently~~	• ~~Many different test/deployment pipelines~~ • Data not always available where you need it • ~~Infrastructural overhead for each service~~ • ~~Network communication needed between services~~ • ~~Large refactors can be tedious~~

Let’s talk implementation details

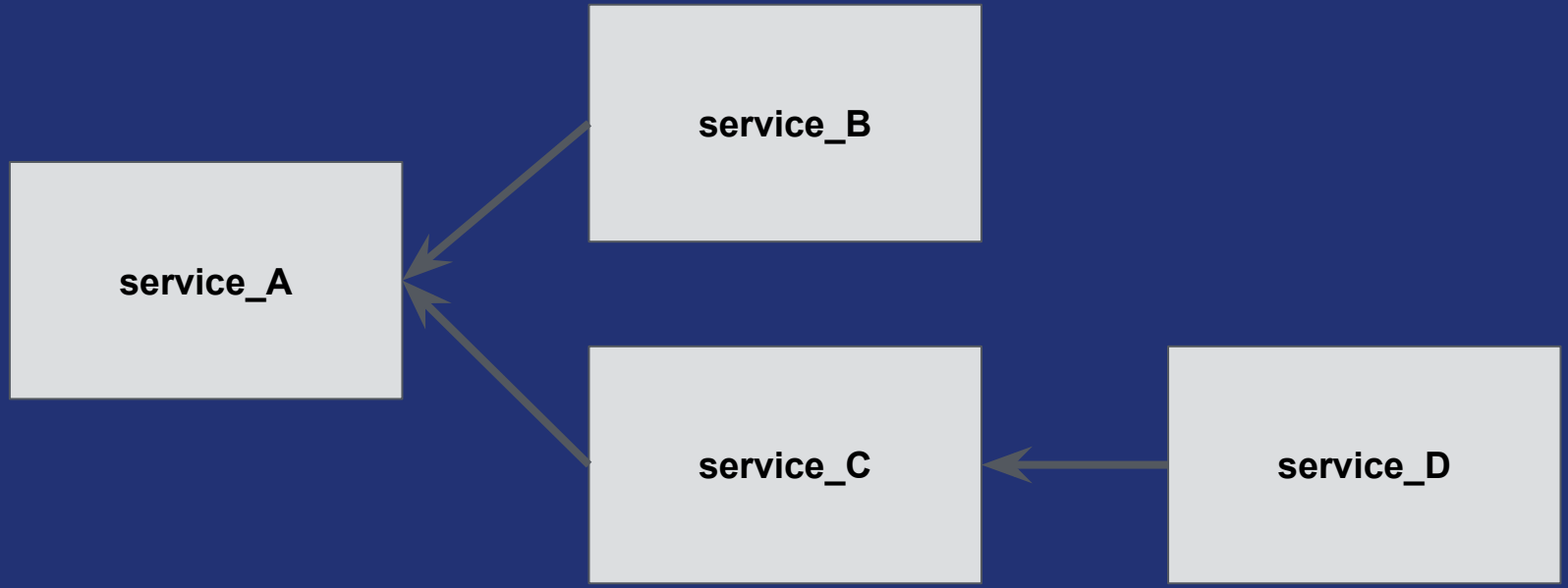
1. Namespace
2. Untangle
3. Enforce boundaries between modules

The Modular Monolith: Rails Architecture

Dan Mange

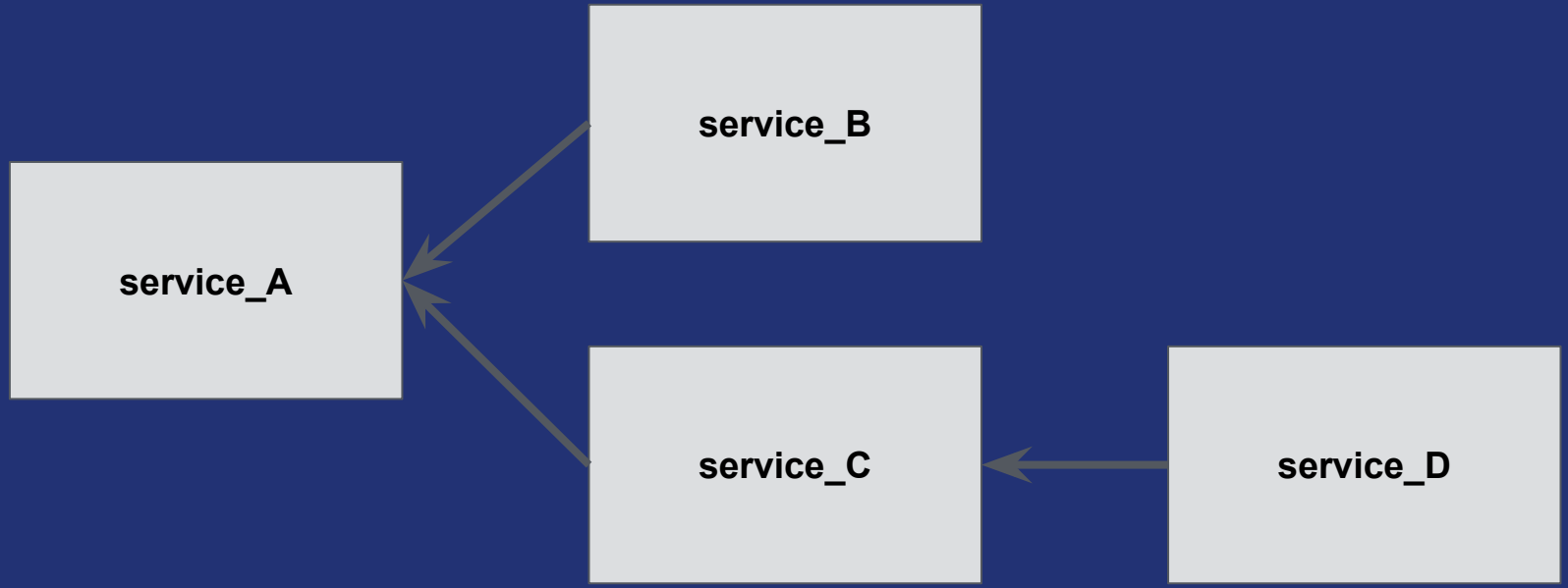

monolith
.git
app
bin
config
db
lib
log
public
test
tmp
vendor
.gitignore
config.ru
Gemfile
package.json
Rakefile
README.md
Modular Monolith
service_A
service_B
service_C
service_D
.git
app
bin
config
db
lib
log
public
test
tmp
vendor
.gitignore
config.ru
Gemfile
package.json
Rakefile
README.md

Dependency Graph



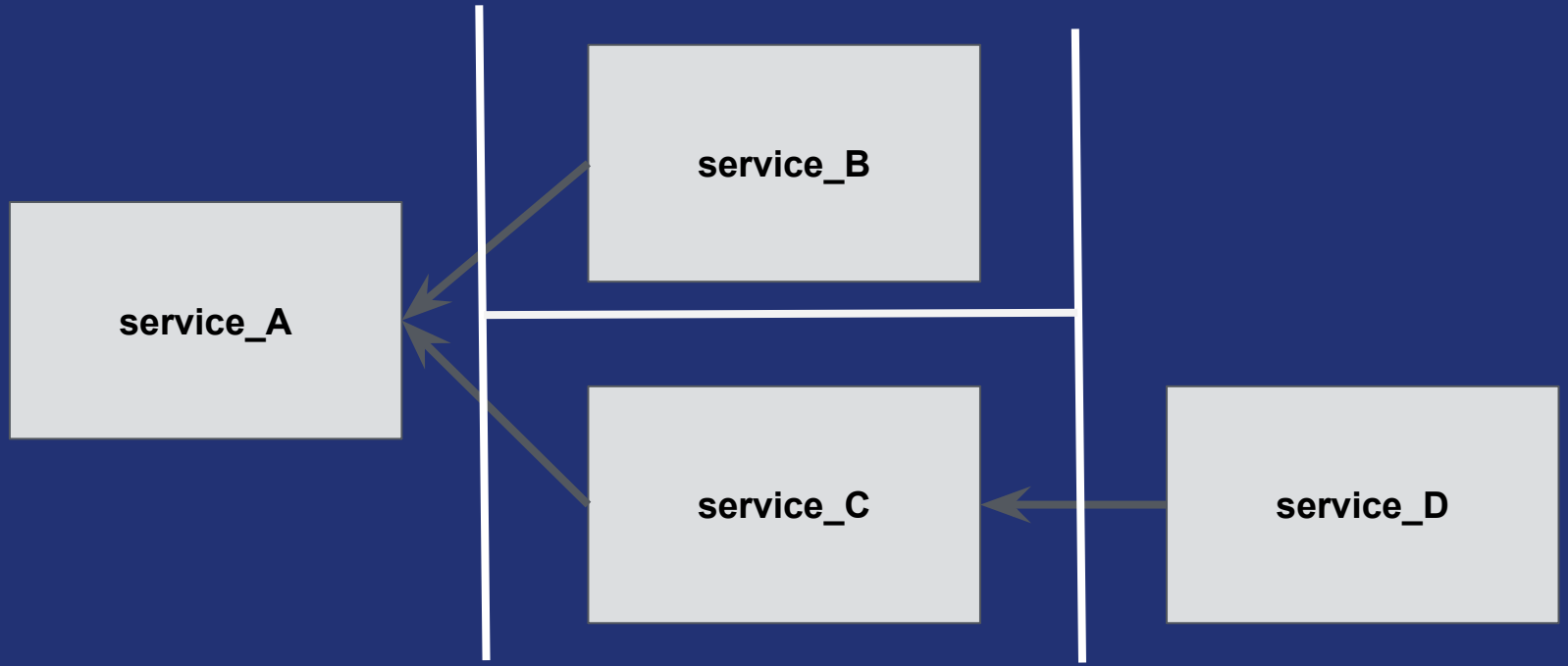
Source: Giphy

Dependency Graph



Source: 123rf

Transitioning to Microservices



Event Driven Architecture drives communication between services and enables near real-time processing.

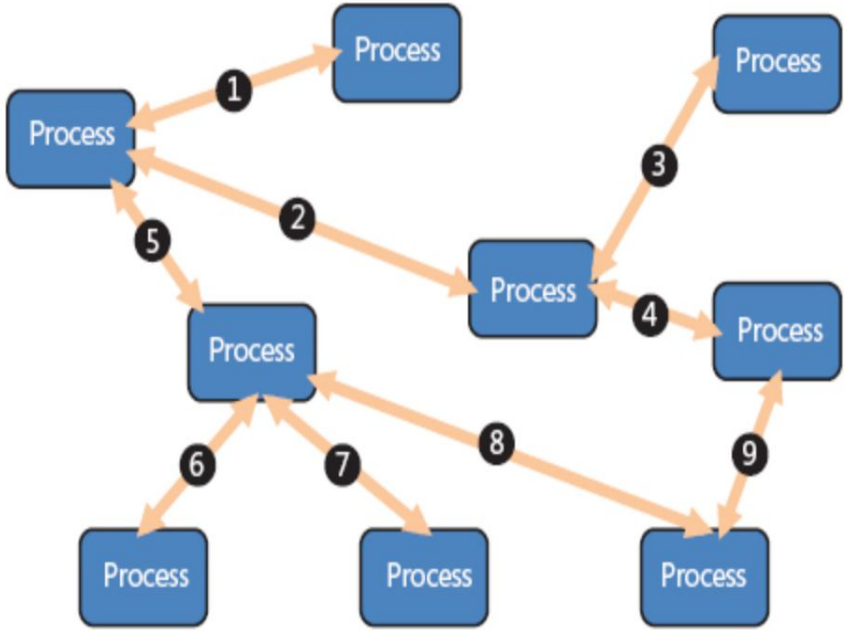


Figure 1. Request-driven architecture (logical view)

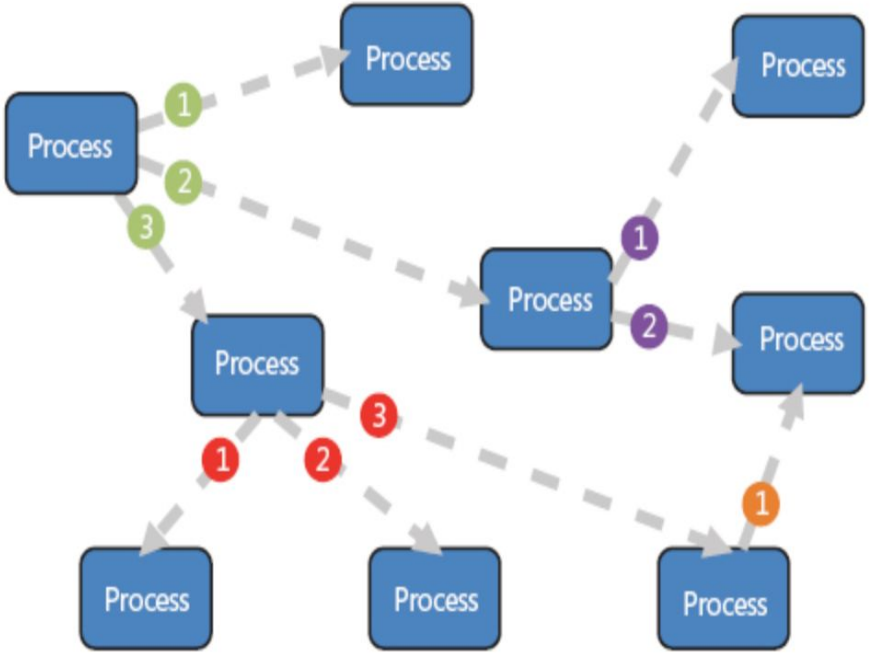
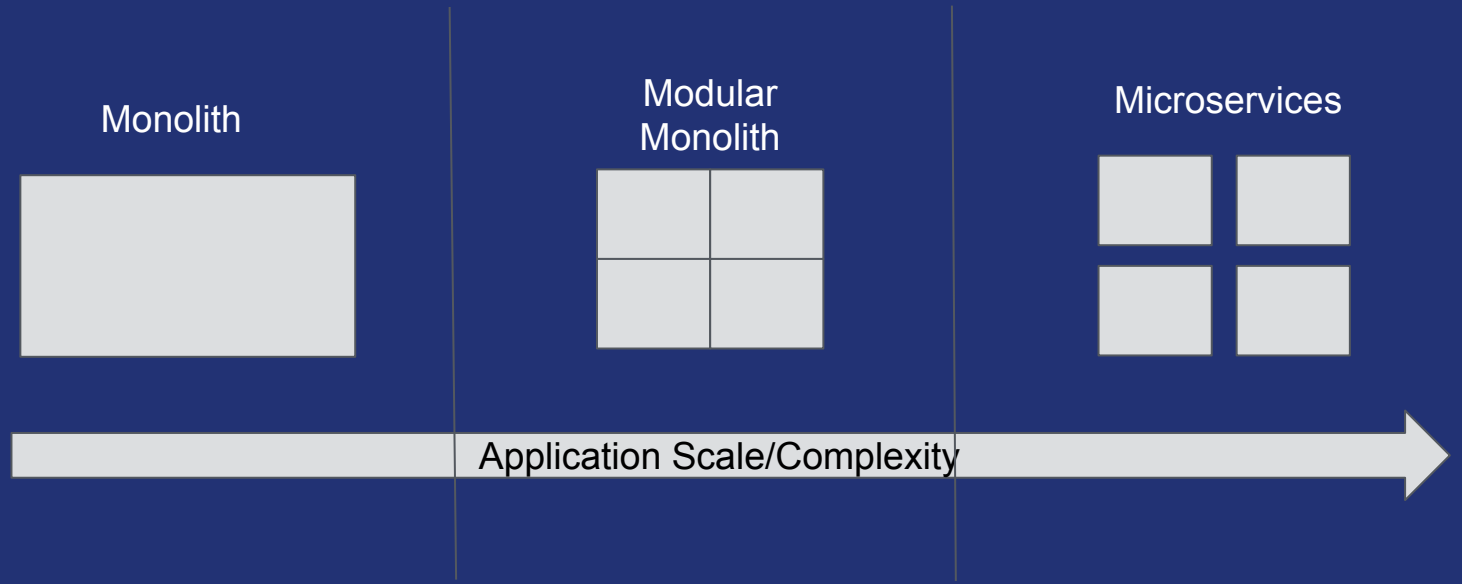


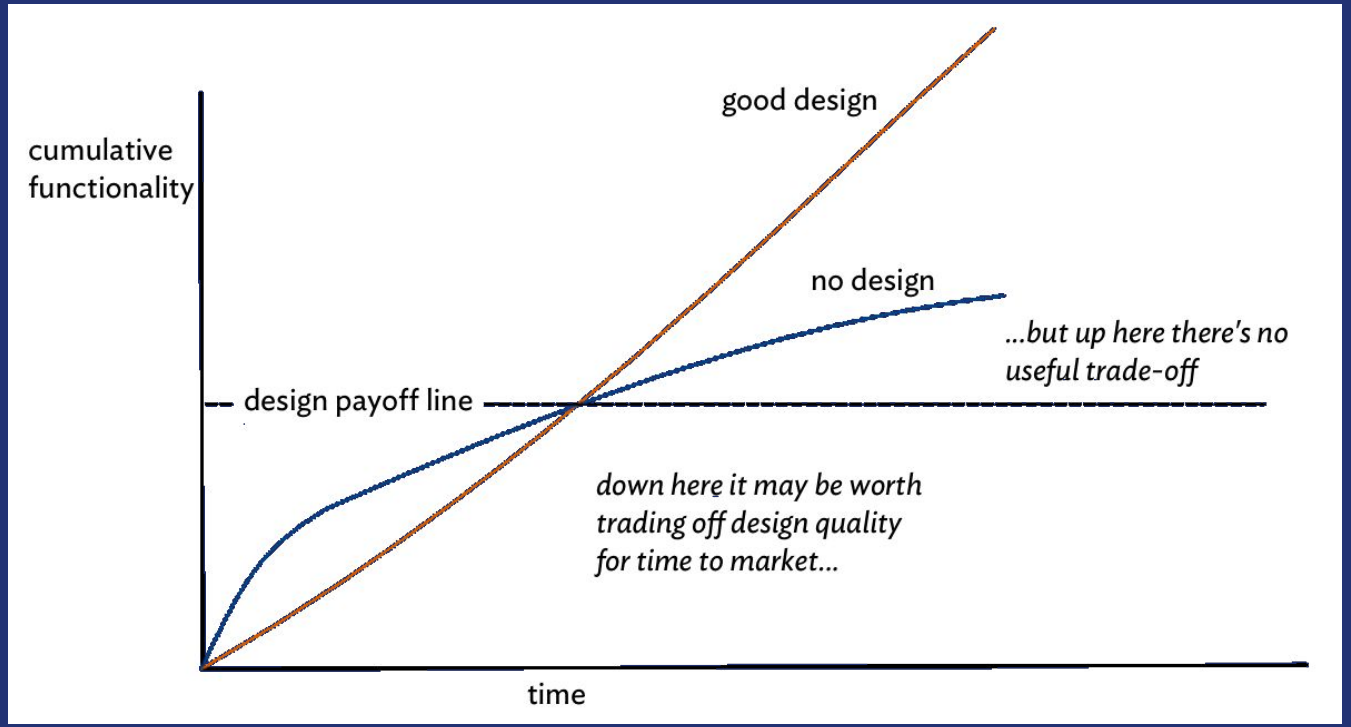
Figure 2. Event-driven architecture (logical view)

Source: Microsoft


Monolith
Modular Monolith
Microservices
Application Scale/Complexity

Design Stamina Hypothesis

Martin Fowler



Source: Martin Fowler

“When you have a hammer
Everything looks like a nail.”

- Abraham Maslow

Thanks!

shopify

Resources

- Wikipedia: Monolithic System: https://en.wikipedia.org/wiki/Monolithic_system
- Smartbear: what are microservices? https://smartbear.com/learn/api-design/what-are-microservices/
- Carnegie Mellon University: “On the Criteria to be used in decomposing systems into modules” by D. L. Parnas https://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci
- Medium: The modular monolith by Dan Menge https://medium.com/@dan_manges/the-modular-monolith-rails-architecture-fb1023826fc4
- Giphy: Boundaries https://media.giphy.com/media/26tn42f4M1jbuAMM0/giphy.gif
- 123rf: Red stop icon https://www.123rf.com/photo_89037069_stock-vector-red-stop-icon-on-transparent-background-no-symbol-vector-illustration.html
- Microsoft: Using events in highly distributed architectures https://msdn.microsoft.com/en-us/library/dd129913.aspx
- Martin Fowler: The Design Stamina Hypothesis https://martinfowler.com/bliki/DesignStaminaHypothesis.html
- Lean IX: 2017 Survey https://cdn2.hubspot.net/hubfs/2570476/Sales%20info/leanIX_Microservices-Study.pdf