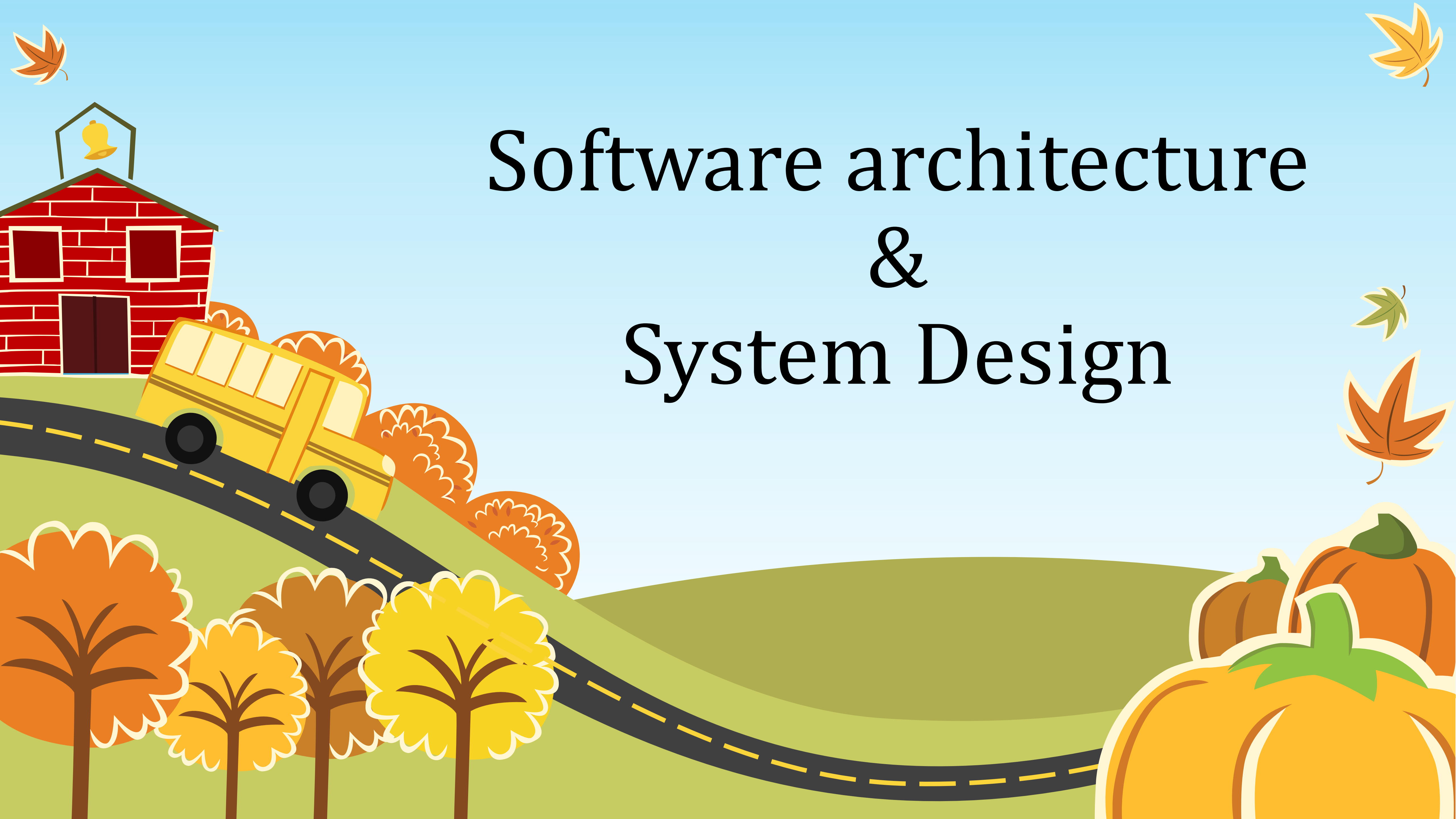
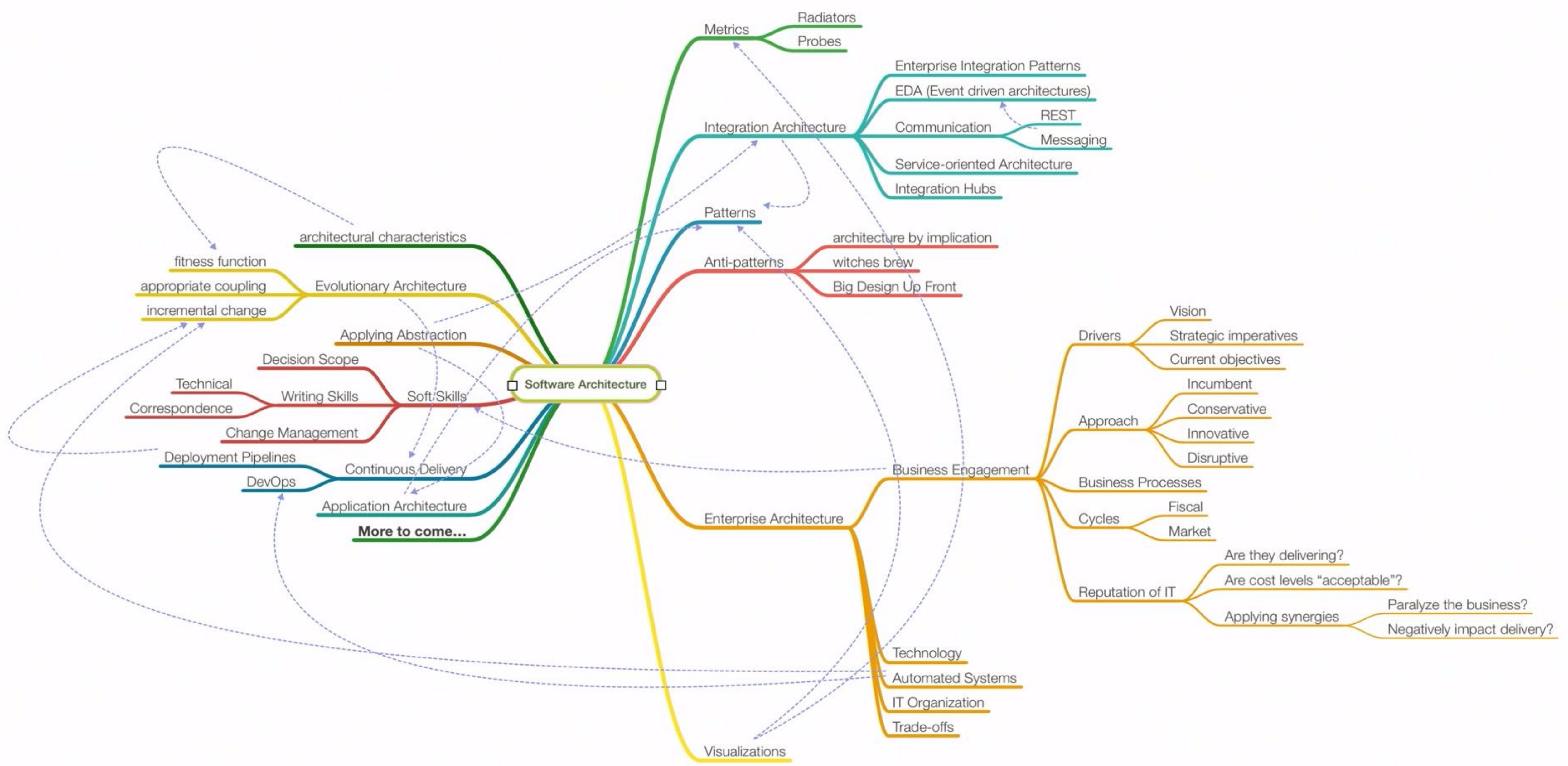
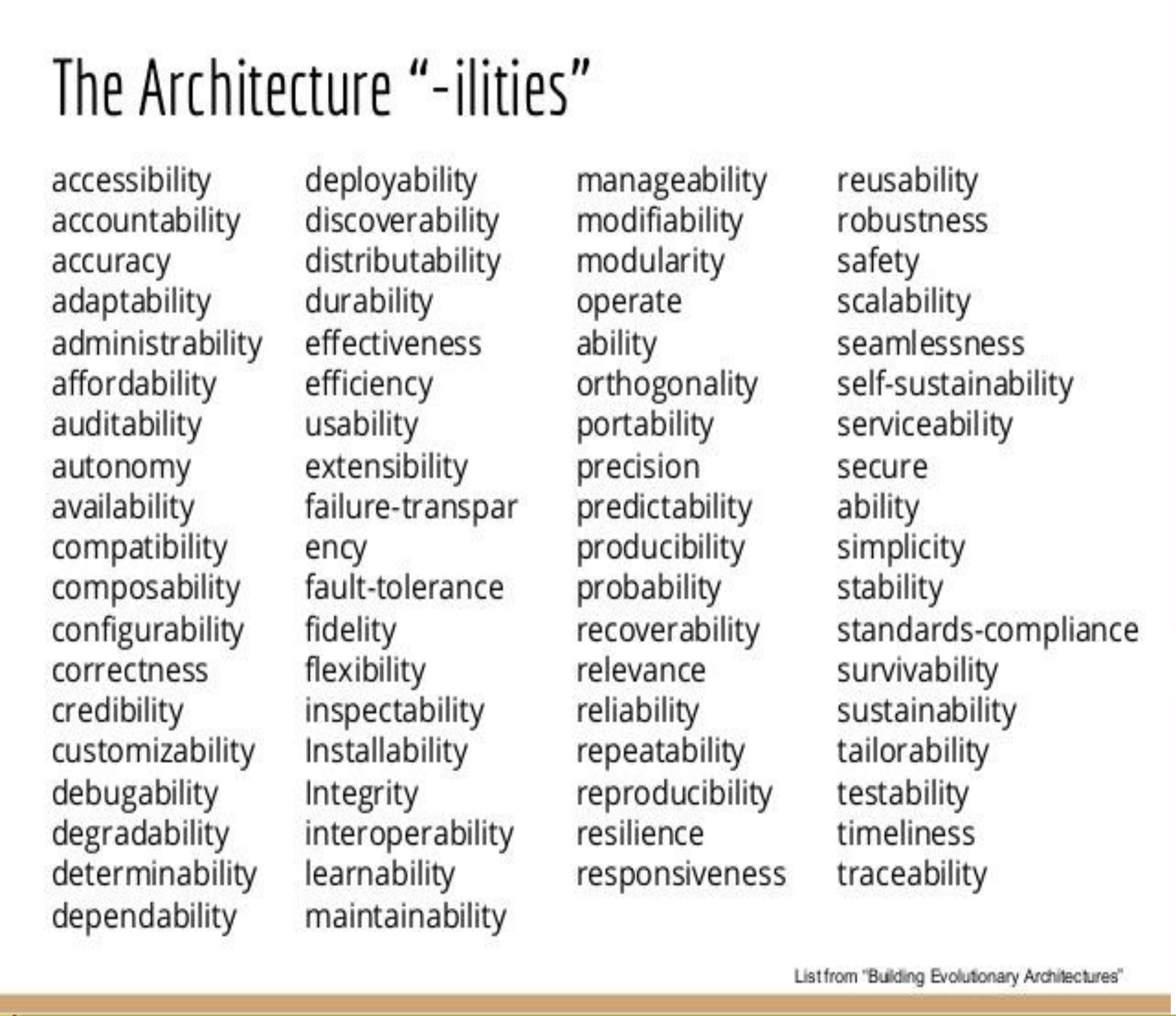
Software architecture & System Design



What is Software Architecture ?



What is Software Architecture ?



The Architecture "-ilities"

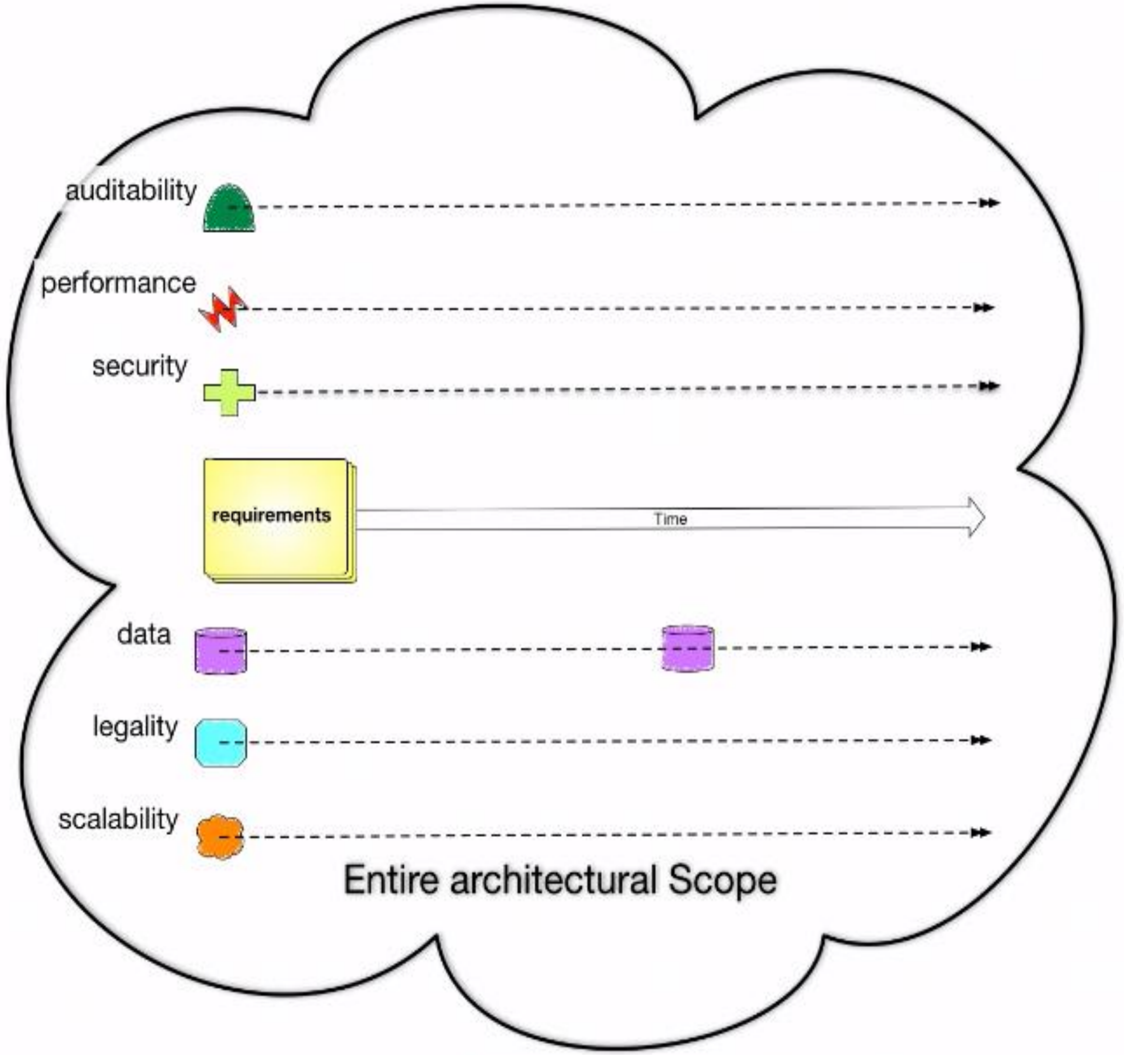
accessibility
accountability
accuracy
adaptability
administrability
affordability
auditability
autonomy
availability
compatibility
composability
configurability
correctness
credibility
customizability
debugability
degradability
determinability
dependability

deployability
discoverability
distributability
durability
effectiveness
efficiency
usability
extensibility
failure-transpar
ency
fault-tolerance
fidelity
flexibility
inspectability
Installability
Integrity
interoperability
learnability
maintainability

manageability
modifiability
modularity
operate
ability
orthogonality
portability
precision
predictability
producibility
probability
recoverability
relevance
reliability
repeatability
reproducibility
resilience
responsiveness

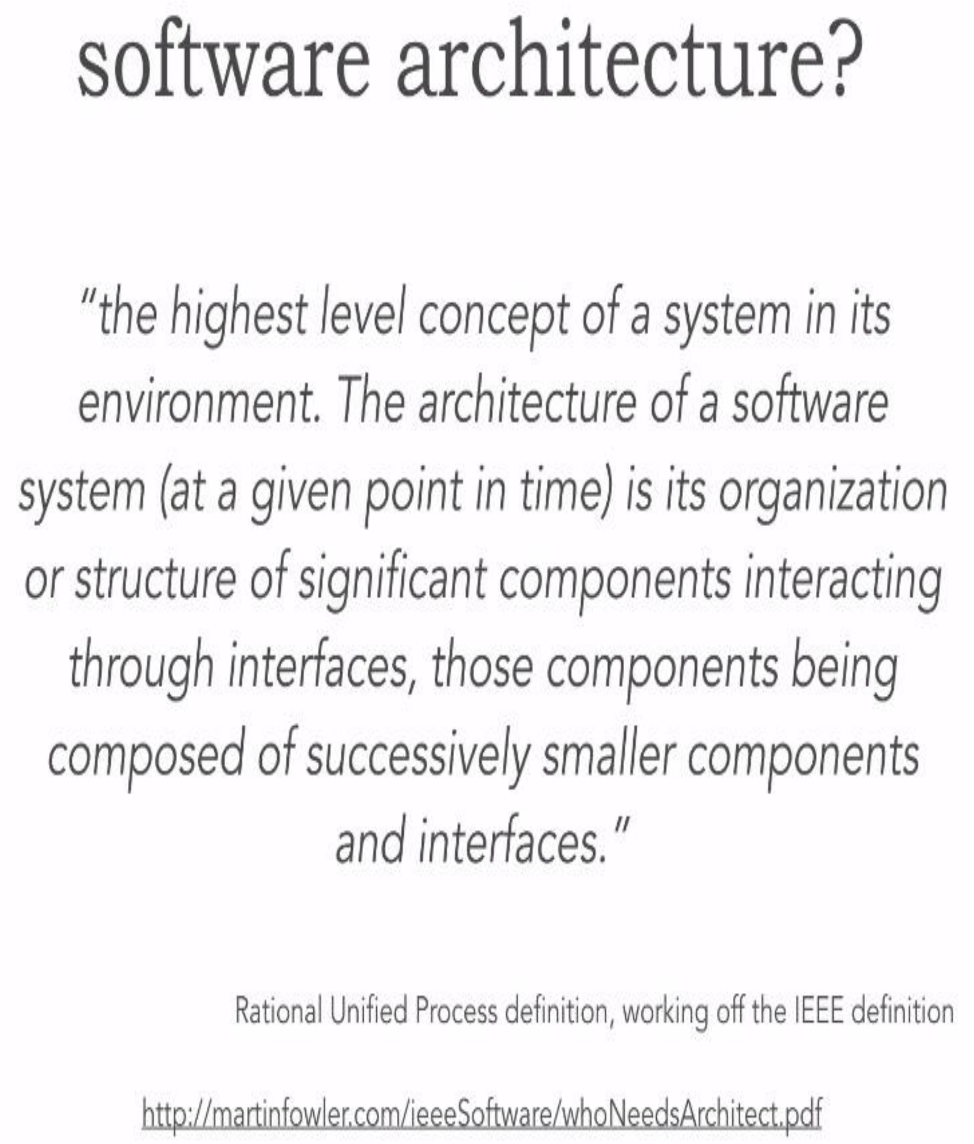
reusability
robustness
safety
scalability
seamlessness
self-sustainability
serviceability
secure
ability
simplicity
stability
standards-compliance
survivability
sustainability
tailorability
testability
timeliness
traceability

List from "Building Evolutionary Architectures"



https://en.wikipedia.org/wiki/List_of_system_quality_attributes

Definition - Software Architecture

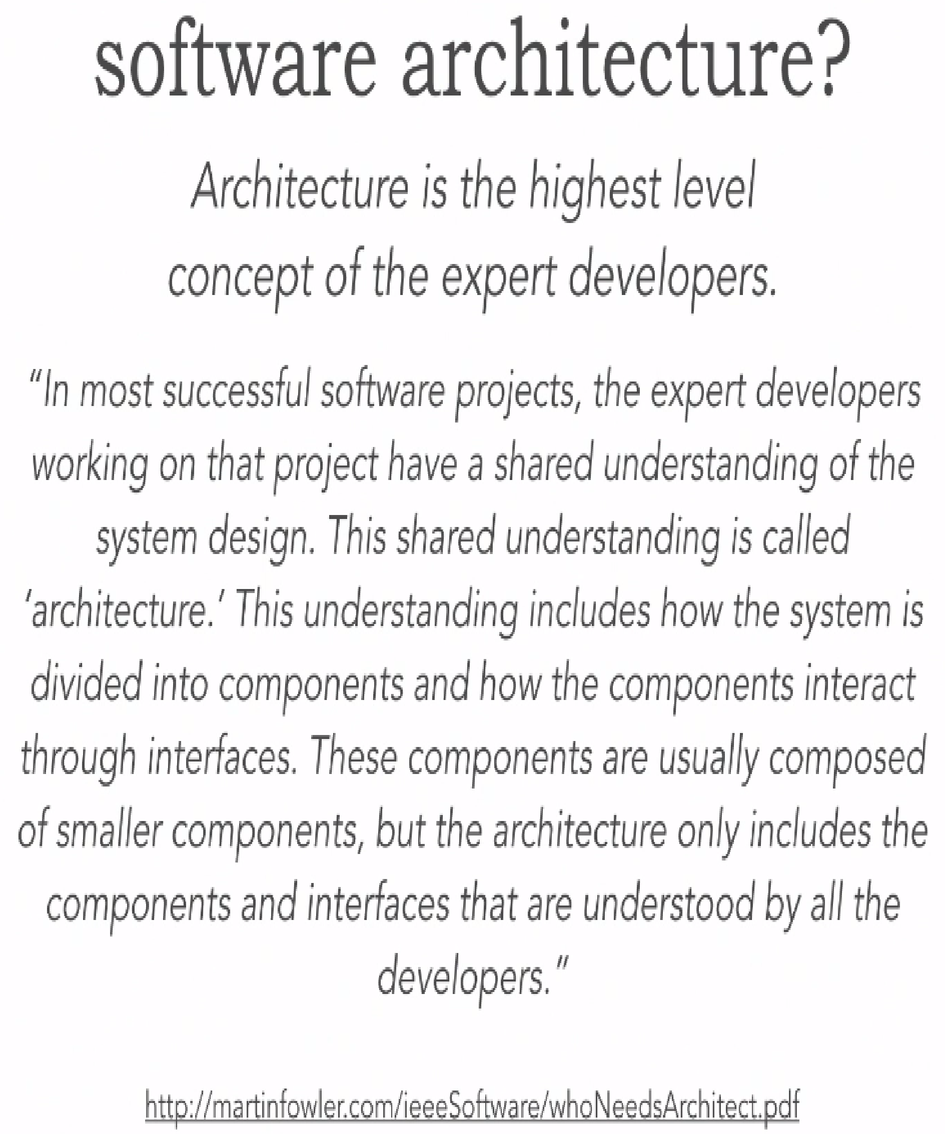


software architecture?

"the highest level concept of a system in its environment. The architecture of a software system (at a given point in time) is its organization or structure of significant components interacting through interfaces, those components being composed of successively smaller components and interfaces."

Rational Unified Process definition, working off the IEEE definition

http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf



software architecture?

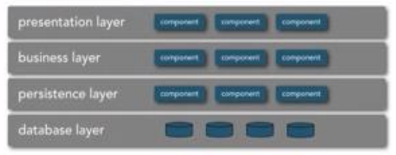
Architecture is the highest level concept of the expert developers.

"In most successful software projects, the expert developers working on that project have a shared understanding of the system design. This shared understanding is called 'architecture.' This understanding includes how the system is divided into components and how the components interact through interfaces. These components are usually composed of smaller components, but the architecture only includes the components and interfaces that are understood by all the developers."

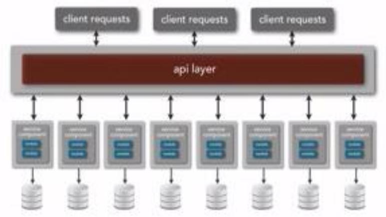
http://martinfowler.com/ieeeSoftware/whoNeedsArchitect.pdf

Architecture Characteristics

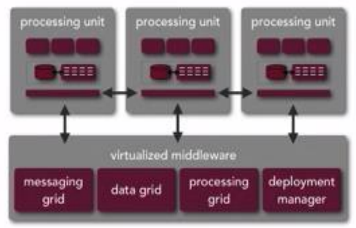
feasibility →



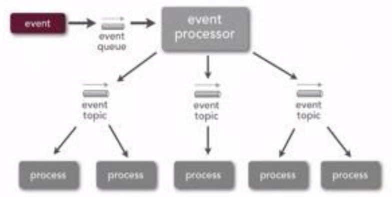
agility →



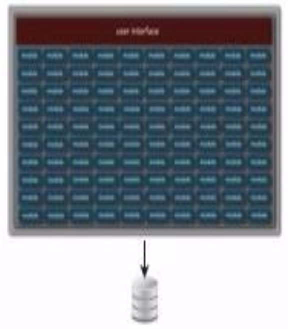
elasticity →

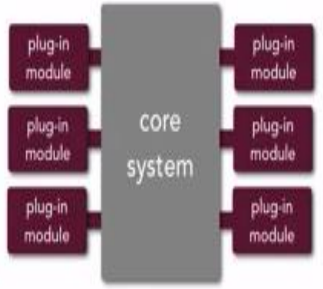


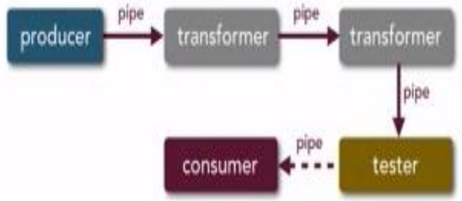
scalability →

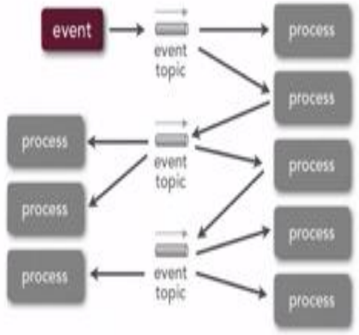


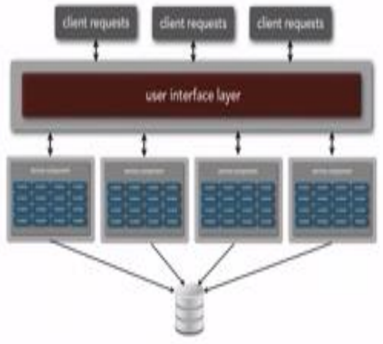
Architecture Patterns

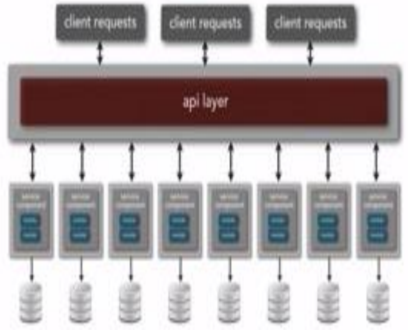


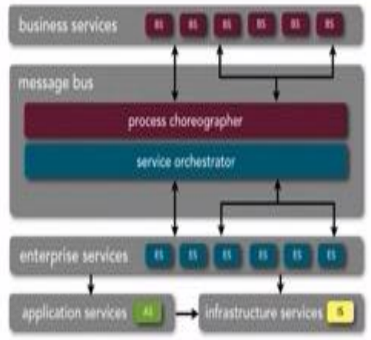


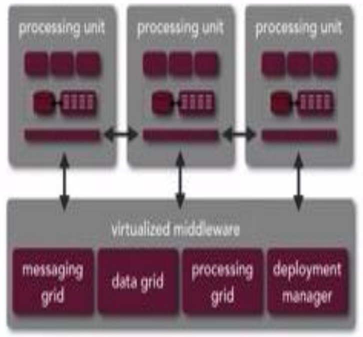












Analyzing Architecture Tradeoff

We Need Lighting-fast response time to keep up with the backlog of the calls -	Performance
Over time we are expecting the entire company to use this system -	Scalability
We are planning to acquire several businesses in next 5 years -	Extensibility , Agility , Maintainability
The budget and time frame are very tight in the project -	Feasibility

Architecture Tradeoff

ATAM – Architecture Tradeoff Analysis Method

- Proposed Architecture
- Business Drivers
- Quality Attributed

CBAM – Cost Benefit Analysis Method

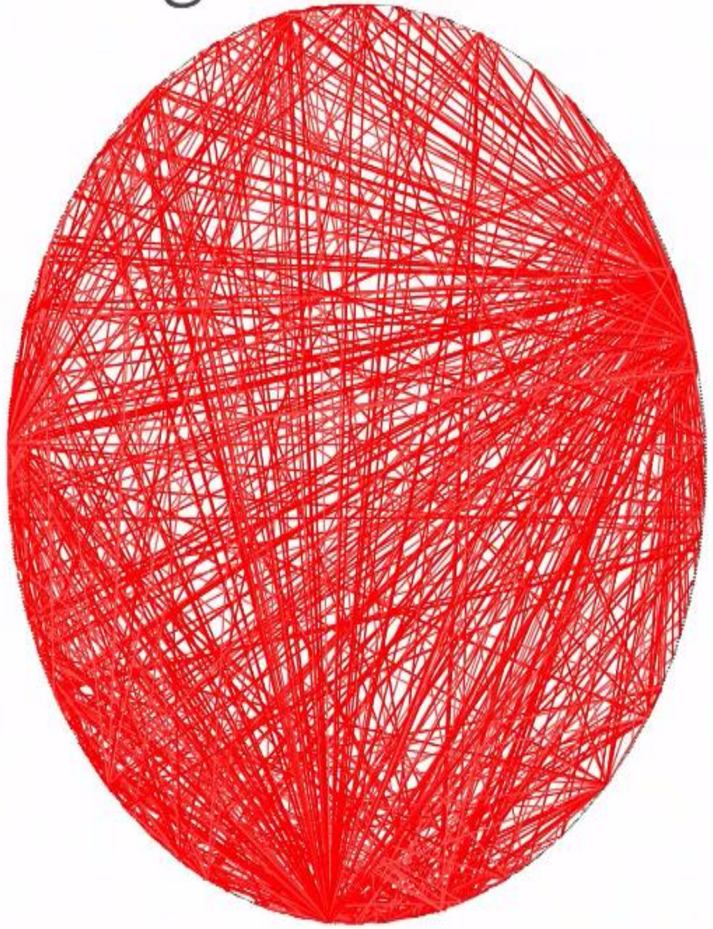
- Business Goals – Performance , Availability , Scalability
- Max the difference b/w cost and benefit

Architecture Patterns

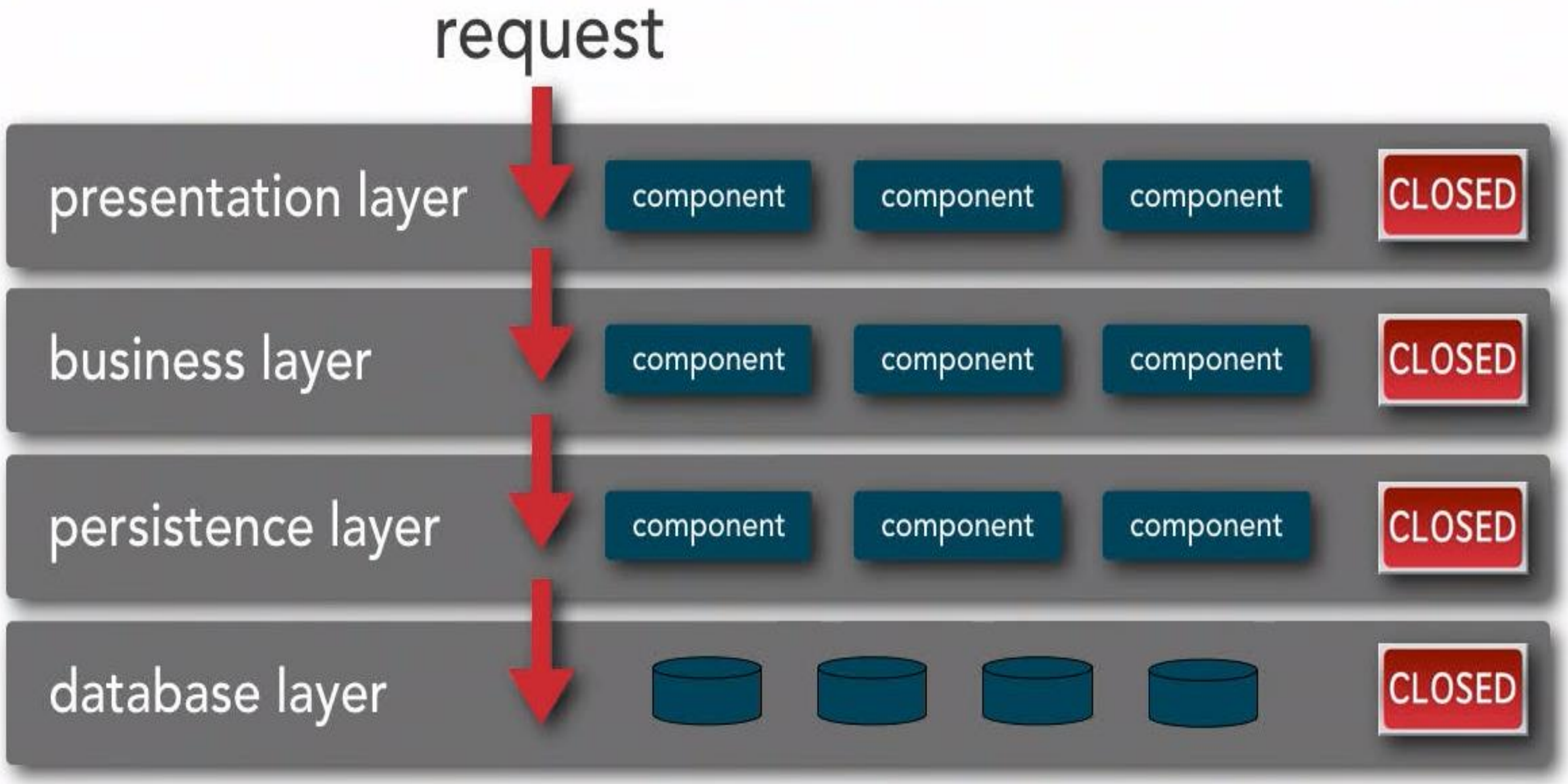
big ball of mud

ttps://en.wikipedia.org/wiki/Big_ball_of_mud

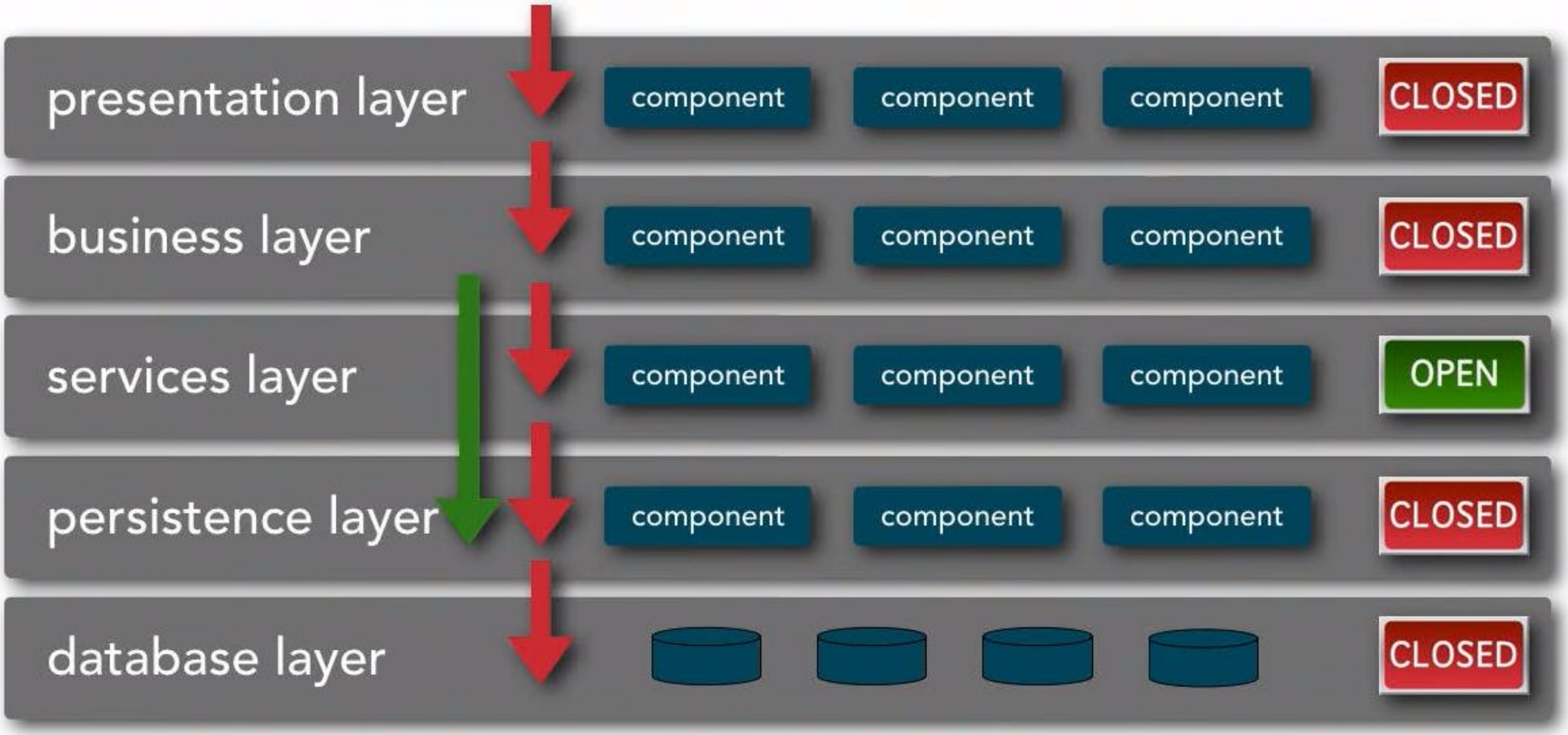
big ball of mud



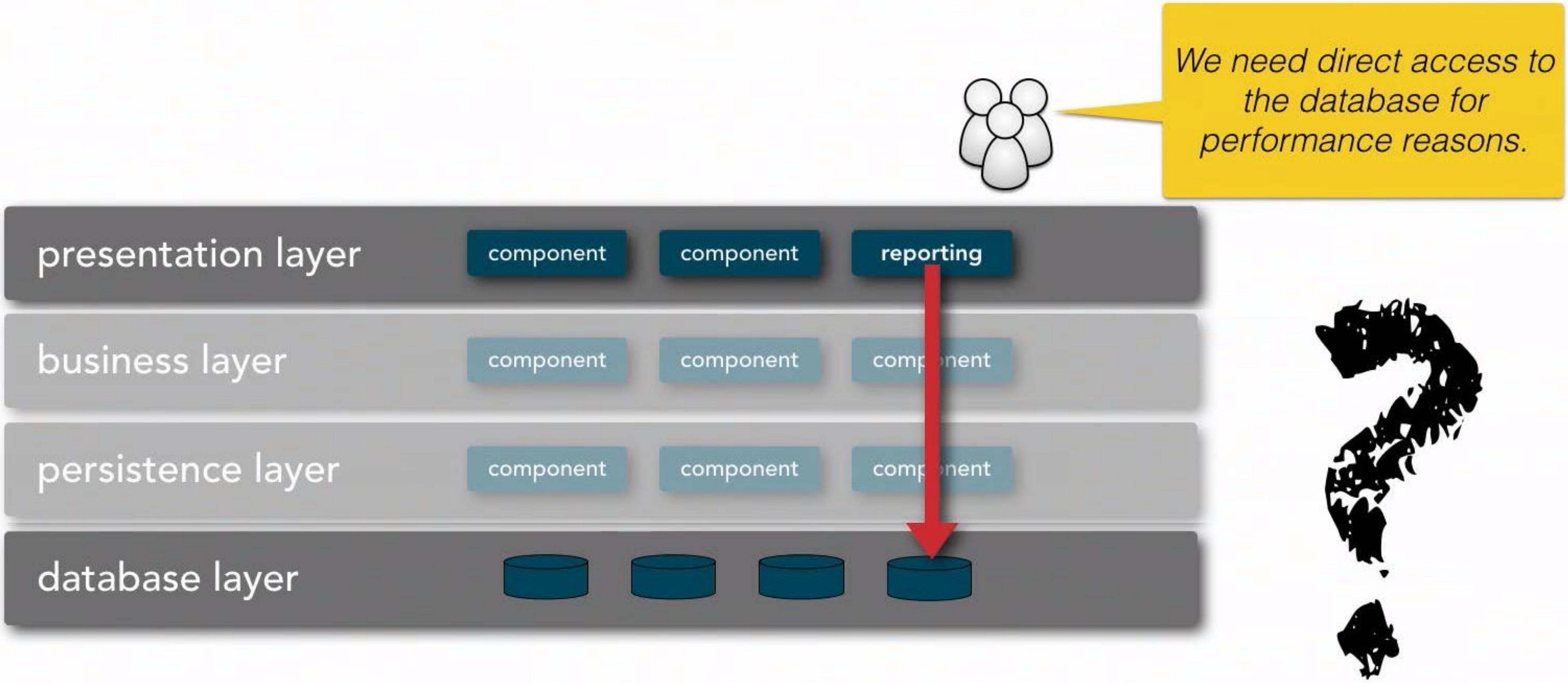
Layered Architecture -Monolith's



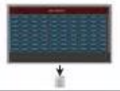
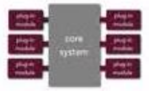
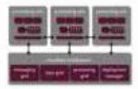
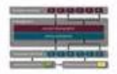
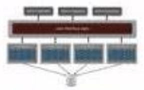
Layered Architecture – Hybrids and Variants



Pattern Governance

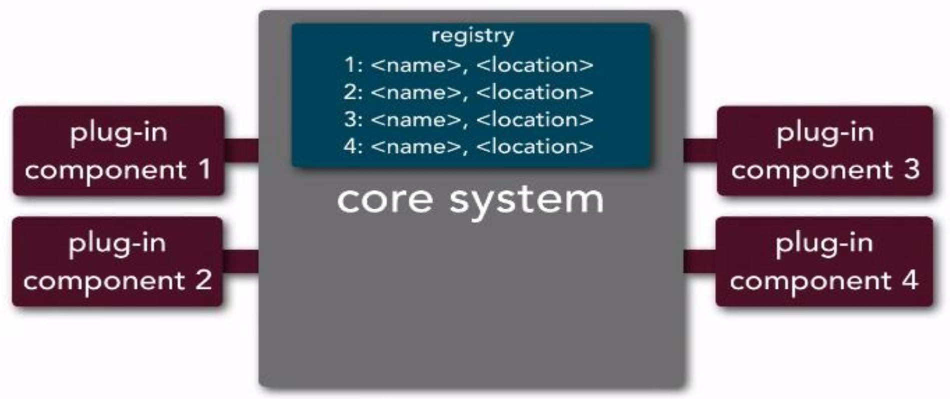


Layered Architecture

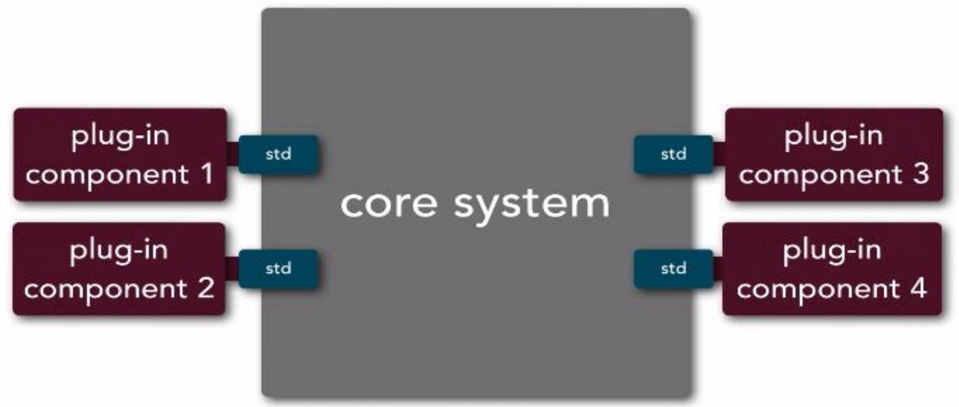
	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👍	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$

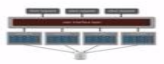
Microkernel Architecture – Modular Monolithic

registry

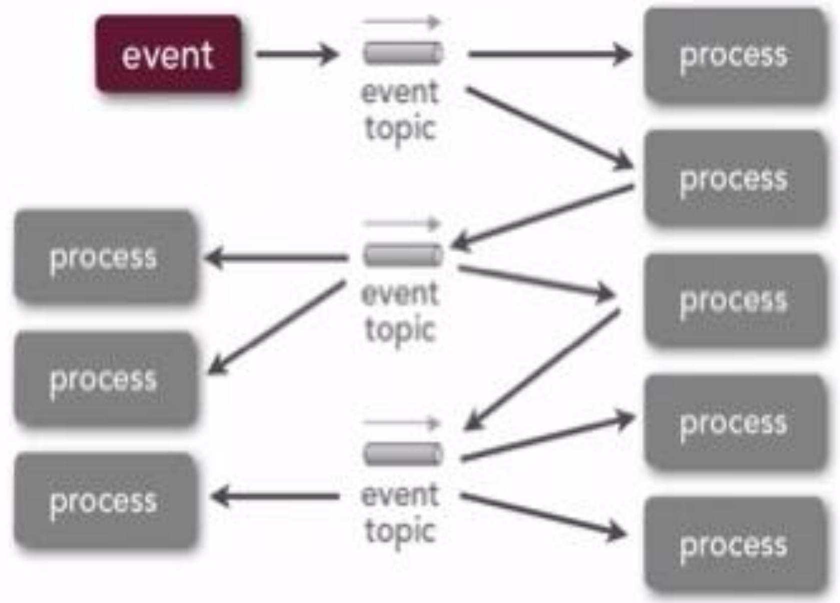


plug-in contracts

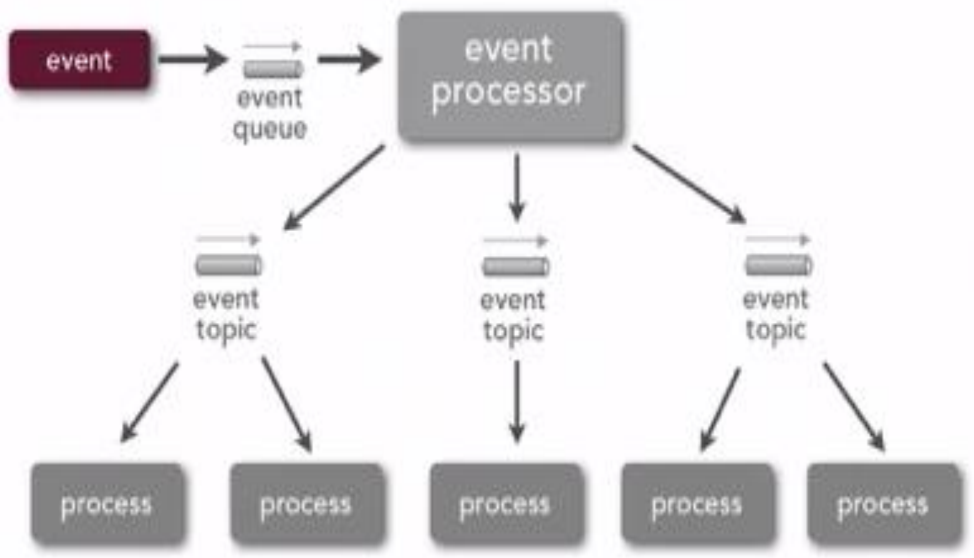


	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👍	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$

Event Driven– Distributed Architecture

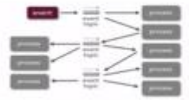
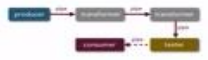
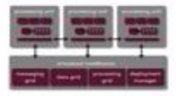
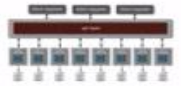
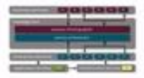
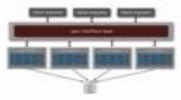


broker topology

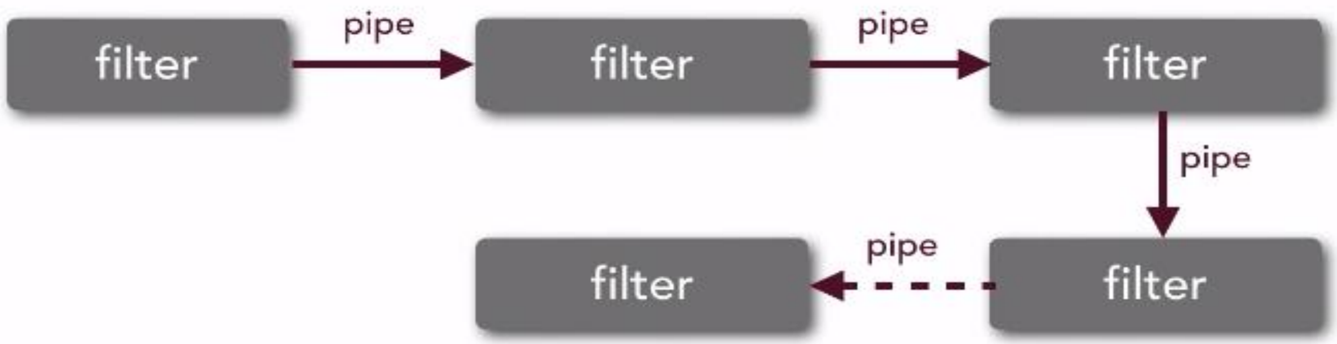


mediator topology

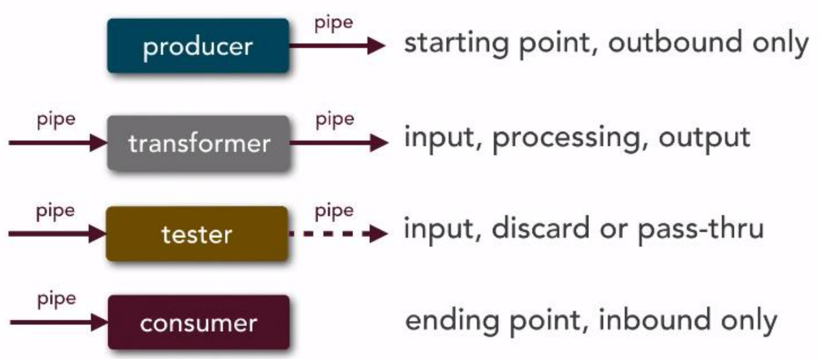
Event Driven Architecture

	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👍	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$

Pipeline Driven– Pipe and Filter

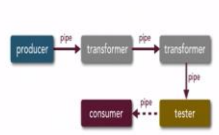


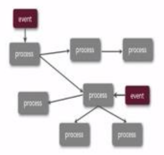
filters



producer	pipe →	starting point, outbound only
transformer	pipe → pipe →	input, processing, output
tester	pipe → pipe ⇢	input, discard or pass-thru
consumer	pipe →	ending point, inbound only

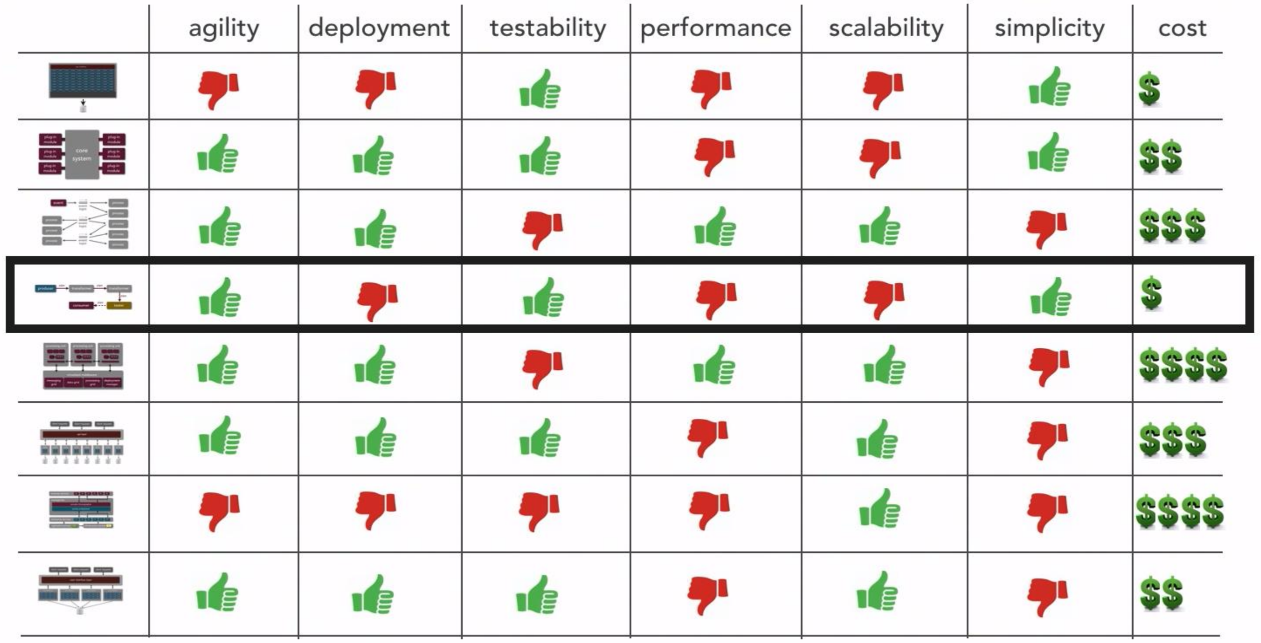
pipeline vs. event-driven





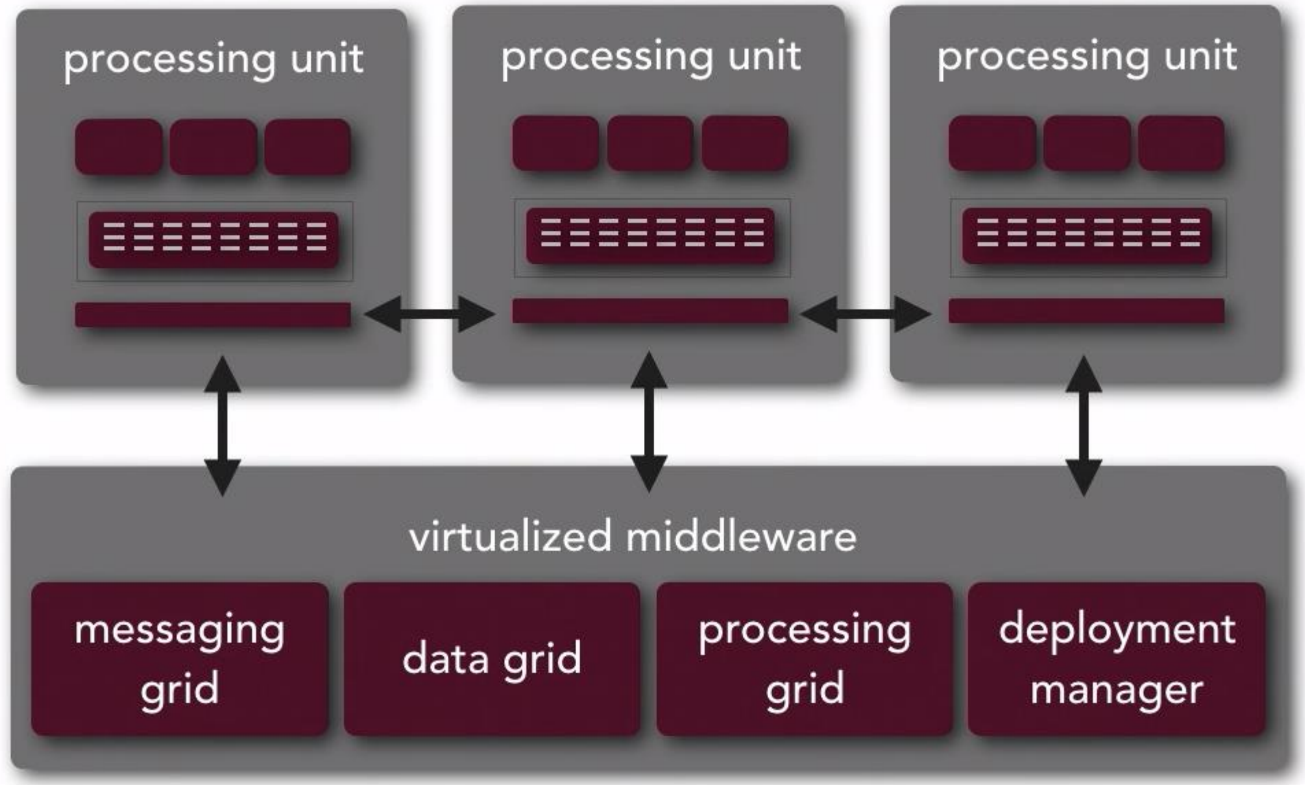
synchronous data filtering	asynchronous event processing
always unidirectional	can be request/reply
simple single purpose filters	complex multi-purpose processors
monolithic architecture	distributed architecture

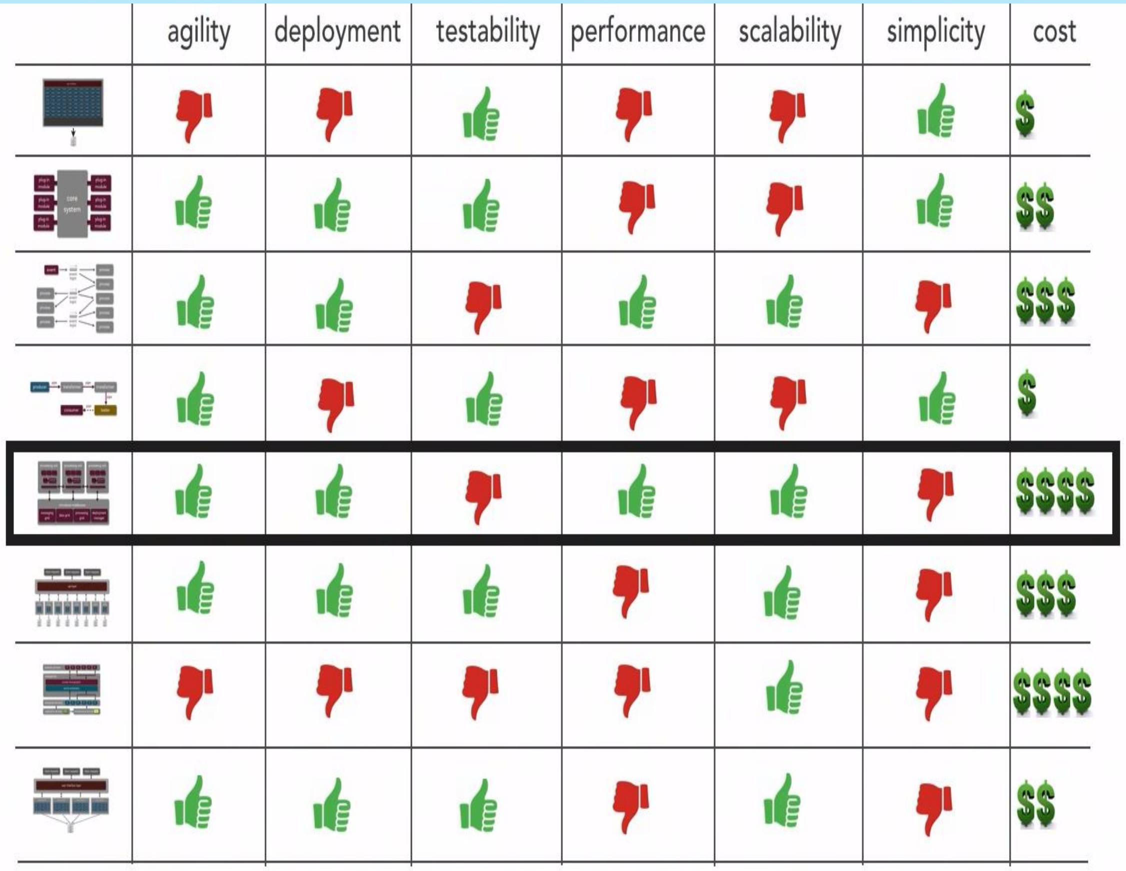
pipeline architecture



	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👍	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$

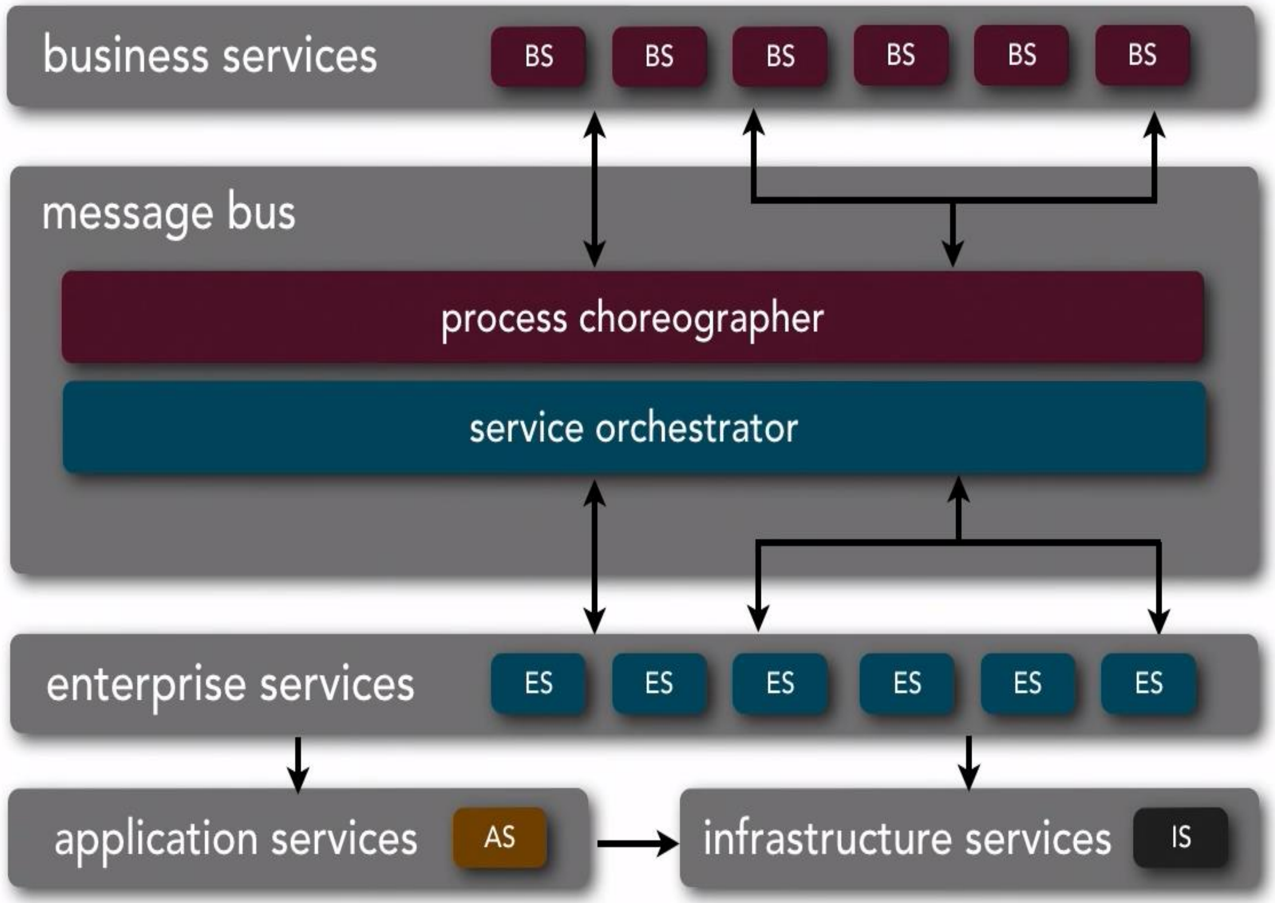
Space-Based Architecture

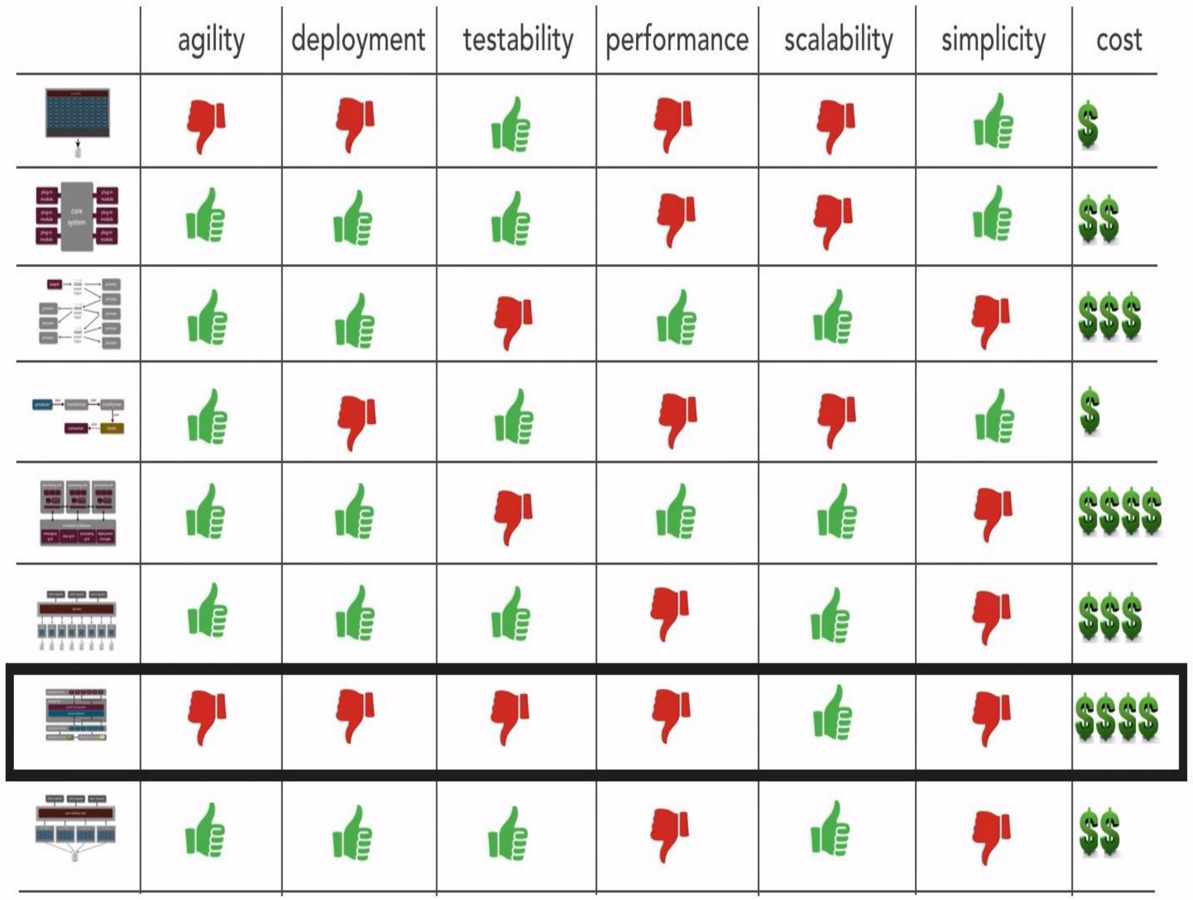




	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👍	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$

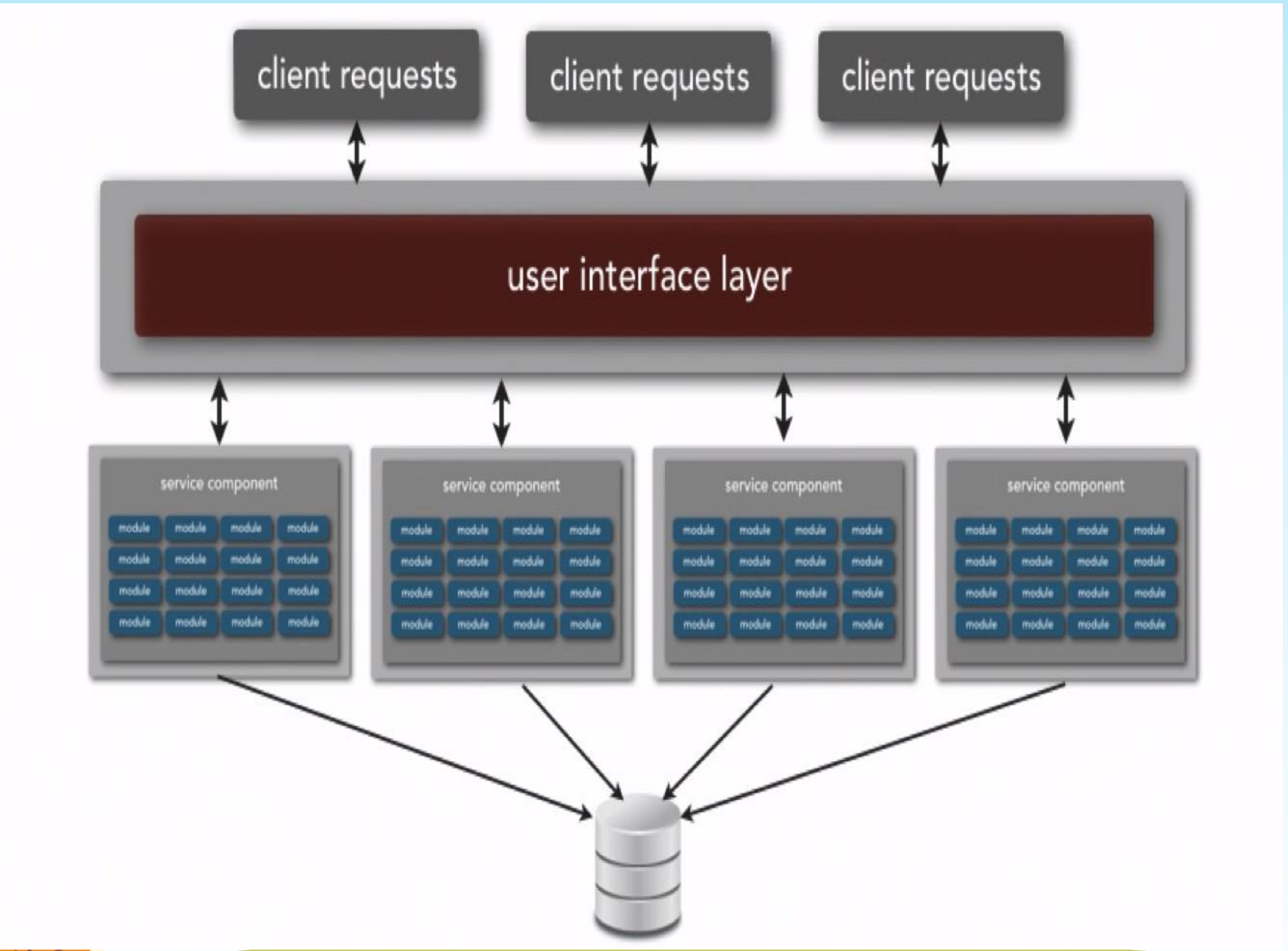
Service Oriented Architecture





	agility	deployment	testability	performance	scalability	simplicity	cost
	👎	👎	👍	👎	👎	👍	$
	👍	👍	👍	👎	👎	👍	$$
	👍	👍	👎	👍	👍	👎	$$$
	👍	👎	👍	👎	👎	👍	$
	👍	👍	👎	👍	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$$
	👎	👎	👎	👎	👍	👎	$$$$
	👍	👍	👍	👎	👍	👎	$$

Service Based Architecture



Reference :

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5177

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=513476

Software Architecture Fundamentals
- Neal Ford
- Mark Richards

https://martinfowler.com/articles/serverless.html