Software Architecture by Example

Mark Richards
Independent Consultant
Hands-on Software Architect, Published Author
Founder, DeveloperToArchitect.com
http://www.wmrichards.com
@markrichardssa

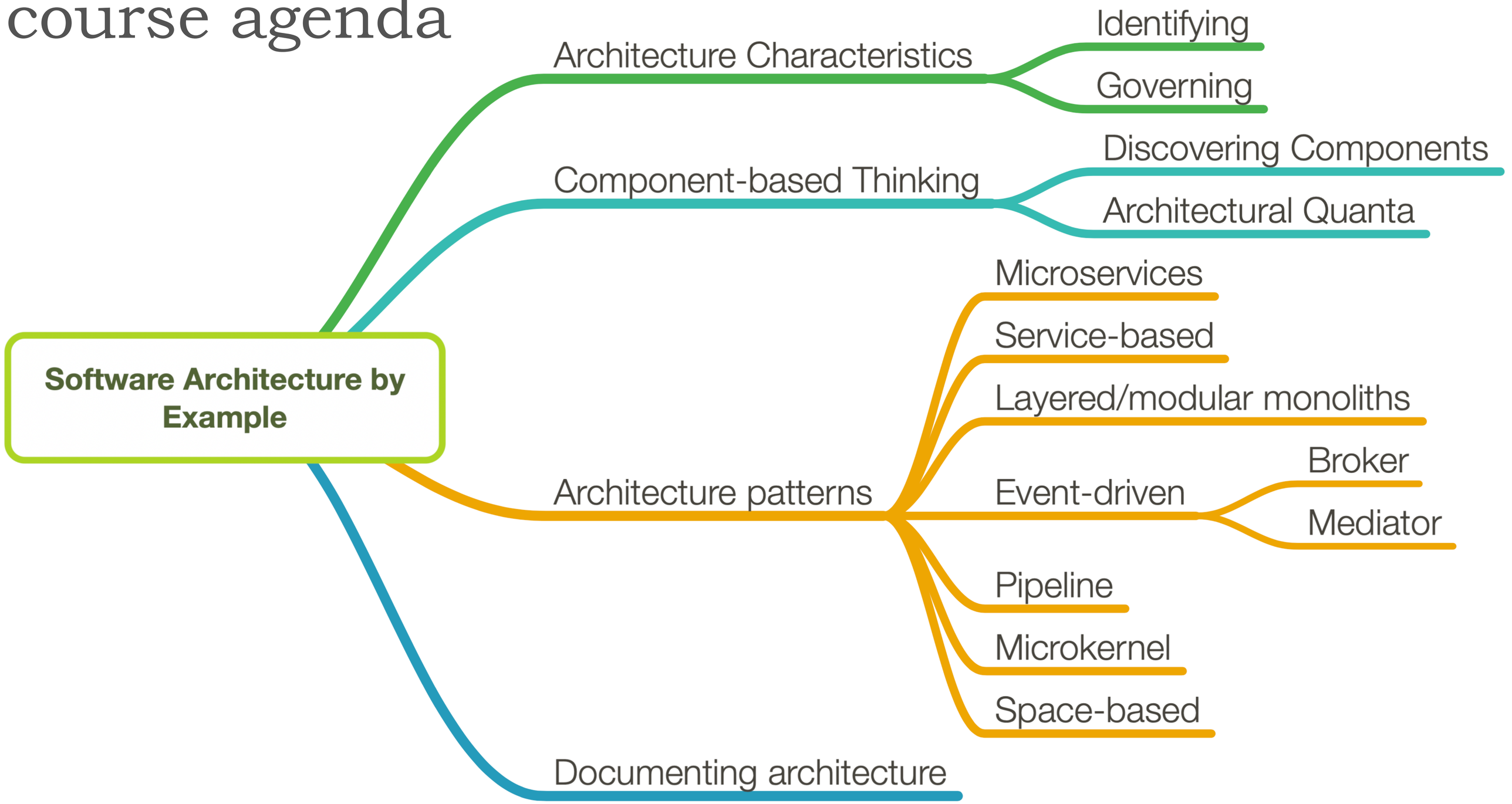
Neal Ford
ThoughtWorks
Director / Software Architect / Meme Wrangler
http://www.nealford.com
@neal4d

O'REILLY®

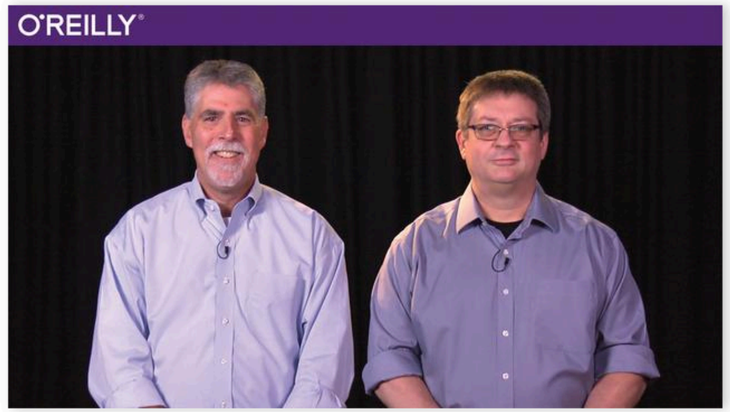
v2020-06-23

introduction

course agenda



course references



Software Architecture Fundamentals, Second Edition

★★★★★ 9 reviews

by Mark Richards, Neal Ford

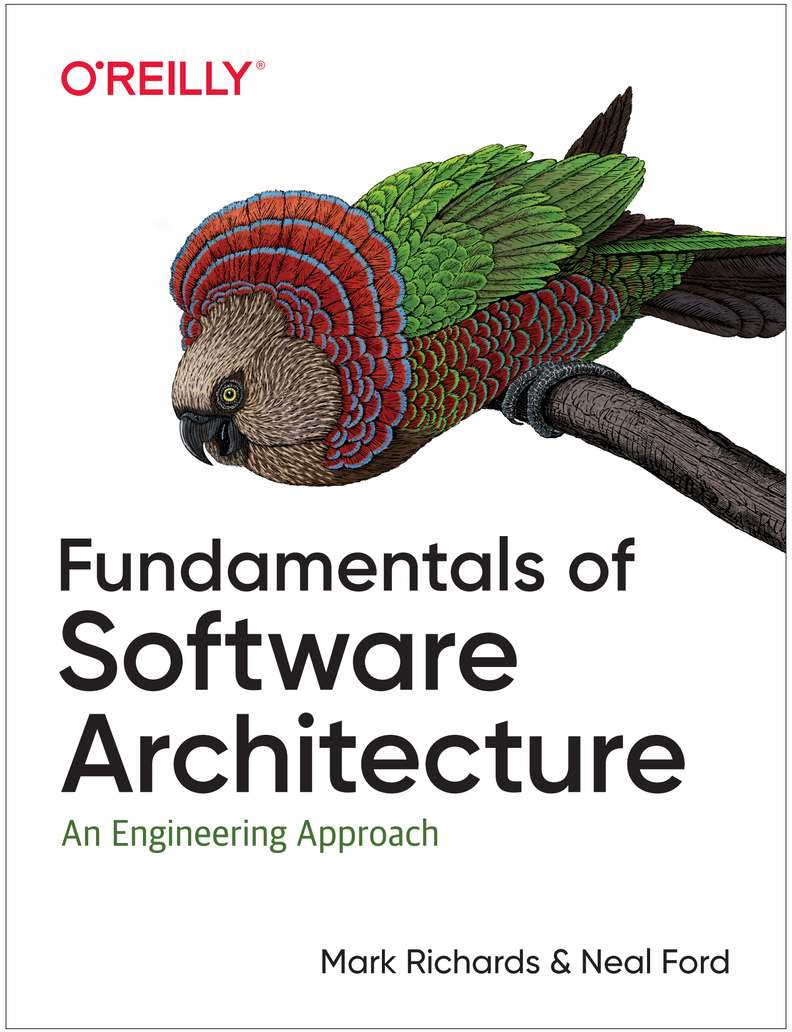
Publisher: O'Reilly Media, Inc.

Release Date: November 2017

ISBN: 9781491998991

https://learning.oreilly.com/library/view/software-architecture-fundamentals/9781491998991/

course references



https://learning.oreilly.com/library/view/fundamentals-of-software/9781492043447/

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable internet-ordering (in addition to their current call-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

First Law of Software Architecture:

Everything in software architecture is a tradeoff.

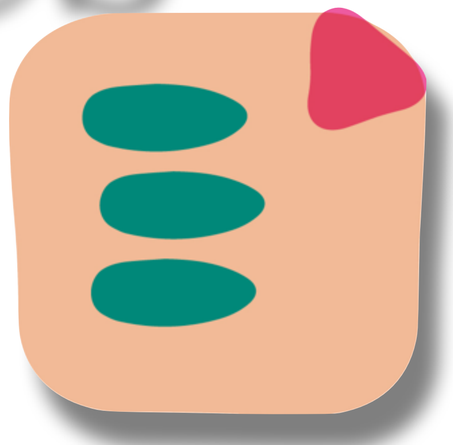
First Law of Software Architecture:

Everything in software architecture is a tradeoff.

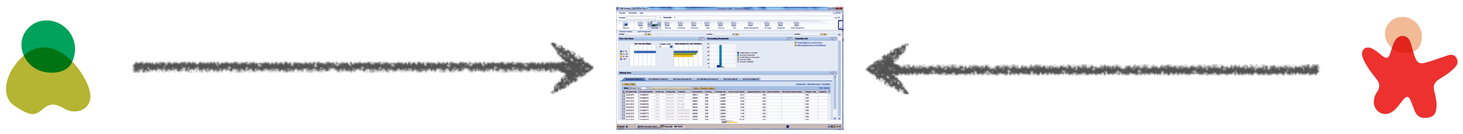
First Corollary:

If you think you've found something that *isn't* a tradeoff, it just means you haven't identified the tradeoff…yet.

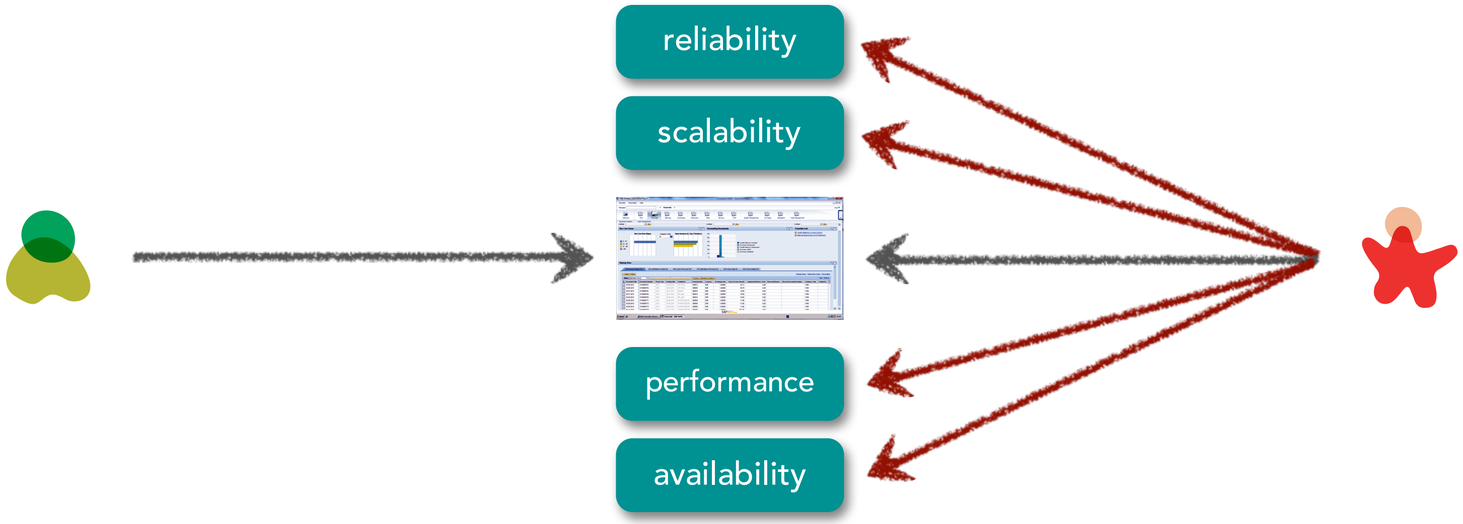
architecture characteristics



architecture characteristics



architecture characteristics

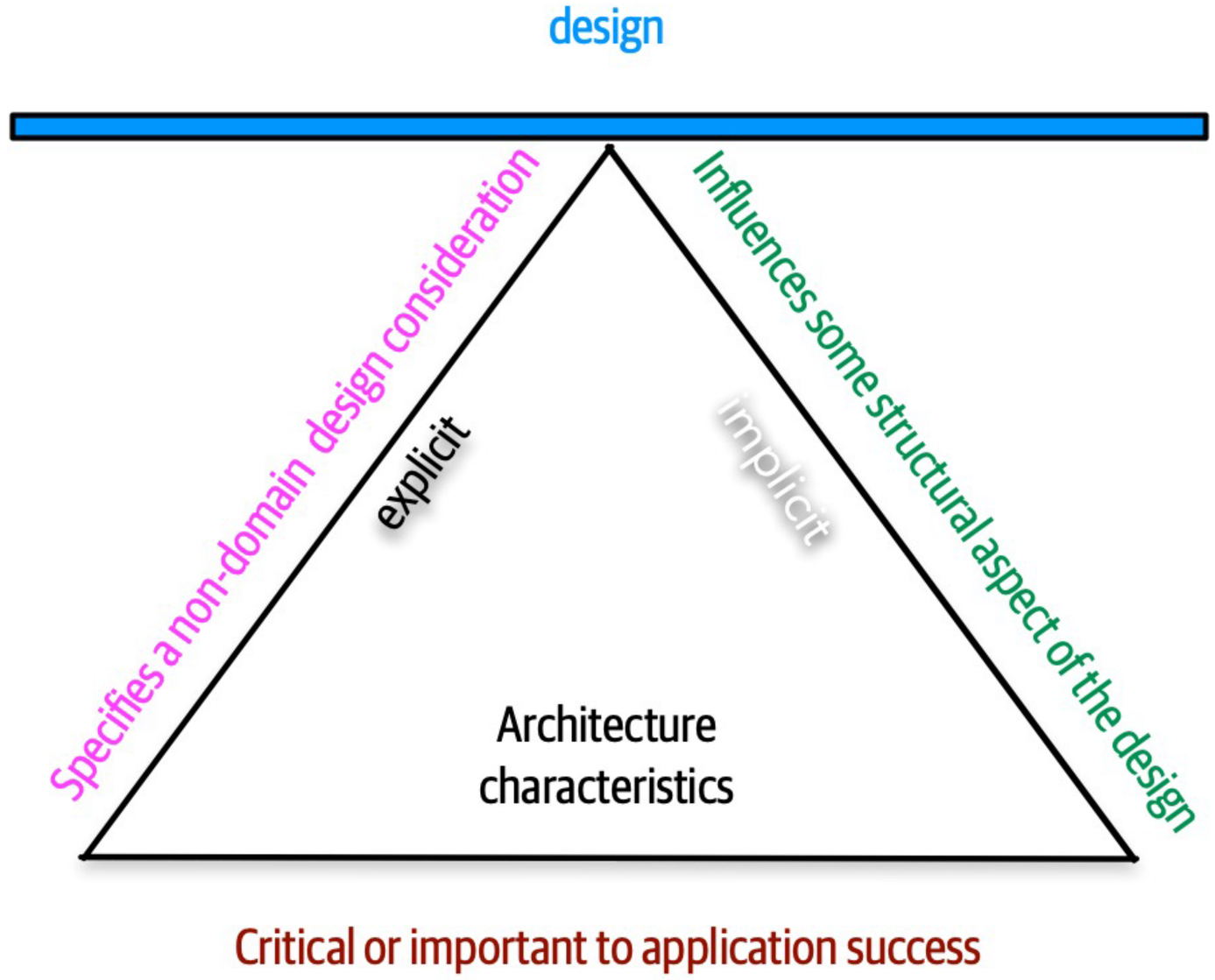


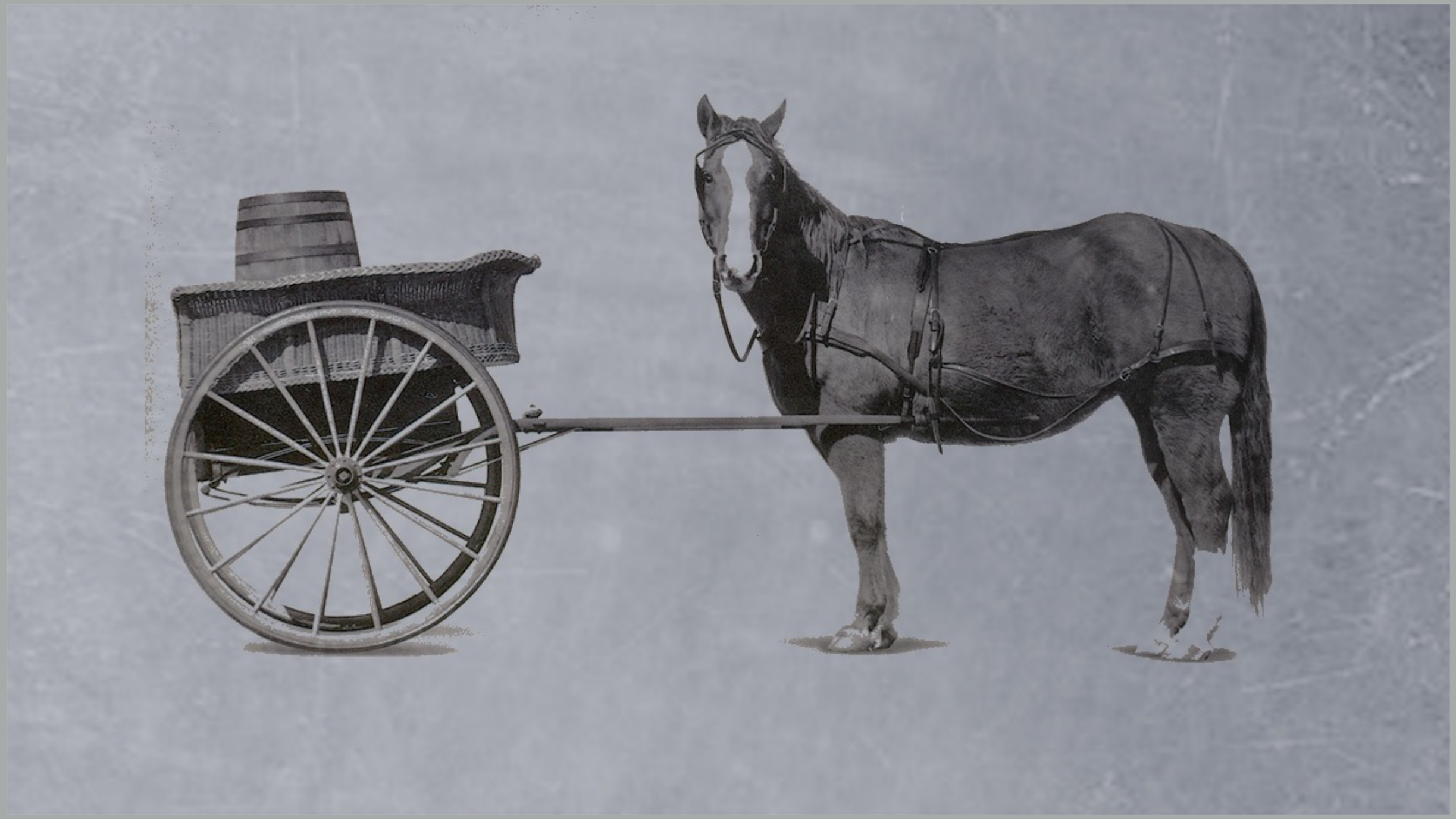
architecture characteristics

accessibility
accountability
accuracy
adaptability
administrability
affordability
agility
auditability
autonomy
availability
compatibility
composability
configurability
correctness
credibility
customizability
debugability
degradability
determinability
demonstrability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency
evolvability
extensibility
failure transparency
fault-tolerance
fidelity
flexibility
inspectability
installability
integrity
interchangeability
interoperability
learnability
maintainability
manageability
mobility
modifiability
modularity
operability
orthogonality
portability
precision
predictability
process capabilities
producibility
provability
recoverability
relevance
reliability
repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness
self-sustainability
serviceability
supportability
securability
simplicity
stability
standards compliance
survivability
sustainability
tailorability
testability
timeliness
traceability
transparency
ubiquity
understandability
upgradability
usability

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

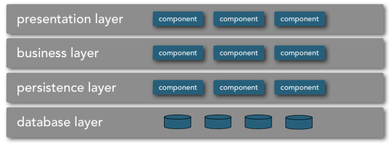
architecture characteristics



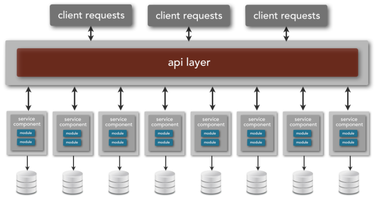


architecture characteristics

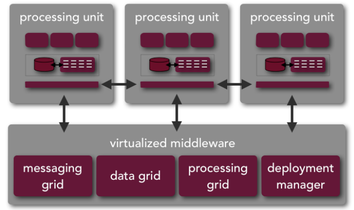
feasibility



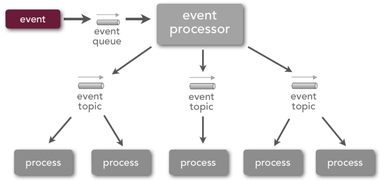
agility



elasticity



scalability



architecture katas

identifying driving characteristics

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - video stream of the action after the fact
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotionals/specials
  - offer local daily promotionals/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

?

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability

Your Architectural Kata is...

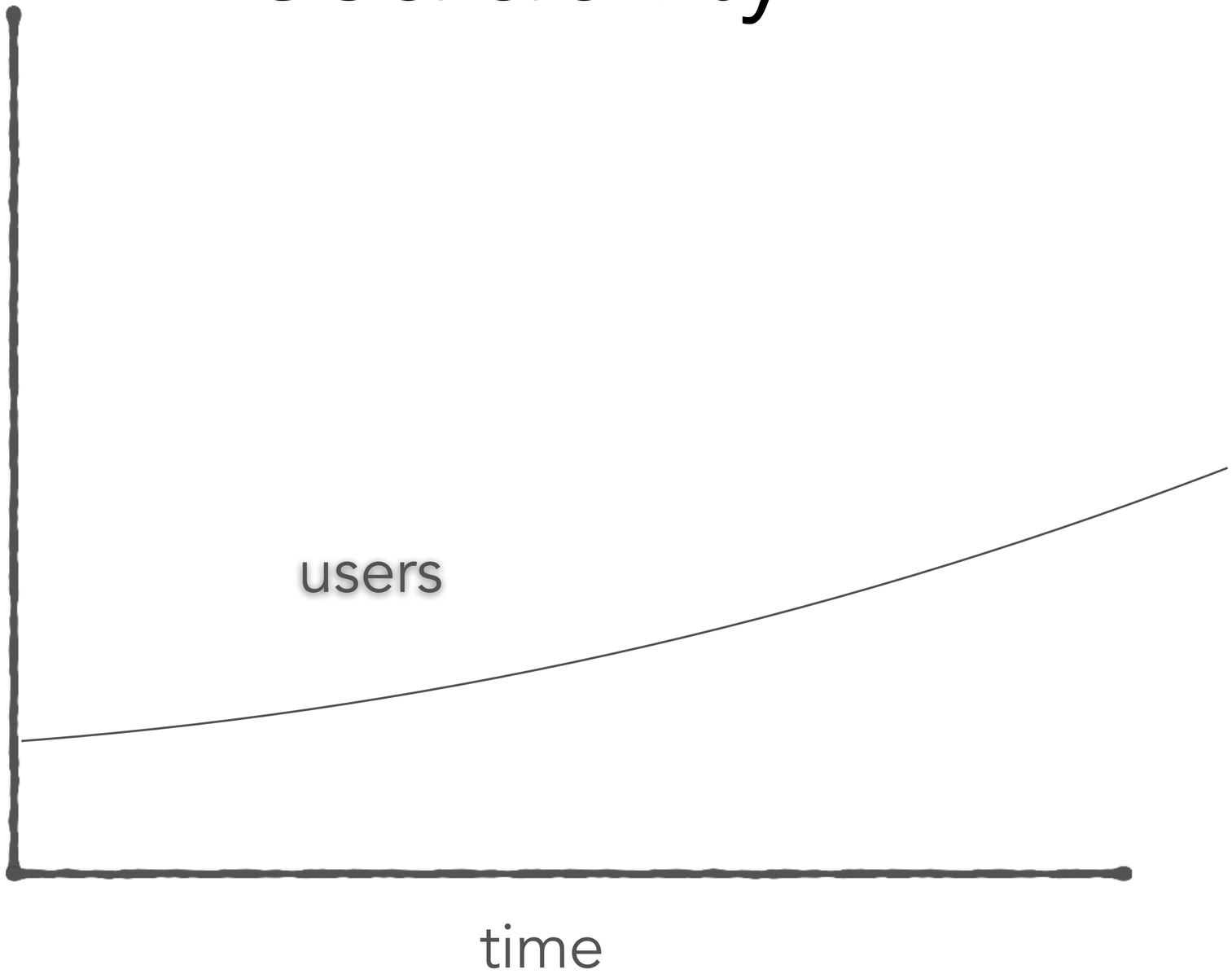
Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

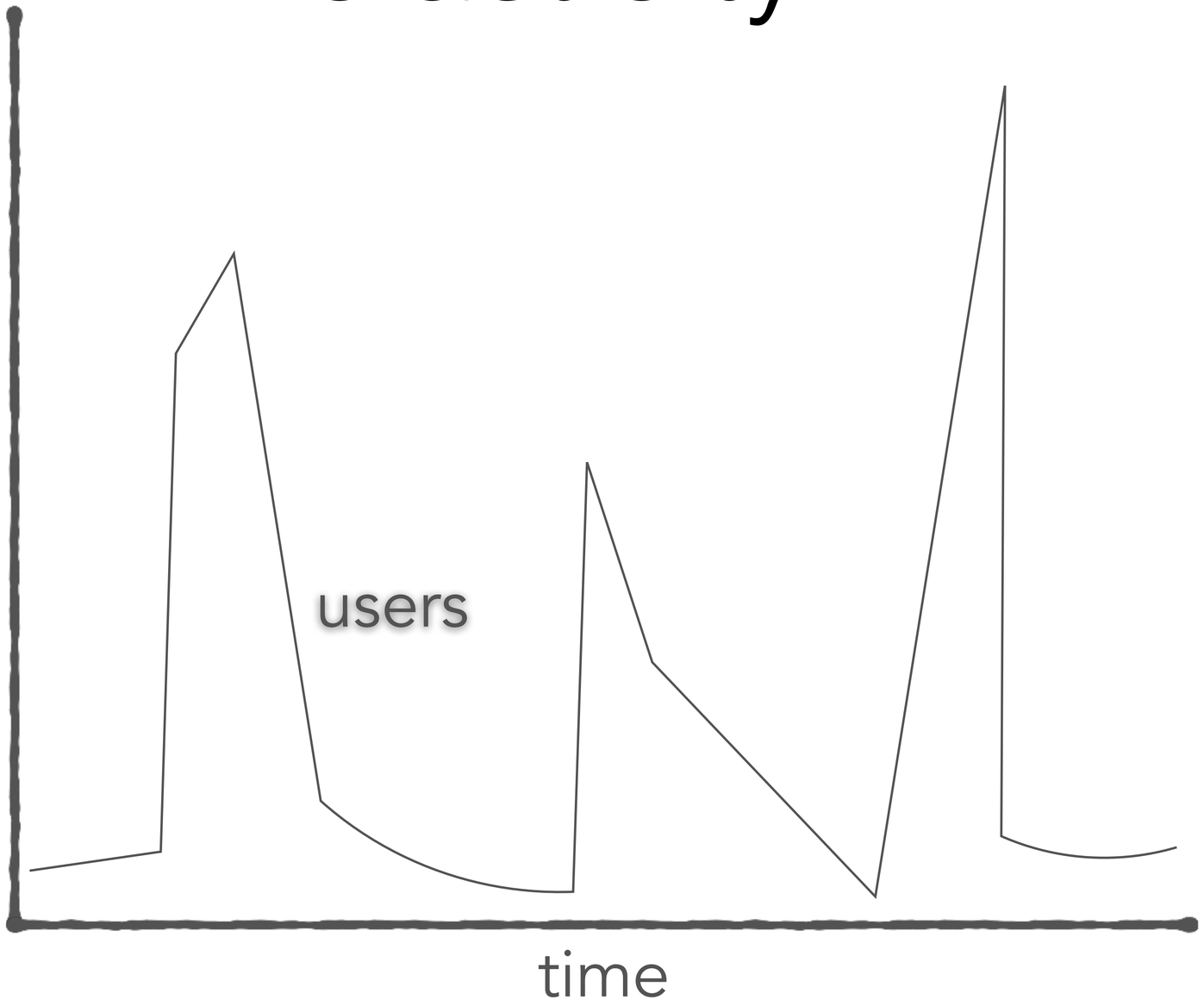
- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity

scalability:



elasticity:



Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

?

availability reliability performance scalability elasticity

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

?

availability reliability performance scalability elasticity

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity (security)

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

availability reliability performance scalability elasticity (security)

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

performance availability reliability

scalability elasticity i18n l10n

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

performance availability reliability

scalability elasticity i18n l10n

Global Customizability Local Customizability

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

performance availability reliability

scalability elasticity

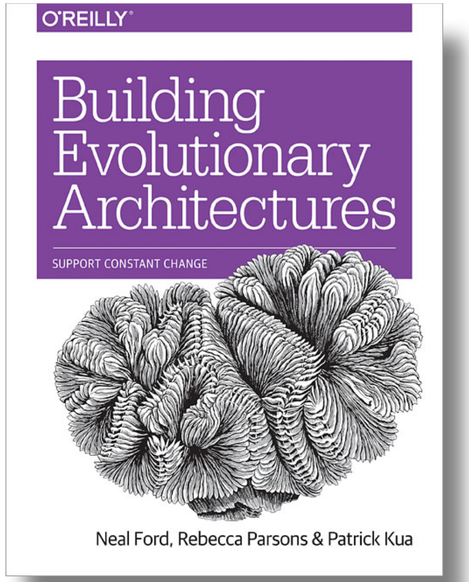
Customizability { location, sales, recipe

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.



building evolutionary architectures

Evolutionary Architecture

An evolutionary architecture supports
guided,
incremental change
across multiple dimensions.

Evolutionary Architecture

An evolutionary architecture supports
guided,
incremental change
across multiple dimensions.

evolutionary computing fitness functions

An objective function used to summarize…how close a given design solution achieves the set goals.

guided

architectural *fitness function*:

An architectural fitness function is any mechanism that provides an objective integrity assessment of some architectural characteristic(s).

guided

***architectural** fitness function*:

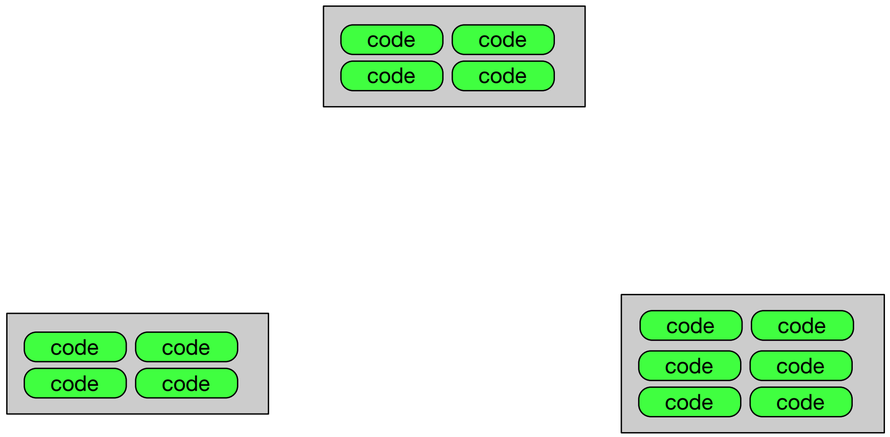
An architectural fitness function is ***any*** mechanism that provides an objective integrity assessment of some architectural characteristic(s).

guided

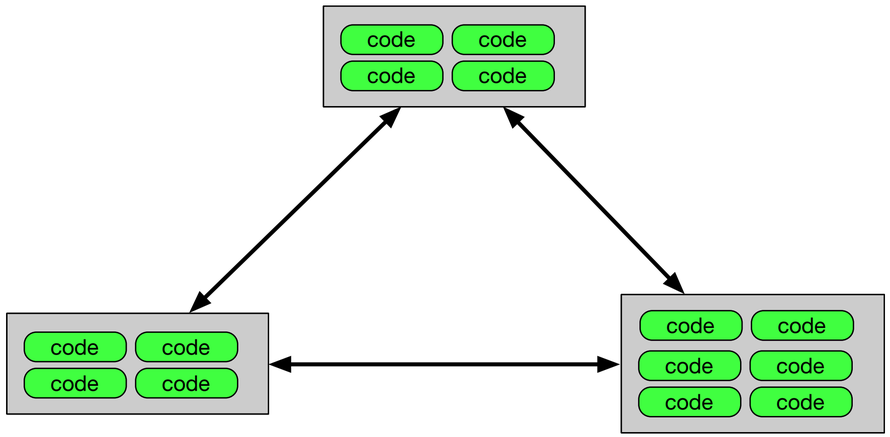
***architectural** fitness function*:

An architectural fitness function is ***any*** mechanism that provides an ***objective*** integrity assessment of some architectural characteristic(s).

cyclic dependencies



cyclic dependencies




```
/**
 * Tests that a package dependency cycle does not
 * exist for any of the analyzed packages.
 */
public void testAllPackages() {

    Collection packages = jdepend.analyze();

    assertEquals("Cycles exist",
                 false, jdepend.containsCycles());
}
```

clarkware.com/software/JDepend.html

maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

immutability

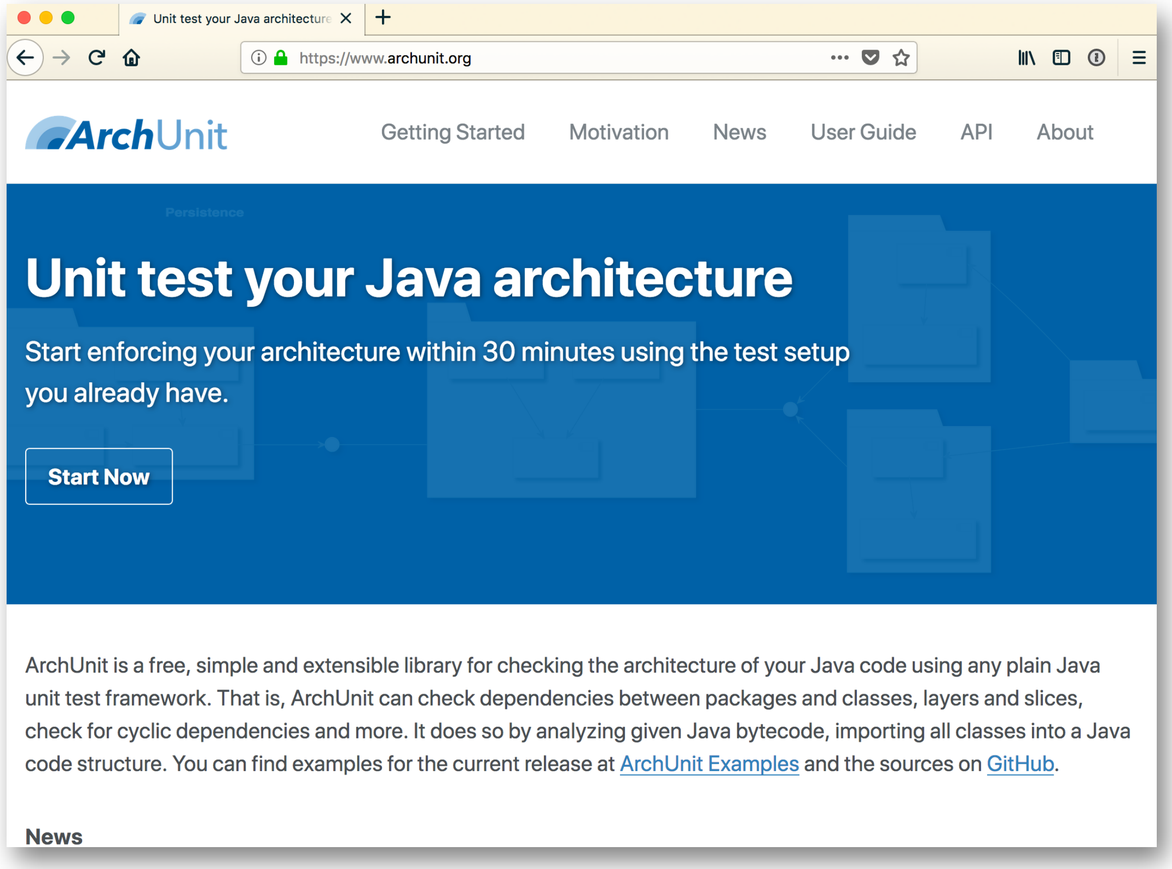
maintainable?

limited inheritance

Controlled afferent/efferent coupling

ArchUnit

https://www.archunit.org/



ArchUnit

https://www.archunit.org/

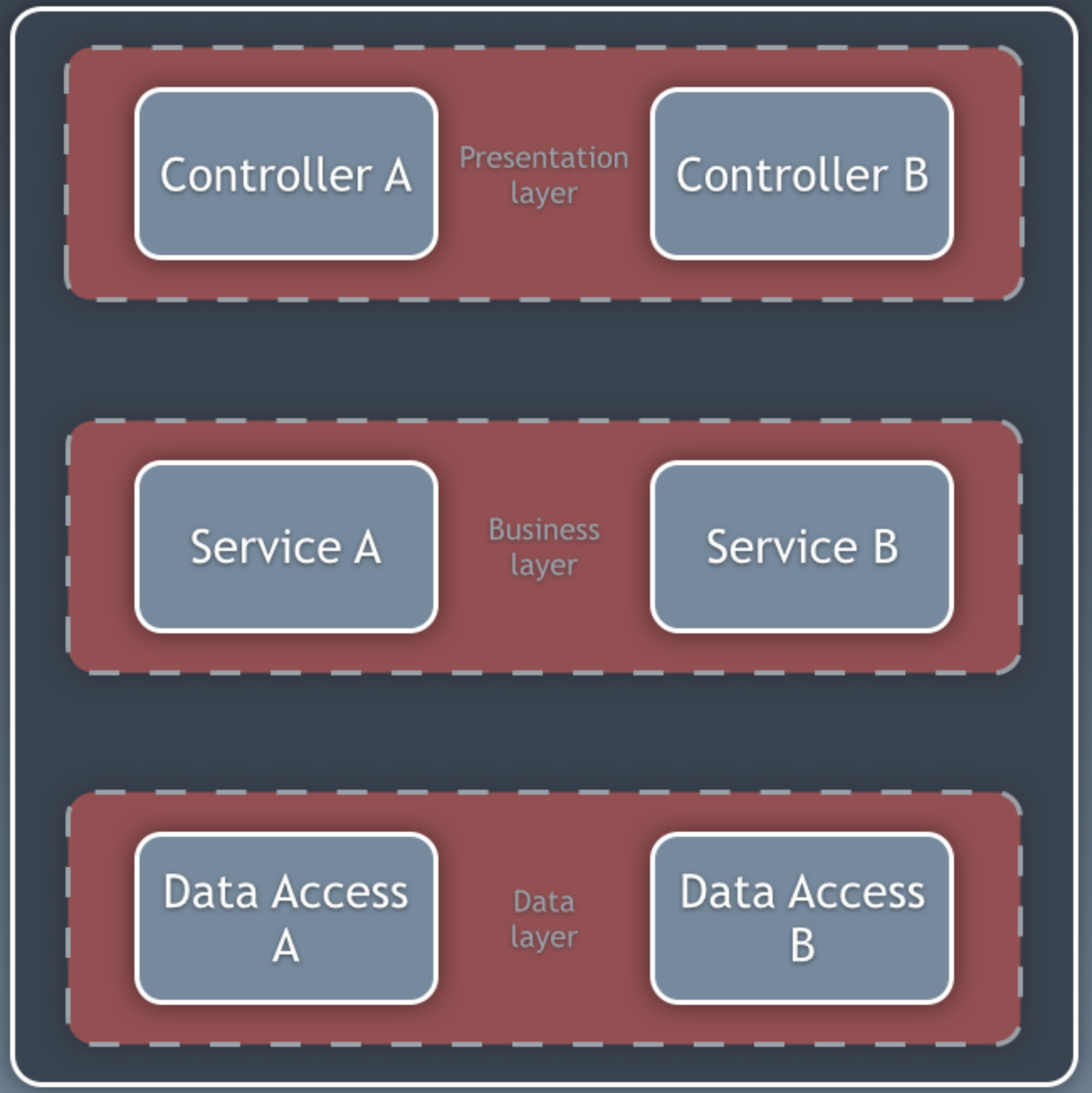
coding rules

```java
import static com.tngtech.archunit.lang.syntax.ArchRuleDefinition.noClasses;
import static com.tngtech.archunit.library.GeneralCodingRules.ACCESS_STANDARD_STREAMS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS;
import static com.tngtech.archunit.library.GeneralCodingRules.NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING;

public class CodingRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void classes_should_not_access_standard_streams_defined_by_hand() {
        noClasses().should(ACCESS_STANDARD_STREAMS).check(classes);
    }

    @Test
    public void classes_should_not_access_standard_streams_from_library() {
        NO_CLASSES_SHOULD_ACCESS_STANDARD_STREAMS.check(classes);
    }

    @Test
    public void classes_should_not_throw_generic_exceptions() {
        NO_CLASSES_SHOULD_THROW_GENERIC_EXCEPTIONS.check(classes);
    }

    @Test
    public void classes_should_not_use_java_util_logging() {
        NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING.check(classes);
    }
}
```

ArchUnit

```java
public class InterfaceRules {

    @Test
    public void interfaces_should_not_have_names_ending_with_the_word_interface() {
        JavaClasses classes = new ClassFileImporter().importClasses(
                SomeBusinessInterface.class,
                SomeDao.class
        );

        noClasses().that().areInterfaces().should().haveNameMatching(".*Interface").check(classes);
    }

    @Test
    public void interfaces_should_not_have_simple_class_names_ending_with_the_word_interface() {
        JavaClasses classes = new ClassFileImporter().importClasses(
                SomeBusinessInterface.class,
                SomeDao.class
        );

        noClasses().that().areInterfaces().should().haveSimpleNameContaining("Interface").check(classes);
    }

    @Test
    public void interfaces_must_not_be_placed_in_implementation_packages() {
        JavaClasses classes = new ClassFileImporter().importPackagesOf(SomeInterfacePlacedInTheWrongPackage.class);

        noClasses().that().resideInAPackage("..impl..").should().beInterfaces().check(classes);
    }
}
```

interface rules



Package by layer (horizontal slicing)

ArchUnit

```java
public class LayerDependencyRulesTest {
    private JavaClasses classes;

    @Before
    public void setUp() throws Exception {
        classes = new ClassFileImporter().importPackagesOf(ClassViolatingCodingRules.class);
    }

    @Test
    public void services_should_not_access_controllers() {
        noClasses().that().resideInAPackage("..service..")
                .should().accessClassesThat().resideInAPackage("..controller..").check(classes);
    }

    @Test
    public void persistence_should_not_access_services() {
        noClasses().that().resideInAPackage("..persistence..")
                .should().accessClassesThat().resideInAPackage("..service..").check(classes);
    }

    @Test
    public void services_should_only_be_accessed_by_controllers_or_other_services() {
        classes().that().resideInAPackage("..service..")
                .should().onlyBeAccessed().byAnyPackage("..controller..", "..service..").check(classes);
    }
}
```

layer dependency

NetArchTest

https://github.com/BenMorris/NetArchTest/

```
// Controllers should not directly reference repositories
var result = Types.InCurrentDomain()
    .That()
    .ResideInNamespace("NetArchTest.SampleLibrary.Presentation")
    .ShouldNot()
    .HaveDependencyOn("NetArchTest.SampleLibrary.Data")
    .GetResult().IsSuccessful;
```

```
// Only classes in the data namespace can have a dependency on System.Data
result = Types.InCurrentDomain()
    .That().HaveDependencyOn("System.Data")
    .And().ResideInNamespace(("ArchTest"))
    .Should().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .GetResult().IsSuccessful;
```

NetArchTest

https://github.com/BenMorris/NetArchTest/

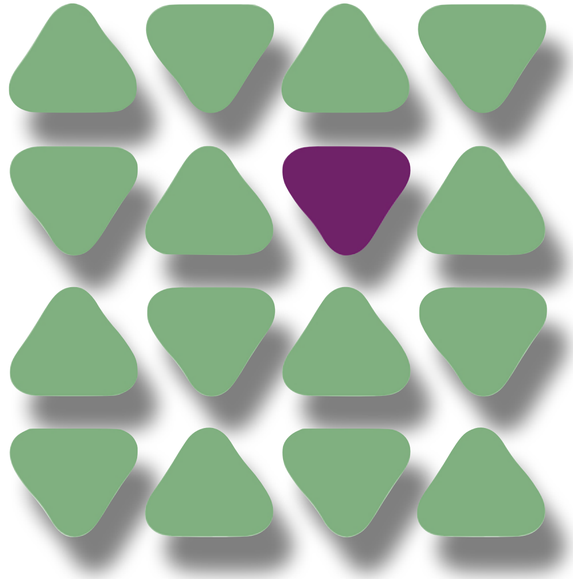
```
// Controllers should not directly reference repositories
var result = Types.InCurrentDomain()
    .That()
    .ResideInNamespace("NetArchTest.SampleLibrary.Presentation")
    .ShouldNot()
    .HaveDependencyOn("NetArchTest.SampleLibrary.Data")
    .GetResult().IsSuccessful;
```

NetArchTest

https://github.com/BenMorris/NetArchTest/

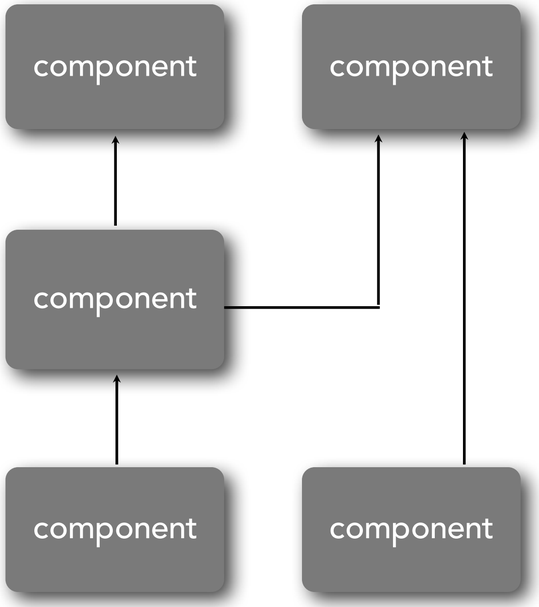
```
// All the classes in the data namespace should implement IRepository
result = Types.InCurrentDomain()
    .That().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .And().AreClasses()
    .Should().ImplementInterface(typeof(IRepository<>))
    .GetResult().IsSuccessful;

// Classes that implement IRepository should have the suffix "Repository"
result = Types.InCurrentDomain()
    .That().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .And().AreClasses()
    .Should().HaveNameEndingWith("Repository")
    .GetResult().IsSuccessful;

// Classes that implement IRepository must reside in the Data namespace
result = Types.InCurrentDomain()
    .That().ImplementInterface(typeof(IRepository<>))
    .Should().ResideInNamespace(("NetArchTest.SampleLibrary.Data"))
    .GetResult().IsSuccessful;

// All the service classes should be sealed
result = Types.InCurrentDomain()
    .That().ImplementInterface(typeof(IWidgetService))
    .Should().BeSealed()
    .GetResult().IsSuccessful;
```

component-based thinking



component identification

as an architect, you should think about the artifacts within the architecture in terms of *components*



component:

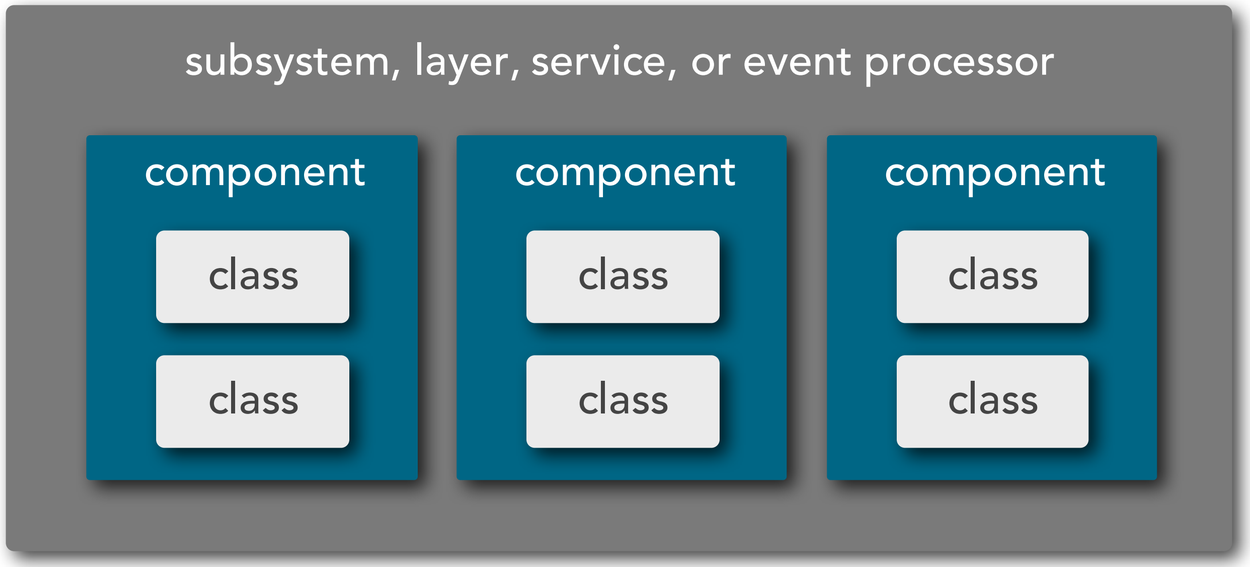
building block of the application

well defined set of operations

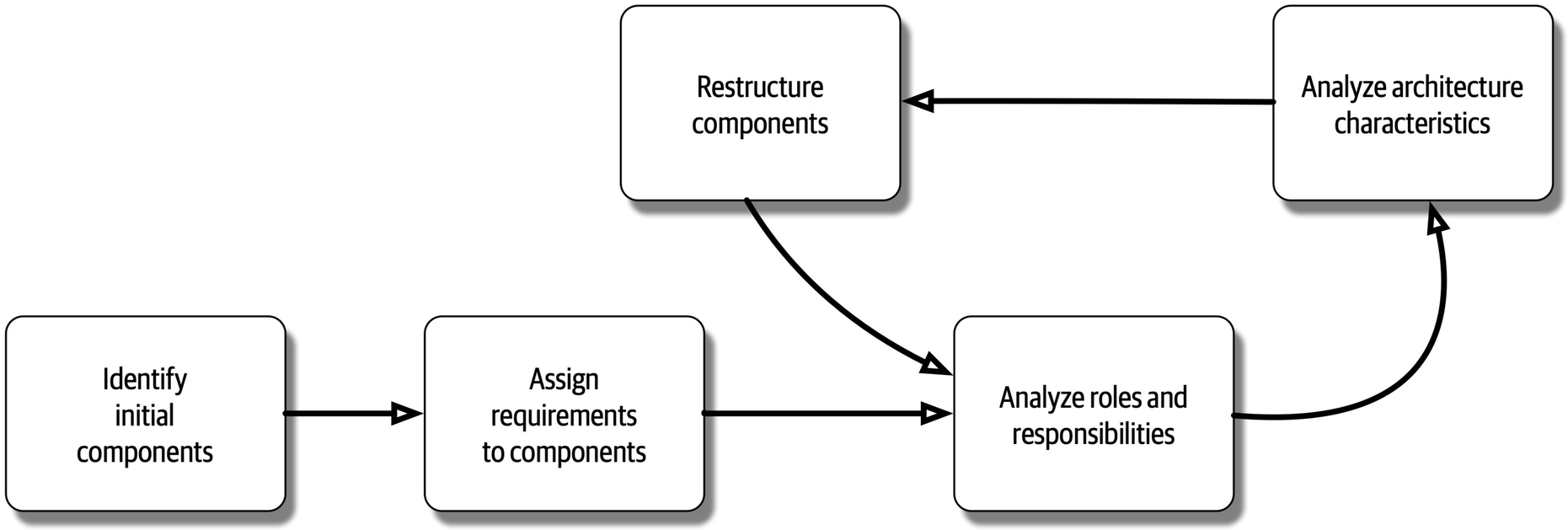
well defined role and responsibility

component identification

component scope

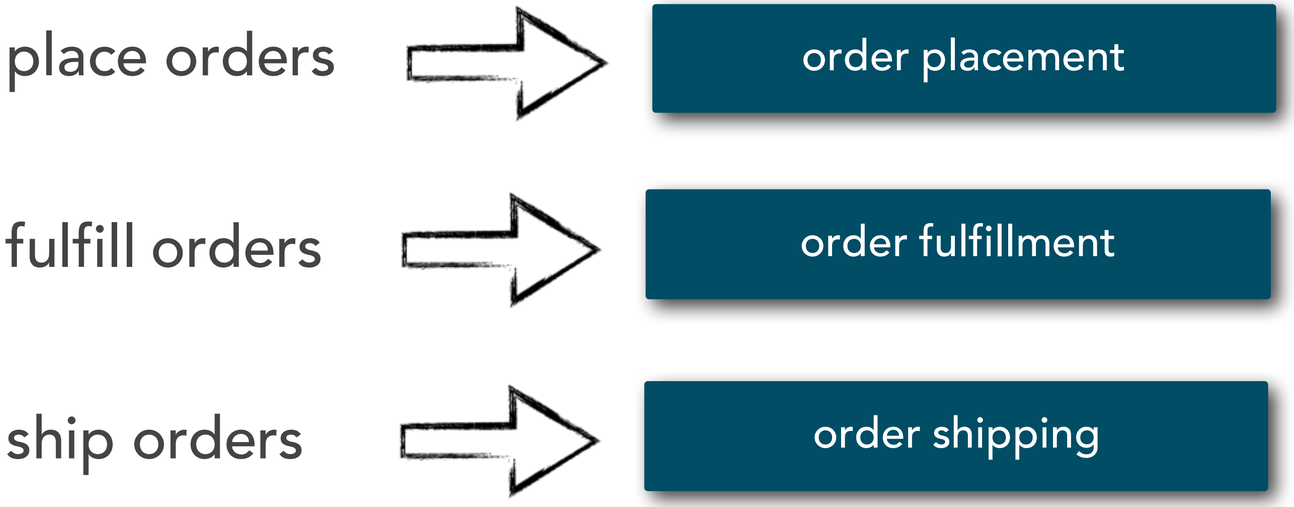


component identification



component identification

identify initial components using core functionality



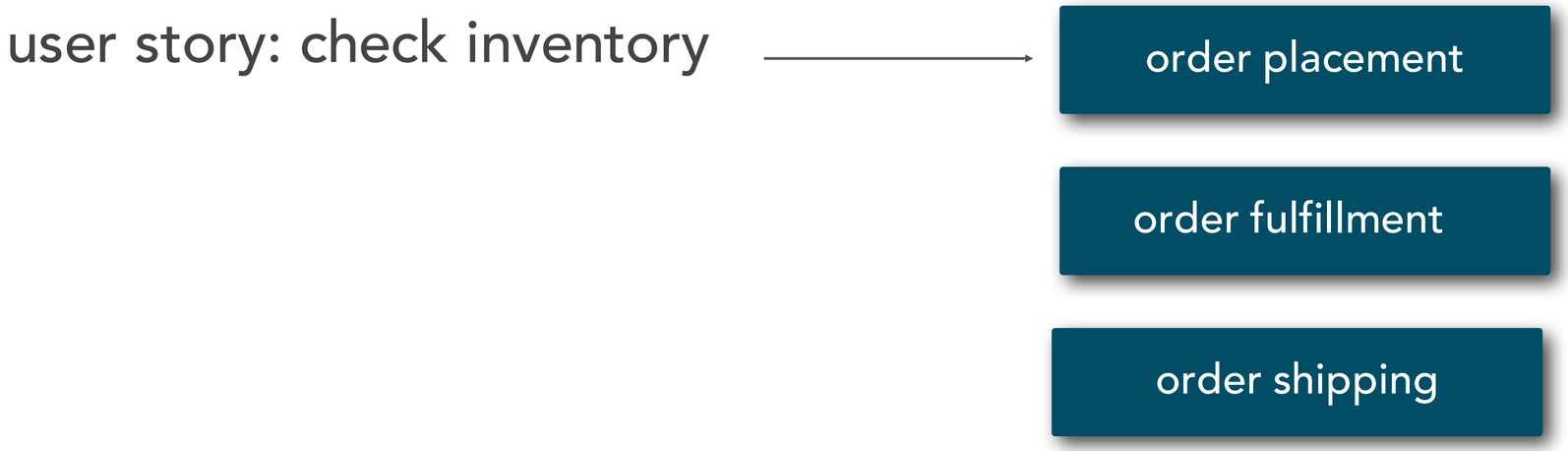
component identification

assign requirements, use cases, or user stories to a component

user story: check inventory

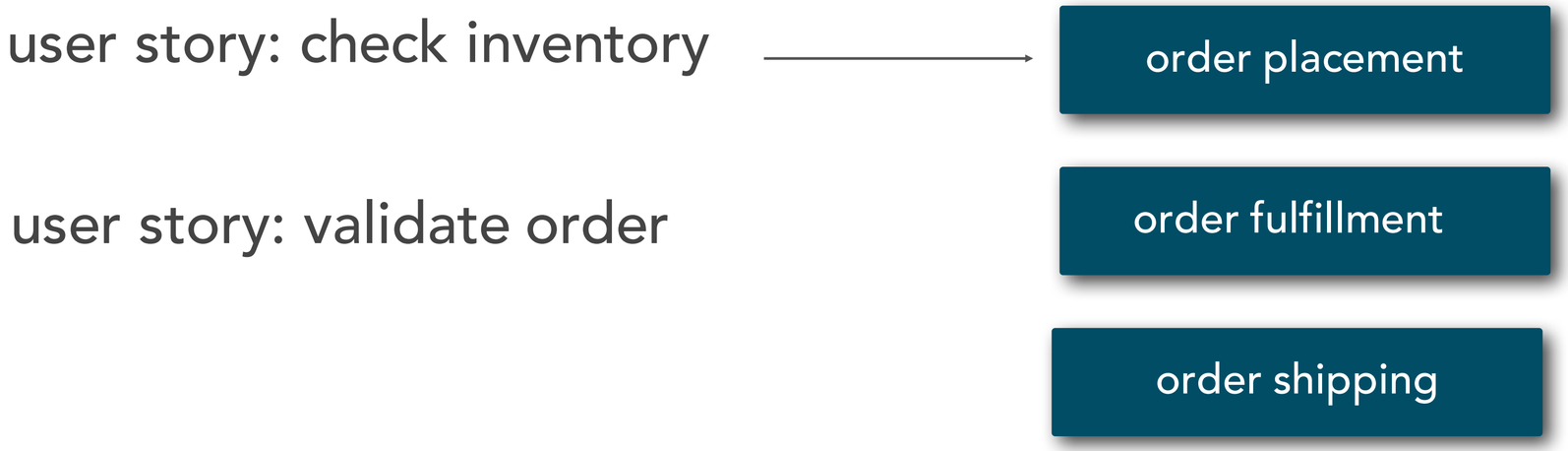
component identification

assign requirements, use cases, or user stories to a component



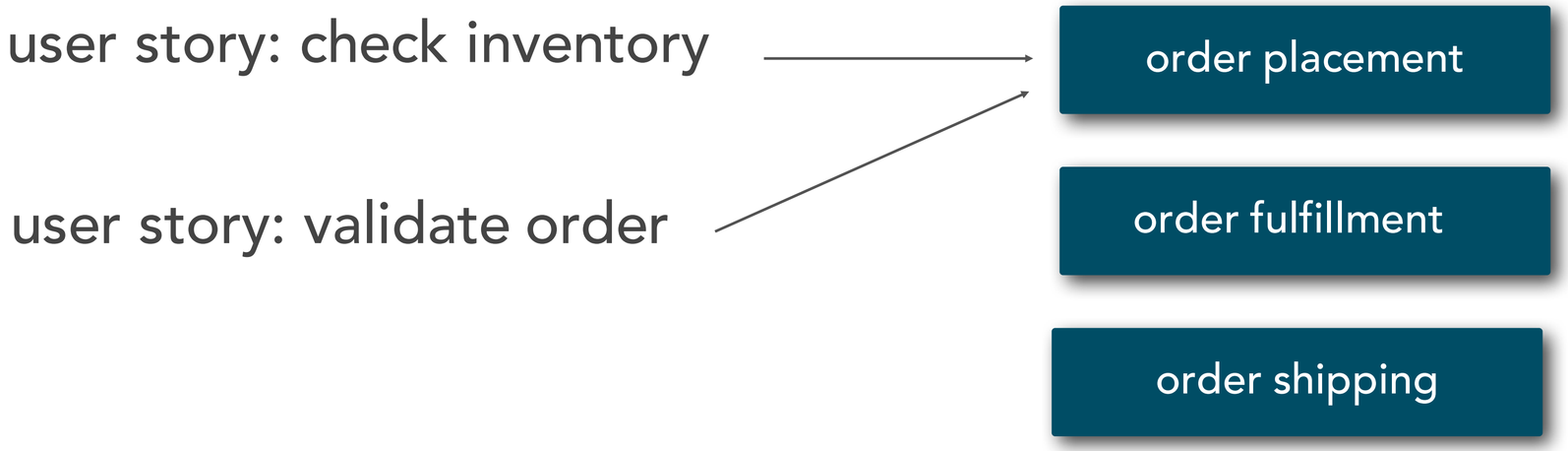
component identification

assign requirements, use cases, or user stories to a component



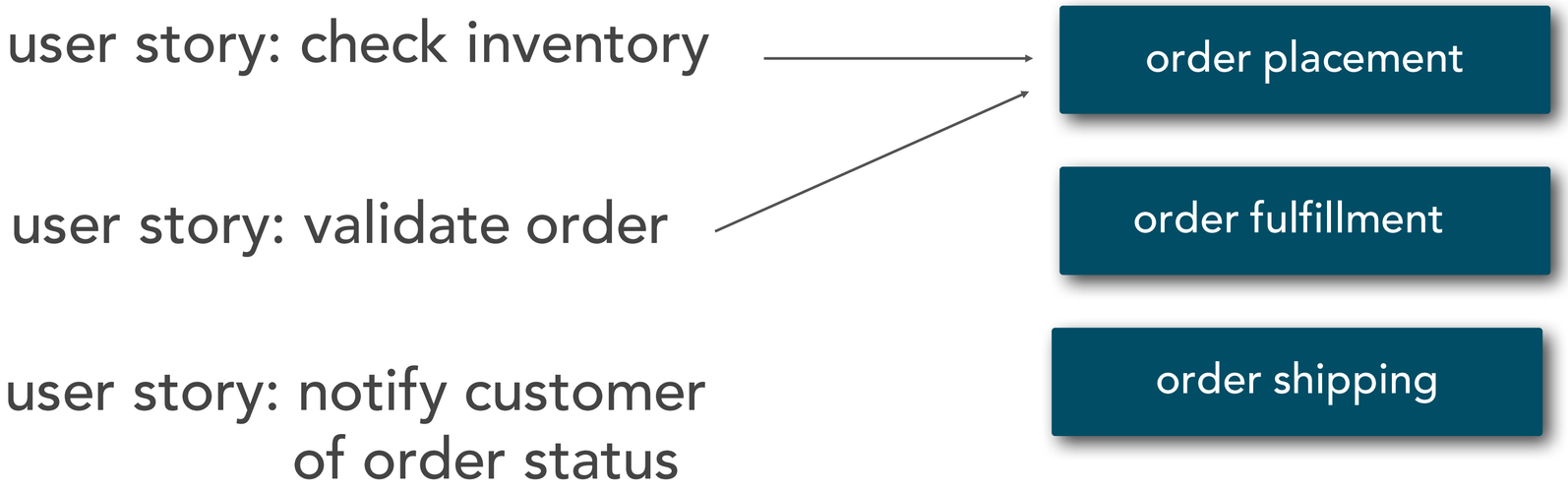
component identification

assign requirements, use cases, or user stories to a component



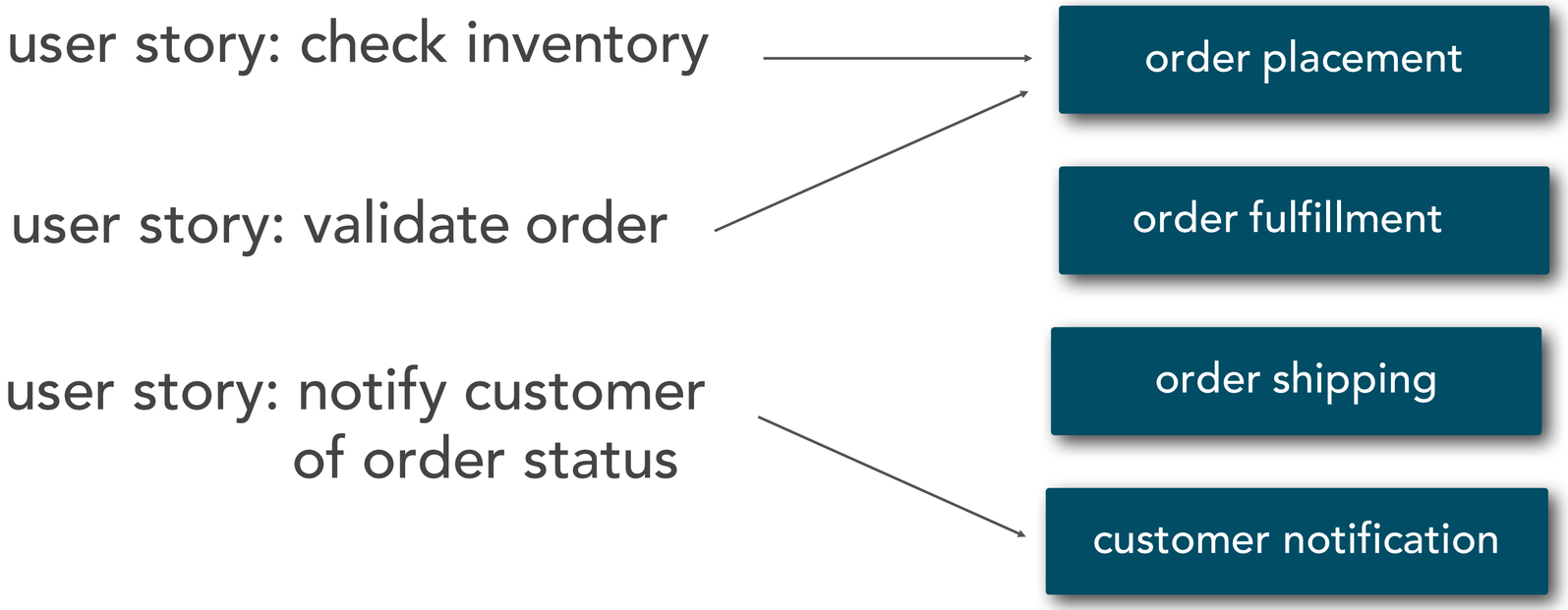
component identification

assign requirements, use cases, or user stories to a component



component identification

assign requirements, use cases, or user stories to a component



component identification

component granularity

responsible for creating, deleting, and updating orders. also responsible for shipping the order and tracking the order once it has been shipped. this component is also responsible for notifying the customer each time the order status changes.

component identification

component granularity

order maintenance	responsible for creating, deleting, and updating orders.
order shipment	responsible for shipping and tracking orders
customer notification	responsible for notifying the customer when the order status changes.

architecture katas

identifying major components

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - video stream of the action after the fact
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotionals/specials
  - offer local daily promotionals/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.

Your Architectural Kata is...

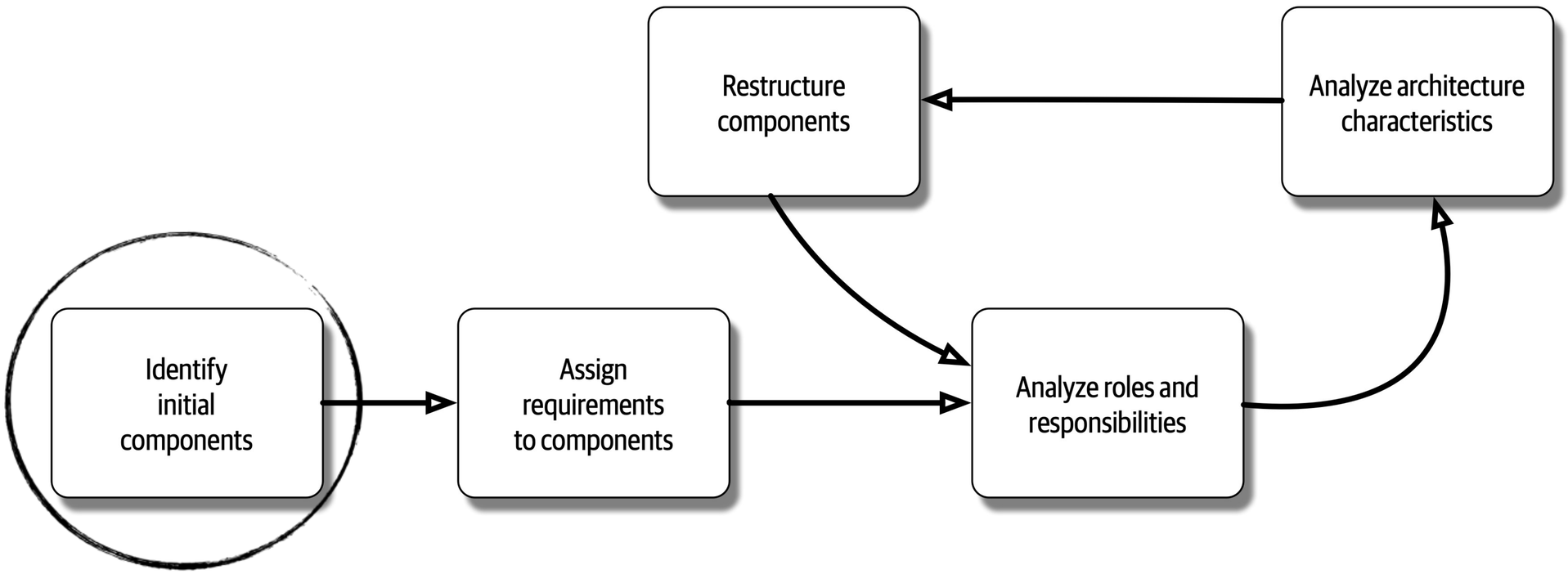
Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Going Going Gone!



Your Architectural Kata is...

Going Going Gone!

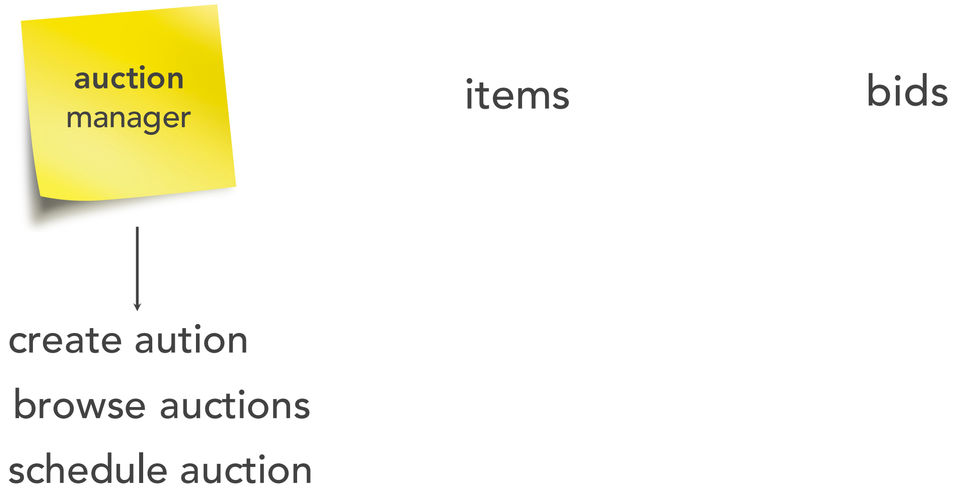
the “entity trap”

auctions items bids

Your Architectural Kata is...

Going Going Gone!

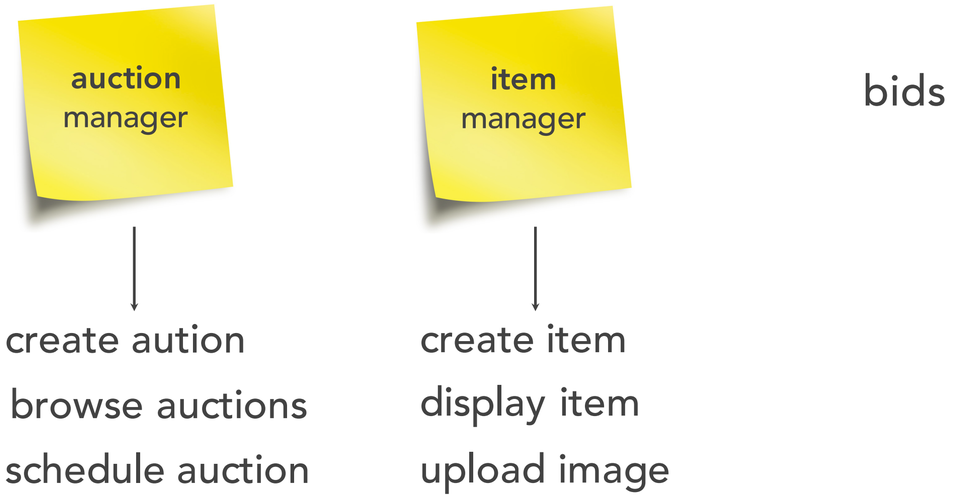
the “entity trap”



Your Architectural Kata is...

Going Going Gone!

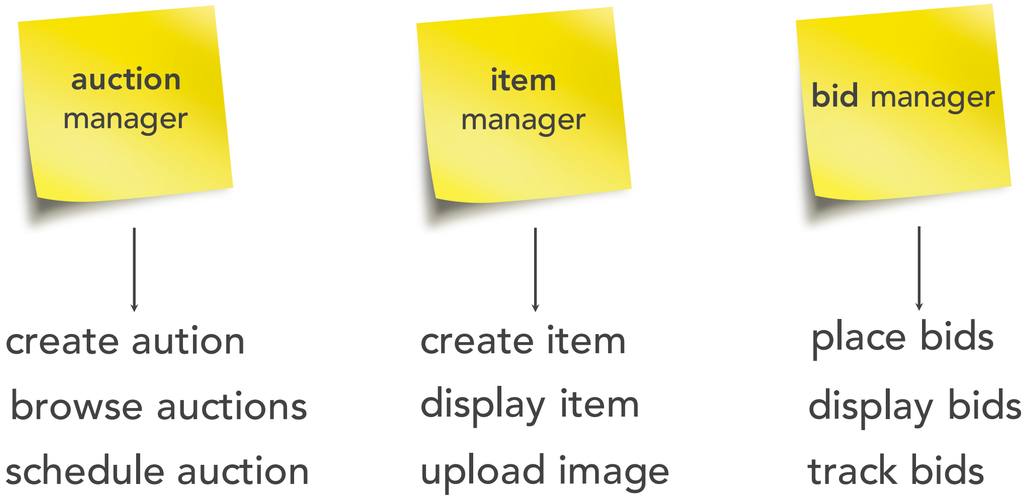
the “entity trap”



Your Architectural Kata is...

Going Going Gone!

the “entity trap”



Your Architectural Kata is...

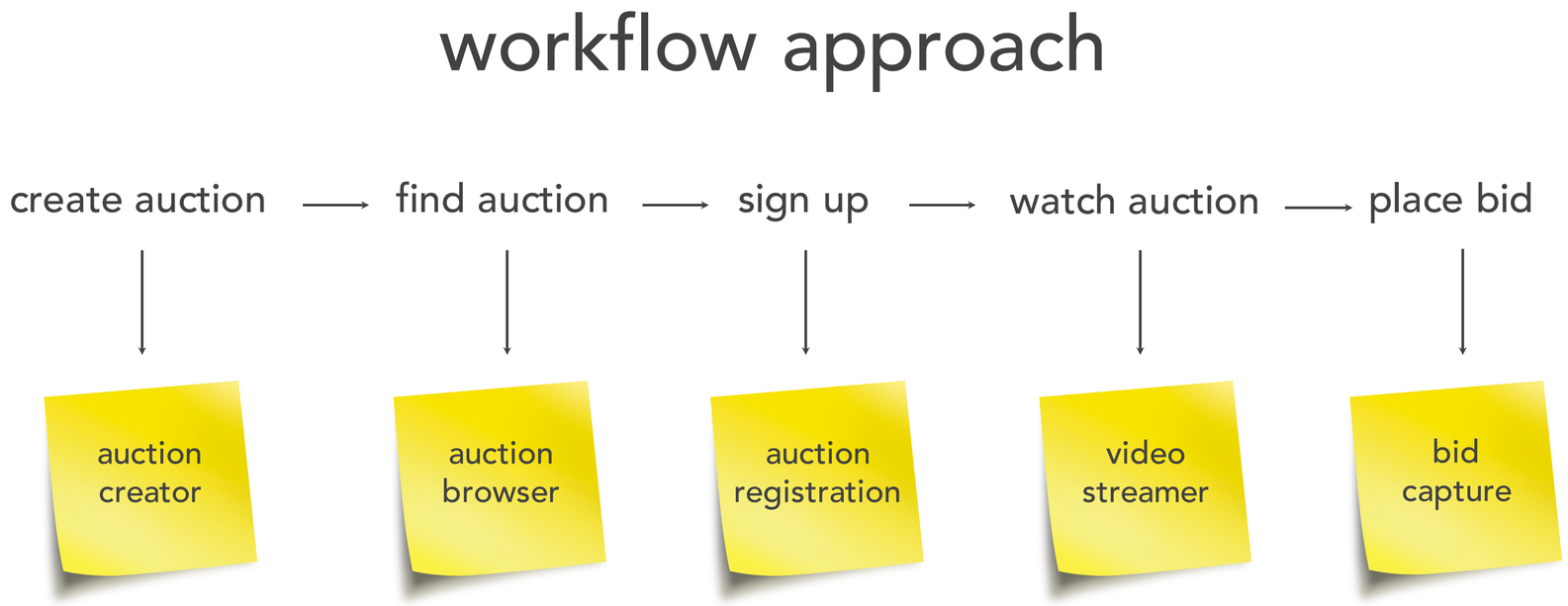
Going Going Gone!

workflow approach

create auction ⟶ find auction ⟶ sign up ⟶ watch auction ⟶ place bid

Your Architectural Kata is...

Going Going Gone!



Your Architectural Kata is...

Going Going Gone!

actor/action approach

bidder

auctioneer

system

Your Architectural Kata is...

Going Going Gone!

actor/action approach

bidder

auctioneer

system

view live video stream

view live bid stream

place a bid

Your Architectural Kata is...

Going Going Gone!

actor/action approach

bidder

auctioneer

system

view live video stream
view live bid stream
place a bid

enter live bids into system
receive online bid
mark item as sold

Your Architectural Kata is...

Going Going Gone!

actor/action approach

bidder

view live video stream

view live bid stream

place a bid

auctioneer

enter live bids into system

receive online bid

mark item as sold

system

start auction

make payment

track bidder activity

Your Architectural Kata is...

Going Going Gone!

bidder

view live video stream
view live bid stream
place a bid

auctioneer

enter live bids into system
receive online bid
mark item as sold

system

start auction
make payment
track bidder activity

Your Architectural Kata is...

Going Going Gone!

bidder

view live video stream
view live bid stream
place a bid

auctioneer

enter live bids into system
receive online bid
mark item as sold

system

start auction
make payment
track bidder activity

Your Architectural Kata is...

Going Going Gone!

bidder

view live video stream
view live bid stream
place a bid

auctioneer

enter live bids into system
receive online bid
mark item as sold

system

start auction
make payment
track bidder activity

auction session

Your Architectural Kata is...

Going Going Gone!

bidder

view live video stream
view live bid stream
place a bid

auctioneer

enter live bids into system
receive online bid
mark item as sold

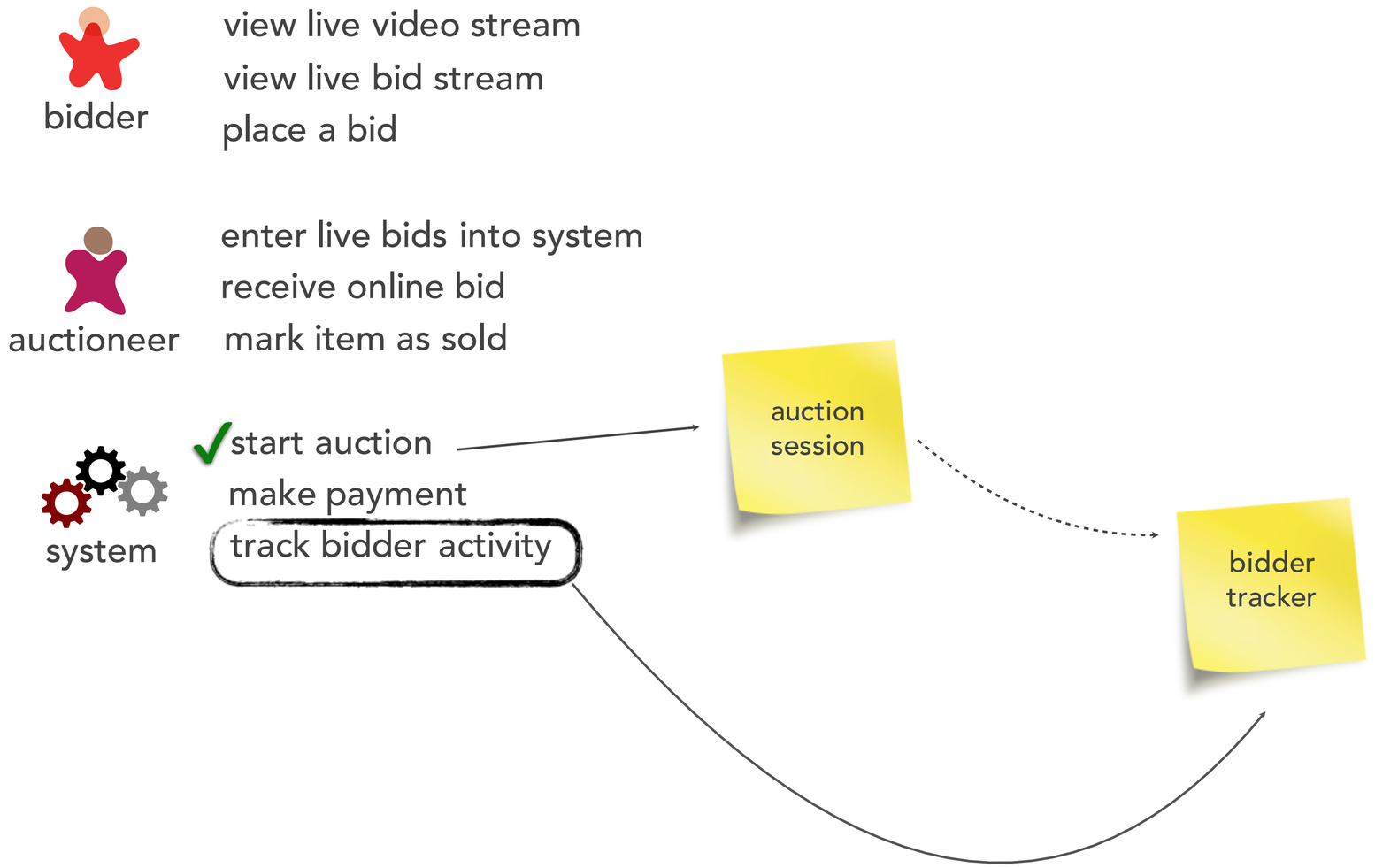
system

✓start auction
make payment
track bidder activity

auction session

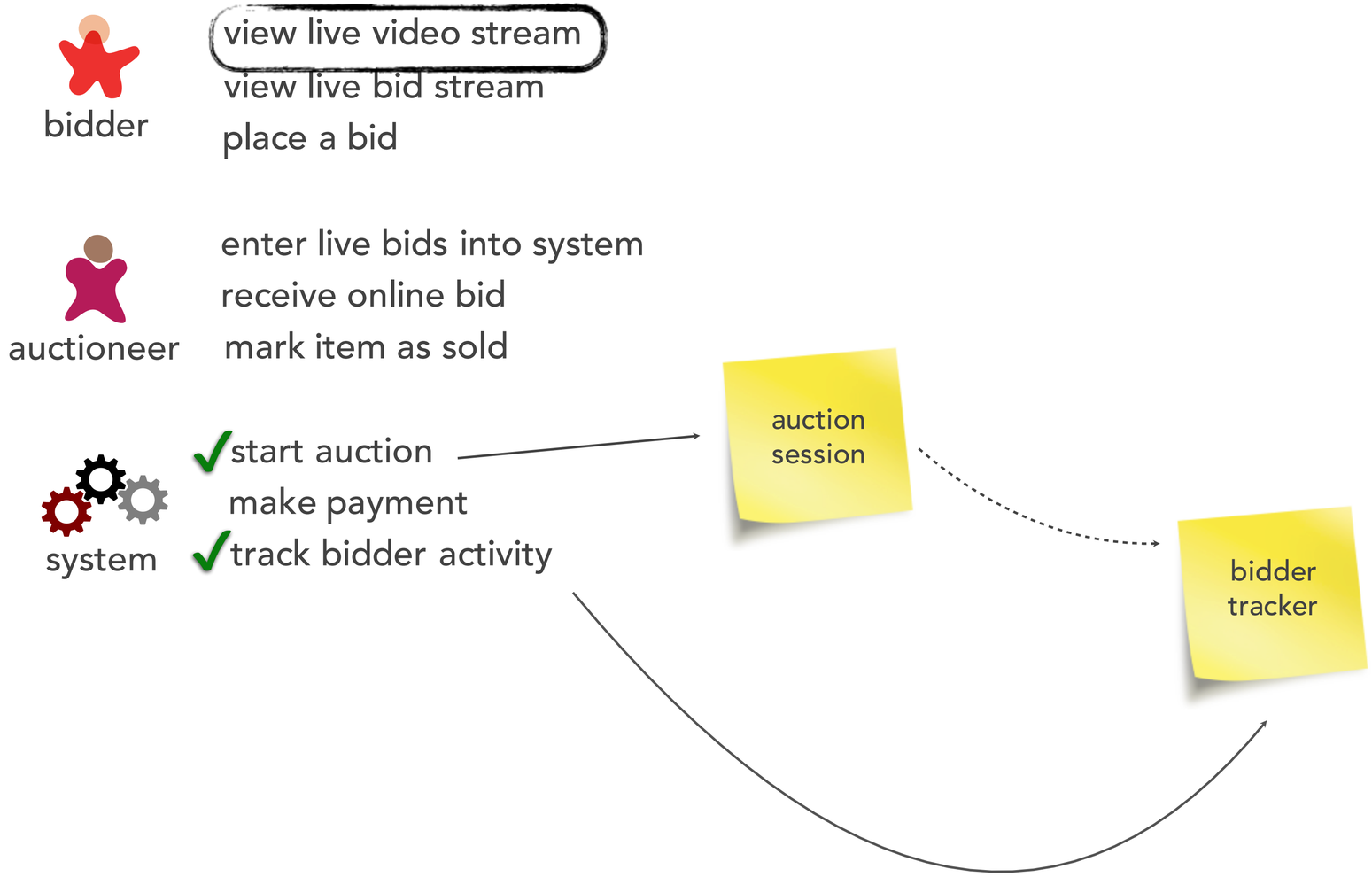
Your Architectural Kata is...

Going Going Gone!



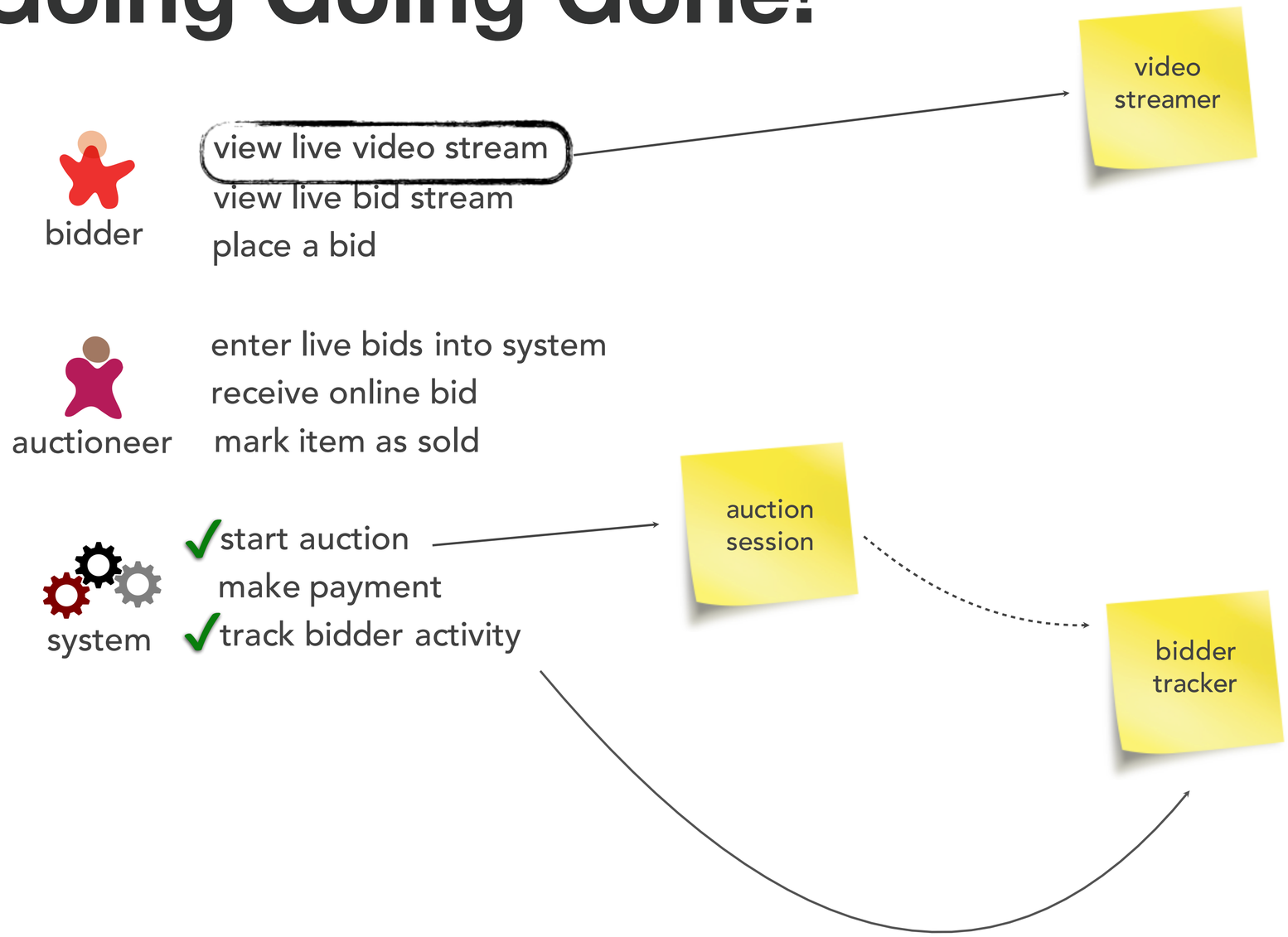
Your Architectural Kata is...

Going Going Gone!



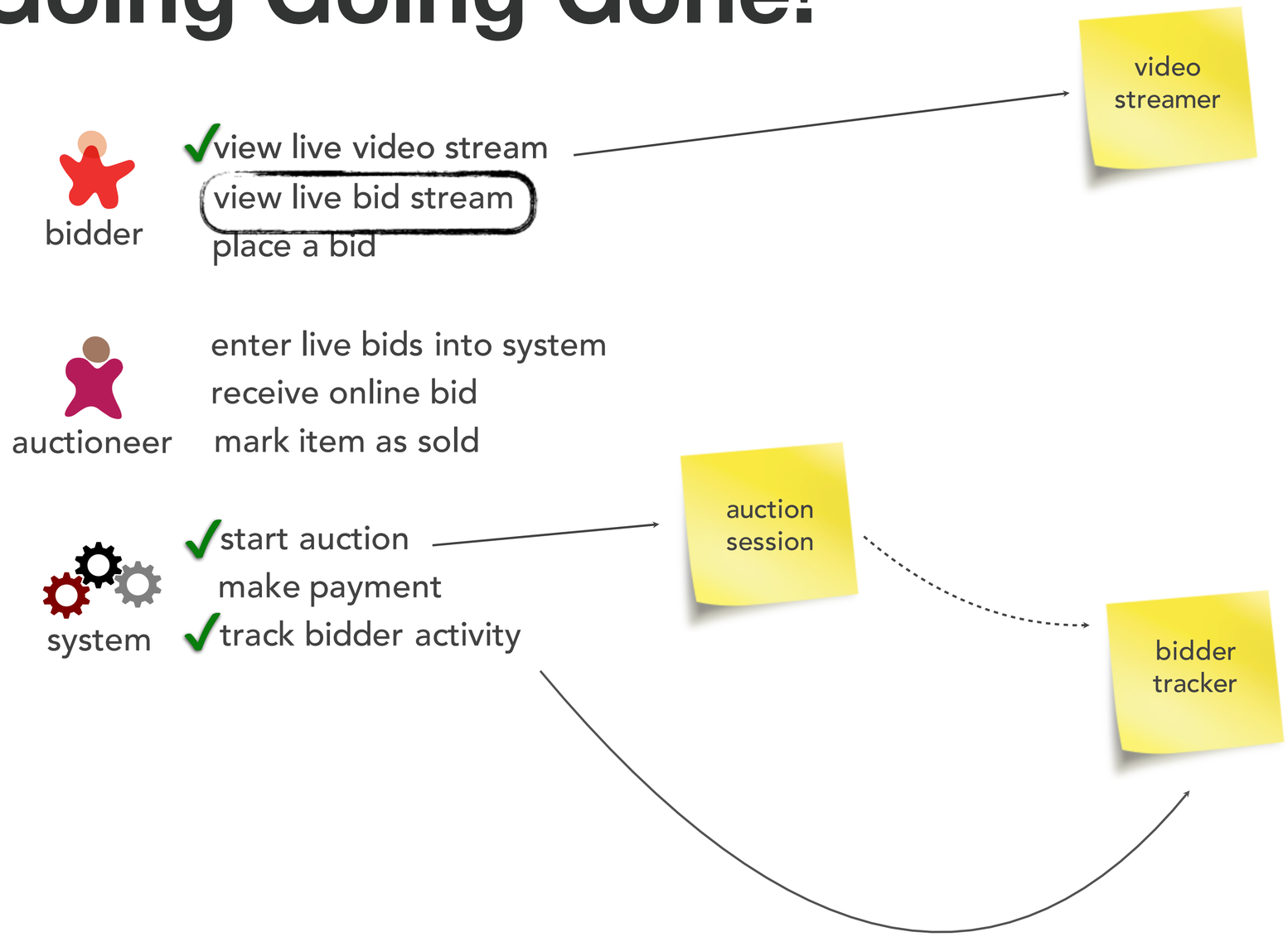
Your Architectural Kata is...

Going Going Gone!



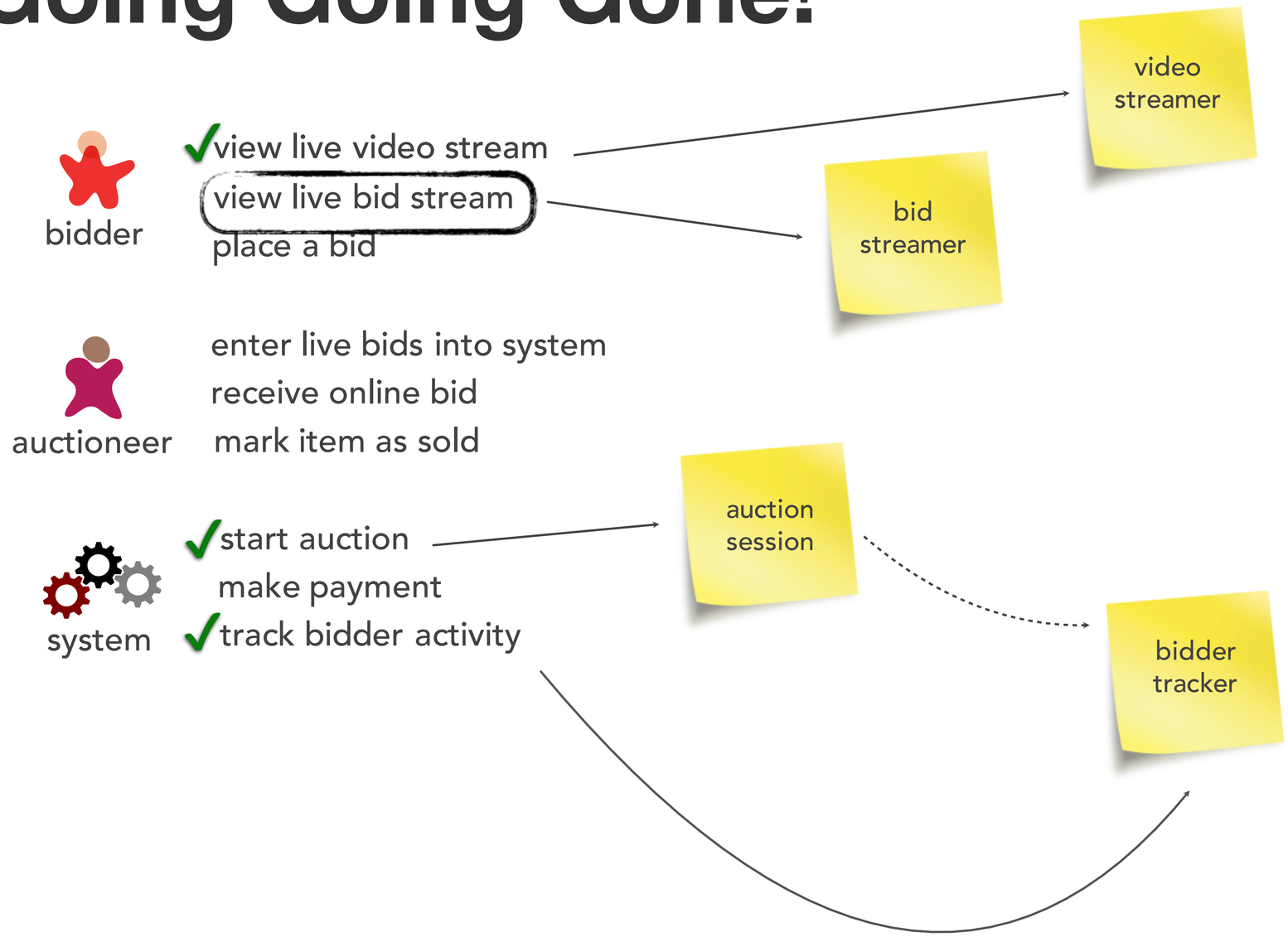
Your Architectural Kata is...

Going Going Gone!



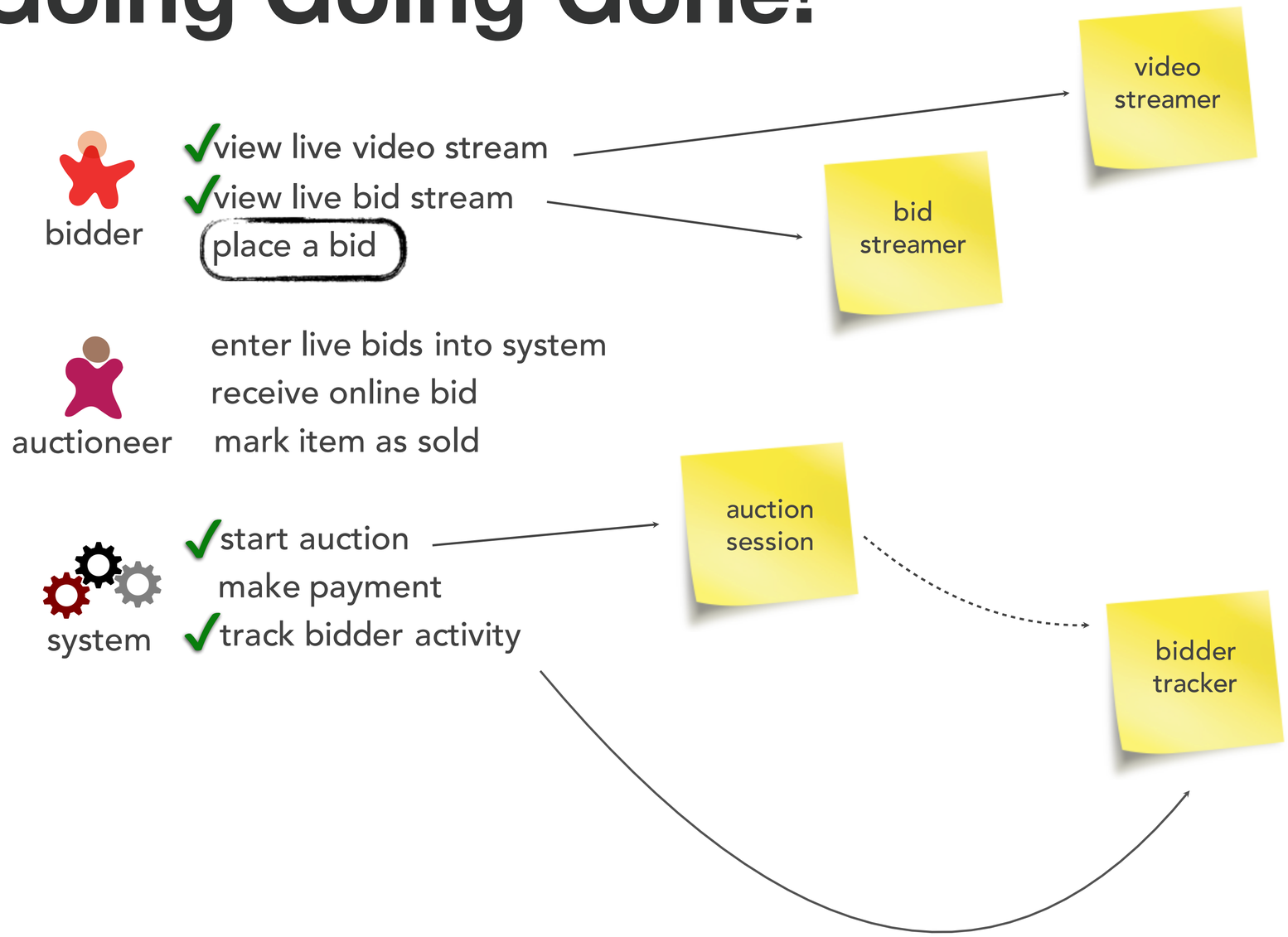
Your Architectural Kata is...

Going Going Gone!



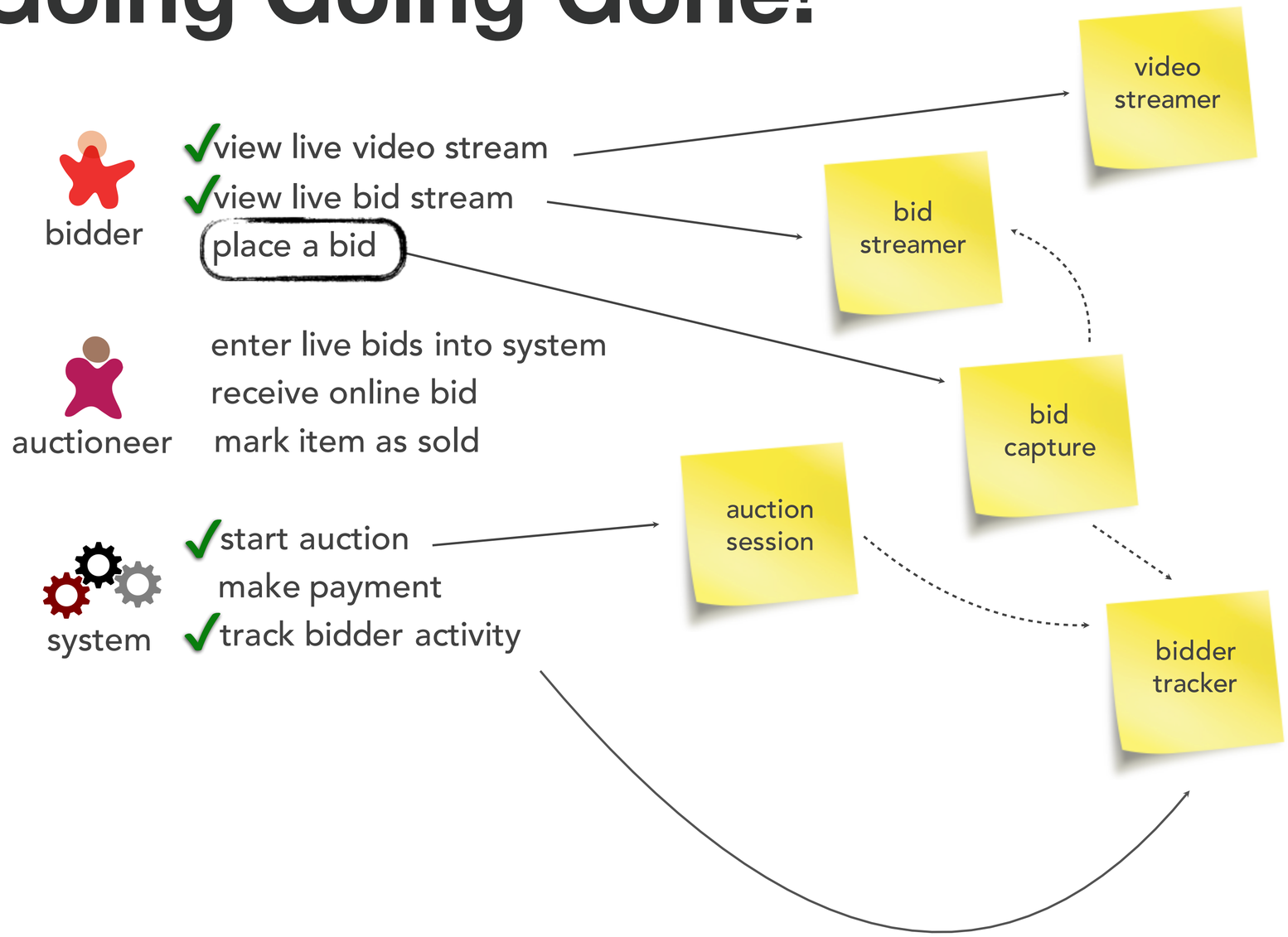
Your Architectural Kata is...

Going Going Gone!



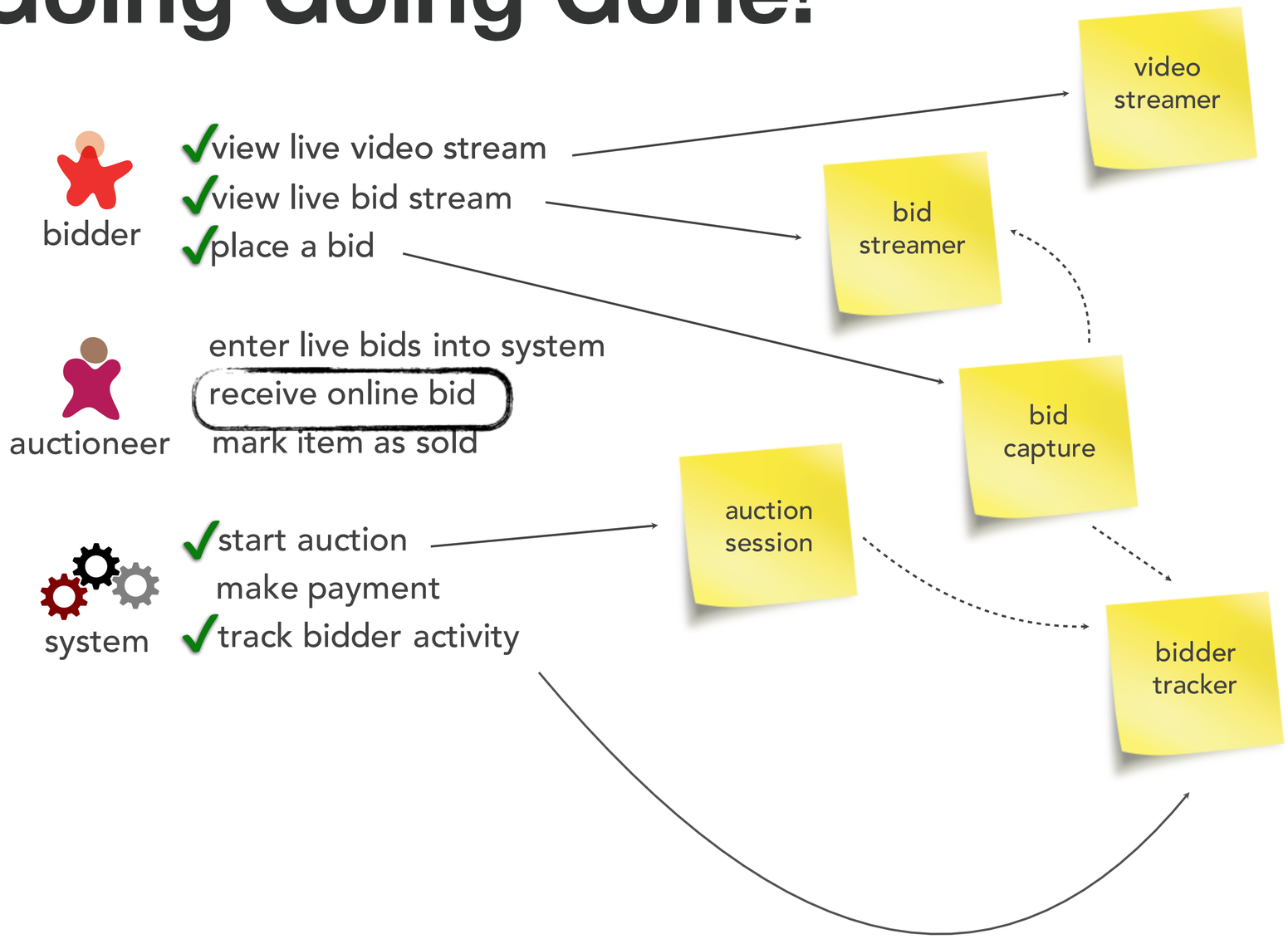
Your Architectural Kata is...

Going Going Gone!



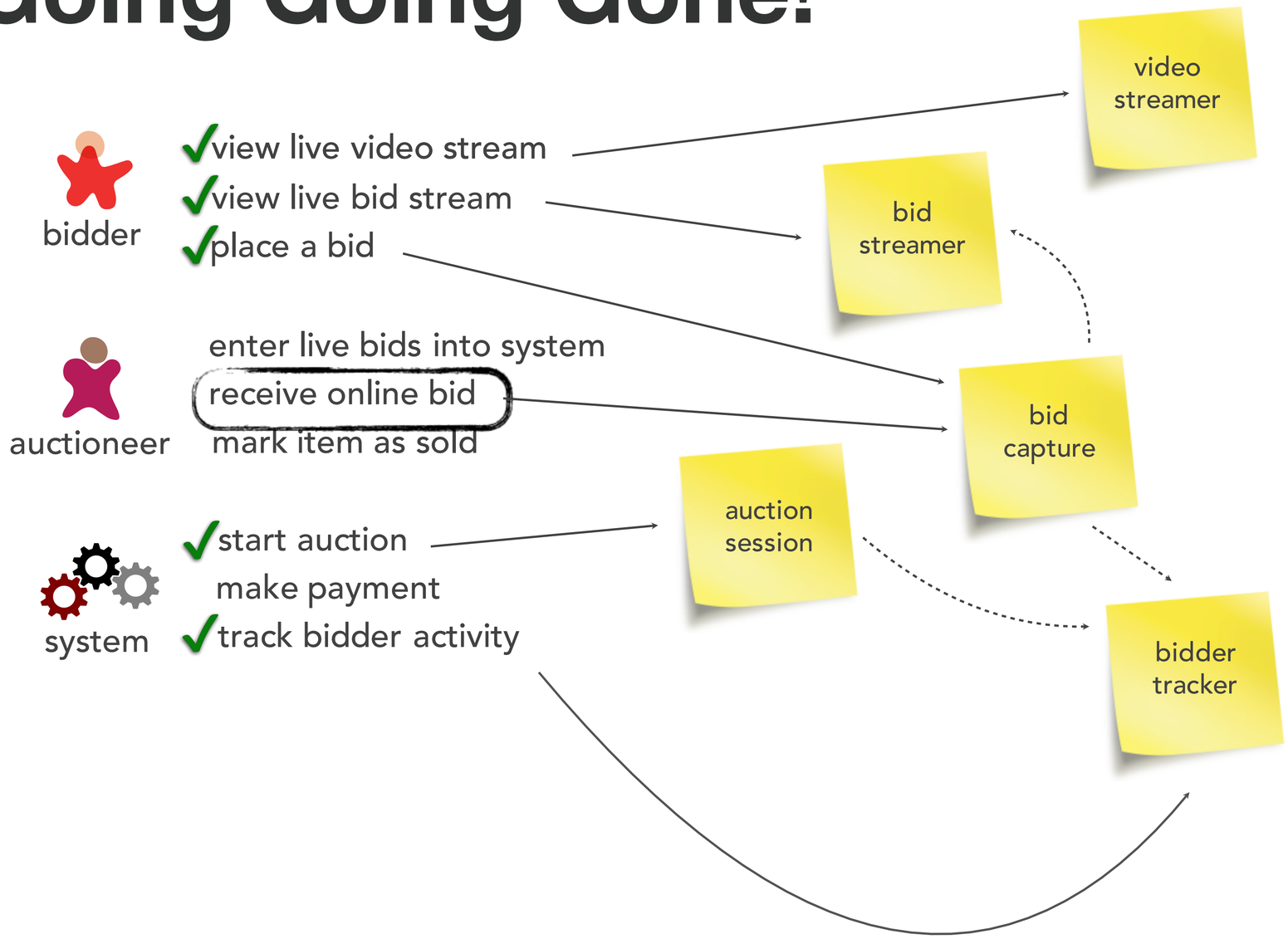
Your Architectural Kata is...

Going Going Gone!



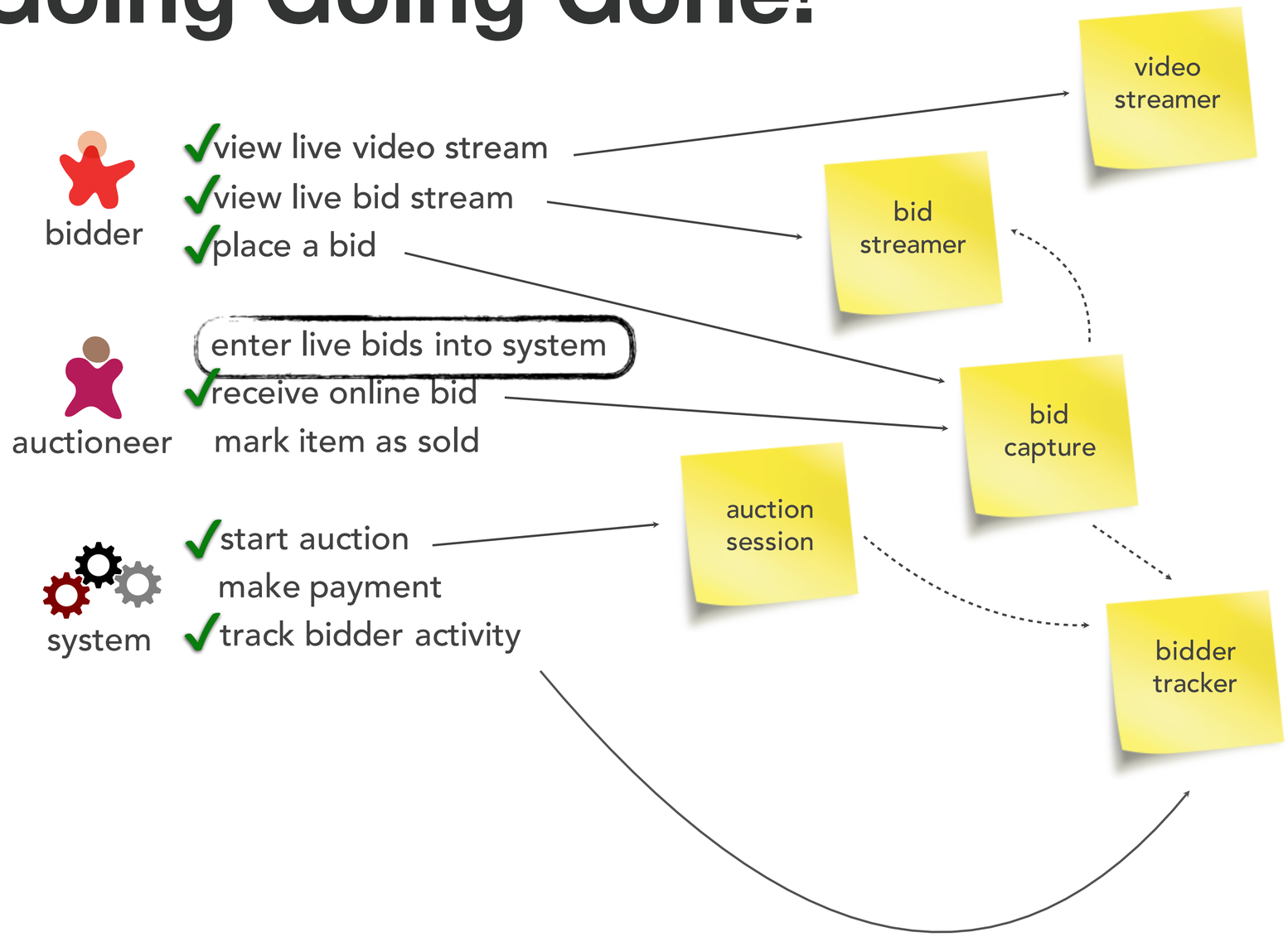
Your Architectural Kata is...

Going Going Gone!



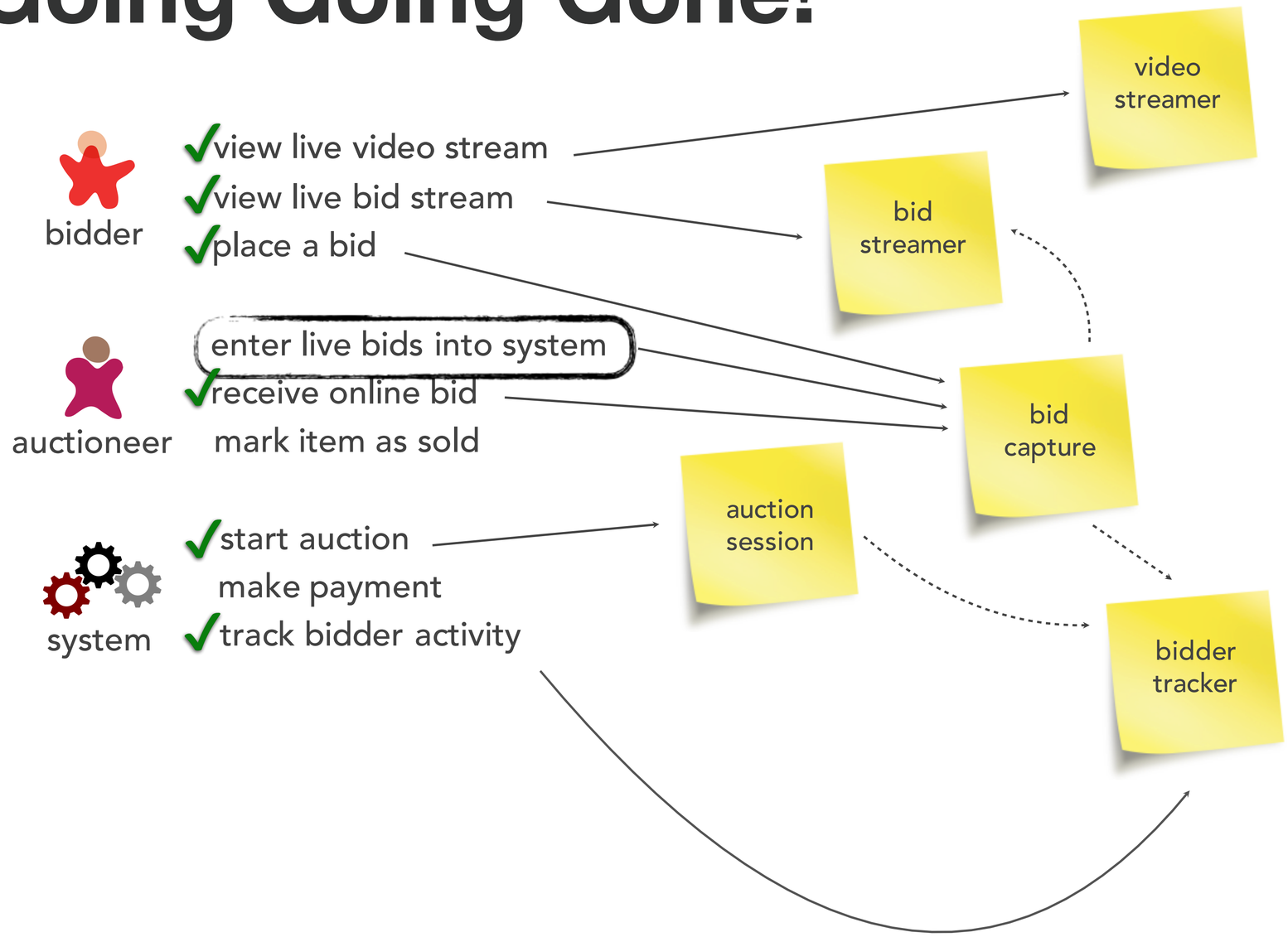
Your Architectural Kata is...

Going Going Gone!



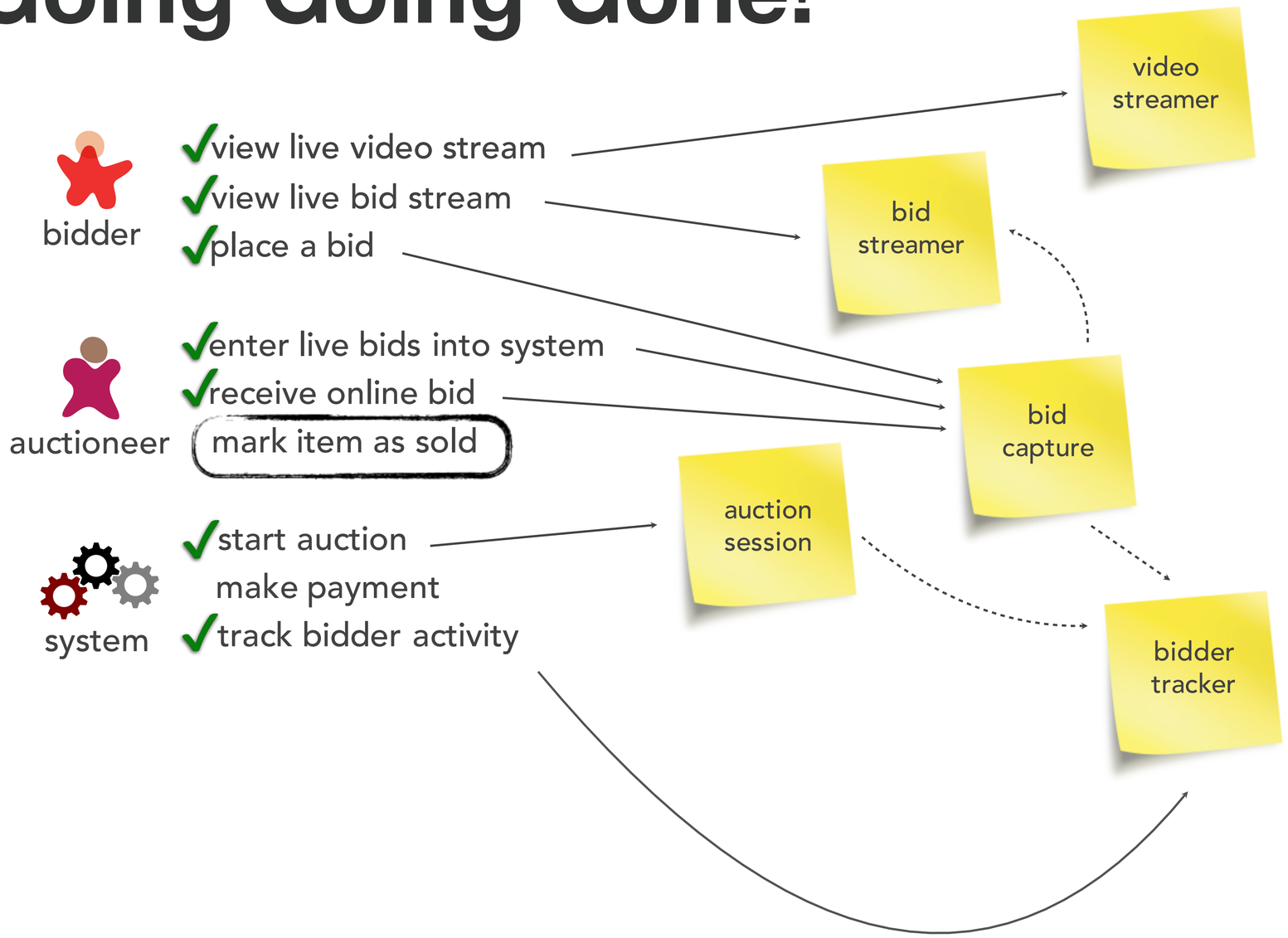
Your Architectural Kata is...

Going Going Gone!



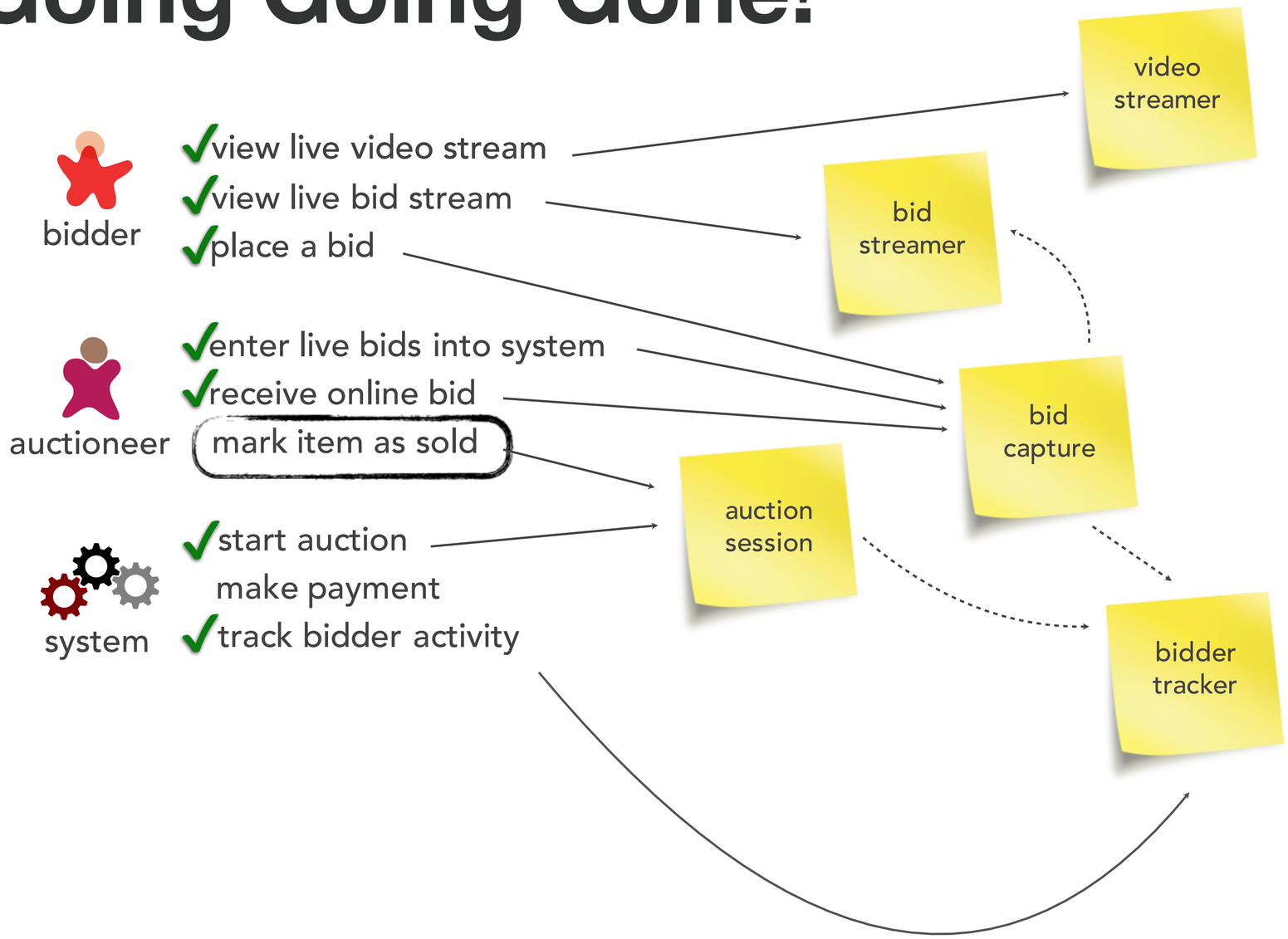
Your Architectural Kata is...

Going Going Gone!



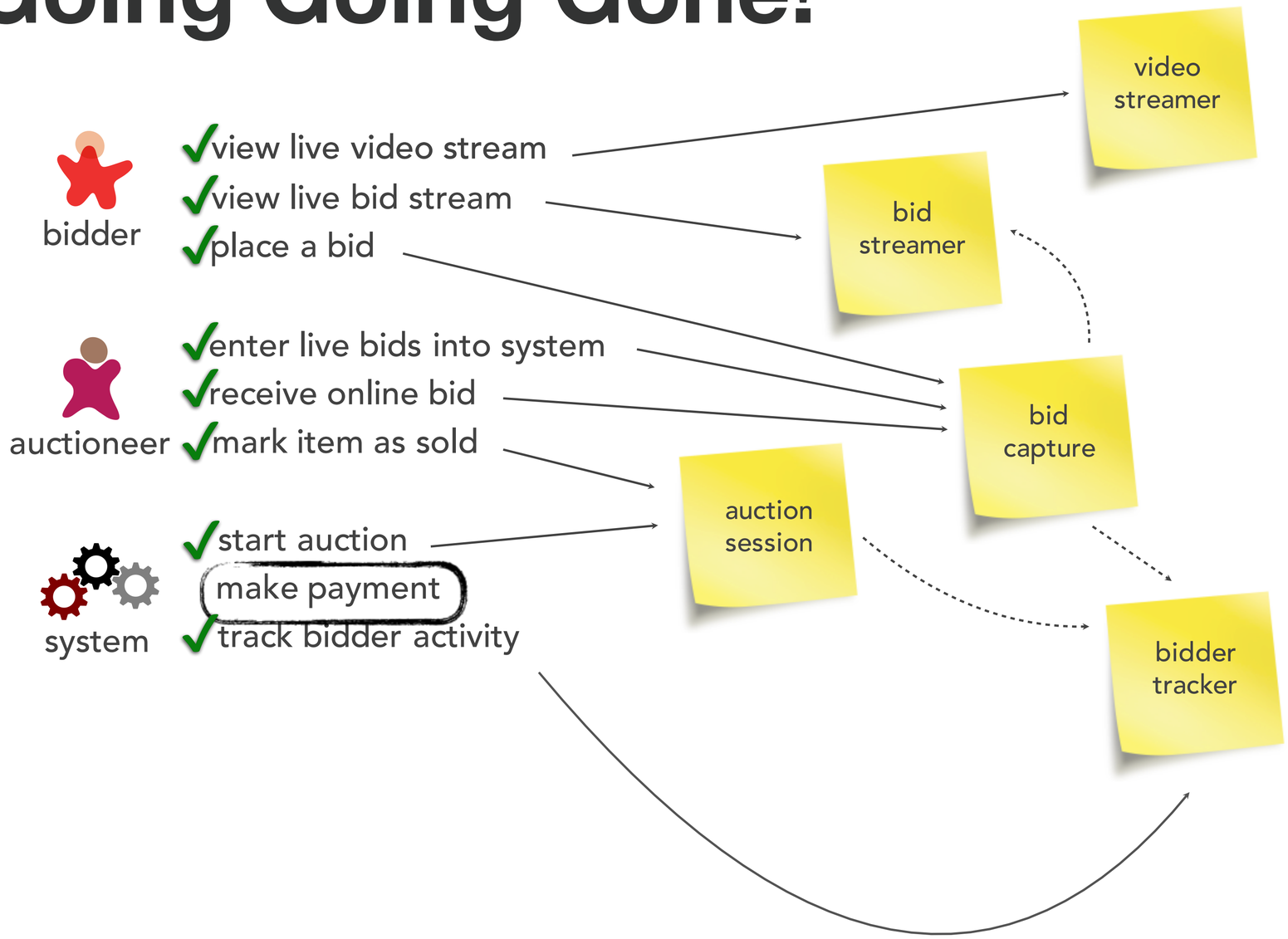
Your Architectural Kata is...

Going Going Gone!



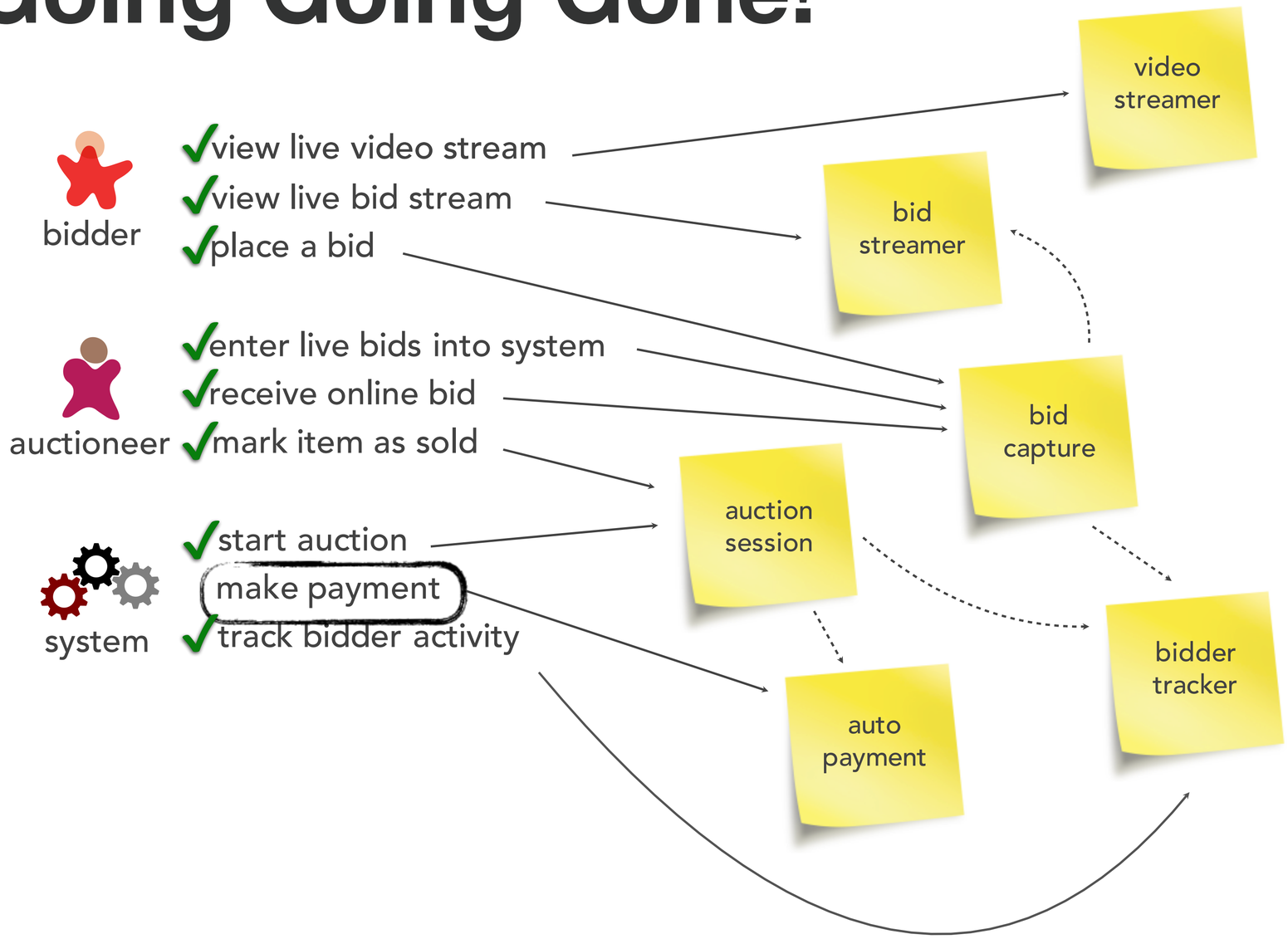
Your Architectural Kata is...

Going Going Gone!



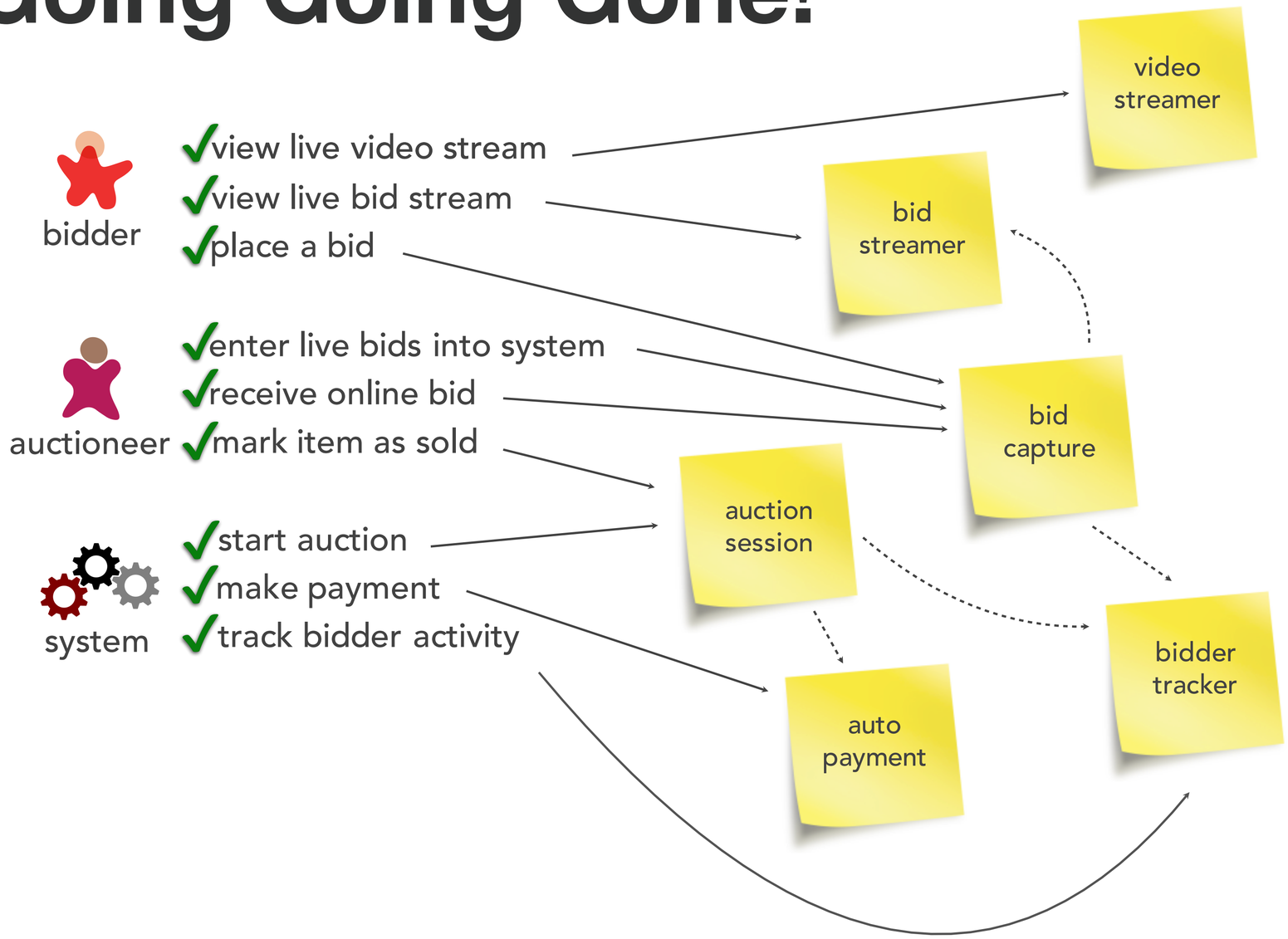
Your Architectural Kata is...

Going Going Gone!



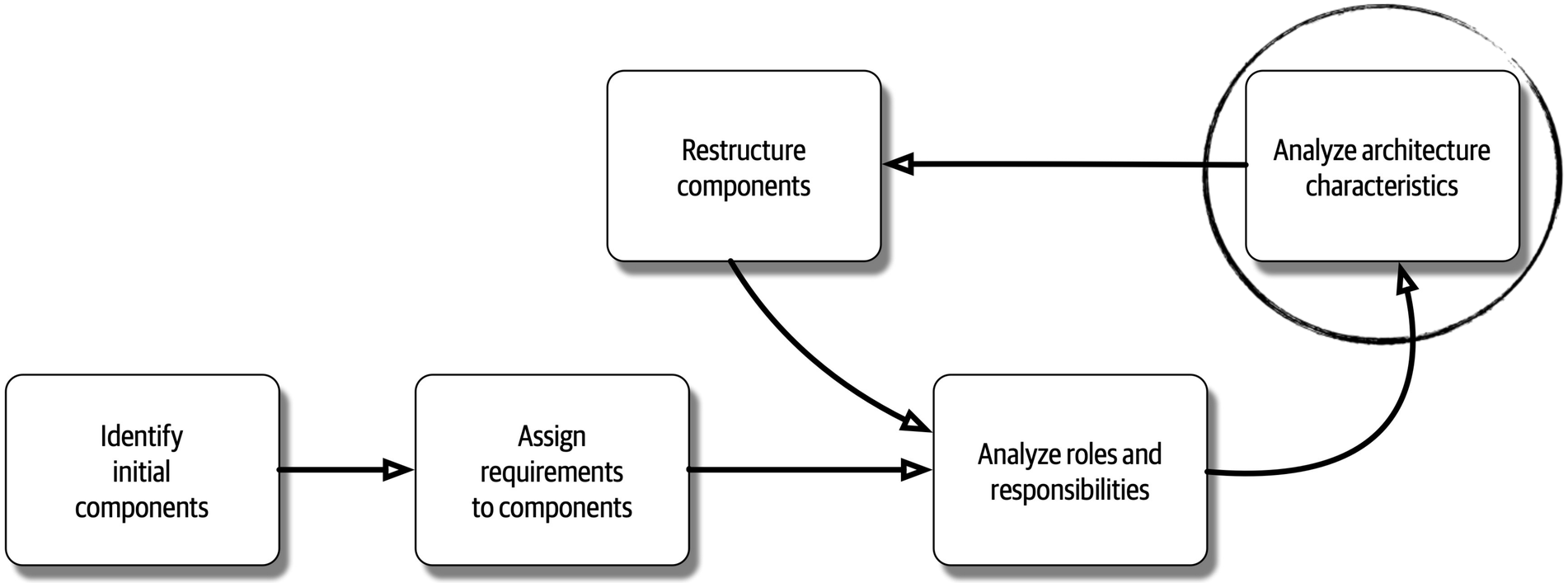
Your Architectural Kata is...

Going Going Gone!



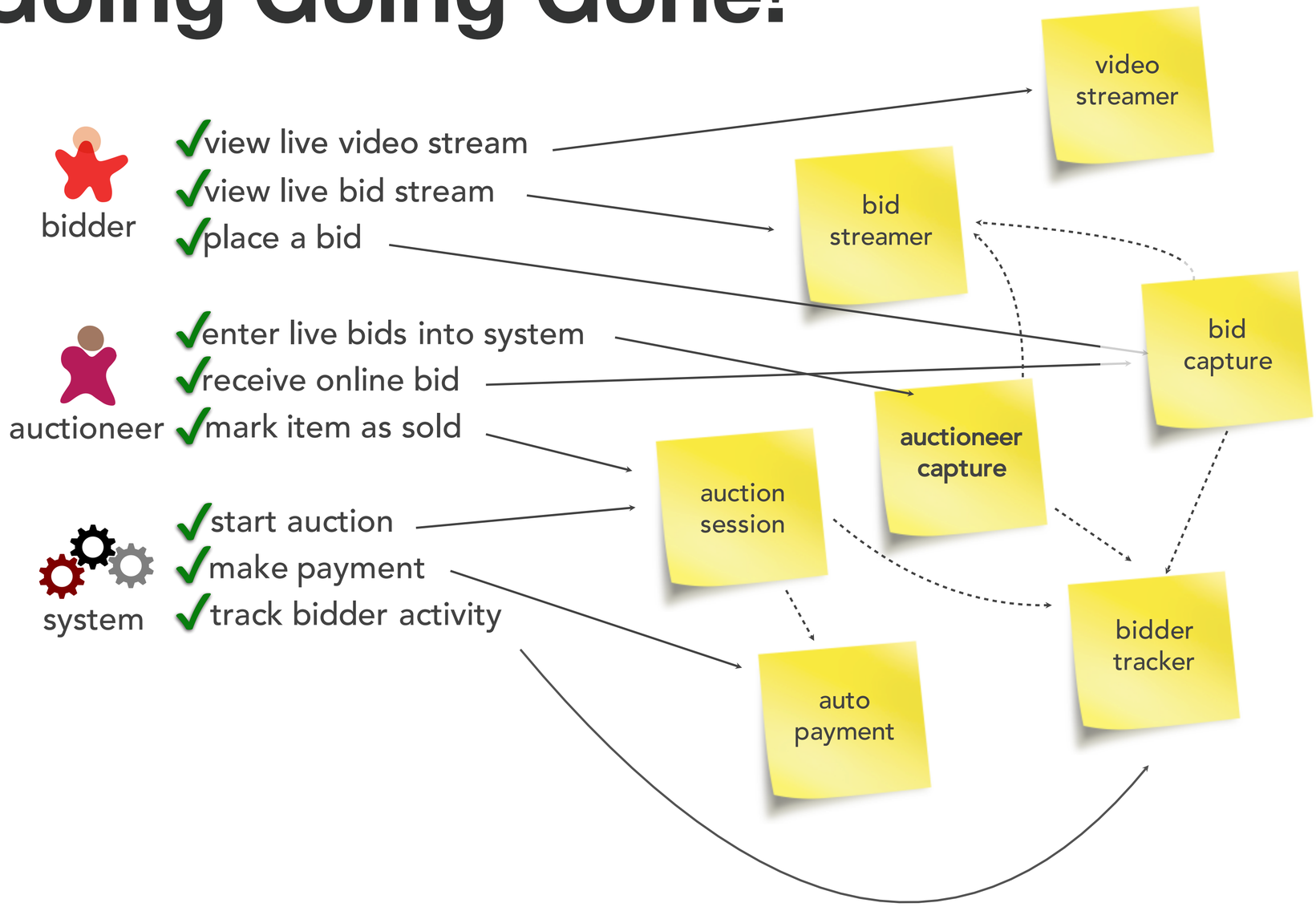
Your Architectural Kata is...

Going Going Gone!



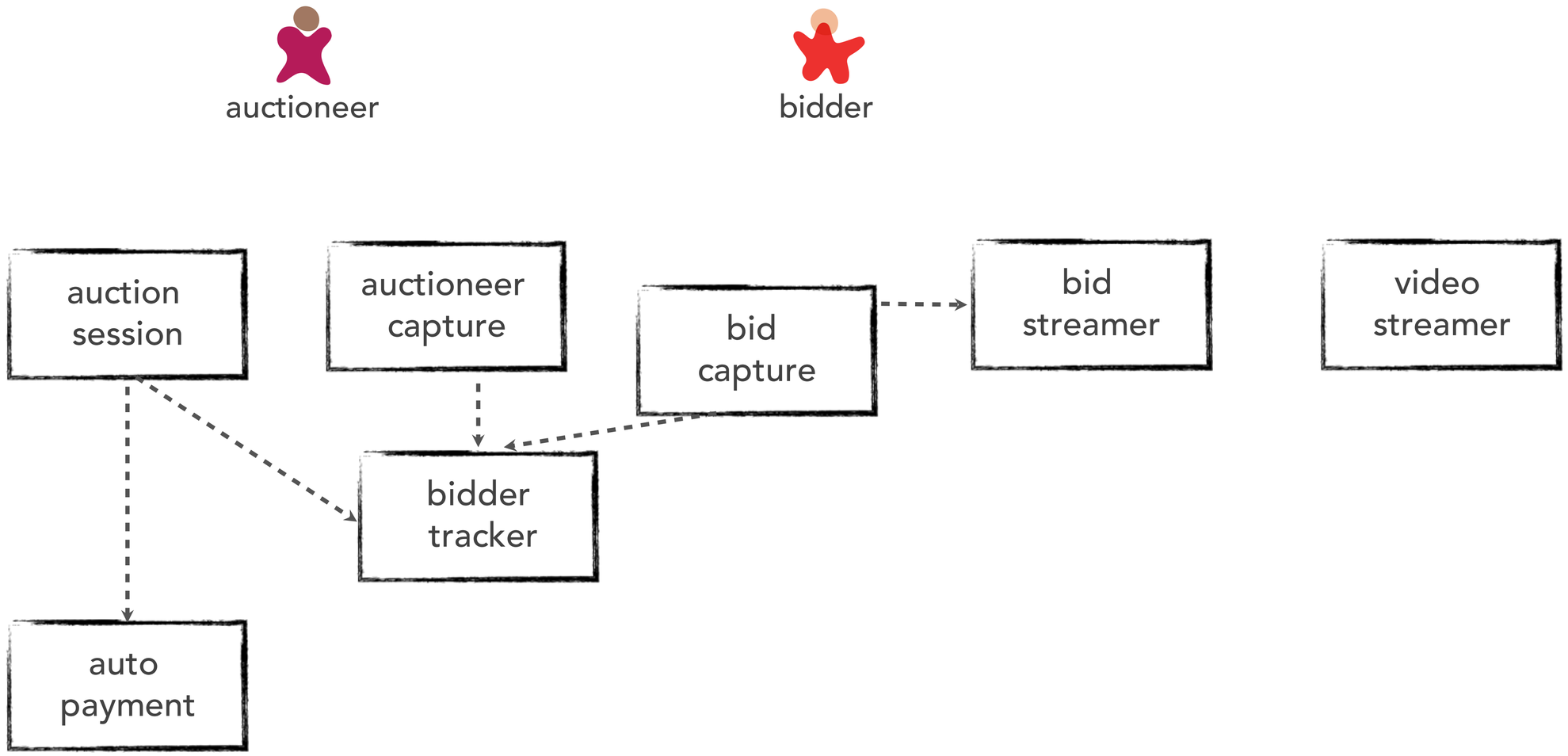
Your Architectural Kata is...

Going Going Gone!



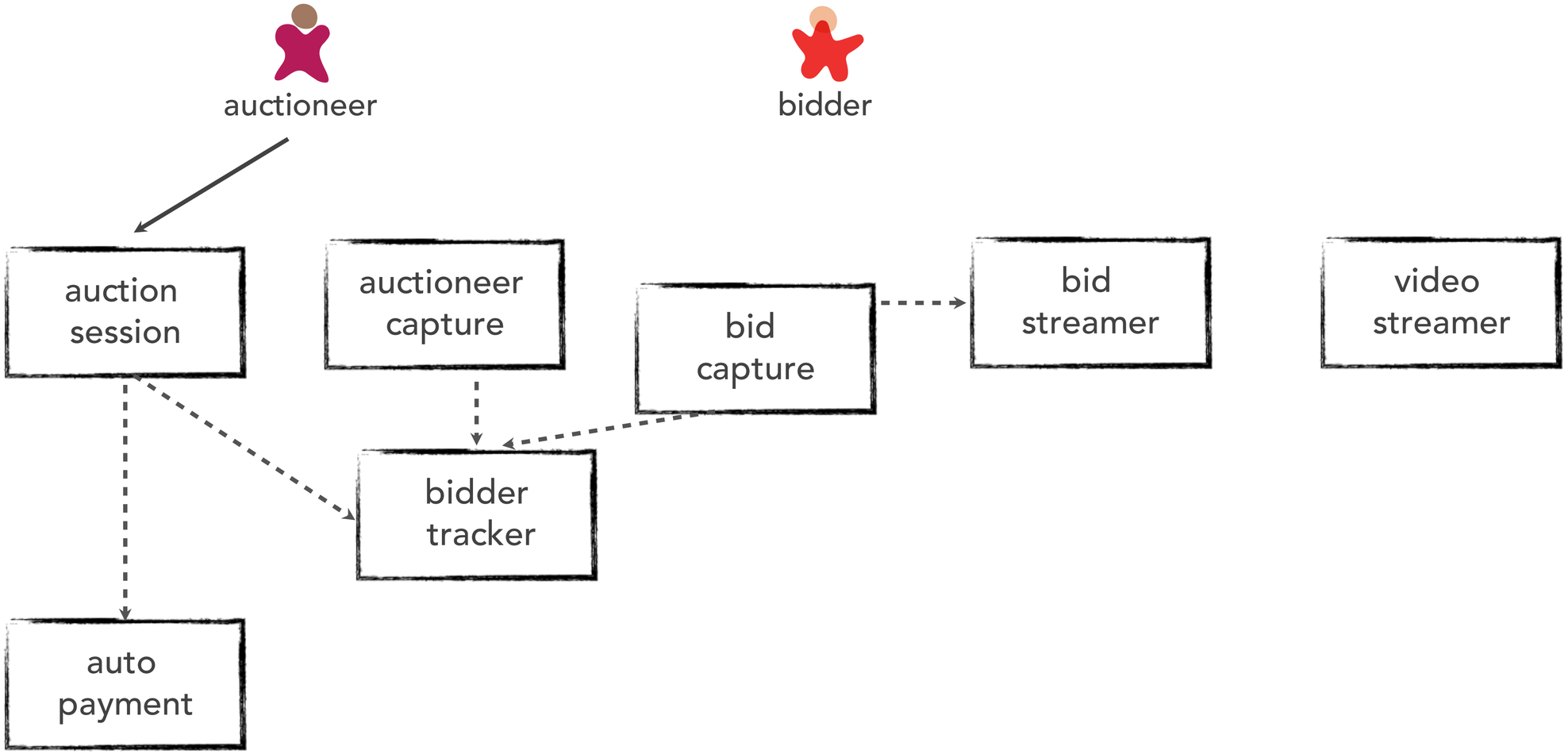
Your Architectural Kata is...

Going Going Gone!



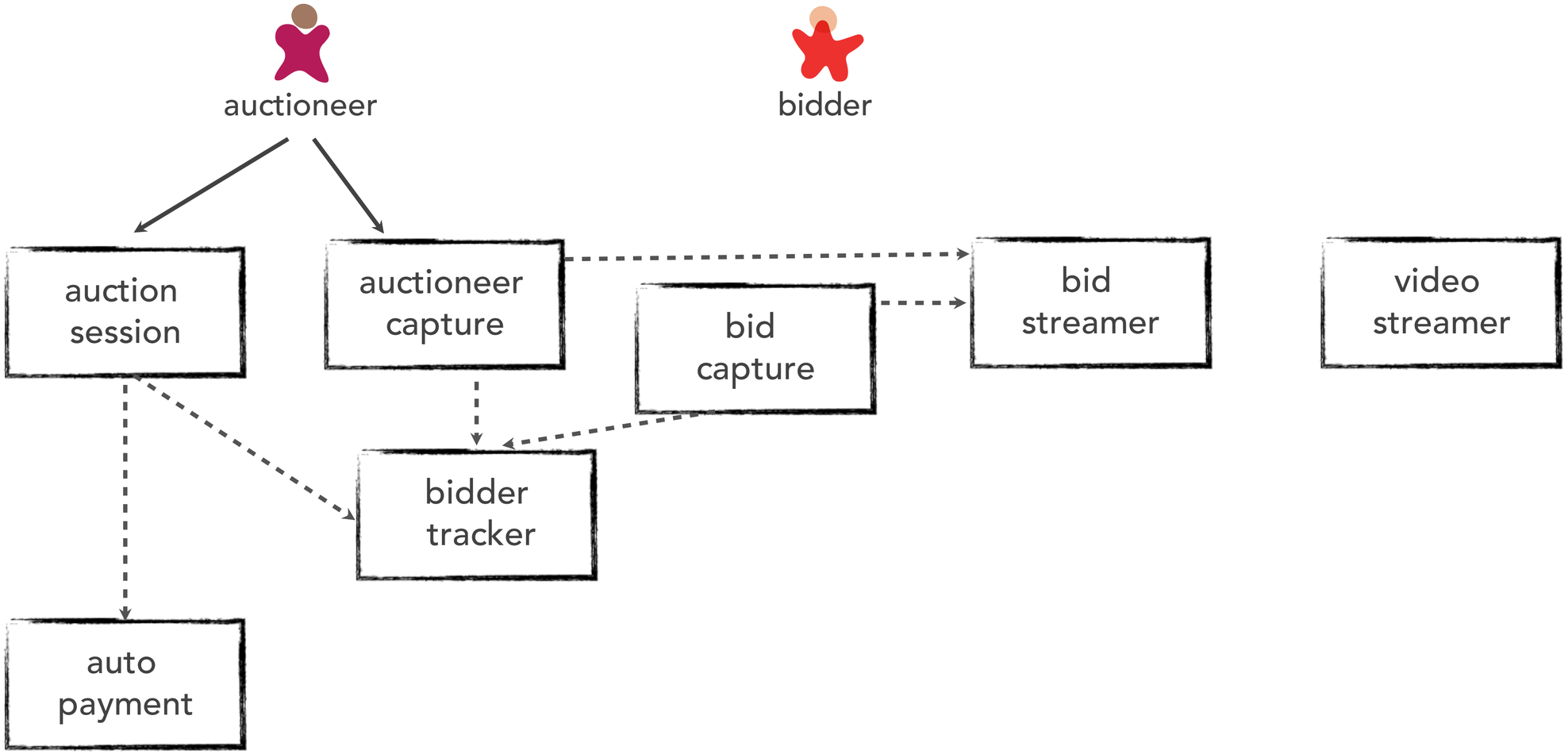
Your Architectural Kata is...

Going Going Gone!



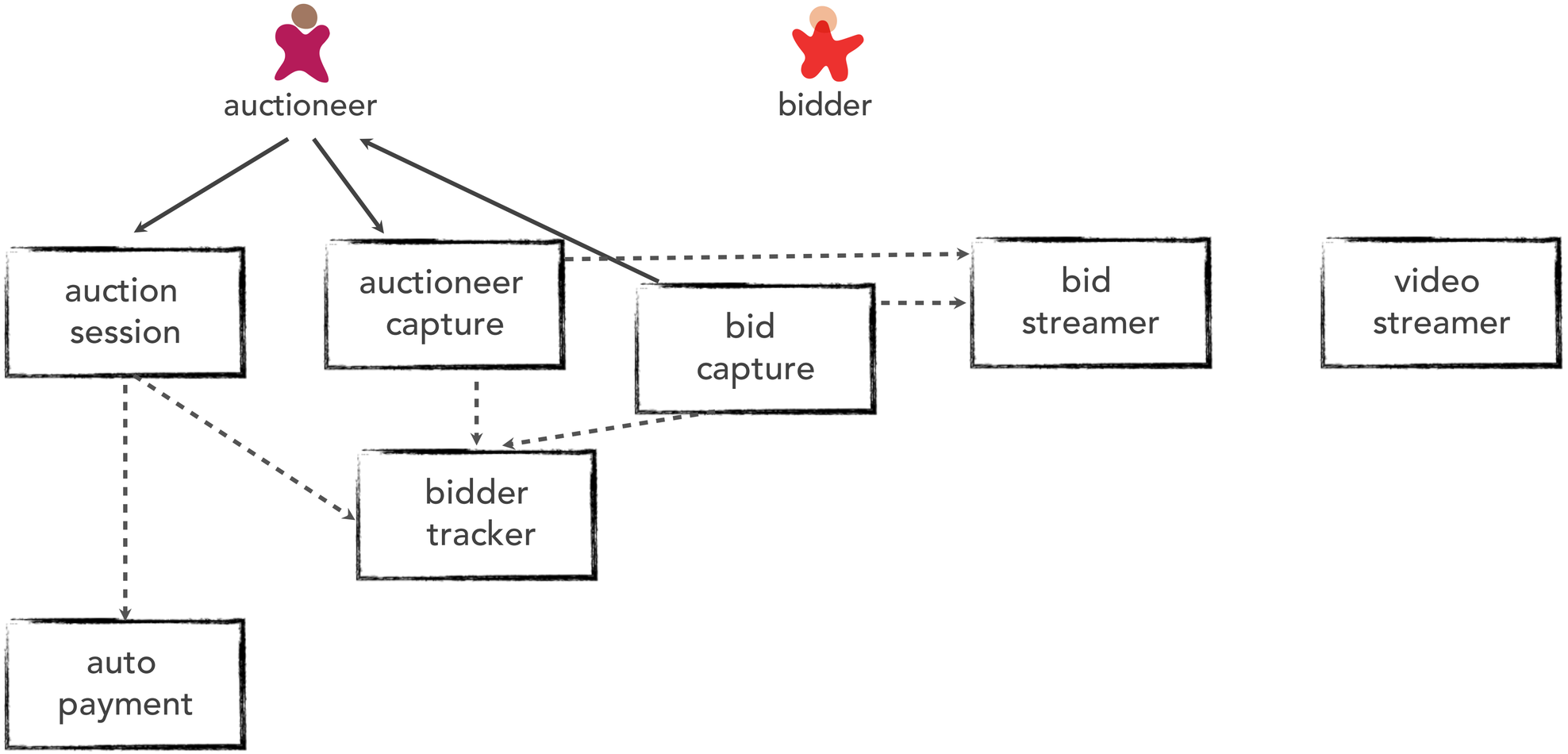
Your Architectural Kata is...

Going Going Gone!



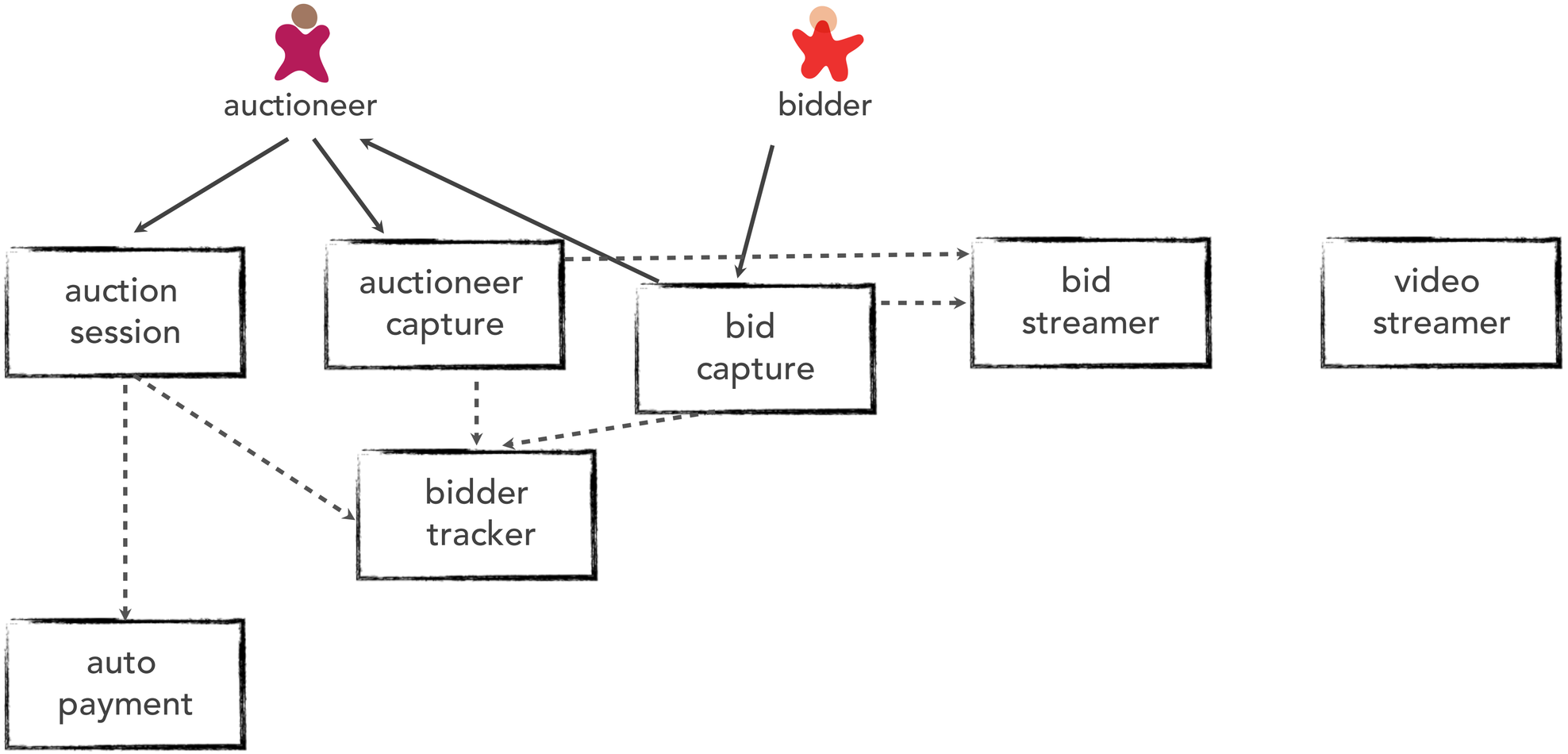
Your Architectural Kata is...

Going Going Gone!



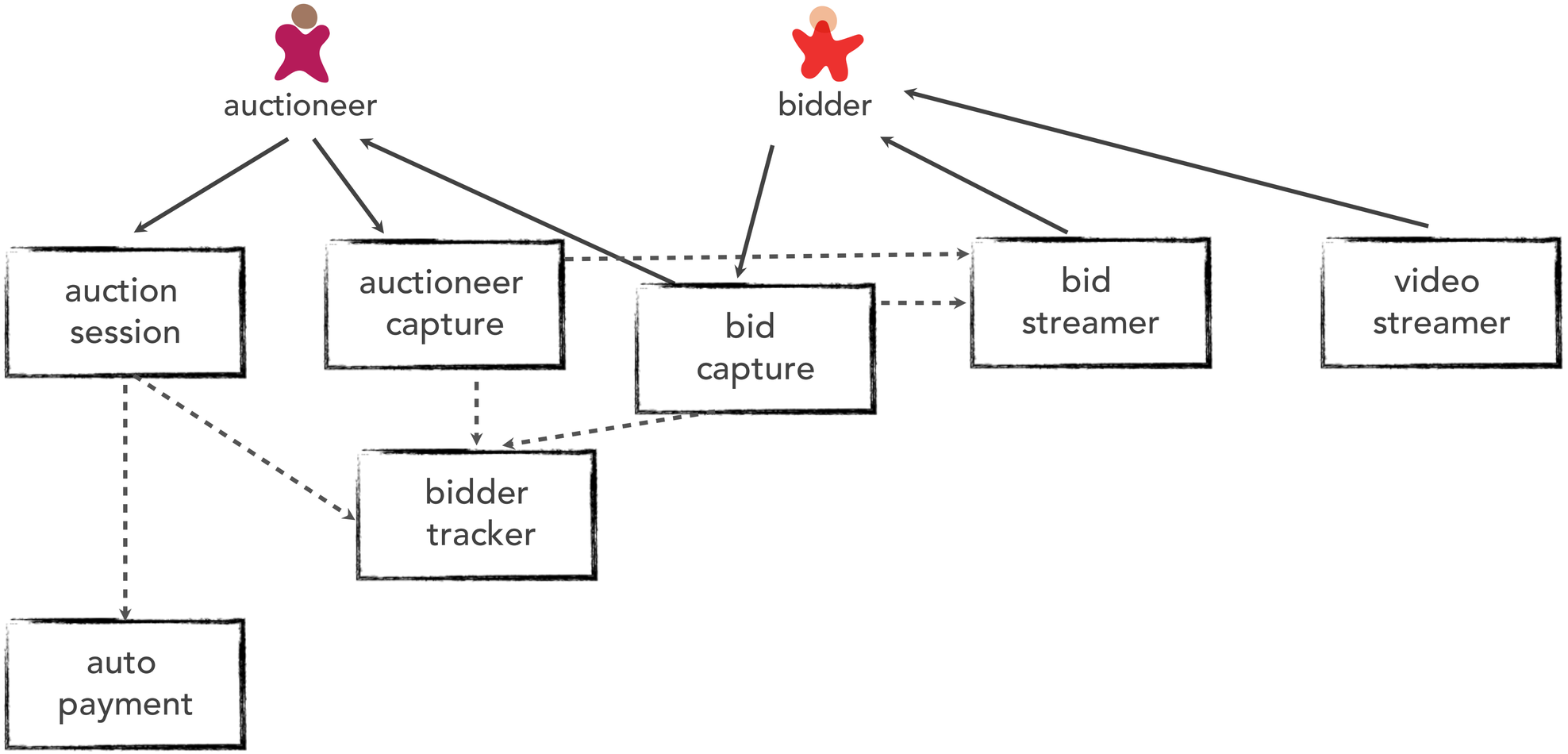
Your Architectural Kata is...

Going Going Gone!



Your Architectural Kata is...

Going Going Gone!



Your Architectural Kata is...

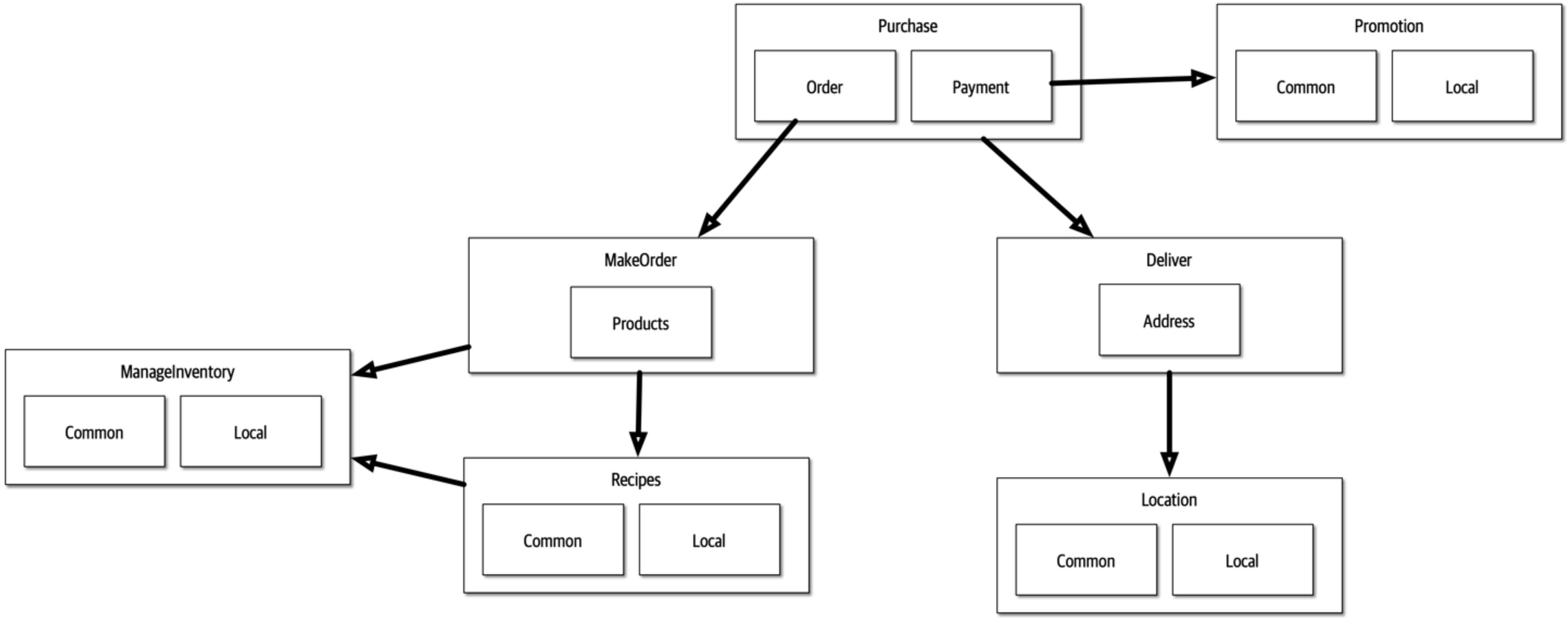
Silicon Sandwiches

A national sandwich shop wants to enable internet-ordering (in addition to their current call-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

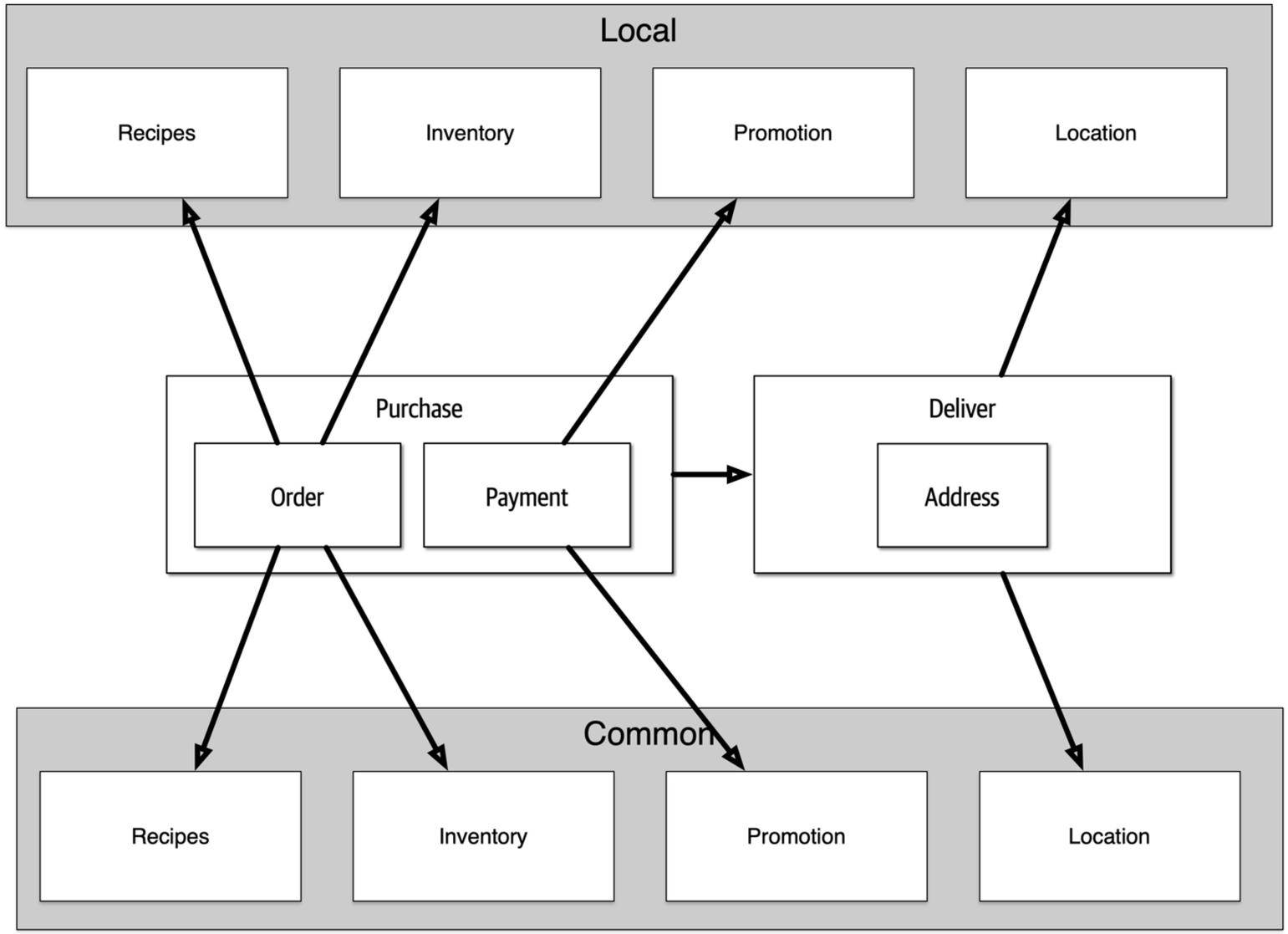
Your Architectural Kata is...

Silicon Sandwiches



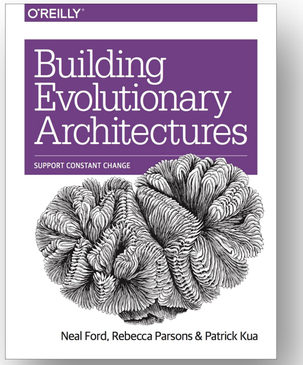
Your Architectural Kata is...

Silicon Sandwiches

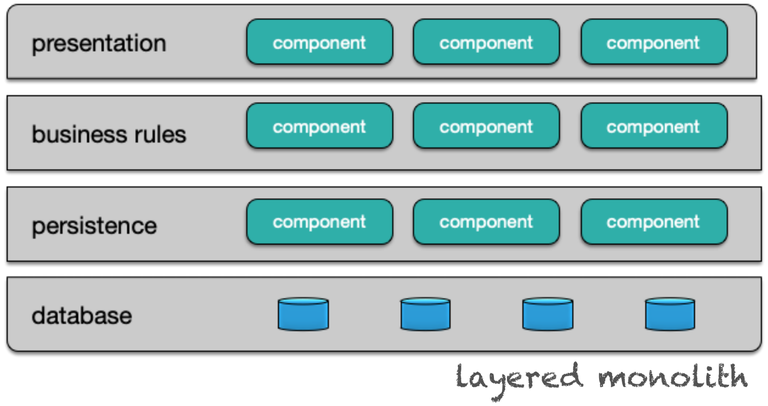


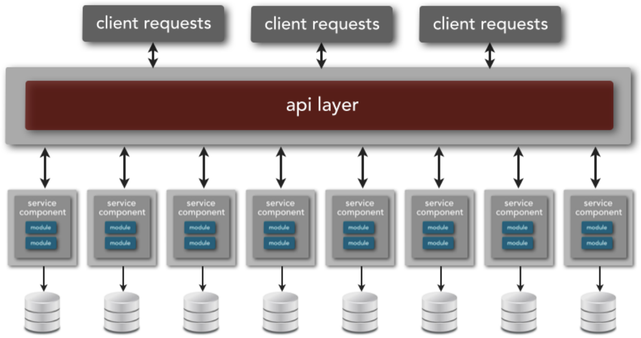
architectural quantum

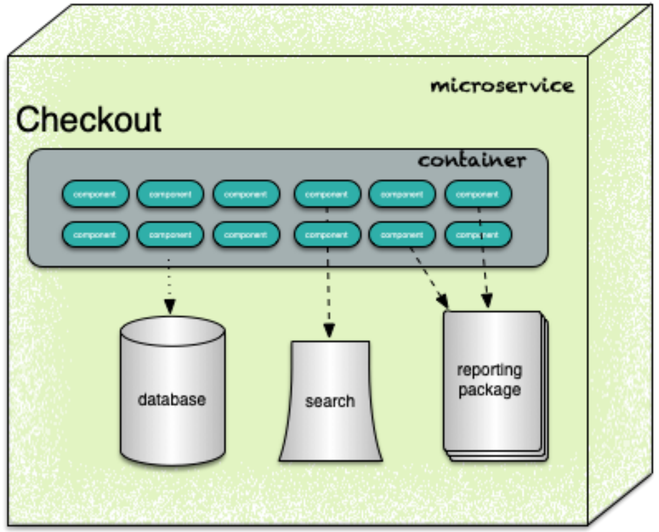
An *architectural quantum* is an independently deployable component with high functional cohesion and synchronous communication coupling.



architectural quantum

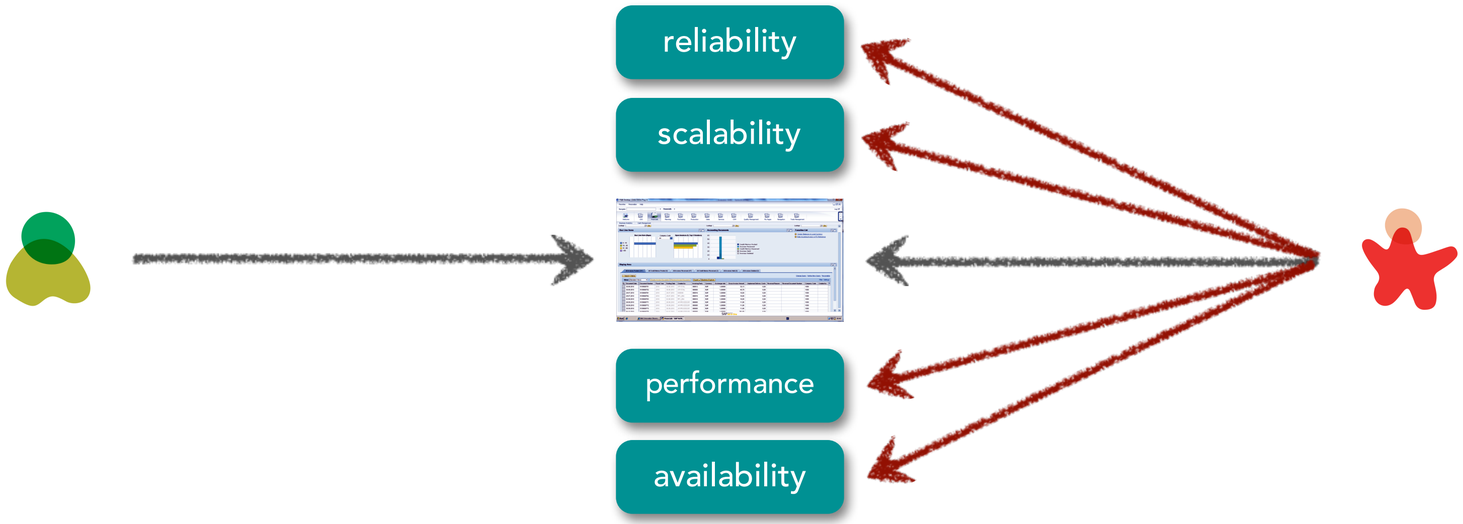






architectural quantum?

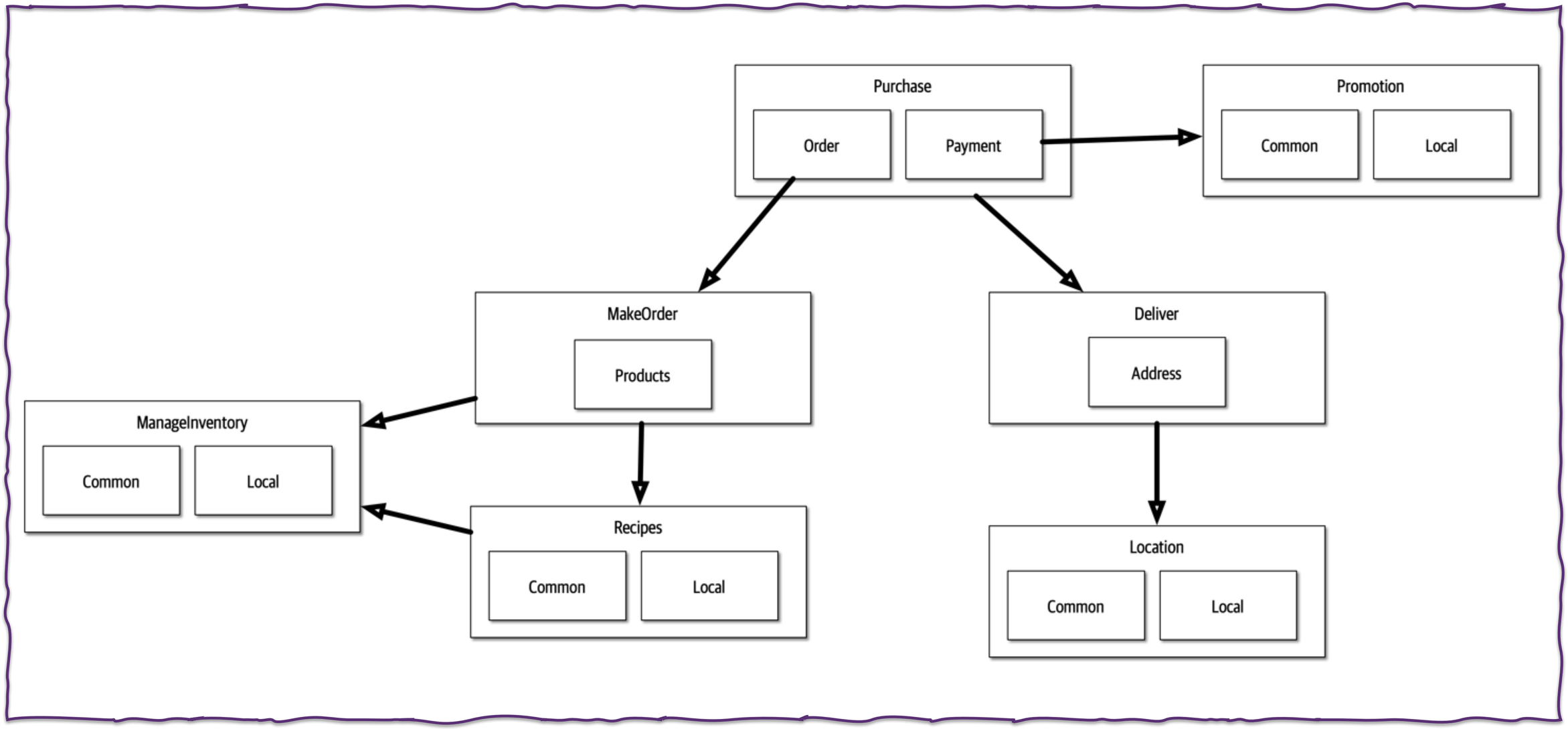
Architectural characteristics live at the quantum level.



Your Architectural Kata is...

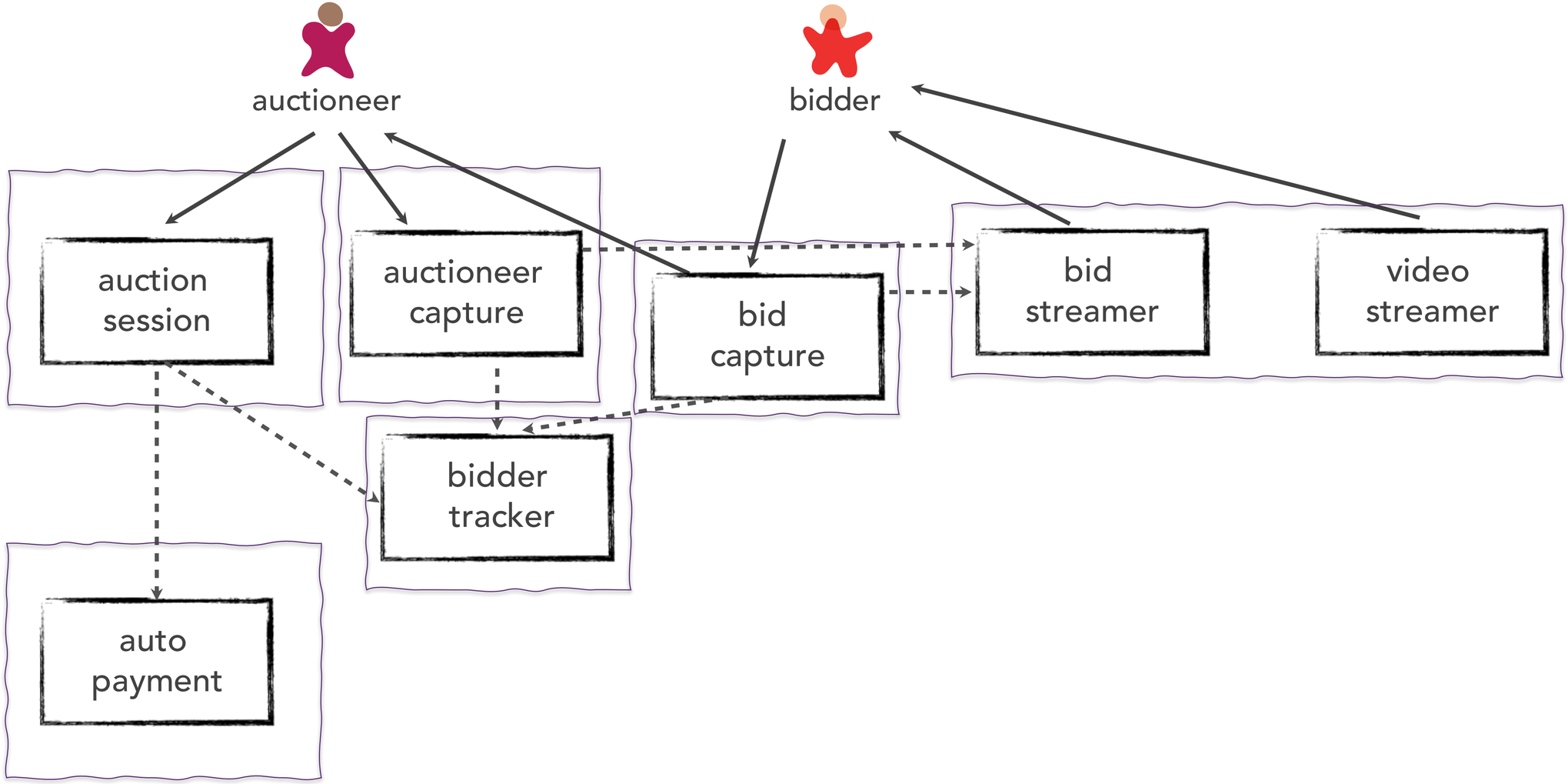
Silicon Sandwiches

quantum

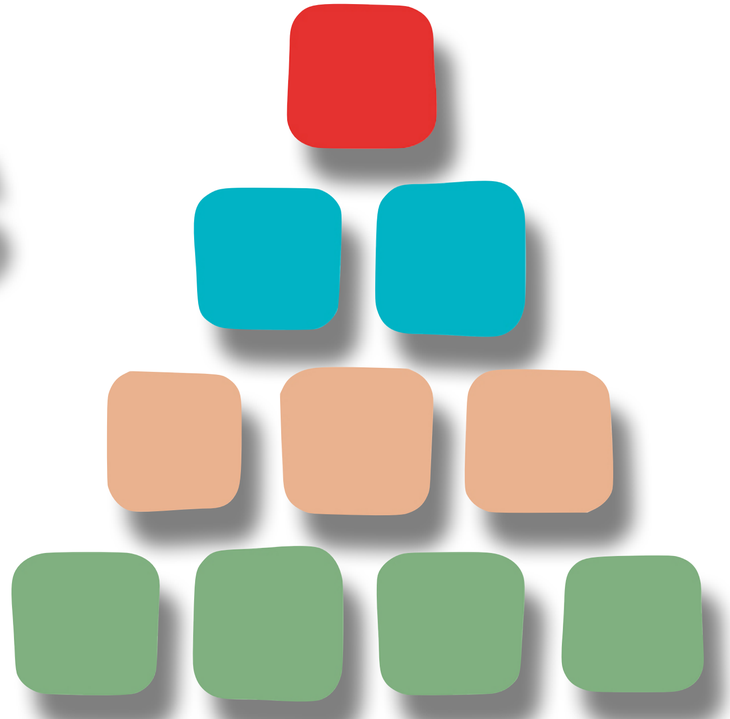


Your Architectural Kata is...

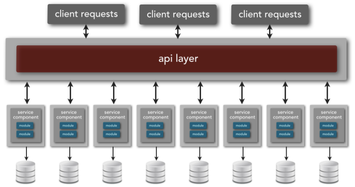
Going Going Gone!



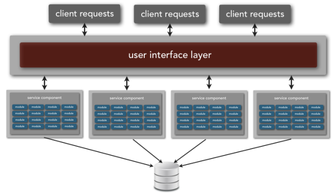
architecture patterns



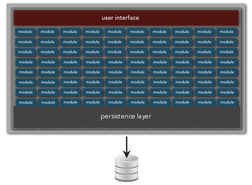
architecture patterns help define the basic characteristics and behavior of an application



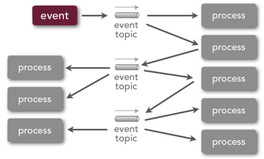
microservices architecture



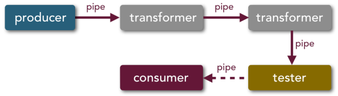
service-based architecture



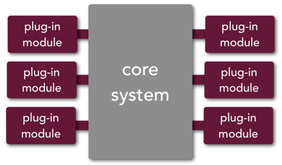
layered architecture



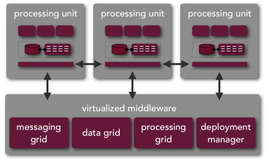
event-driven architecture



pipeline architecture

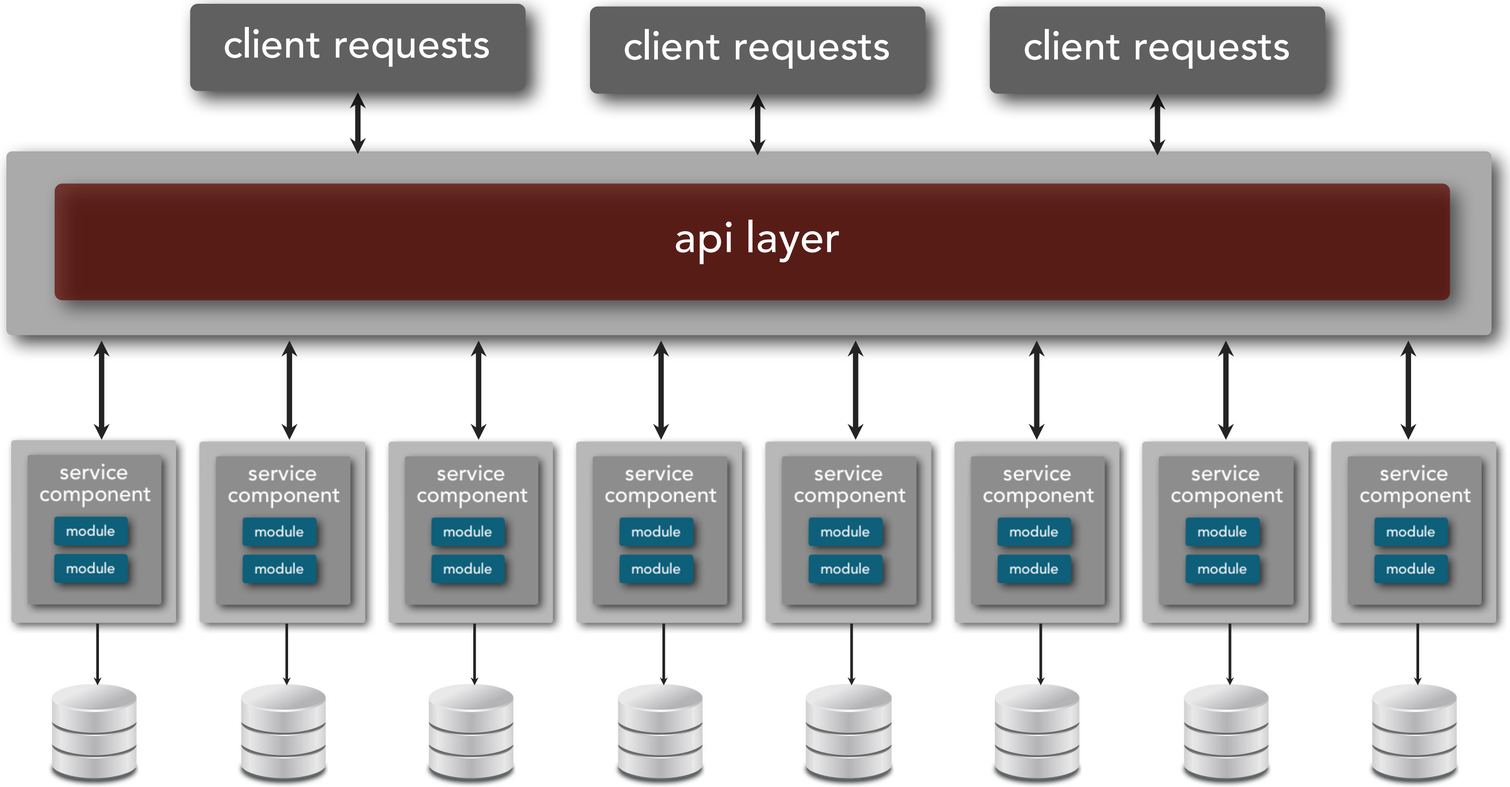


microkernel architecture



space-based architecture

microservices architecture



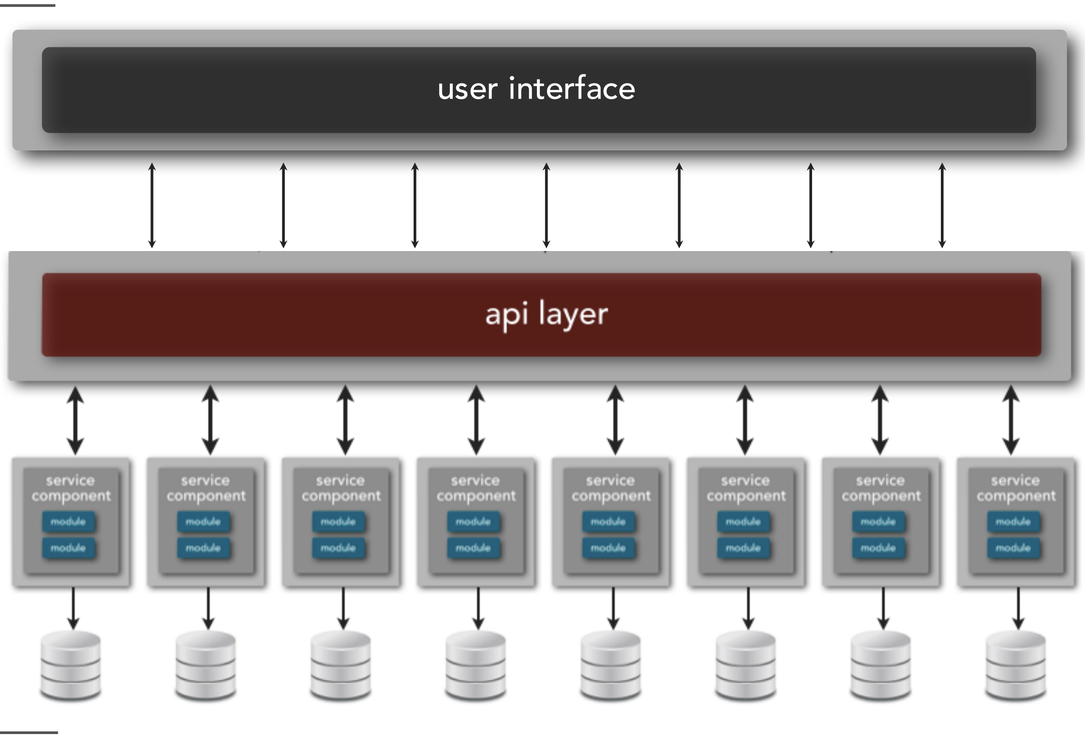
microservices is a label, not a description

microservices architecture

architectural quantum

monolithic user interface

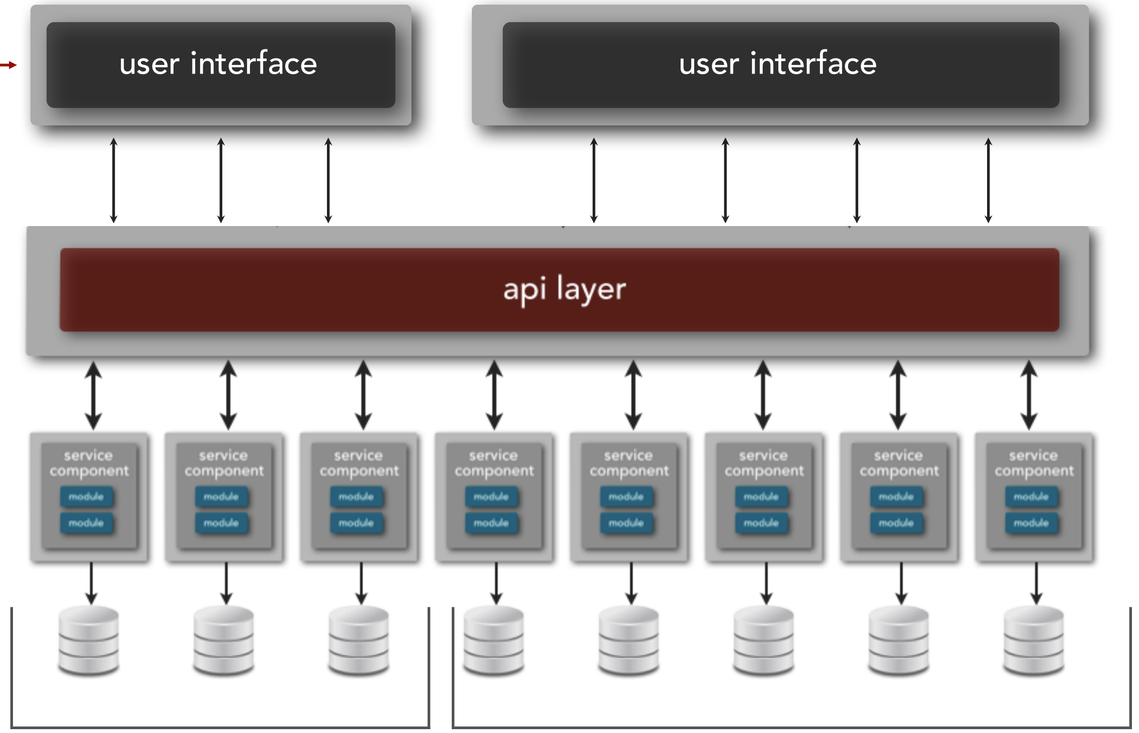
system level



microservices architecture

architectural quanta

partitioned user interface

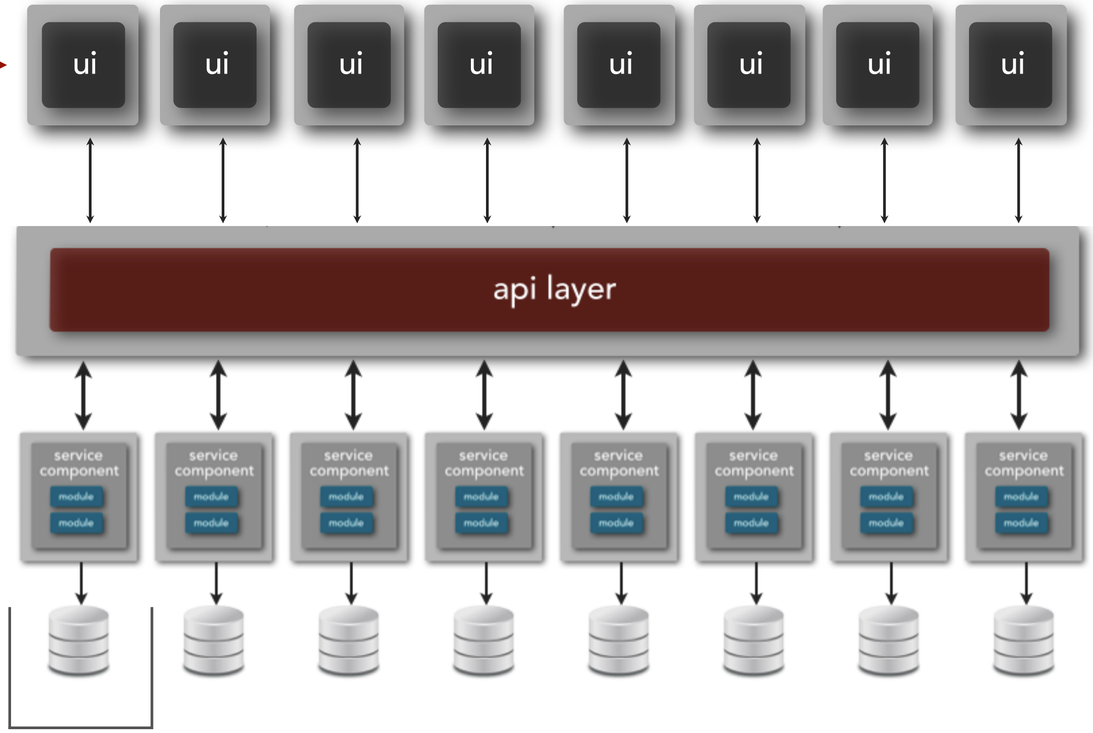


sub-system level

microservices architecture

architectural quanta

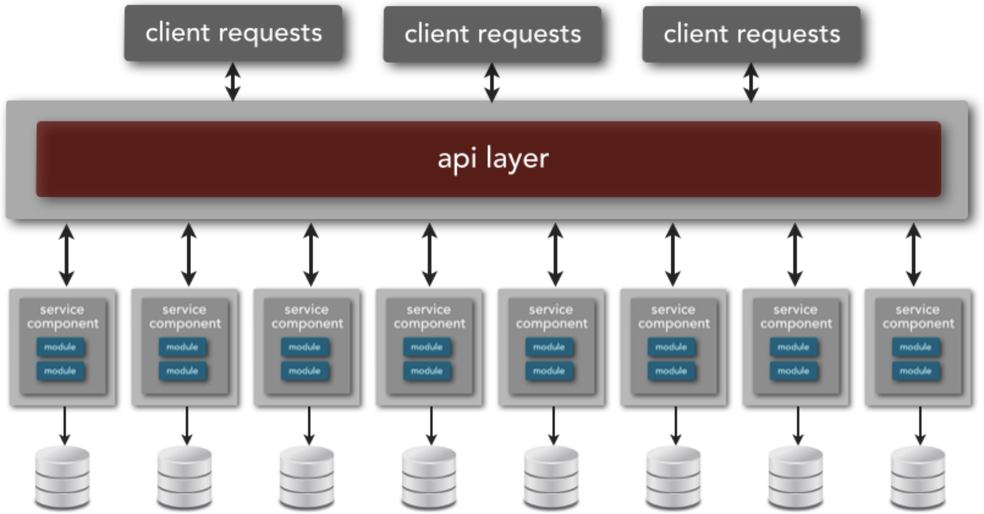
micro front-end



service level

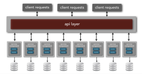
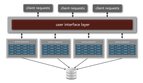
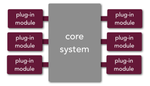
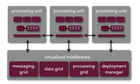
microservices architecture

drivers

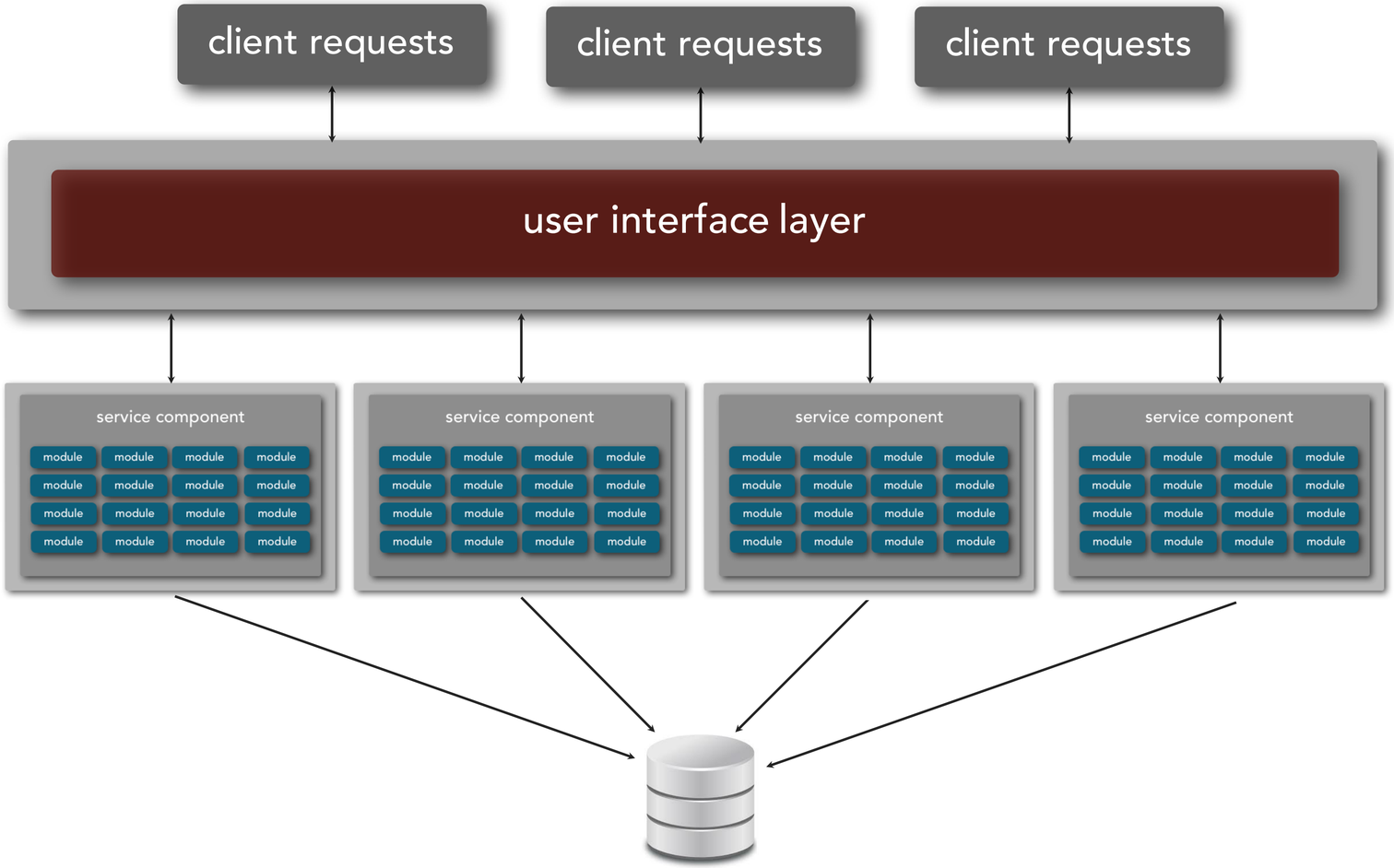


- ✓ modularity
- ✓ agility
- ✓ fault-tolerance
- ✓ scalability
- ✓ testability
- ✓ deployability
- ✓ evolvability

microservices architecture

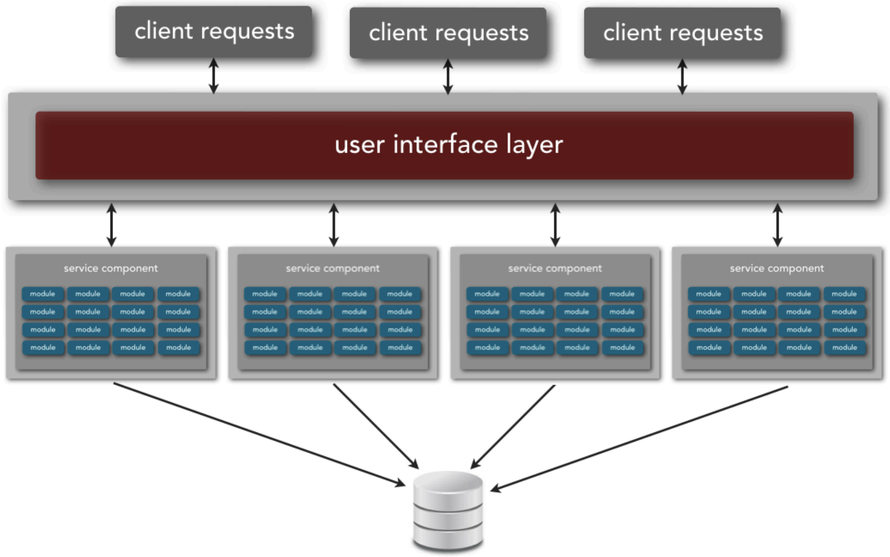
	agility	deployment	testability	performance	scalability	simplicity	cost
	👍👍👍	👍👍👍	👍👍👍	👎👎	👍👍👍	👎👎👎	$$$
							
							
							
							
							
							

service-based architecture



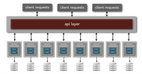
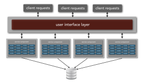
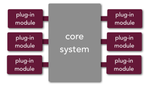
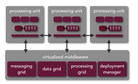
service-based architecture

drivers

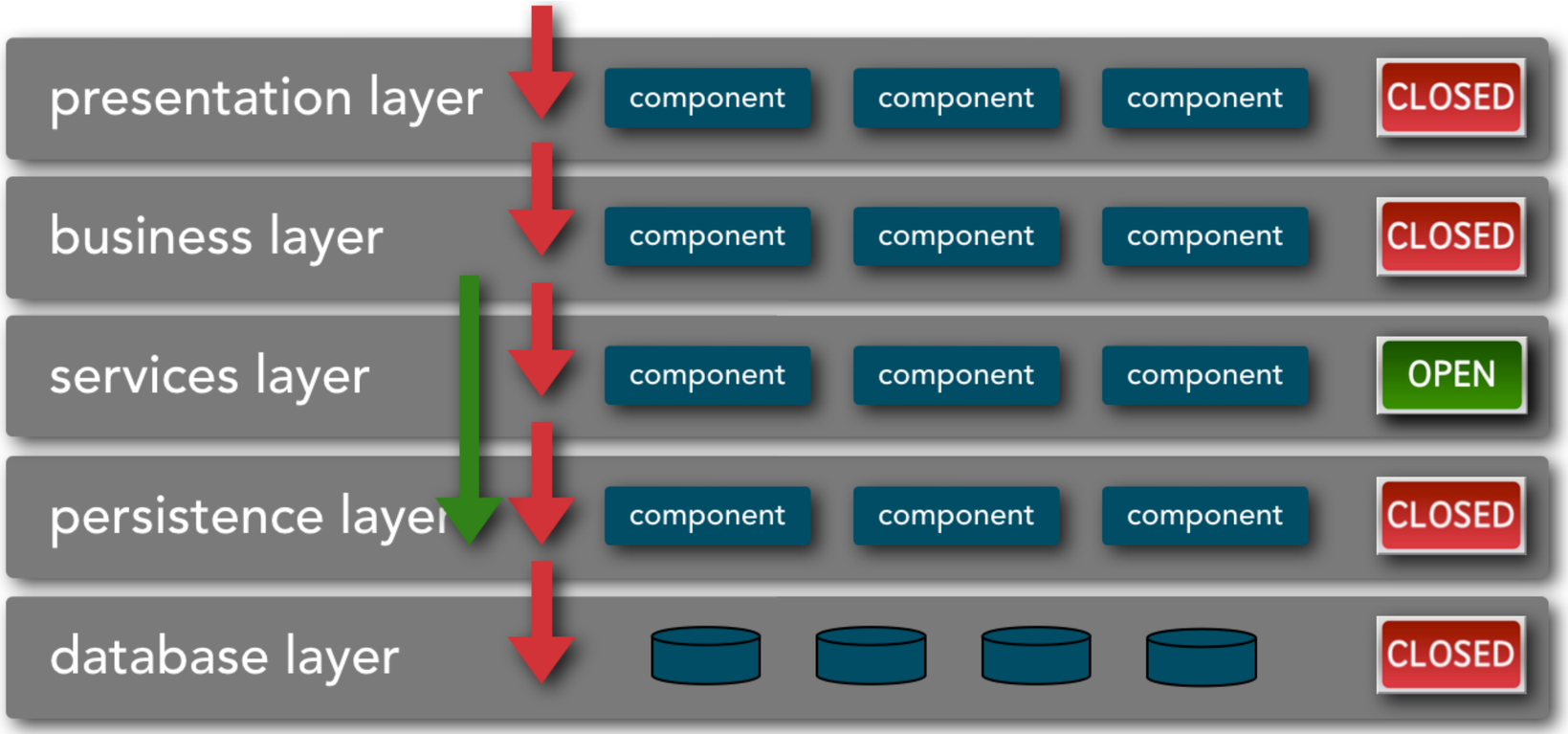


- ✓ modularity
- ✓ agility
- ✓ fault-tolerance
- ✓ scalability
- ✓ testability
- ✓ deployability
- ✓ simplicity / cost
- ✓ evolvability

service-based architecture

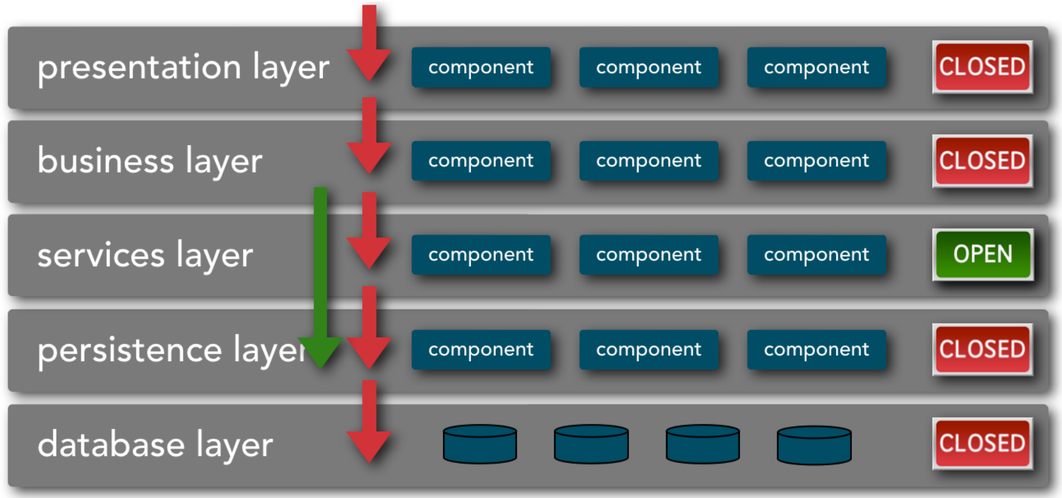
	agility	deployment	testability	performance	scalability	simplicity	cost
	👍👍👍	👍👍👍	👍👍👍	👎👎	👍👍👍	👎👎👎	$$$
	👍👍	👍👍	👍👍	👎	👍👍	👎	$$
							
							
							
							
							

layered architecture



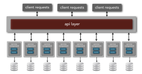
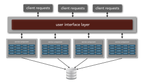
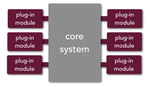
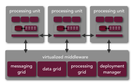
layered architecture

drivers

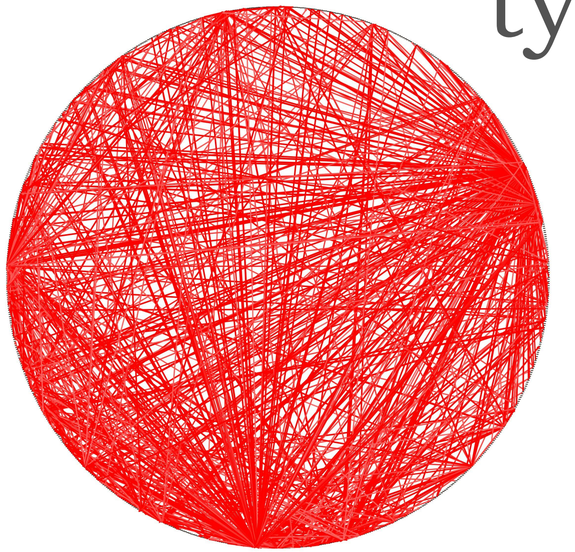


- ✓ simplicity
- ✓ cost

layered architecture

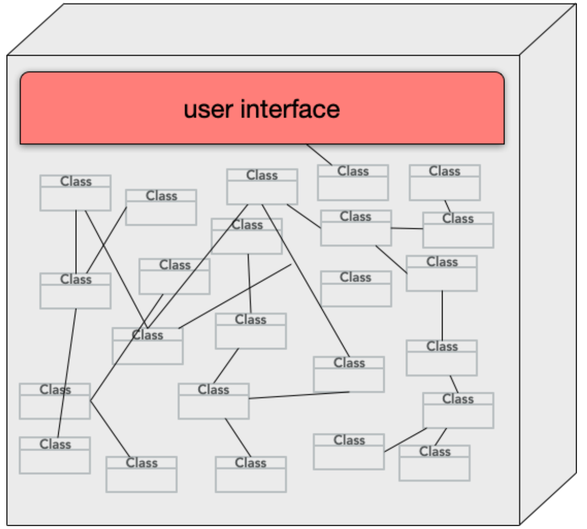
	agility	deployment	testability	performance	scalability	simplicity	cost
	👍👍👍	👍👍👍	👍👍👍	👎👎	👍👍👍	👎👎👎	$$$
	👍👍	👍👍	👍👍	👎	👍👍	👎	$$
	👎	👎	👍	👎	👎	👍👍👍	$
							
							
							
							

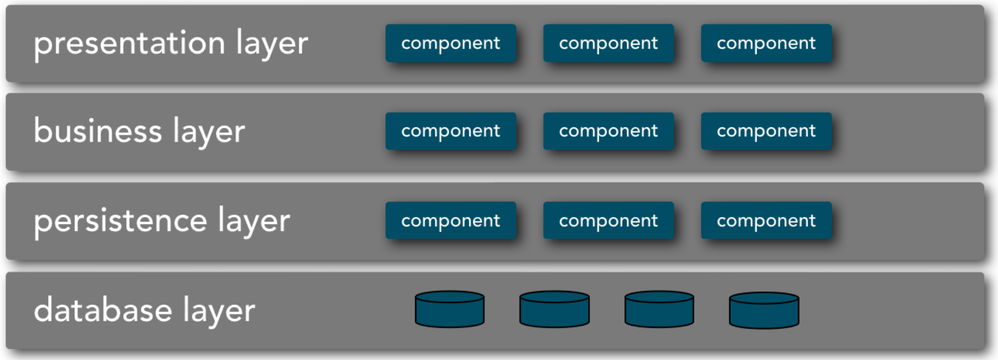
types of monoliths



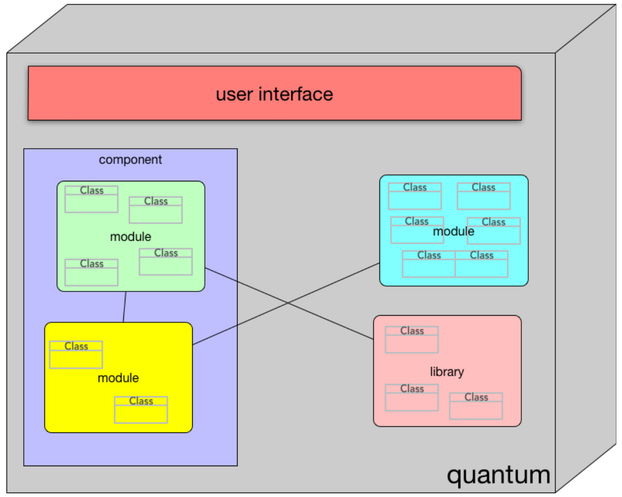
"big ball of mud"

unstructured monolith



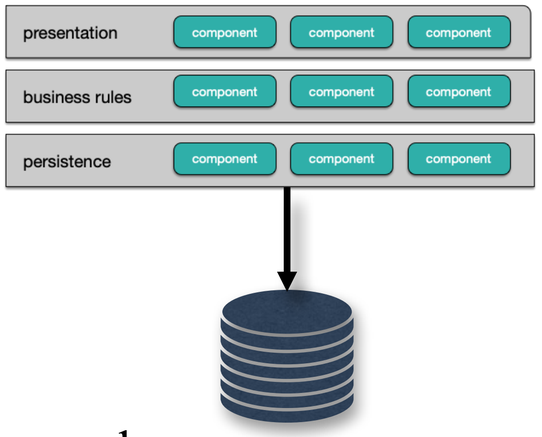


layered monolith

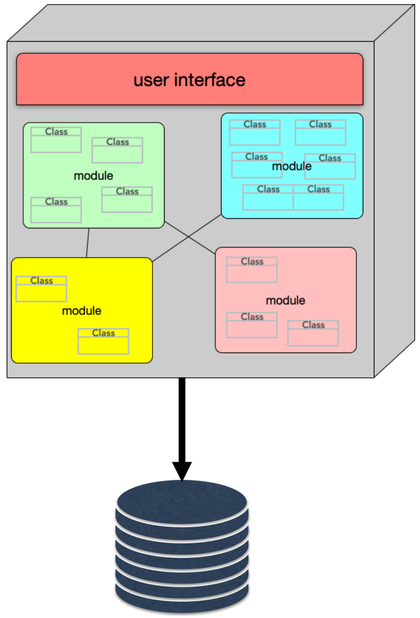


modular monolith

types of monoliths



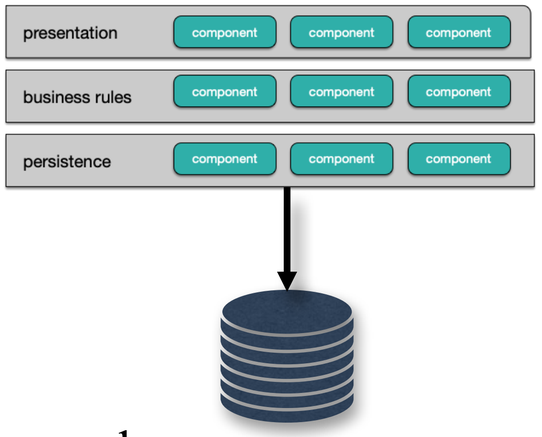
layered architecture



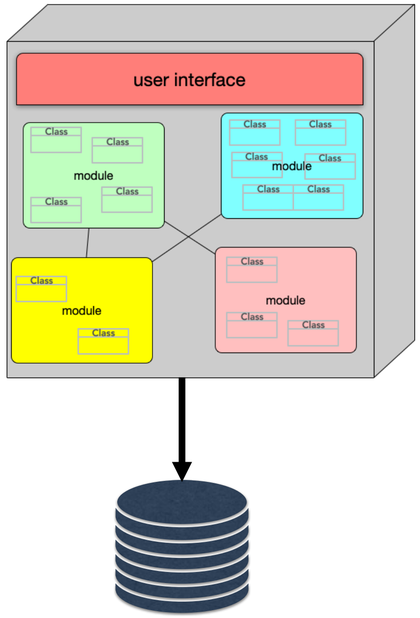
modular monolith

types of monoliths

technical partitioning



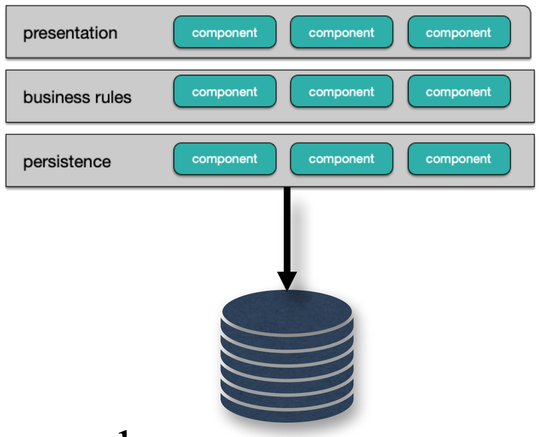
layered architecture



modular monolith

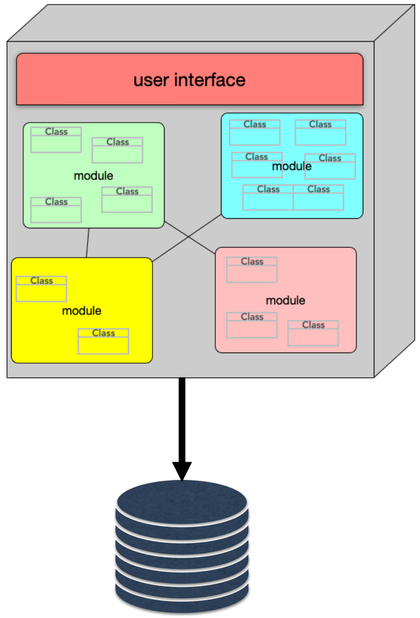
types of monoliths

technical partitioning



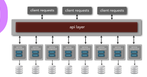
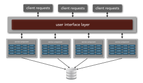
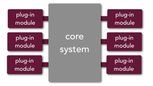
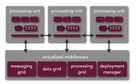
layered architecture

domain partitioning

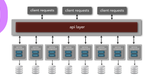
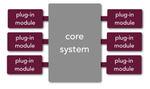
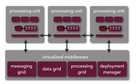


modular monolith

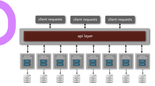
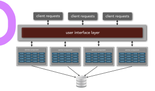
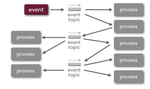
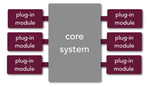
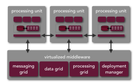
layered architecture

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

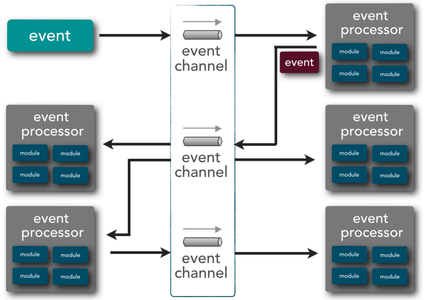
layered architecture

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

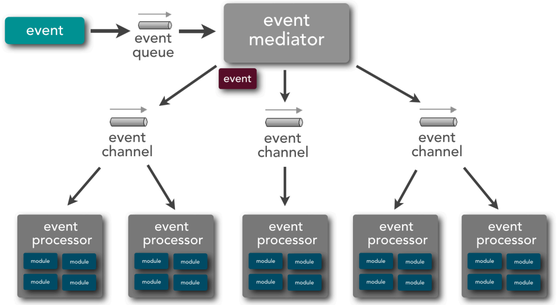
layered architecture

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

event-driven architecture



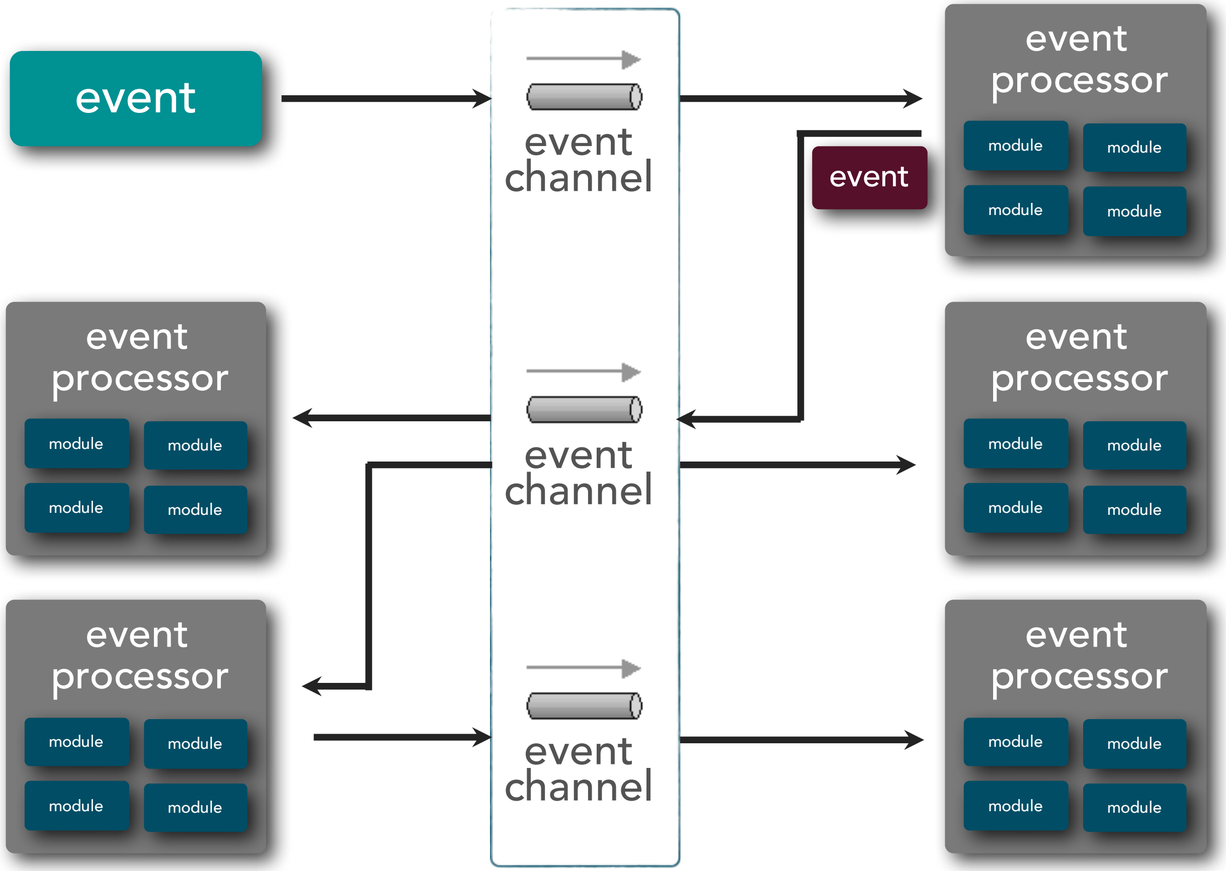
broker topology



mediator topology

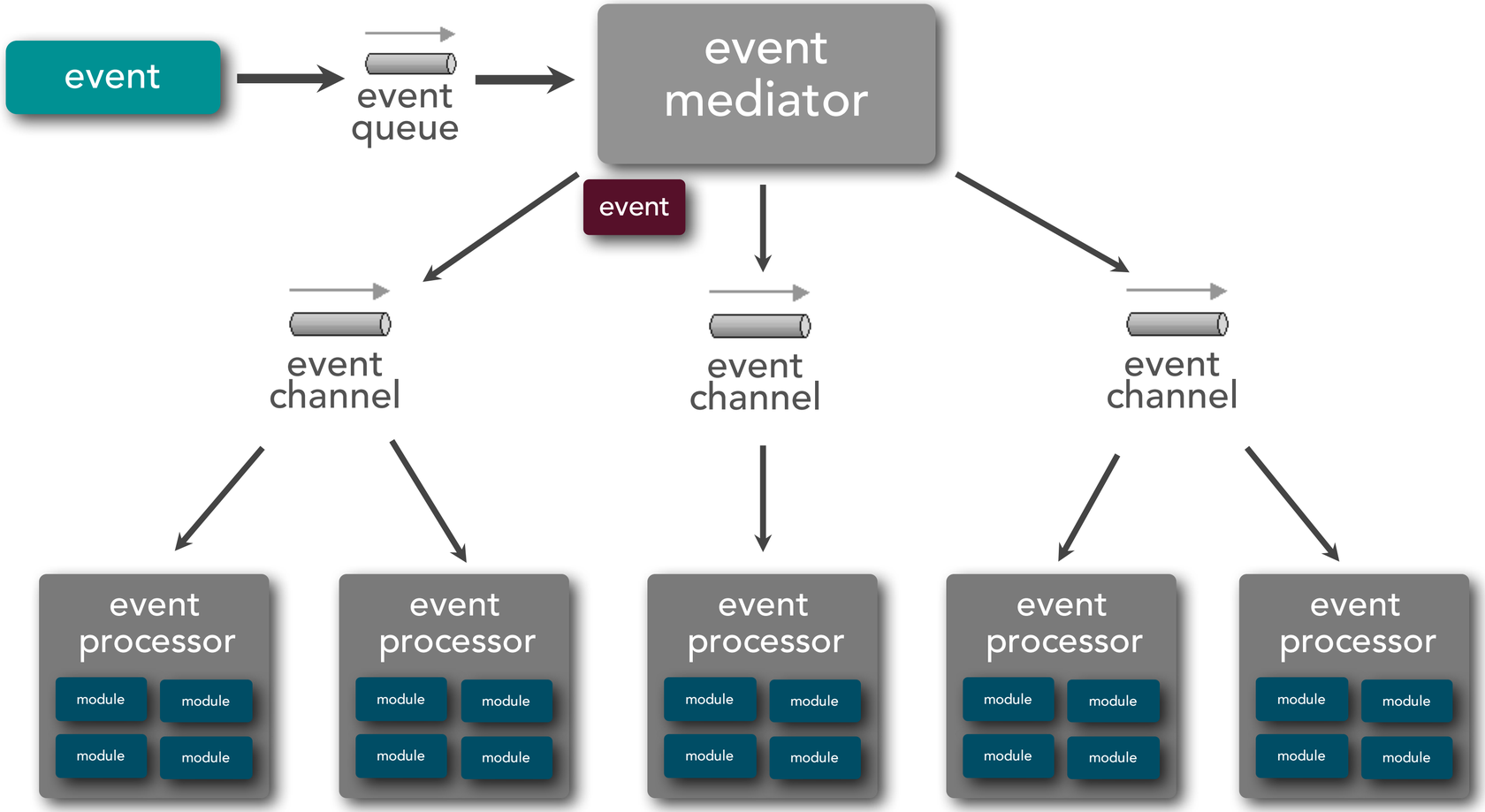
event-driven architecture

broker topology



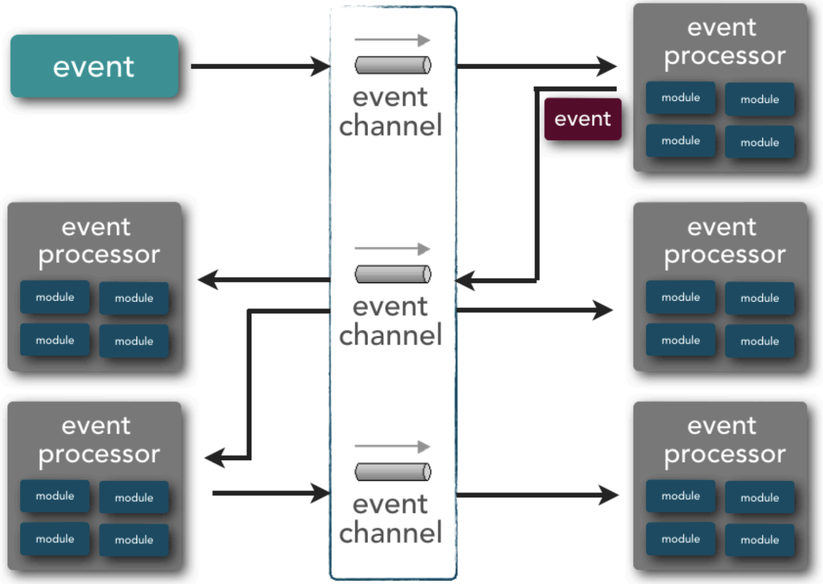
event-driven architecture

mediator topology



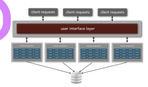
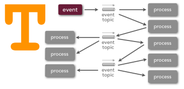
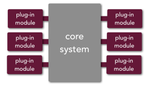
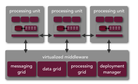
event-driven architecture

drivers



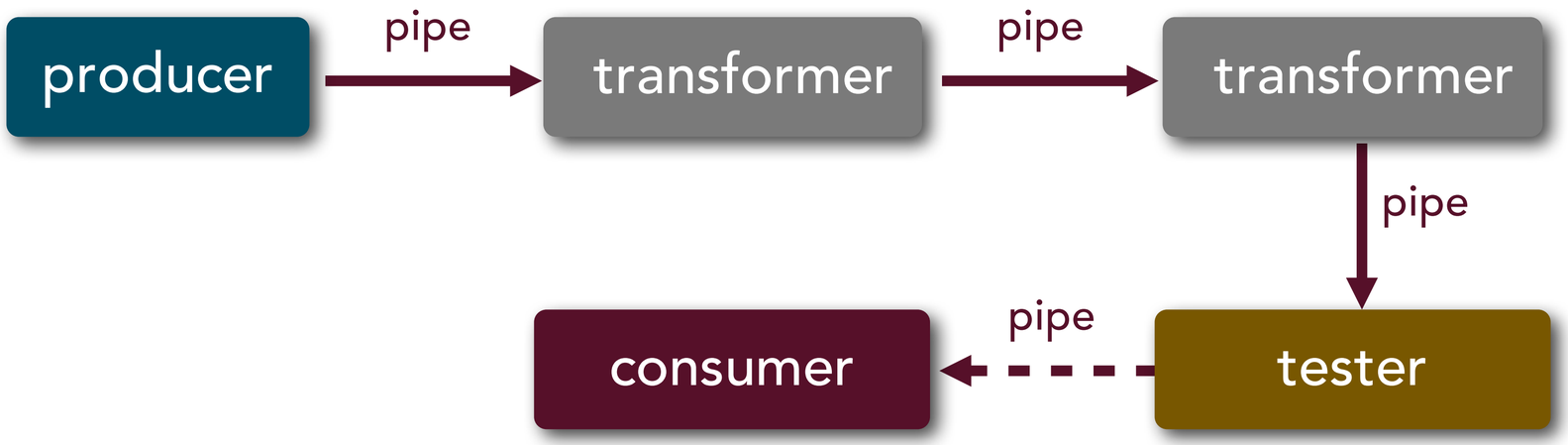
- ✓ modularity
- ✓ agility
- ✓ fault-tolerance
- ✓ scalability
- ✓ performance
- ✓ elasticity
- ✓ adaptability
- ✓ evolvability

event-driven architecture

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

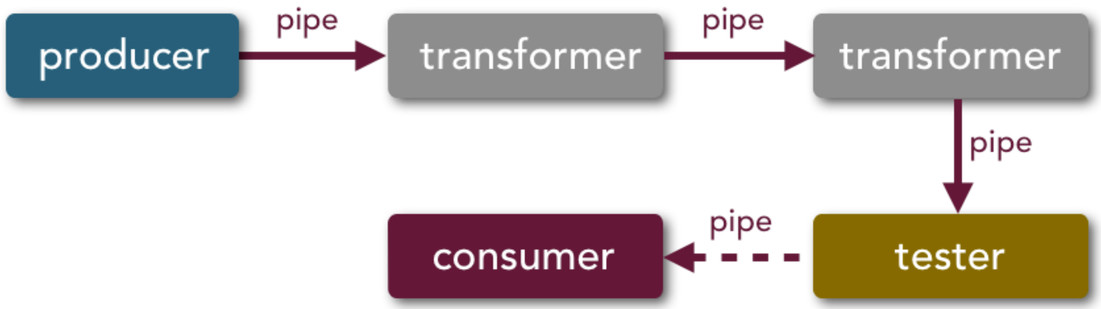
pipeline architecture

a.k.a. pipes and filters architecture



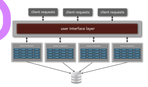
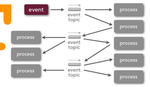
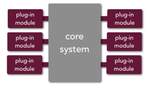
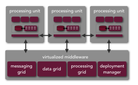
pipeline architecture

drivers

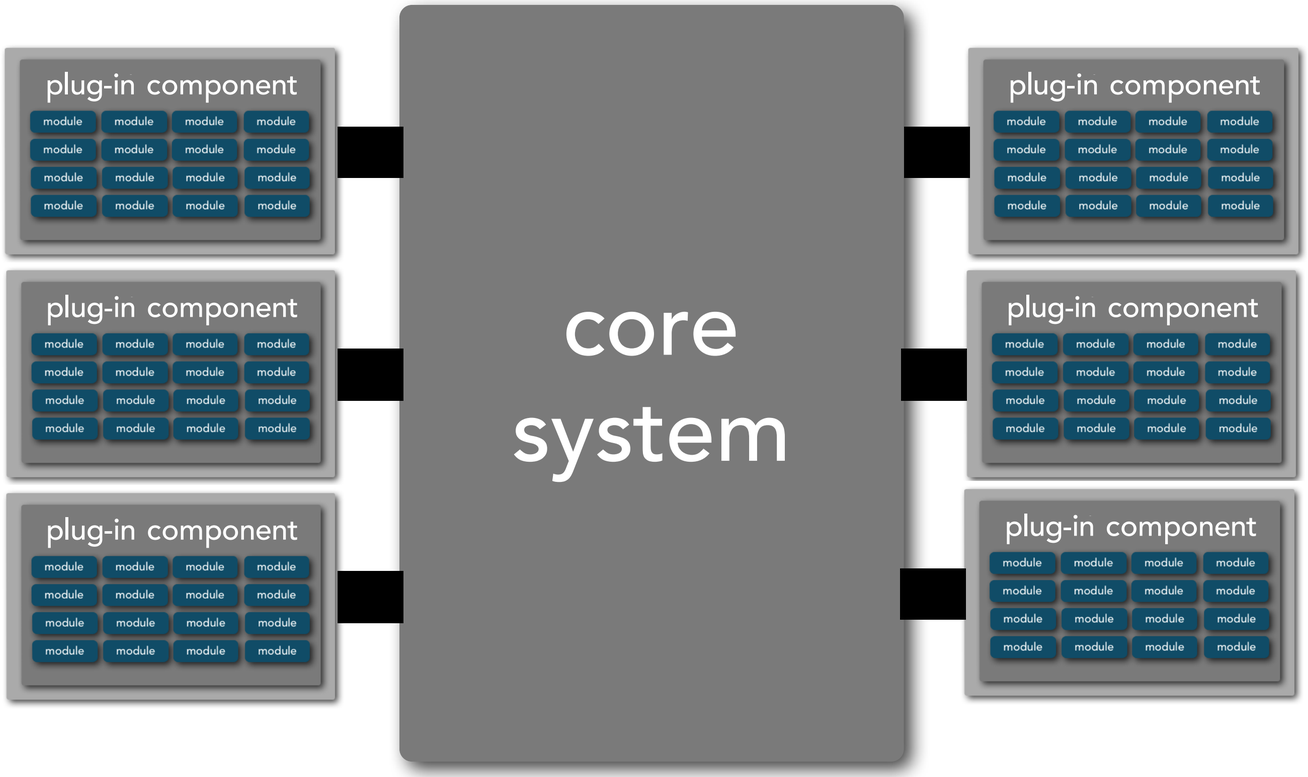


- ✓ modularity
- ✓ simplicity
- ✓ cost
- ✓ agility
- ✓ adaptability
- ✓ evolvability

pipeline architecture

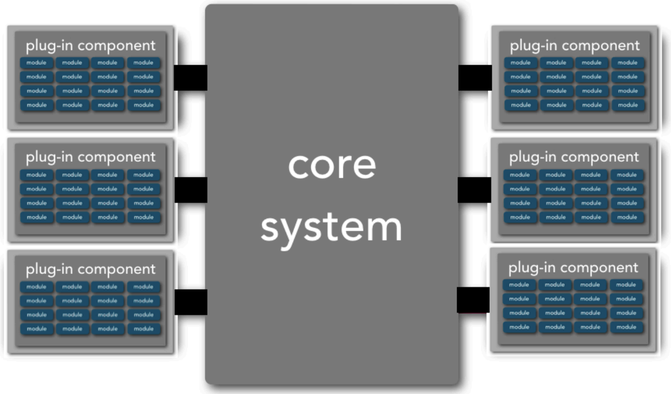
	agility	deployment	testability	performance	scalability	simplicity	cost
D 	👍👍👍	👍👍👍	👍👍👍	👎👎	👍👍👍	👎👎👎	$$$
D 	👍👍	👍👍	👍👍	👎	👍👍	👎	$$
T 	👎	👎	👍	👎	👎	👍👍👍	$
T 	👍👍	👍👍	👎👎	👍👍👍	👍👍	👎👎	$$$
T 	👍	👎	👍	👎	👎	👍👍	$
							
							

microkernel architecture



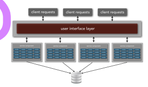
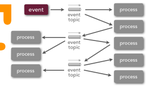
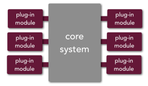
microkernel architecture

drivers

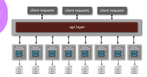
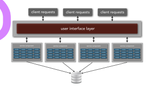
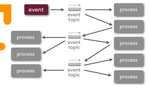
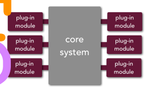


- ✓ modularity
- ✓ simplicity
- ✓ cost
- ✓ adaptability
- ✓ evolvability
- ✓ agility
- ✓ testability
- ✓ deployability

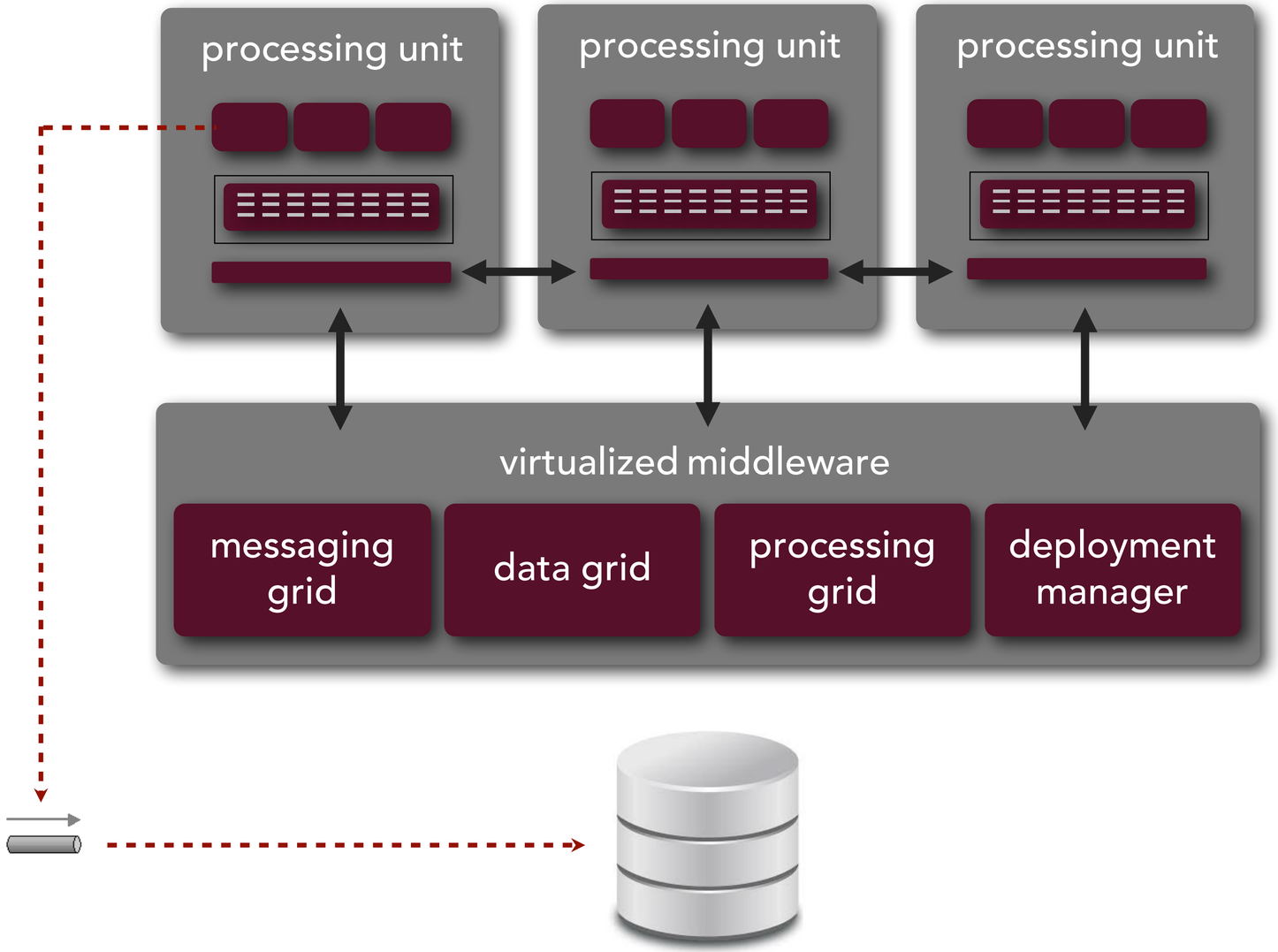
microkernel architecture

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

microkernel architecture

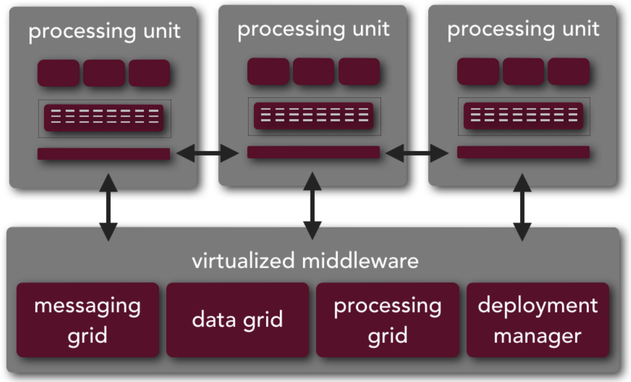
	agility	deployment	testability	performance	scalability	simplicity	cost
D 	👍👍👍	👍👍👍	👍👍👍	👎👎	👍👍👍	👎👎👎	$$$
D 	👍👍	👍👍	👍👍	👎	👍👍	👎	$$
T 	👎	👎	👍	👎	👎	👍👍👍	$
T 	👍👍	👍👍	👎👎	👍👍👍	👍👍	👎👎	$$$
T 	👍	👎	👍	👎	👎	👍👍	$
T D 	👍	👍	👍	👎	👎	👍	$$
							

space-based architecture



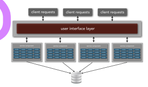
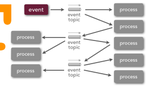
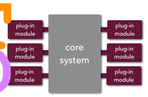
space-based architecture

drivers



- ✓ scalability
- ✓ elasticity
- ✓ performance

space-based architecture

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

architecture katas

identifying architecture patterns

Your Architectural Kata is...

Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - video stream of the action after the fact
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotionals/specials
  - offer local daily promotionals/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.

Your Architectural Kata is...

Going Going Gone!

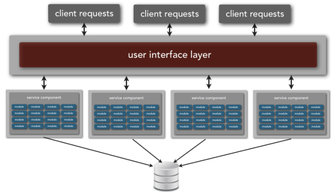
An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

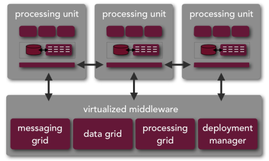
availability reliability performance scalability elasticity (security)

Your Architectural Kata is...

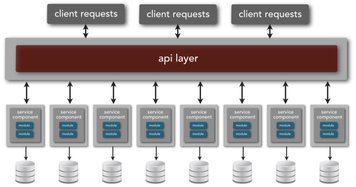
Going Going Gone!



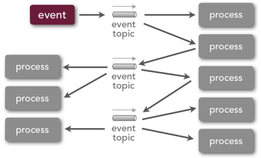
service-based architecture



space-based architecture



microservices architecture



event-driven architecture

performance

availability

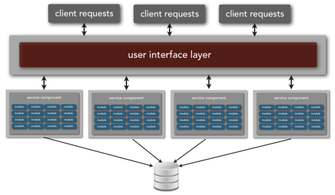
reliability

scalability

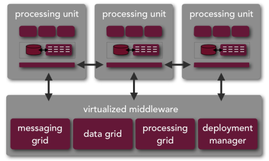
elasticity

Your Architectural Kata is...

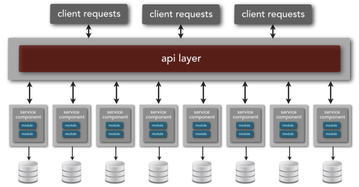
Going Going Gone!



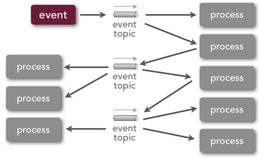
service-based architecture



space-based architecture



microservices architecture

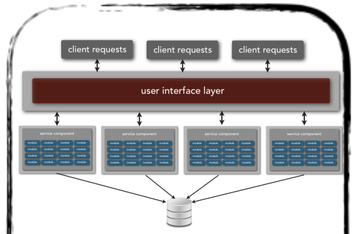


event-driven architecture

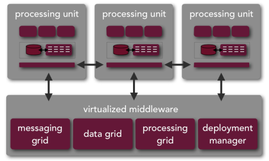
	service-based architecture	space-based architecture	microservices architecture	event-driven architecture
performance	👎	👍👍👍	👎👎	👍👍👍
availability	👍	👍	👍👍👍	👍
reliability	👍	—	—	👎
scalability	👍👍	👍👍👍	👍👍👍	👍👍
elasticity	👎	👍👍👍	👍	👍👍

Your Architectural Kata is...

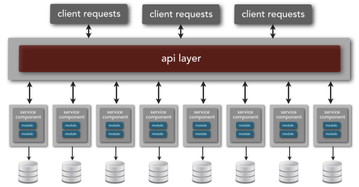
Going Going Gone!



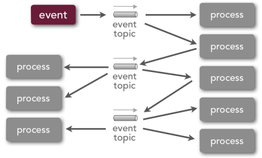
service-based architecture



space-based architecture



microservices architecture



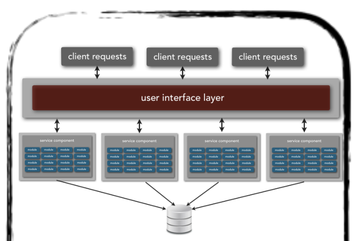
event-driven architecture

	service-based architecture	space-based architecture	microservices architecture	event-driven architecture
performance	👎	👍👍👍	👎👎	👍👍👍
availability	👍	👍	👍👍👍	👍
reliability	👍	—	—	👎
scalability	👍👍	👍👍👍	👍👍👍	👍👍
elasticity	👎	👍👍👍	👍	👍👍

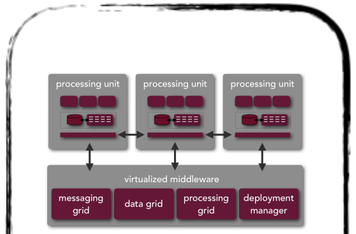
option 1

Your Architectural Kata is...

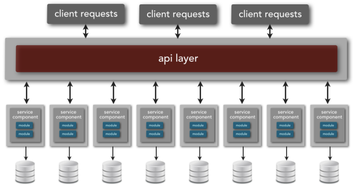
Going Going Gone!



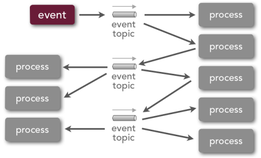
service-based architecture



space-based architecture



microservices architecture



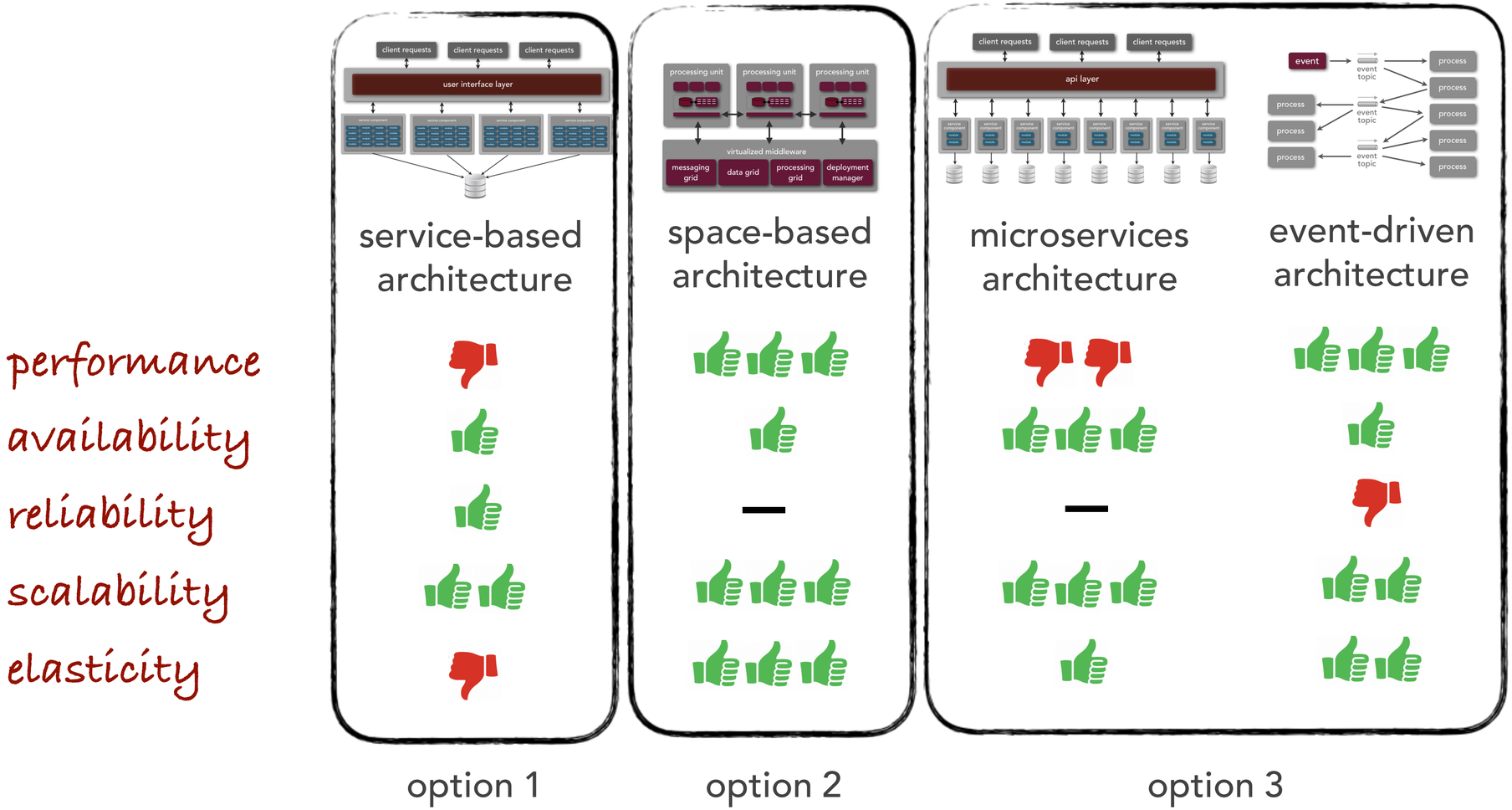
event-driven architecture

	service-based architecture	space-based architecture	microservices architecture	event-driven architecture
performance	👎	👍👍👍	👎👎	👍👍👍
availability	👍	👍	👍👍👍	👍
reliability	👍	—	—	👎
scalability	👍👍	👍👍👍	👍👍👍	👍👍
elasticity	👎	👍👍👍	👍	👍👍

option 1 option 2

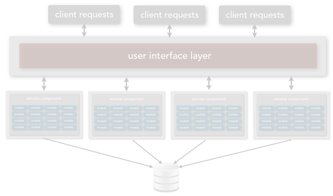
Your Architectural Kata is...

Going Going Gone!

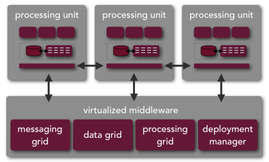


Your Architectural Kata is...

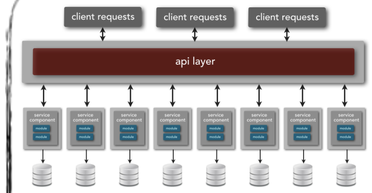
Going Going Gone!



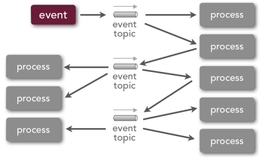
service-based architecture



space-based architecture



microservices architecture



event-driven architecture

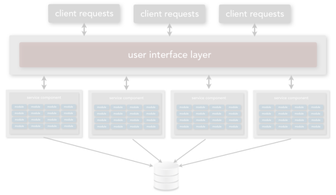
	service-based architecture	space-based architecture	microservices architecture	event-driven architecture
performance	👎	👍👍👍	👎👎	👍👍👍
availability	👍	👍	👍👍👍	👍
reliability	👍	—	—	👎
scalability	👍👍👍	👍👍👍	👍👍👍	👍👍
elasticity	👎	👍👍👍	👍	👍👍

option 2

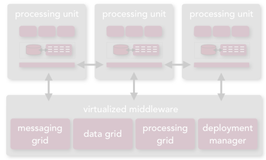
option 3

Your Architectural Kata is...

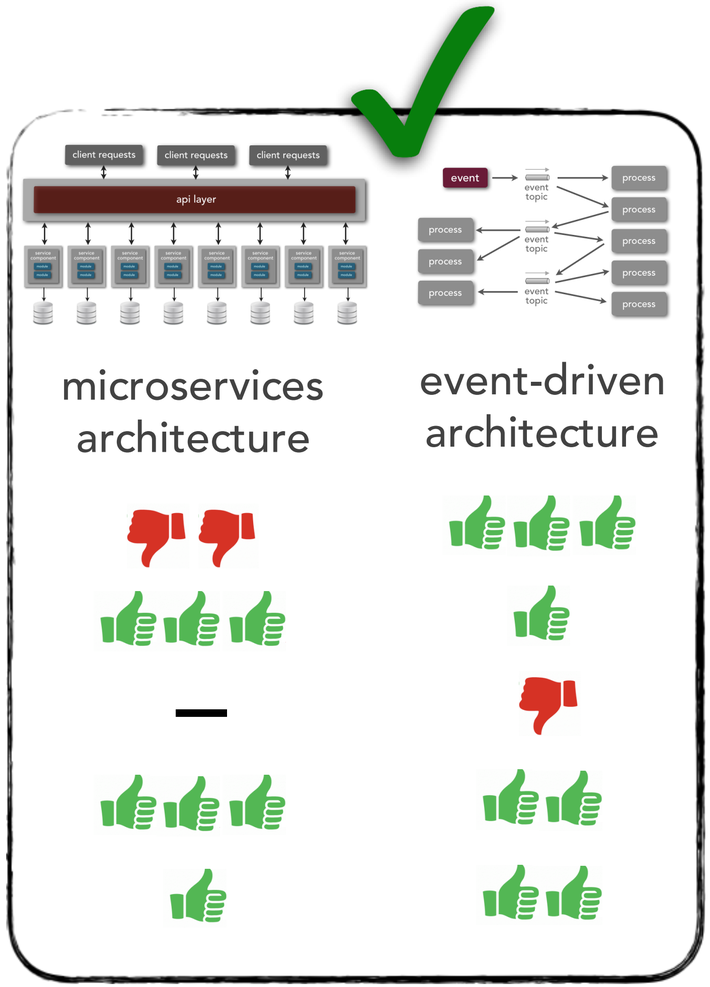
Going Going Gone!



service-based architecture



space-based architecture



microservices architecture

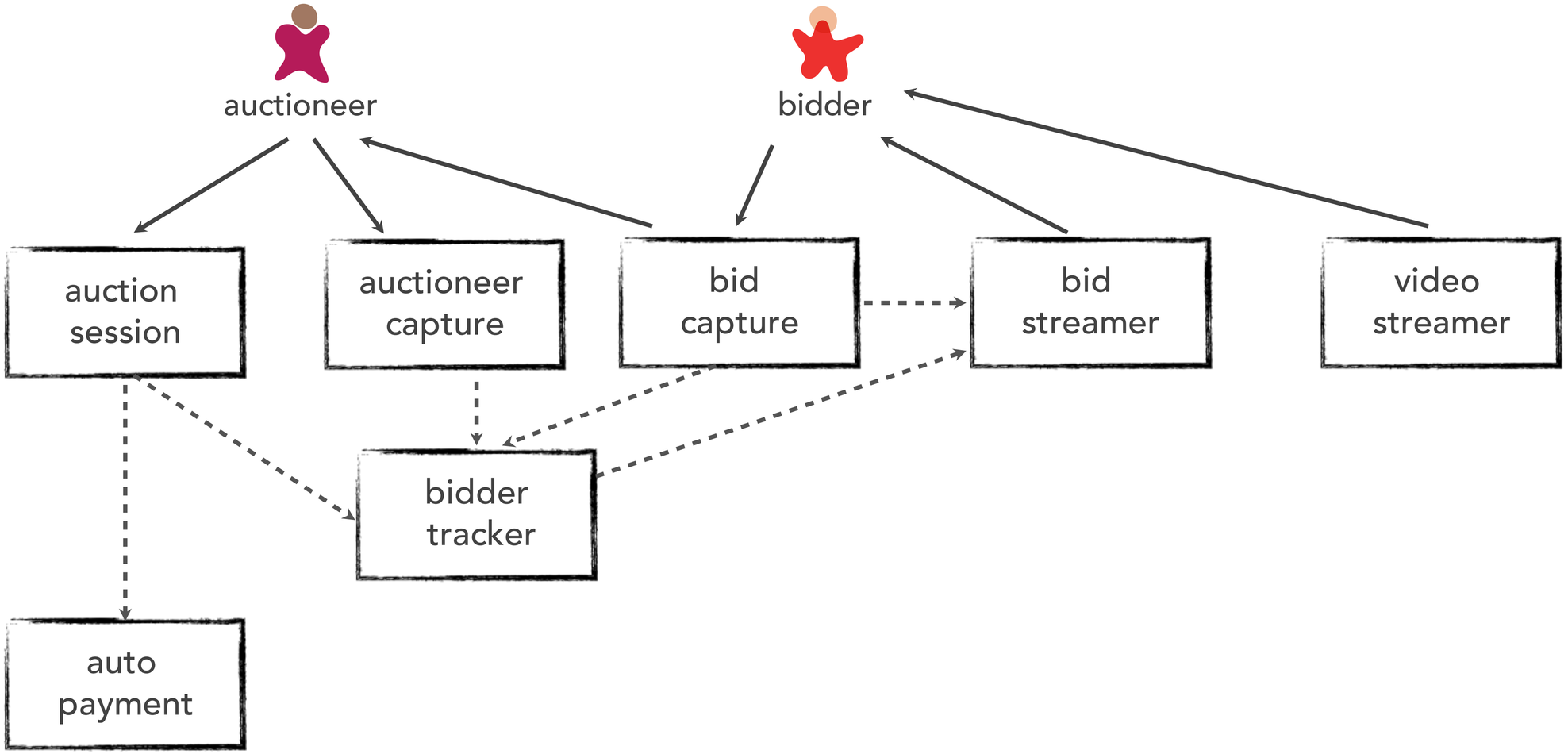
event-driven architecture

	service-based architecture	space-based architecture	microservices architecture	event-driven architecture
performance	👎	👍👍👍	👎👎	👍👍👍
availability	👍	👍	👍👍👍	👍
reliability	👍	—	—	👎
scalability	👍👍	👍👍👍	👍👍👍	👍👍
elasticity	👎	👍👍👍	👍	👍👍

option 3

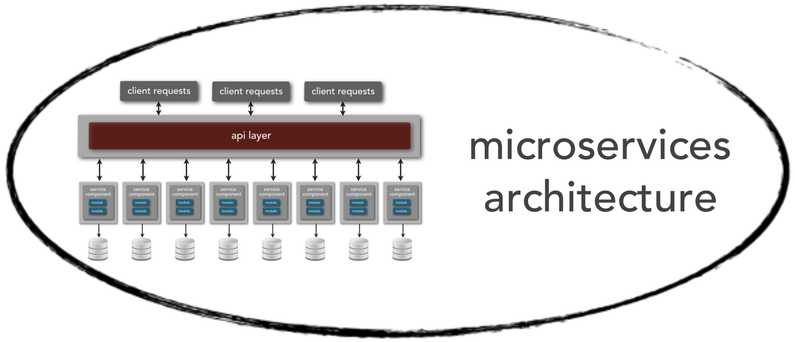
Your Architectural Kata is...

Going Going Gone!

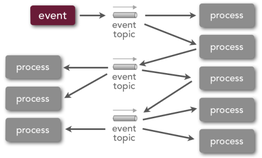


Your Architectural Kata is...

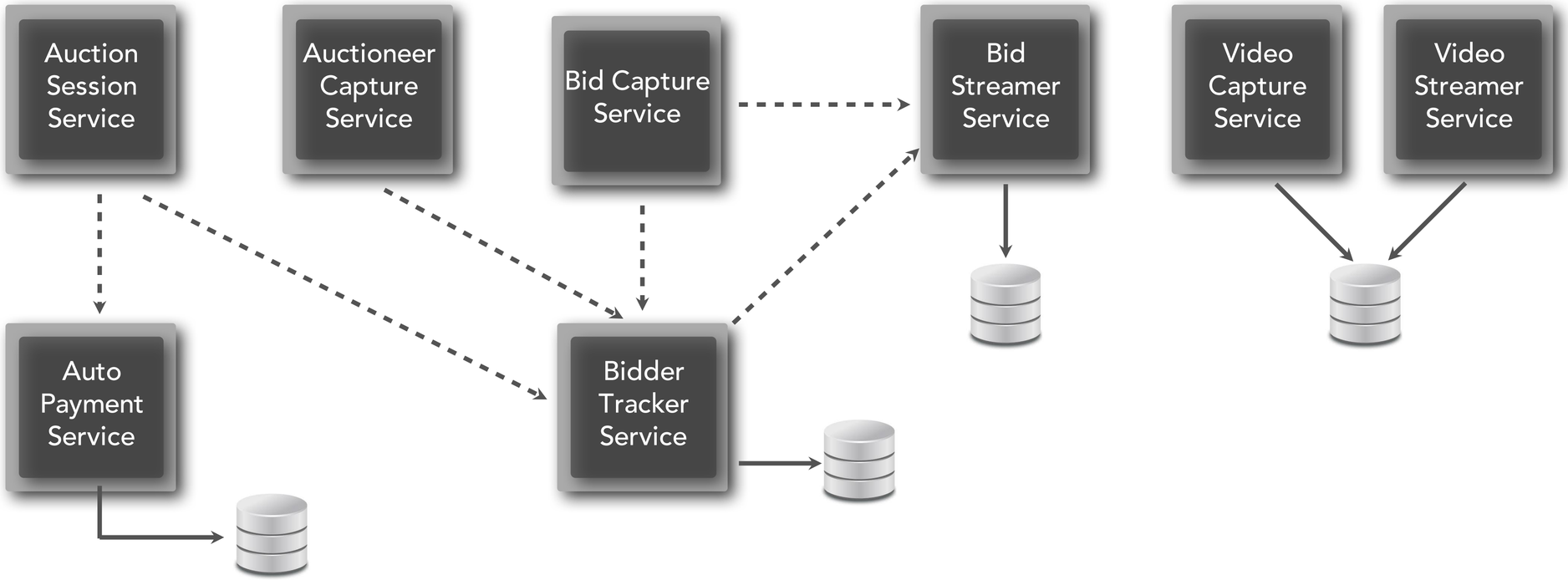
Going Going Gone!



microservices architecture

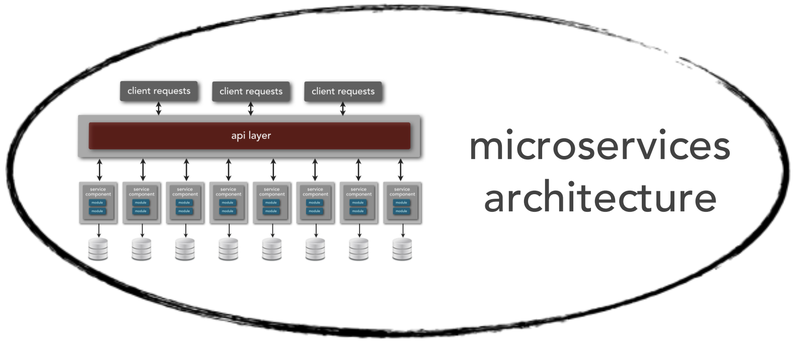


event-driven architecture

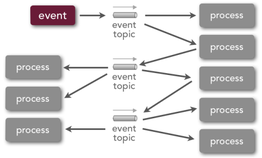


Your Architectural Kata is...

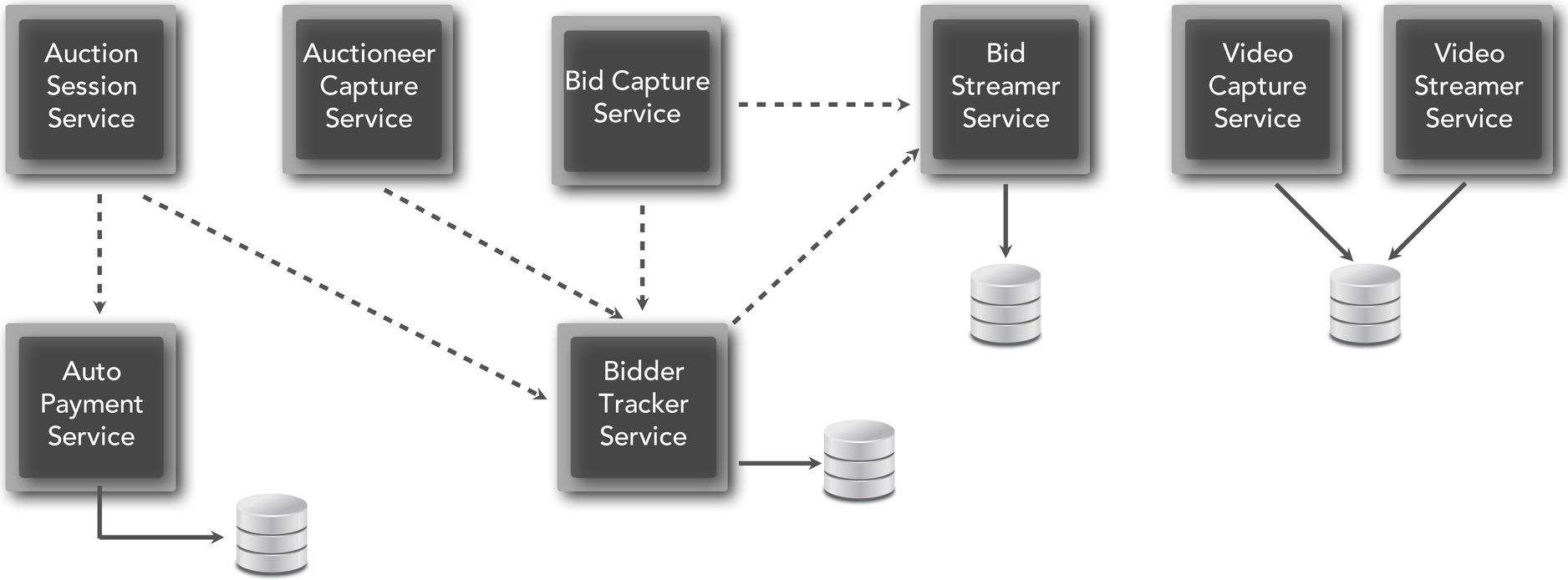
Going Going Gone!



microservices architecture

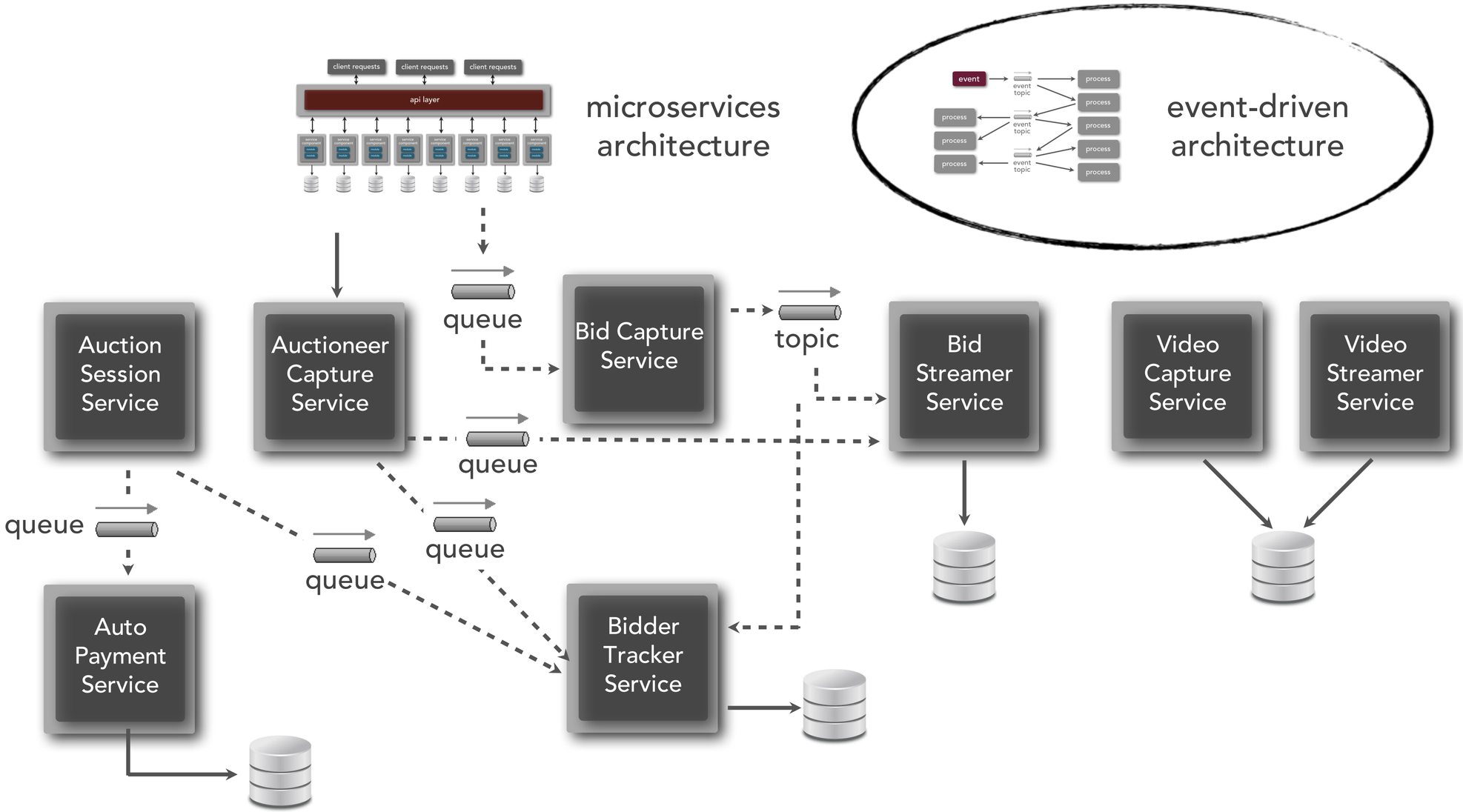


event-driven architecture



Your Architectural Kata is...

Going Going Gone!



Your Architectural Kata is...

Silicon Sandwiches

A national sandwich shop wants to enable 'fax in your order' but over the Internet instead (in addition to their current fax-in service)

- ***Users:*** thousands, perhaps one day millions
- ***Requirements:***
  - users will place their order, then be given a time to pick up their sandwich and directions to the shop (which must integrate with several external mapping services that include traffic information)
  - if the shop offers a delivery service, dispatch the driver with the sandwich to the user
  - mobile-device accessibility
  - offer national daily promotions/specials
  - offer local daily promotions/specials
  - accept payment online or in person/on delivery
- ***Additional Context:***
  - Sandwich shops are franchised, each with a different owner.
  - Parent company has near-future plans to expand overseas.
  - Corporate goal is to hire inexpensive labor to maximize profit.
  - Time to market is critical.

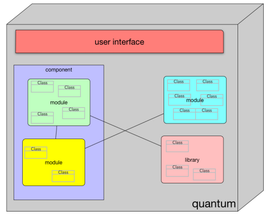
availability reliability

scalability

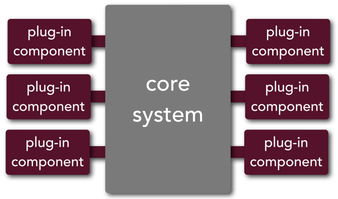
customizability { location, sales, recipe

Your Architectural Kata is...

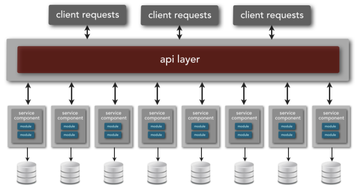
Silicon Sandwiches



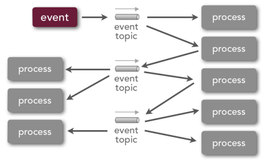
modular monolith



microkernel



microservices



event-driven

availability

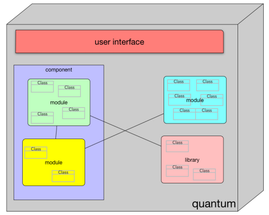
reliability

scalability

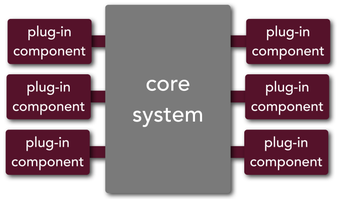
customizability

Your Architectural Kata is...

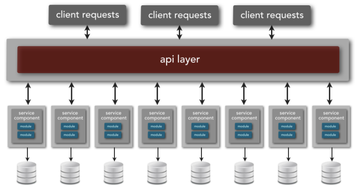
Silicon Sandwiches



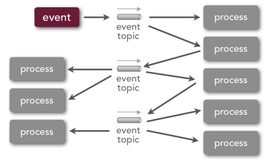
modular monolith



microkernel



microservices



event-driven

availability

reliability

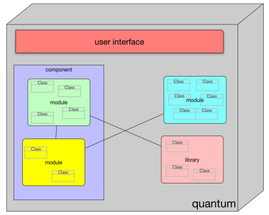
scalability

customizability

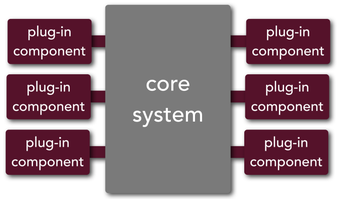
simplicity

Your Architectural Kata is...

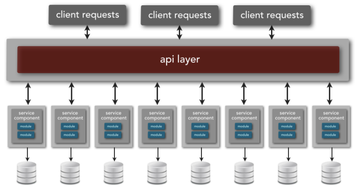
Silicon Sandwiches



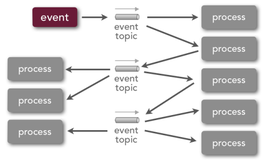
modular monolith



microkernel



microservices



event-driven

availability

reliability

scalability

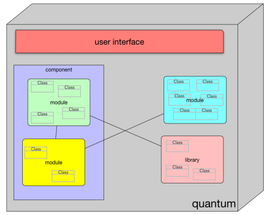
customizability

simplicity

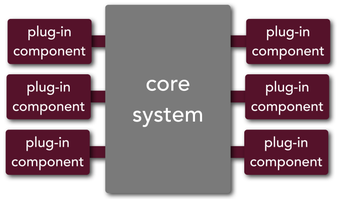
domain isomorphism

Your Architectural Kata is...

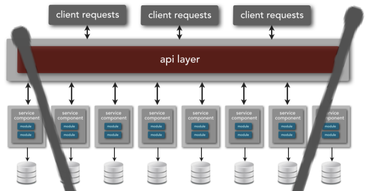
Silicon Sandwiches



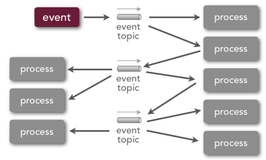
modular monolith



microkernel



microservices



event-driven

availability

reliability

scalability

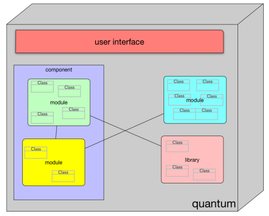
customizability

simplicity

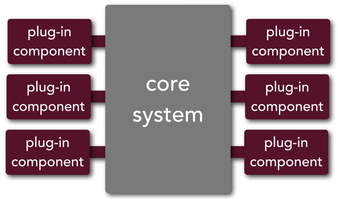
domain isomorphism

Your Architectural Kata is...

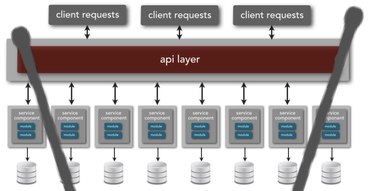
Silicon Sandwiches



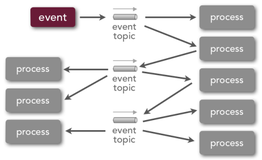
modular monolith



microkernel



microservices



event-driven

availability

reliability

scalability

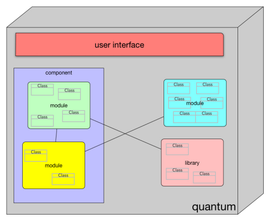
customizability

simplicity

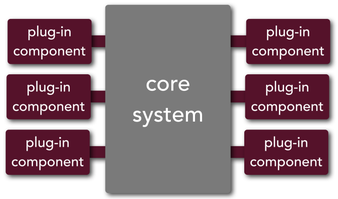
domain isomorphism

Your Architectural Kata is...

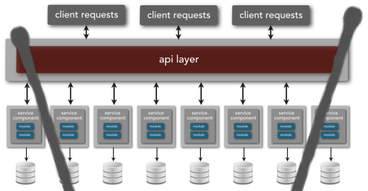
Silicon Sandwiches



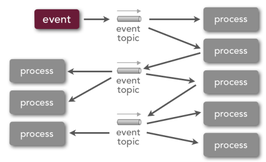
modular monolith



microkernel



microservices



event-driven

availability

reliability

scalability

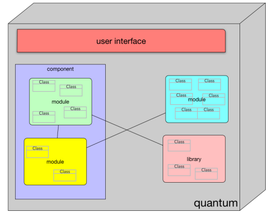
customizability

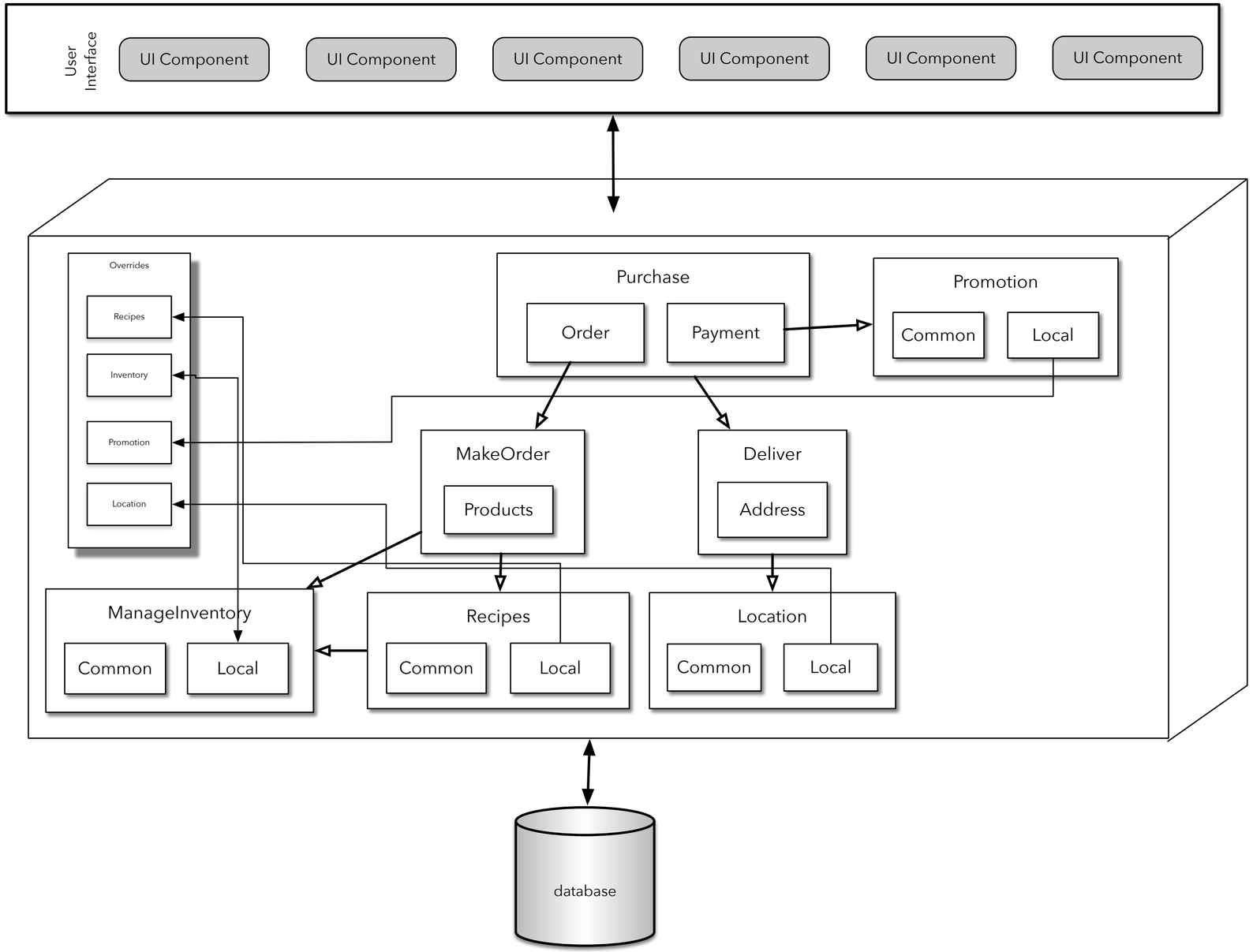
simplicity

domain isomorphism

Your Architectural Kata is...

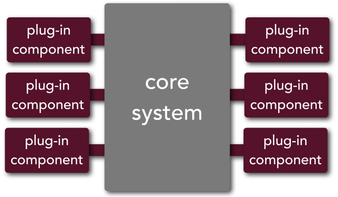
Silicon Sandwiches

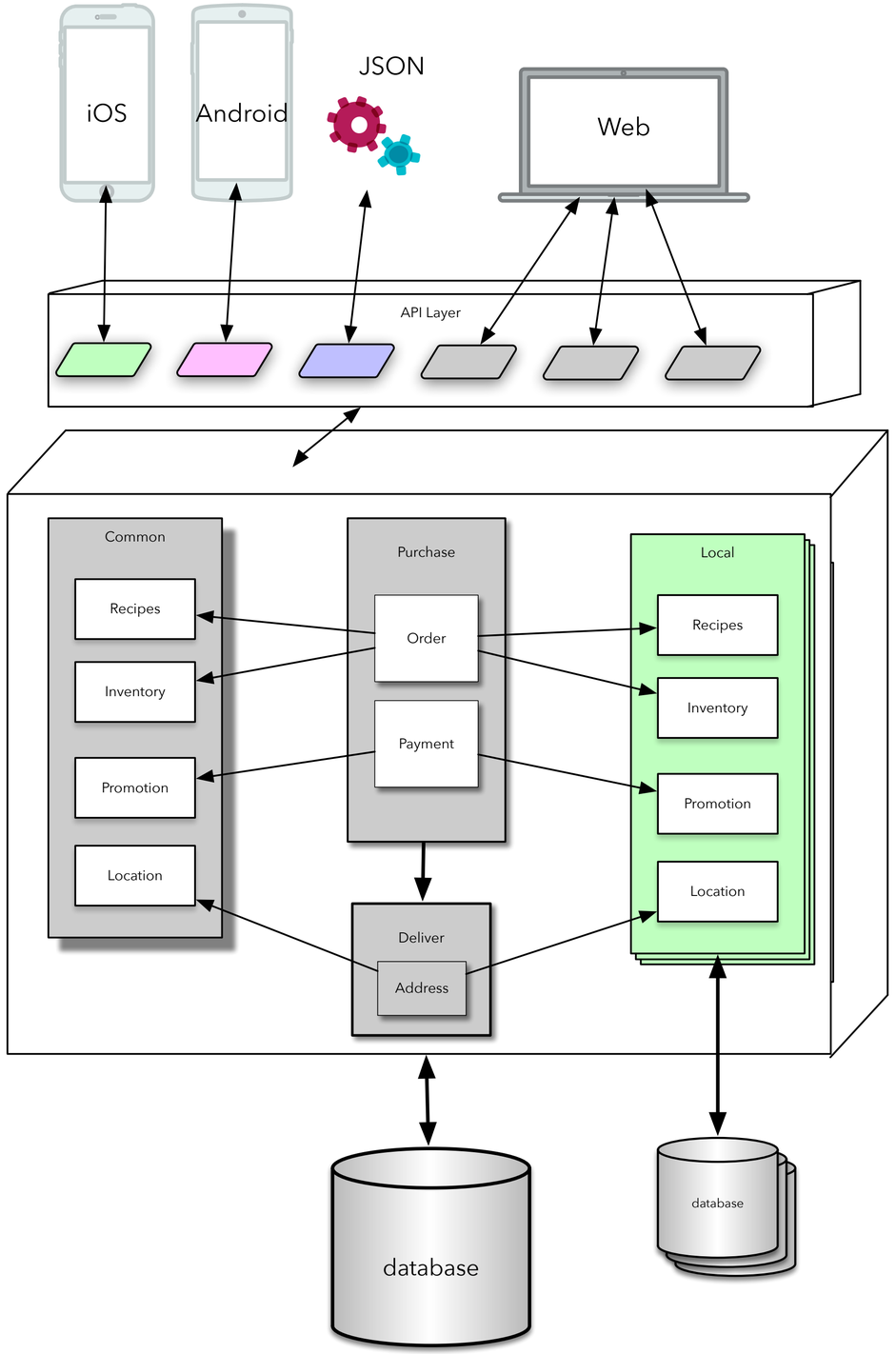




Your Architectural Kata is...

Silicon Sandwiches

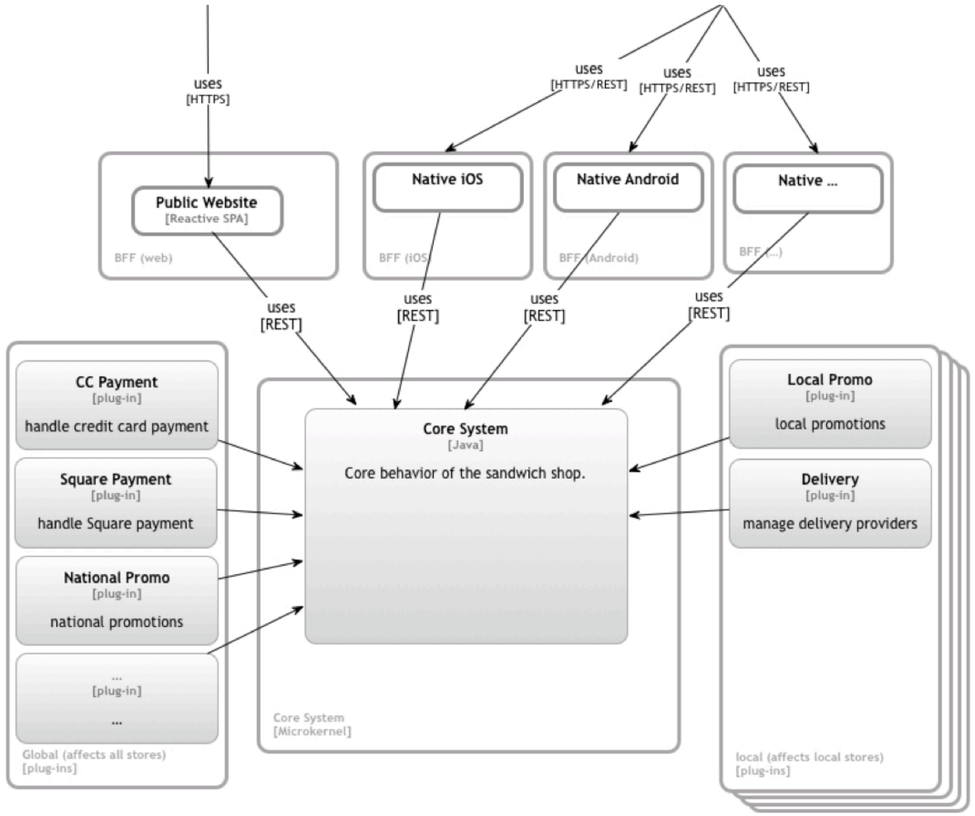




documenting software architecture

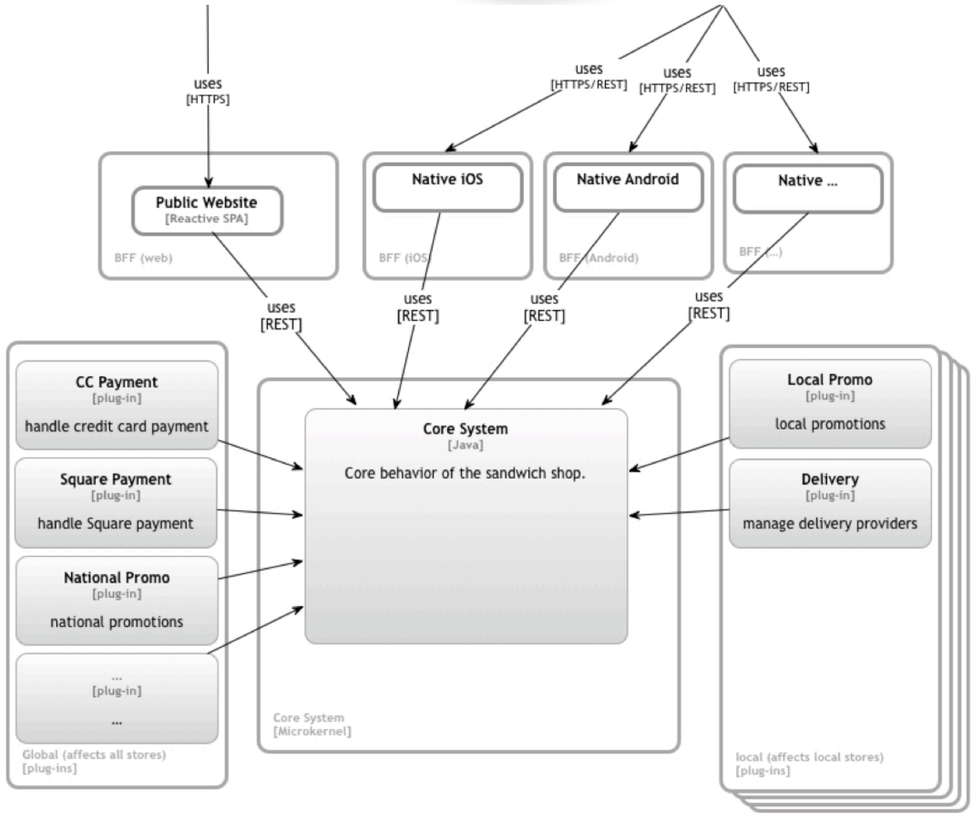
architecture decisions

an architect is responsible for defining the architecture and design principles used to guide technology decisions



architecture decisions

an architect is responsible for defining the architecture and design principles used to guide technology decisions



architecture decisions

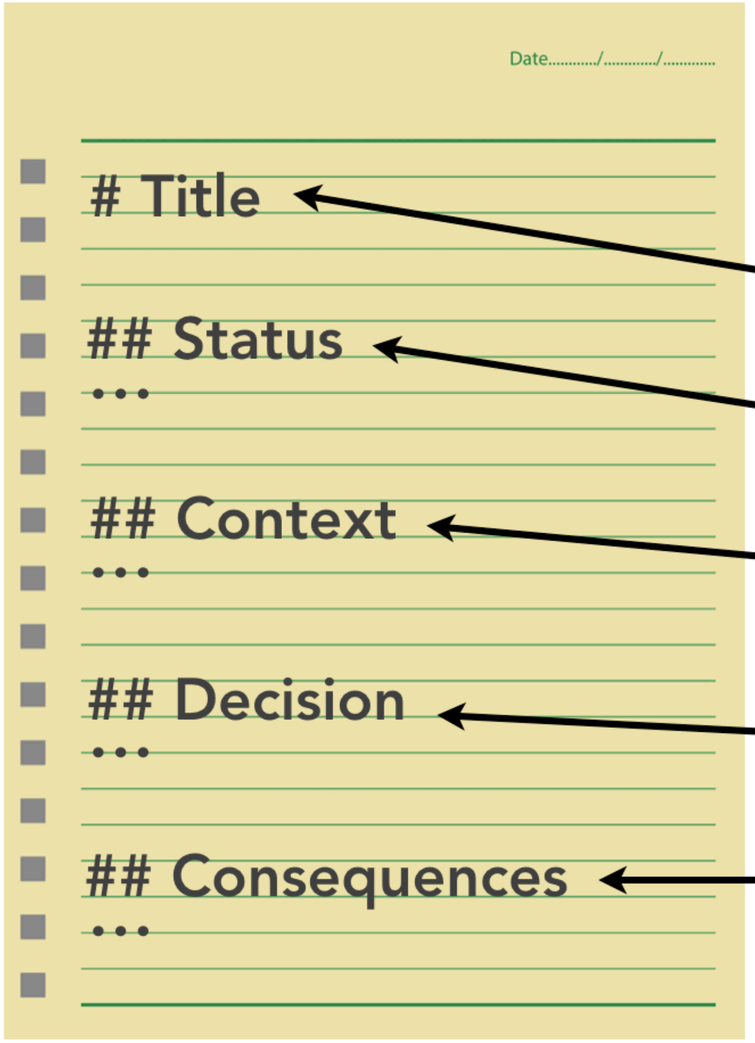
architecturally significant

"We will keep a collection of records for *architecturally significant decisions*: those that affect the structure, non-functional characteristics, dependencies, interfaces, or construction techniques."

- Michael Nygard

http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions

Architecture Decision Records



short text file; 1-2 pages long, one file per decision
markdown, textile, asciidoc, plaintext, etc.

short noun phrase

proposed, accepted, superseded

forces at play

response to forces with justification

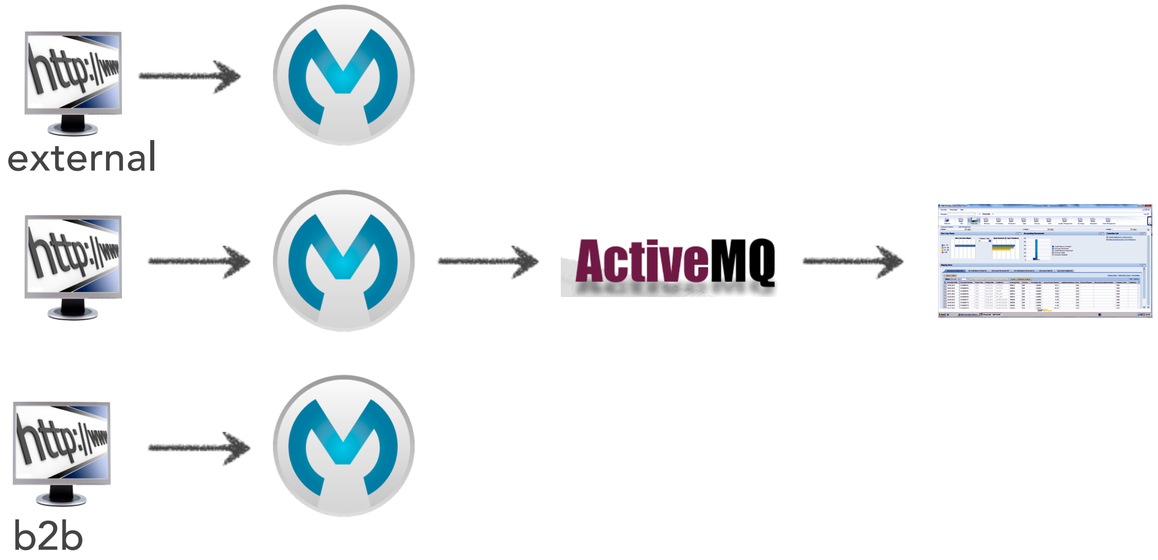
context after decision is applied

justifying decisions

the scenario

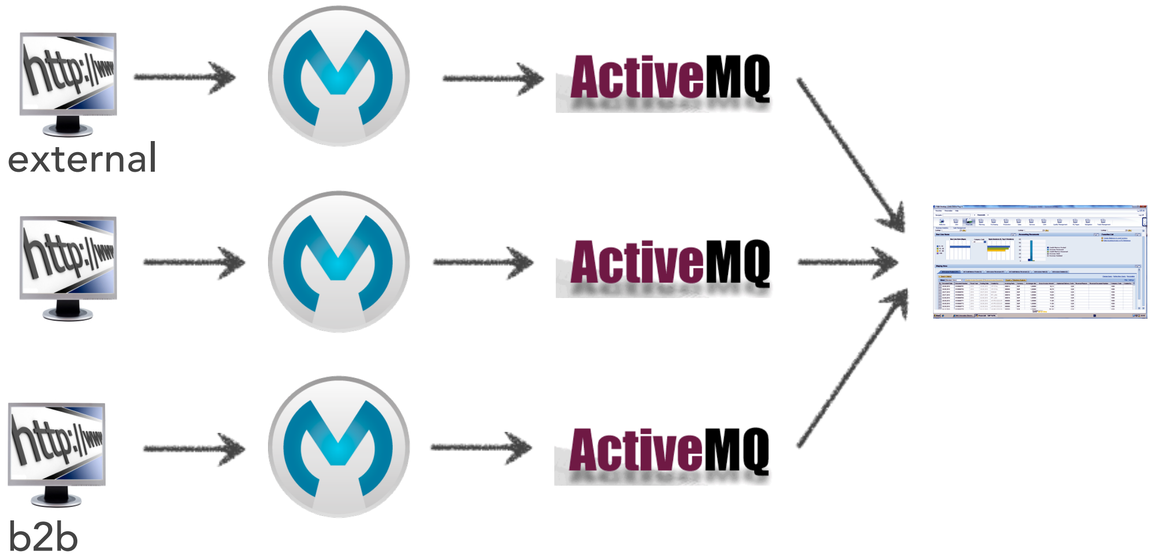
justifying decisions

the requirement: you need to federate the hub



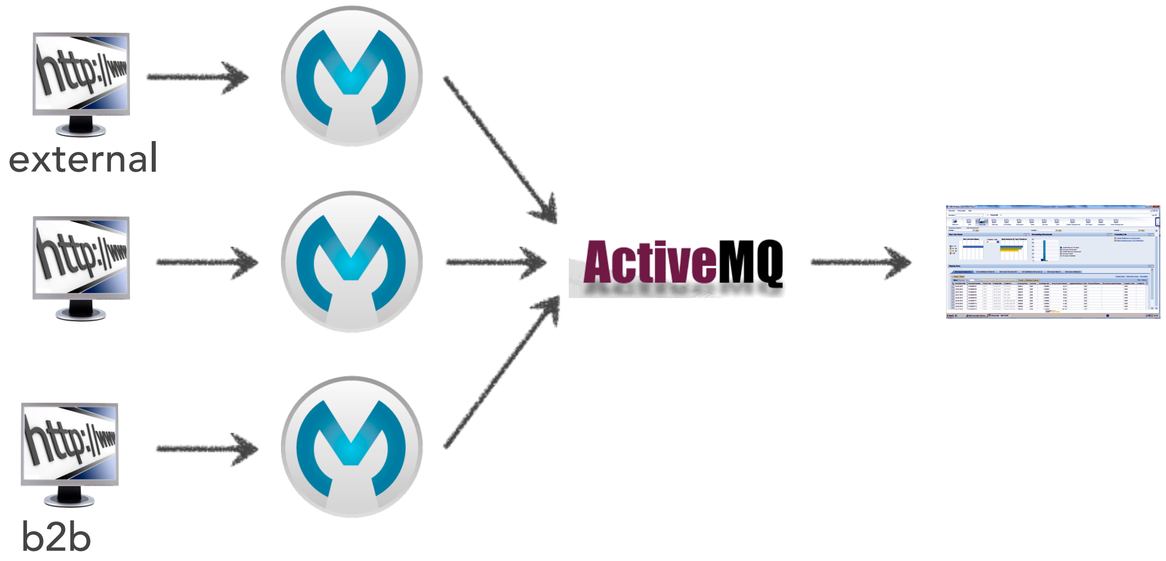
justifying decisions

the decision: dedicated broker instances?



justifying decisions

the decision: centralized broker



21. Centralized message broker for federated gateway hub

Status

Accepted

Context

The incoming gateway hub is federated into 3 separate hubs. Access to the customer information functionality in the application is only through JMS using ActiveMQ via the hub. The two options are to use a centralized broker or to use dedicated broker instances for each hub instance.

Decision

We will use a centralized message broker instance for all federated gateway hubs.

We currently have low request volumes (200 requests/sec) and anticipate this will remain stable for the foreseeable future. Therefore we do not see a performance bottleneck issue.

We will leverage the ActiveMQ failover protocol coupled with clustered ActiveMQ broker instances to address any single point of failure issues.

Consequences

The customer information application will only require a single connection to the ActiveMQ broker instance.

The gateway hubs can be expanded and consolidated without any coding changes to the application.

The application doesn’t need to be concerned about where the request came from.

Federating the Hub

Context
The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:

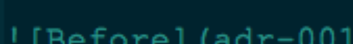


Figure 1: Before new clients added

New requirements require developers to add two new access types: _external_ and _b2b_.

Considerations:

* broker only used for hub access
* low transaction volumes expected
* application logic may be shared between different types of client applications (e.g., internal and external)

Two options exist:

Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

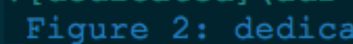


Figure 2: dedicated broker instances

Identified issues to address:

* **throughput**:
Dedicated broker instances provide better throughput because no message contention exists.
* **internal application coupling**:
This approach requires changes to the internal client application to
"understand" the broker. Internal app must now connect to three brokers and know context of request.
* **changes to client**: additional brokers added requires additional changes to client.
* **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
* **performance**: multiple broker instances should protect against aggregated performance problems

Federating the Hub

Context

The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:

Figure 1: Before new clients added

New requirements require developers to add two new access types: *external* and *b2b*.

Considerations:

- broker only used for hub access
- low transaction volumes expected
- application logic may be shared between different types of client applications (e.g., internal and external)

Two options exist:

Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

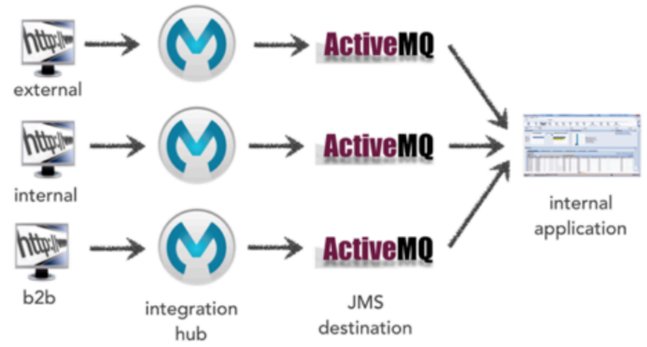


Figure 2: dedicated broker instances

Identified issues to address:

- **throughput**: Dedicated broker instances provide better throughput because no message contention exists.
- **internal application coupling**: This approach requires changes to the internal client application to "understand" the broker. Internal app must now connect to three brokers and know context of request.
- **changes to client**: additional brokers added requires additional changes to client.
- **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
- **performance**: multiple broker instances should protect against aggregated performance problems.

Centralized Broker

Using dedicated broker instances creates the architecture shown in Figure 3.

Figure 3: centralized broker

Identified issues to address:

- **throughput**: Centralized broker potentially creates a throughput bottleneck. However, developers analysed past and expected future usage, and this shouldn't create problem.
- **internal application coupling**: Loose, with only one connection; app doesn't know about broker instances. The internal application doesn't need to know where the request originated.
- **changes to client**: additional brokers do not require changes to client.
- **single point of failure**: mitigated by clustering and failover provided by tools
- **performance**: because we expect low transaction volumes, performance should be sufficient with a single queue.

Decision

We chose a **centralized broker**.

Status

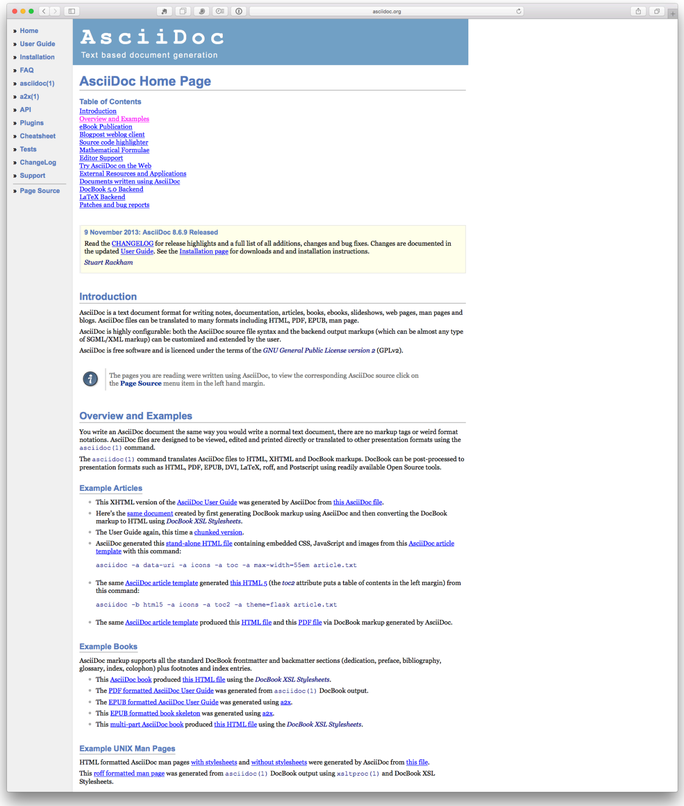
proposed

Consequences

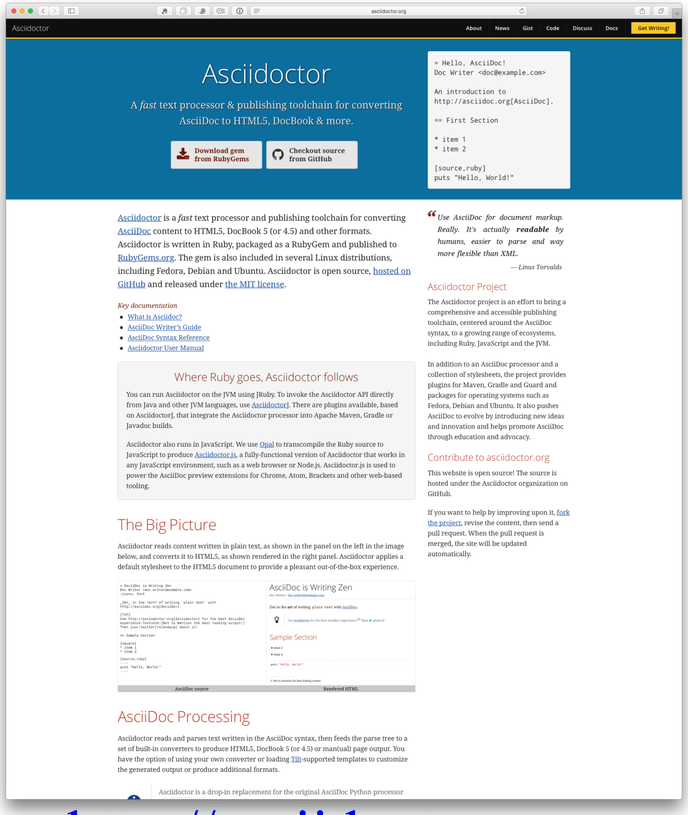
The internal applications should not have to know from which broker instance the request came from.

Only a single broker connection is needed, allowing for the expansion of additional hub instances with no application changes.

the case for Asciidoc(tor)

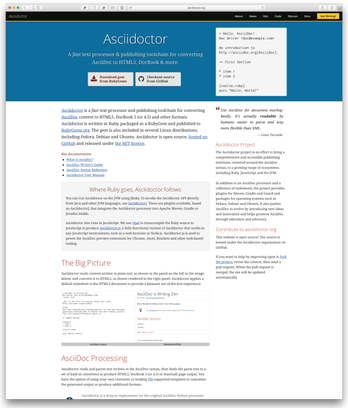


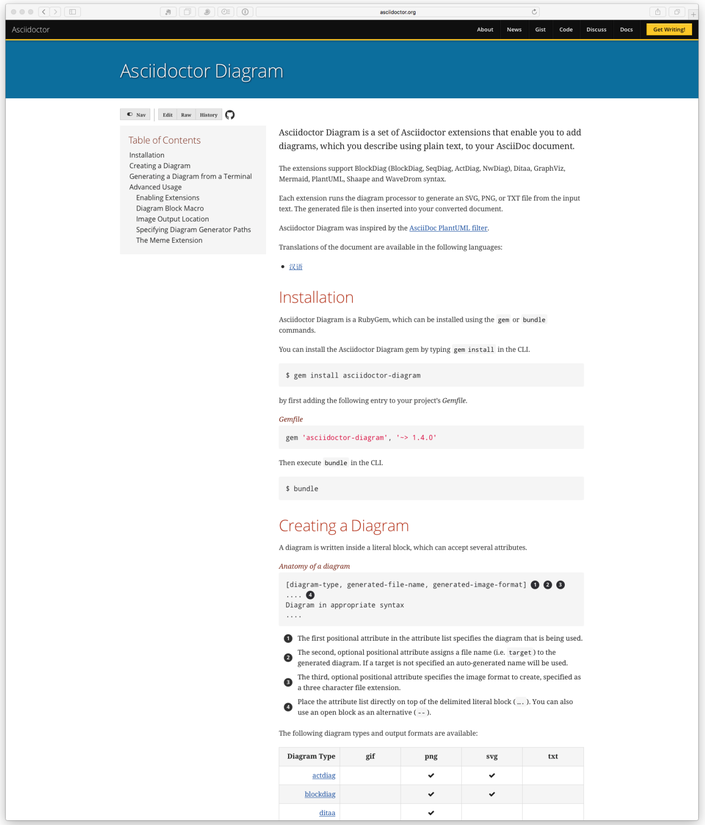
http://asciidoc.org



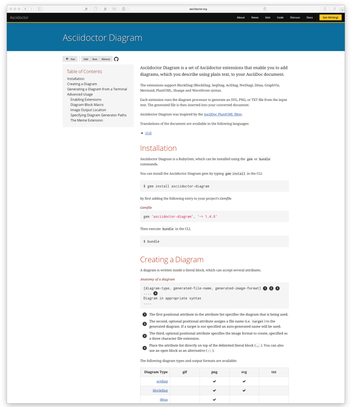
http://asciidoctor.org

the case for Asciidoc(tor)



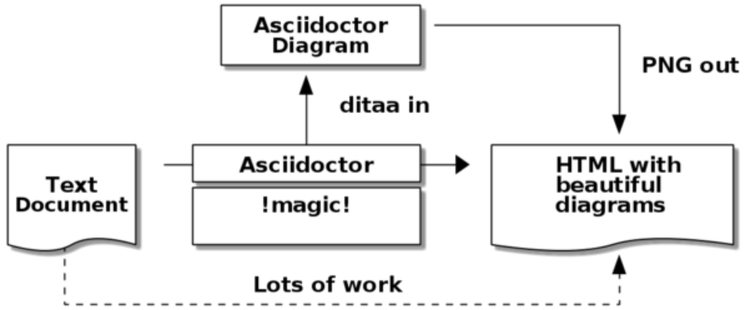


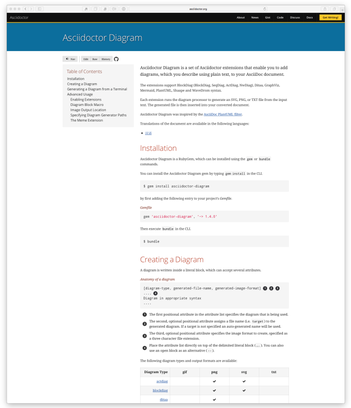
http://asciidoctor.org/docs/asciidoctor-diagram/



the case for Asciidoc(tor)

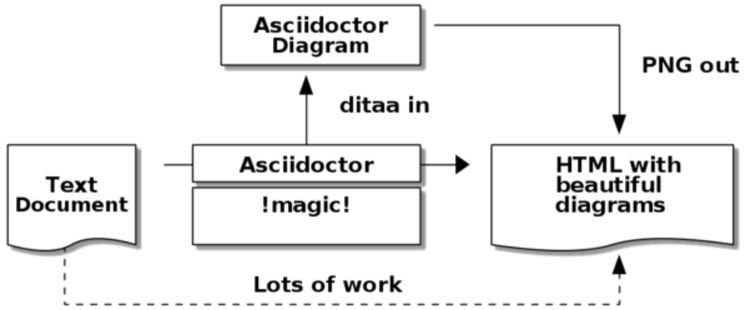
```
[ditaa]
....
                    +-------------+
                    | Asciidoctor |-------+
                    |   diagram   |       |
                    +-------------+       | PNG out
                        ^                 |
                        | ditaa in        |
                        |                 v
 +--------+   +--------+----+    /---------------\
 |        | --+ Asciidoctor +--> |               |
 |  Text  |   +-------------+    |   Beautiful   |
 |Document|   |   !magic!   |    |    Output     |
 |     {d}|   |             |    |               |
 +---+----+   +-------------+    \---------------/
     :                                   ^
     |          Lots of work             |
     +-----------------------------------+
....
```



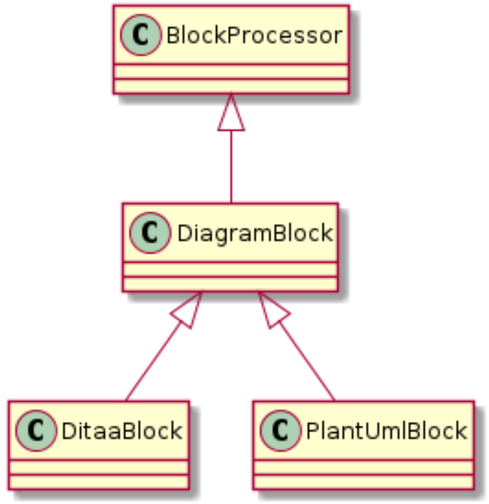


the case for Asciidoc(tor)

```
[ditaa]
....
                    +--------------+
                    | Asciidoctor  |-------+
                    |   diagram    |       |
                    +--------------+       | PNG out
                        ^                  |
                        | ditaa in         |
                        |                  v
 +--------+   +--------+----+    /---------------\
 |        | --+ Asciidoctor +--> |               |
 |  Text  |   +-------------+    |   Beautiful   |
 |Document|   |   !magic!   |    |    Output     |
 |     {d}|   |             |    |               |
 +---+----+   +-------------+    \---------------/
     :                                   ^
     |          Lots of work             |
     +-----------------------------------+
....
```




```
[plantuml, diagram-classes, png]
....
class BlockProcessor
class DiagramBlock
class DitaaBlock
class PlantUmlBlock

BlockProcessor <|-- DiagramBlock
DiagramBlock <|-- DitaaBlock
DiagramBlock <|-- PlantUmlBlock
....
```

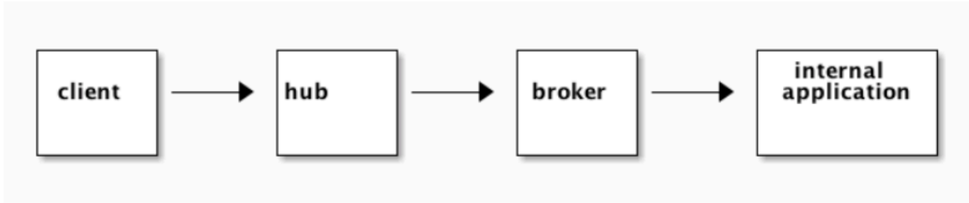


Federating the Hub

Context

The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:



New requirements require developers to add two new access types: *external* and *b2b*.

Considerations:

- broker only used for hub access
- low transaction volumes expected
- application logic may be shared between different types of client applications (e.g., internal and external)

Two options exist:

Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

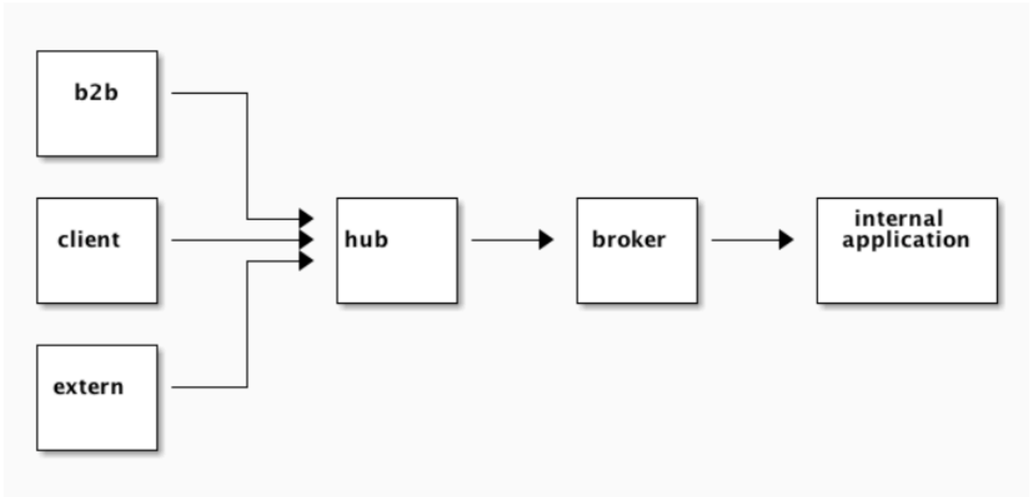


Figure 2: dedicated broker instances

Identified issues to address:

- **throughput**: Dedicated broker instances provide better throughput because no message contention exists.
- **internal application coupling**: This approach requires changes to the internal client application to "understand" the broker. Internal app must now connect to three brokers and know context of request.
- **changes to client**: additional brokers added requires additional changes to client.
- **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
- **performance**: multiple broker instances should protect against aggregated performance problems.

Centralized Broker

Using dedicated broker instances creates the architecture shown in Figure 3.

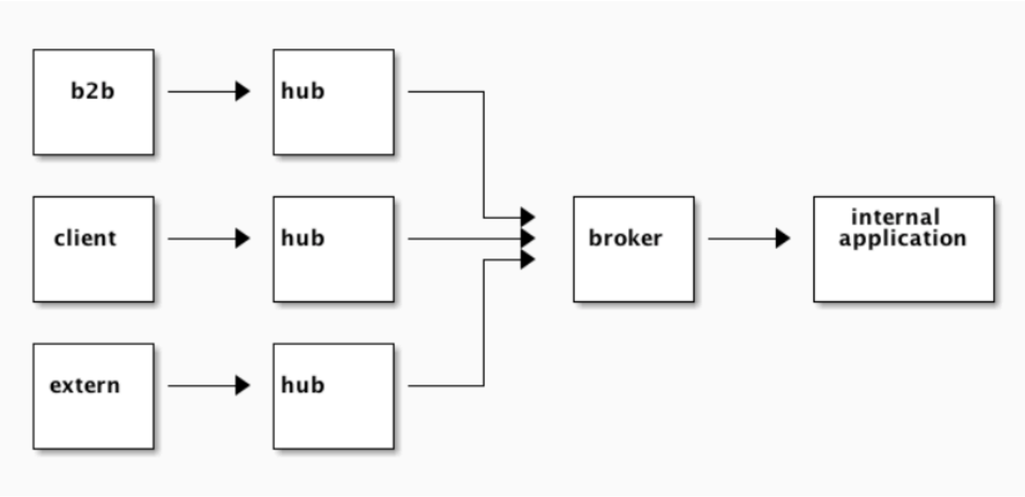


Figure 3: centralized broker

Identified issues to address:

- **throughput**: Centralized broker potentially creates a throughput bottleneck. However, developers analysed past and expected future usage, and this shouldn't create problem.
- **internal application coupling**: Loose, with only one connection; app doesn't know about broker instances. The internal application doesn't need to know where the request originated.
- **changes to client**: additional brokers do not require changes to client.
- **single point of failure**: mitigated by clustering and failover provided by tools
- **performance**: because we expect low transaction volumes, performance should be sufficient with a single queue.

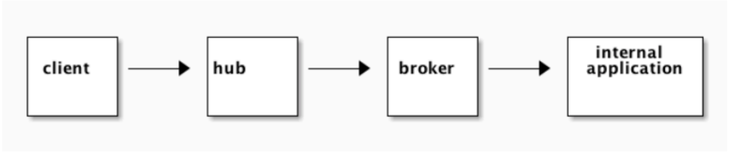
```
_Considerations_:

* broker only used for hub access
* low transaction volumes expected
* application logic may be shared between different types of client app

Two options exist:

### Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figu

[ditaa]
....
+-------+
|       |
|  b2b  |-----+
|       |     |
|       |     |
+-------+     |
              |
+-------+     |     +-------+       +-------+       +-----------+
|       |     |---> |       |       |       |       | internal  |
|client |-----------> |hub    |-----> |broker |-----> |application|
|       |     |---> |       |       |       |       |           |
|       |     |     |       |       |       |       |           |
+-------+     |     +-------+       +-------+       +-----------+
              |
+-------+     |
|       |     |
|extern |-----+
|       |
|       |
+-------+
....

_Figure 2: dedicated broker instances_

_Identified issues to address:_

* **throughput**:
Dedicated broker instances provide better throughput because no message
* **internal application coupling**:
This approach requires changes to the internal client application to
"understand" the broker. Internal app must now connect to three brokers
* **changes to client**: additional brokers added requires additional c
* **single point of failure**: redundancy prevents a single failure fro
* **performance**: multiple broker instances should protect against agg


### Centralized Broker

Using dedicated broker instances creates the architecture shown in Figu

[ditaa]
....
+-------+       +-------+
|       |       |       |
|  b2b  |-----> |hub    |-----+
|       |       |       |     |
|       |       |       |     |
+-------+       +-------+     |
                              |
+-------+       +-------+     |     +-------+       +-----------+
|       |       |       |     |---> |       |       | internal  |
|client |-----> |hub    |-------->  |broker |-----> |application|
|       |       |       |     |---> |       |       |           |
|       |       |       |     |     |       |       |           |
+-------+       +-------+     |     +-------+       +-----------+
                              |
+-------+       +-------+     |
|       |       |       |     |
|extern |-----> |hub    |-----+
|       |       |       |
|       |       |       |
+-------+       +-------+

....
```

Federating the Hub

Context

The AcmeWidgets application currently utilizes an integration hub to allow internal applications to connect to it.

Figure 1 illustrates the existing scenario:



New requirements require developers to add two new access types: *external* and *b2b*.

Considerations:

- broker only used for hub access
- low transaction volumes expected
- application logic may be shared between different types of client applications (e.g., internal and external)

Two options exist:

Dedicated Broker Instances

Using dedicated broker instances creates the architecture shown in Figure 2.

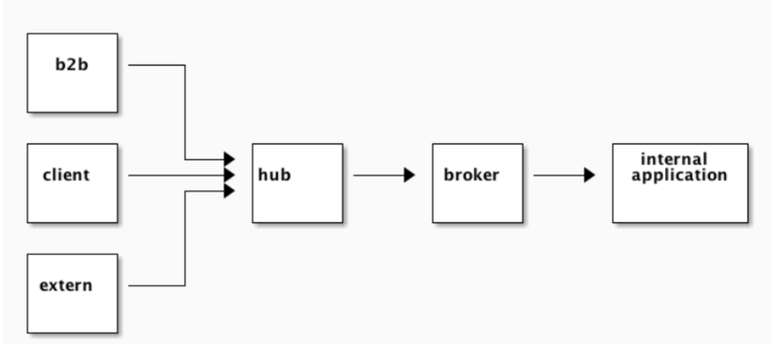


Figure 2: dedicated broker instances

Identified issues to address:

- **throughput**: Dedicated broker instances provide better throughput because no message contention exists.
- **internal application coupling**: This approach requires changes to the internal client application to "understand" the broker. Internal app must now connect to three brokers and know context of request.
- **changes to client**: additional brokers added requires additional changes to client.
- **single point of failure**: redundancy prevents a single failure from disabling all integration architecture.
- **performance**: multiple broker instances should protect against aggregated performance problems.

Centralized Broker

Using dedicated broker instances creates the architecture shown in Figure 3.

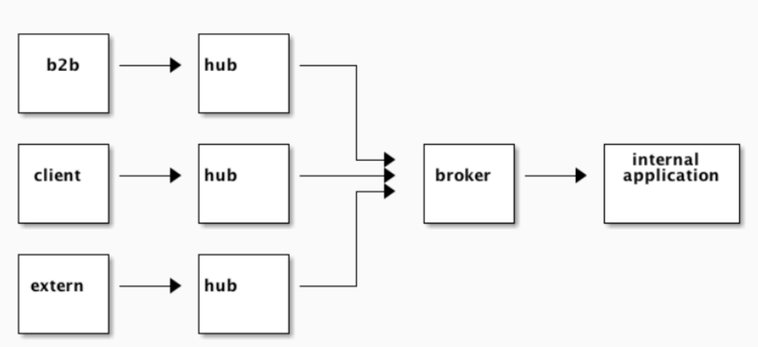


Figure 3: centralized broker

Identified issues to address:

- **throughput**: Centralized broker potentially creates a throughput bottleneck. However, developers analysed past and expected future usage, and this shouldn't create problem.
- **internal application coupling**: Loose, with only one connection; app doesn't know about broker instances. The internal application doesn't need to know where the request originated.
- **changes to client**: additional brokers do not require changes to client.
- **single point of failure**: mitigated by clustering and failover provided by tools
- **performance**: because we expect low transaction volumes, performance should be sufficient with a single queue.

Your Architectural Kata is...

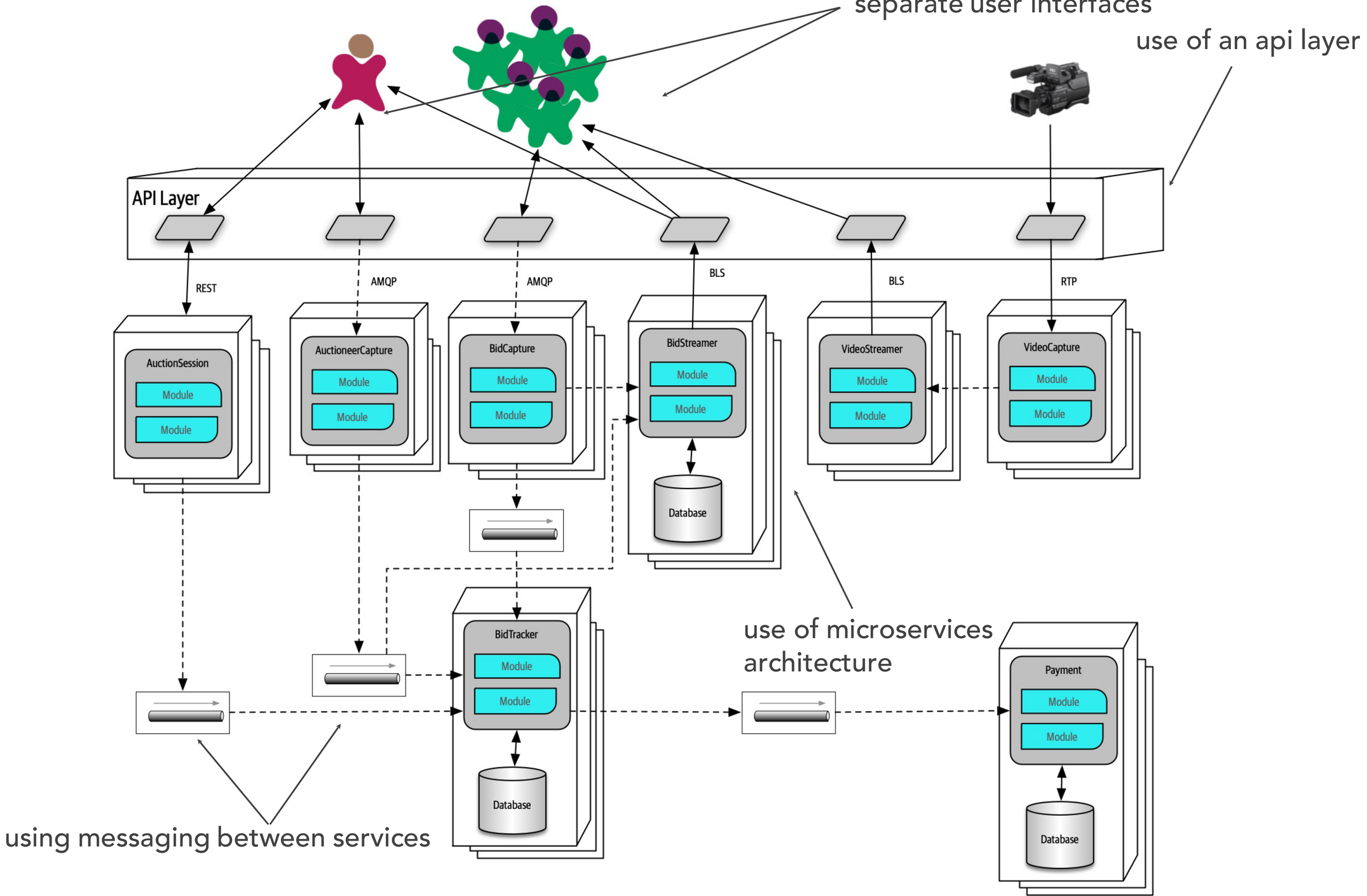
Going Going Gone!

An auction company wants to take their auctions online to a nationwide scale--customers choose the auction to participate in, wait until the auction begins, then bid during the live auction as if they were there in the room, with the auctioneer.

- **Users**: scale up to hundreds of participants (per auction), potentially up to thousands of participants, and as many simultaneous auctions as possible
- **Requirements**:
  - bidders can see a live video stream of the auction and see all bids as they occur
  - auctions must be as real-time as possible
  - both online and live bids must be received in the order in which they are placed
  - bidders register with credit card; system automatically charges card if bidder wins
  - participants must be tracked via a reputation index
- **Additional Context:**
  - auction company is expanding aggressively by merging with smaller competitors
  - if nationwide auction is a success, replicate the model overseas
  - budget is not constrained--this is a strategic direction
  - company just exited a lawsuit where they settled a suit alleging fraud

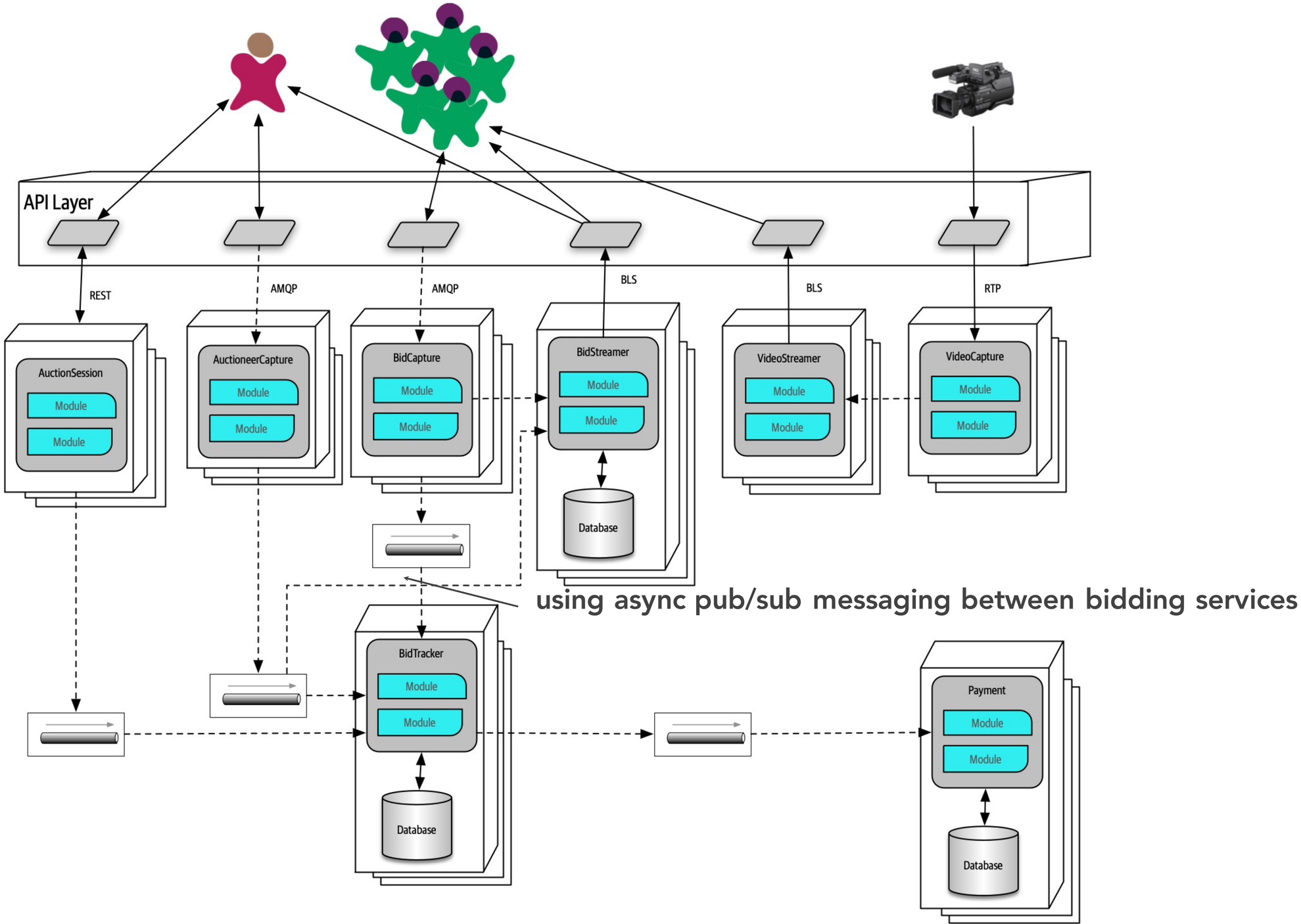
Your Architectural Kata is...

Going Going Gone!



Your Architectural Kata is...

Going Going Gone!



1. Asynchronous messaging between bidding services

Status

Proposed

Context

Decision

Consequences

1. Asynchronous messaging between bidding services

Status

Proposed

Context

The Bid Capture Service, upon receiving a bid from an online bidder or from a live bidder via the auctioneer, must forward that bid onto the Bid Streamer Service and the Bidder Tracker Service. This could be done using async p2p or pub/sub messaging or REST via the Online Auction API Layer.

Decision

Consequences

1. Asynchronous messaging between bidding services

Status

Proposed

Context

The Bid Capture Service, upon receiving a bid from an online bidder or from a live bidder via the auctioneer, must forward that bid onto the Bid Streamer Service and the Bidder Tracker Service. This could be done using async p2p or pub/sub messaging or REST via the Online Auction API Layer.

Decision

We will use asynchronous messaging between the Bid Capture Service, Bid Streamer Service, and the Bidder Tracker Service.

The Bid Capture Service does not need any information back from the Bid Streamer Service or Bidder Tracker Service.

The Bid Streamer Service must receive bids in the exact order they were accepted by the Bid Capture Service. Using messaging and queues automatically guarantees the bid order for the stream.

Using async pub/sub messaging will increase the performance of the bidding process and allow for extensibility of bidding information.

Consequences

1. Asynchronous messaging between bidding services

Status

Proposed

Context

The Bid Capture Service, upon receiving a bid from an online bidder or from a live bidder via the auctioneer, must forward that bid onto the Bid Streamer Service and the Bidder Tracker Service. This could be done using async p2p or pub/sub messaging or REST via the Online Auction API Layer.

Decision

We will use asynchronous messaging between the Bid Capture Service, Bid Streamer Service, and the Bidder Tracker Service.

The Bid Capture Service does not need any information back from the Bid Streamer Service or Bidder Tracker Service.

The Bid Streamer Service must receive bids in the exact order they were accepted by the Bid Capture Service. Using messaging and queues automatically guarantees the bid order for the stream.

Using async pub/sub messaging will increase the performance of the bidding process and allow for extensibility of bidding information.

Consequences

We will require clustering and high availability of the message queues

Internal bid events will be bypassing security checks done in the API layer.

Use of Micro-kernel Architecture

Status
PROPOSED

Context
Two key requirements of the system (_promotions_ anc

The current design features a modular monolith archite

![modular monolith](fig1_modular_monolith.jpg)
Figure 1: the current state architecture

Currently, stores must specify custom behavior (produc

All local customizations reside in one service and in on

To allow stores to most easily add and customize local

![microkernel architecture](fig2_microkernel.jpg)
Figure 2: proposed microkernel architecture

The new design allows easy update of global policy (pr

Decision
The architects decided to migrate the current monolith

Consequences
The architects take advantage of the restructuring oppo

The new design also incorporates the BFF patterns, dis

The new design will greatly improve the customization

- the local store plug-in architecture certifies customiza
- promotions within threshold values go live within 15 n
- all stores work with generic workflows via the core sys
  - promotions
  - location exemptions
  - local products

quirements.

e 1_.

FTP. Operations must certify the JAR, leading to dela

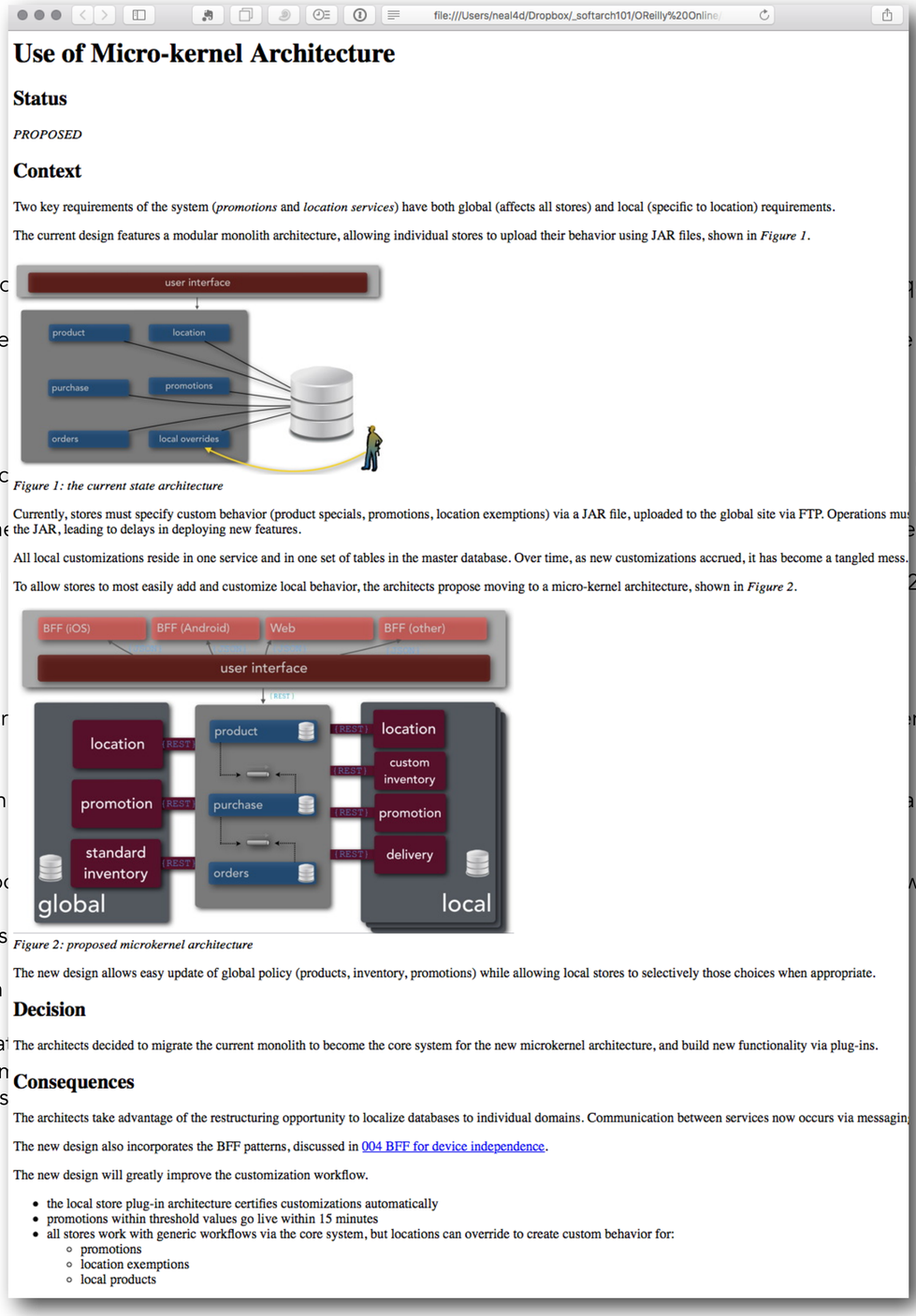
ecome a tangled mess.

2_.

n appropriate.

ality via plug-ins.

w occurs via messaging.



Use c

Statu
_PROP(

Con
Two ke

The cur

![modu
_Figure

Current

All loca

To allo

![microl
_Figure

The nev

Dec
The arc

Con
The arc

The nev

The nev

- the lo
- promo
- all sto
  - prom
  - locat
  - local

file:///Users/neal4d/Dropbox/_softarch101/OReilly%20Online/

Use of Micro-kernel Architecture

Status

PROPOSED

Context

Two key requirements of the system (*promotions* and *location services*) have both global (affects all stores) and local (specific to location) requirements.

The current design features a modular monolith architecture, allowing individual stores to upload their behavior using JAR files, shown in *Figure 1*.

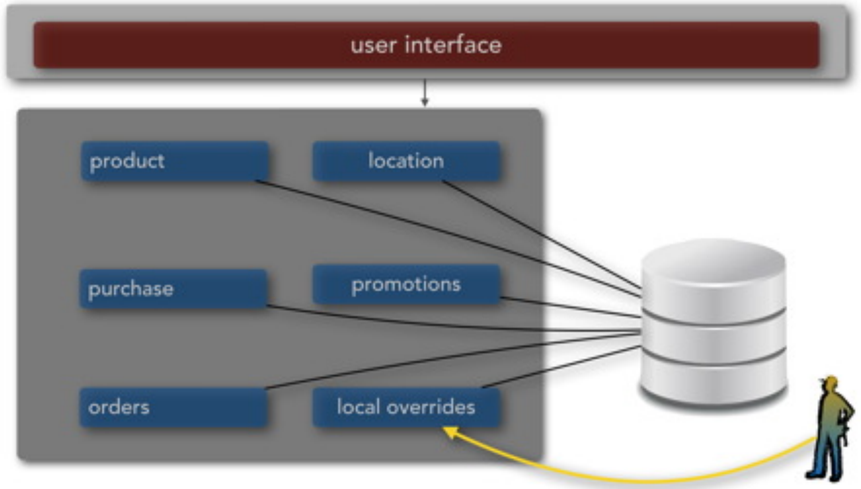


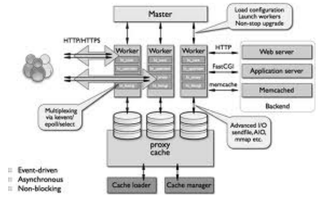
Figure 1: the current state architecture

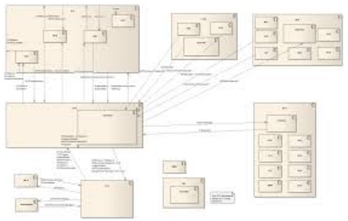
Currently, stores must specify custom behavior (product specials, promotions, location exemptions) via a JAR file, uploaded to the global site via FTP. Operations n
the JAR, leading to delays in deploying new features.

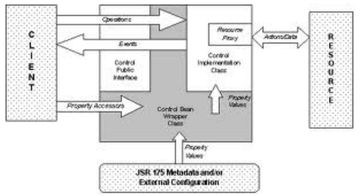
All local customizations reside in one service and in one set of tables in the master database. Over time, as new customizations accrued, it has become a tangled me

summary

summary

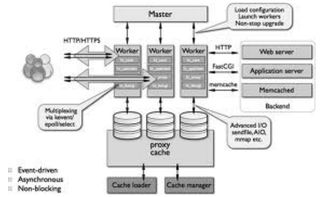


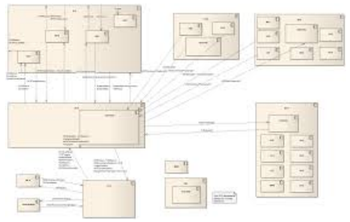


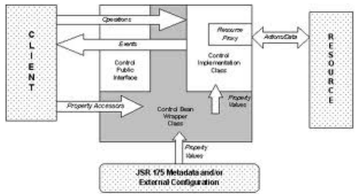


There are no right or wrong answers in architecture; rather, it's always about ***tradeoffs***

summary







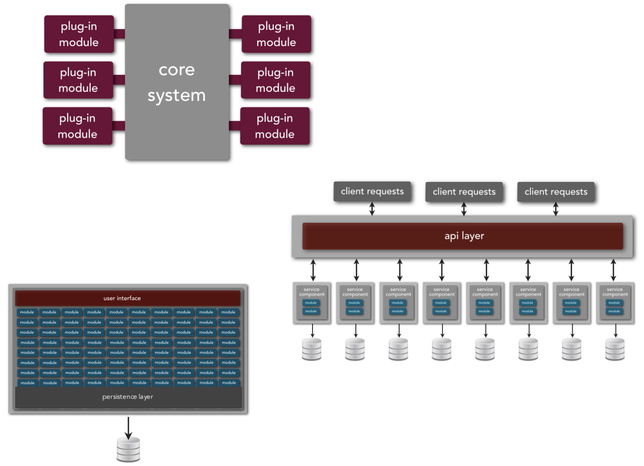
There are no right or wrong answers in architecture; ***only expensive ones***

summary

scalability

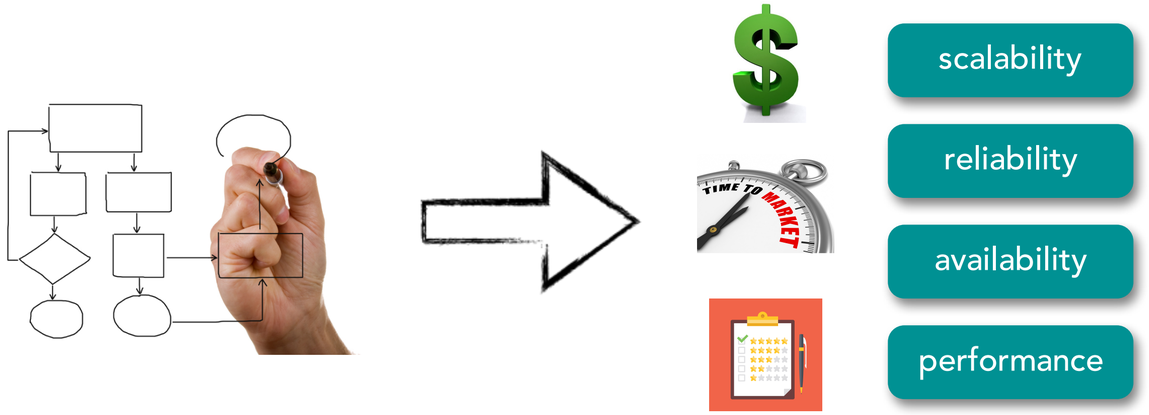
reliability

availability



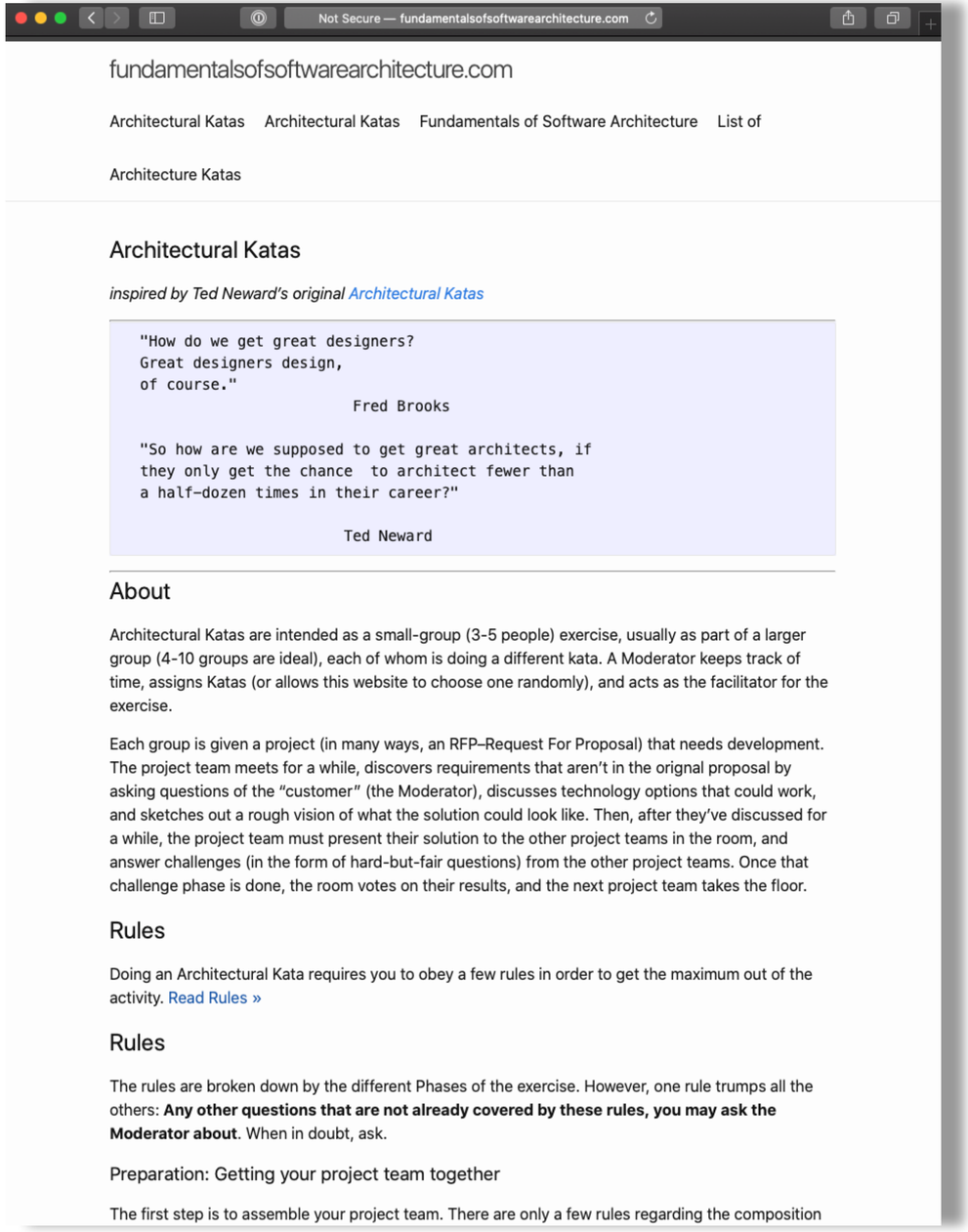
business needs and ***architectural characteristics*** are what drive the architecture!

summary



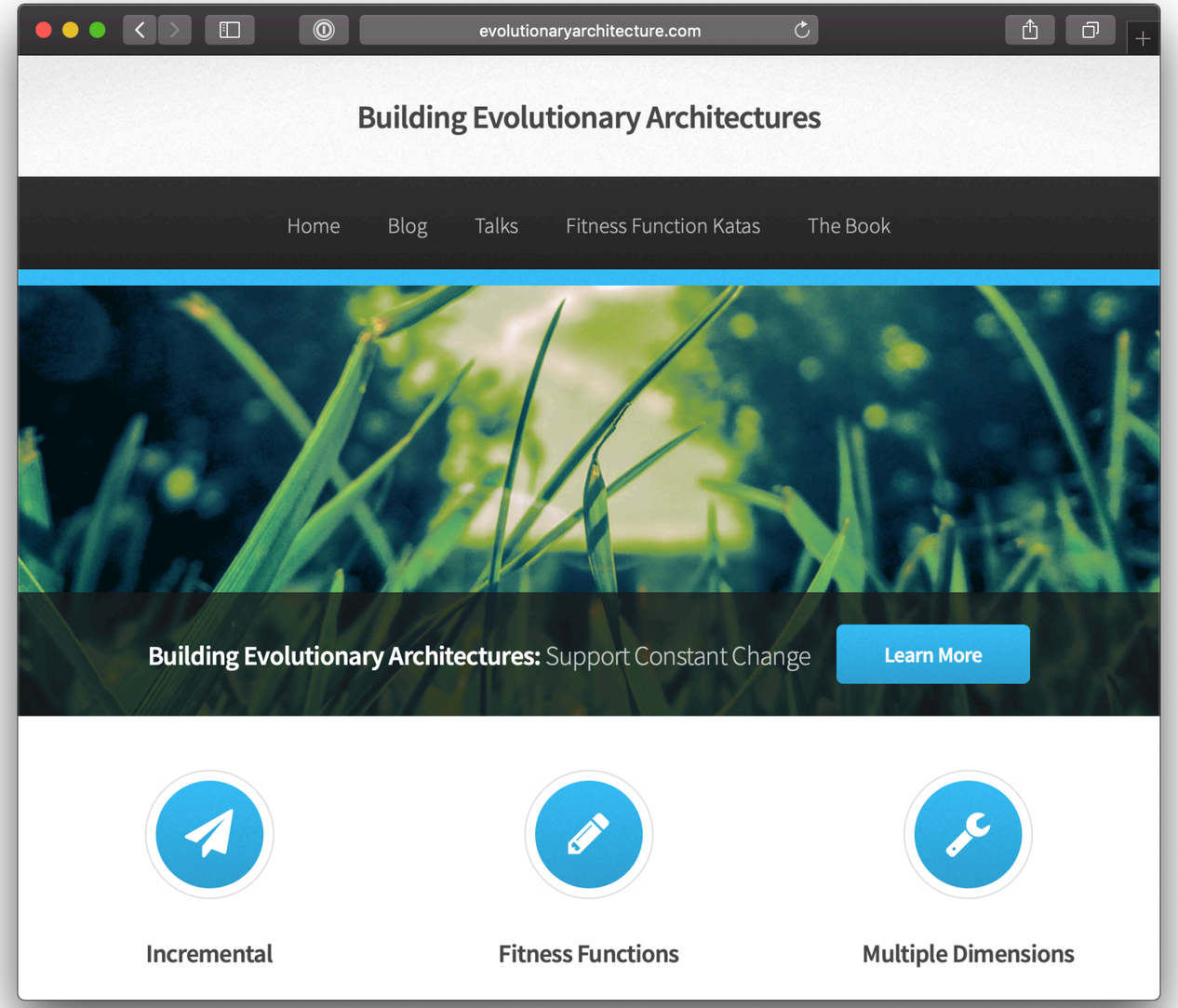
every architecture decision should be accompanied with a ***technical and business justification***

more resources



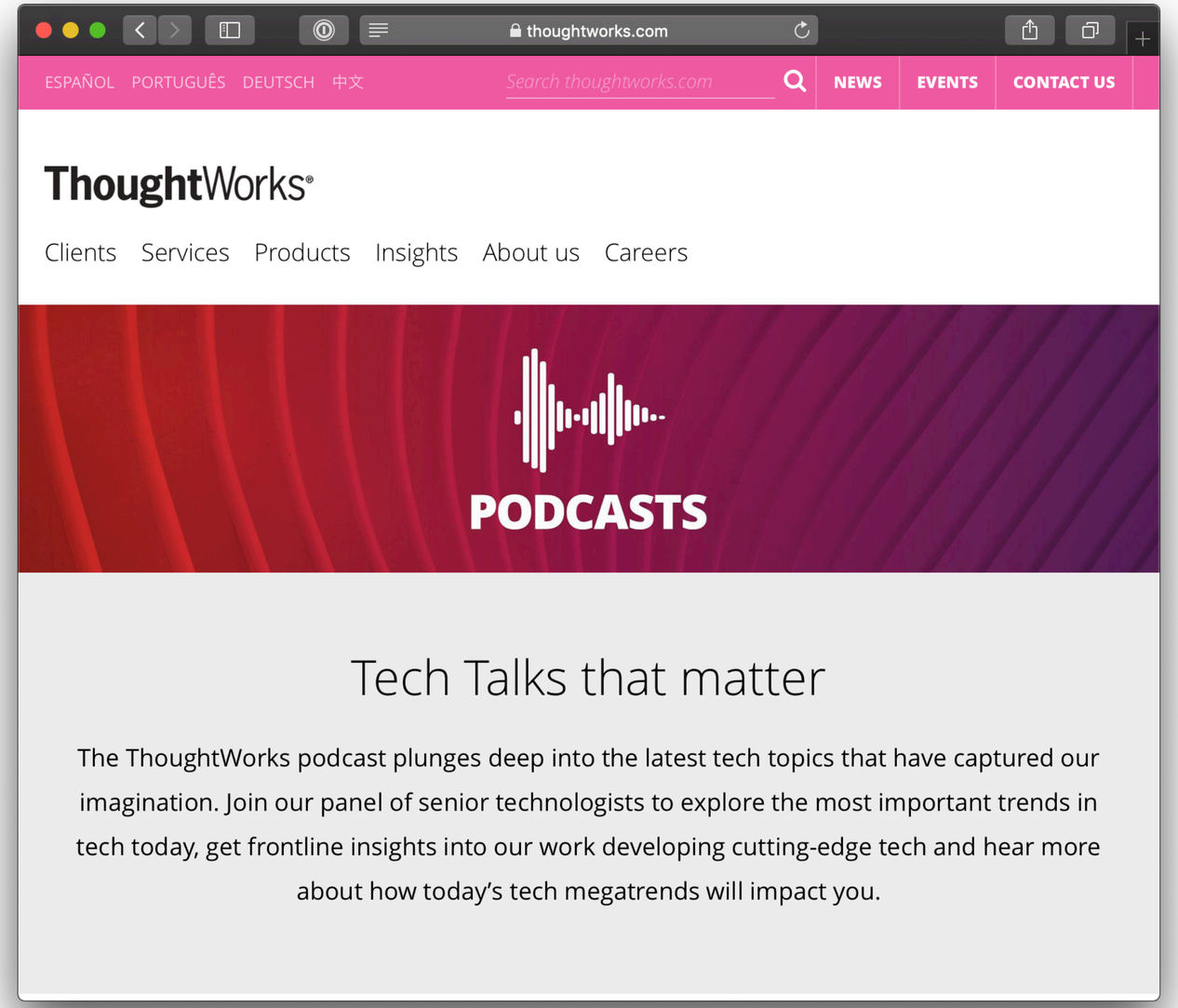
http://fundamentalsofsoftwarearchitecture.com/katas/

more resources



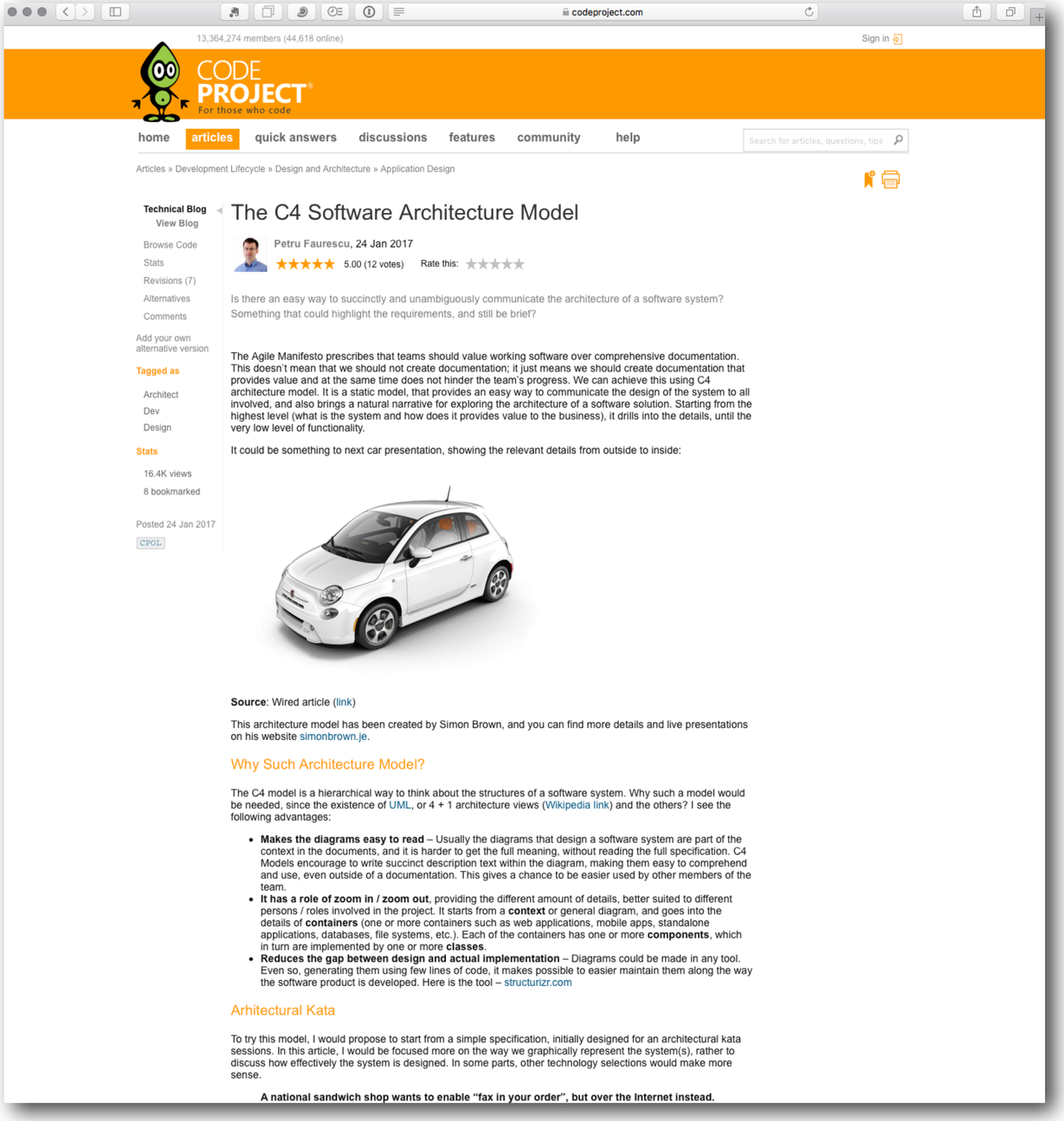
http://evolutionaryarchitecture.com

more resources



https://www.thoughtworks.com/podcasts

more resources



codeproject.com

13,364,274 members (44,618 online) Sign in

CODE PROJECT For those who code

home articles quick answers discussions features community help Search for articles, questions, tips

Articles » Development Lifecycle » Design and Architecture » Application Design

Technical Blog
View Blog

Browse Code
Stats
Revisions (7)
Alternatives
Comments

Add your own alternative version

Tagged as

Architect
Dev
Design

Stats

16.4K views
8 bookmarked

Posted 24 Jan 2017

CPOL

The C4 Software Architecture Model

Petru Faurescu, 24 Jan 2017

5.00 (12 votes) Rate this:

Is there an easy way to succinctly and unambiguously communicate the architecture of a software system? Something that could highlight the requirements, and still be brief?

The Agile Manifesto prescribes that teams should value working software over comprehensive documentation. This doesn't mean that we should not create documentation; it just means we should create documentation that provides value and at the same time does not hinder the team's progress. We can achieve this using C4 architecture model. It is a static model, that provides an easy way to communicate the design of the system to all involved, and also brings a natural narrative for exploring the architecture of a software solution. Starting from the highest level (what is the system and how does it provides value to the business), it drills into the details, until the very low level of functionality.

It could be something to next car presentation, showing the relevant details from outside to inside:

Source: Wired article (link)

This architecture model has been created by Simon Brown, and you can find more details and live presentations on his website simonbrown.je.

Why Such Architecture Model?

The C4 model is a hierarchical way to think about the structures of a software system. Why such a model would be needed, since the existence of UML, or 4 + 1 architecture views (Wikipedia link) and the others? I see the following advantages:

- **Makes the diagrams easy to read** – Usually the diagrams that design a software system are part of the context in the documents, and it is harder to get the full meaning, without reading the full specification. C4 Models encourage to write succinct description text within the diagram, making them easy to comprehend and use, even outside of a documentation. This gives a chance to be easier used by other members of the team.
- **It has a role of zoom in / zoom out**, providing the different amount of details, better suited to different persons / roles involved in the project. It starts from a **context** or general diagram, and goes into the details of **containers** (one or more containers such as web applications, mobile apps, standalone applications, databases, file systems, etc.). Each of the containers has one or more **components**, which in turn are implemented by one or more **classes**.
- **Reduces the gap between design and actual implementation** – Diagrams could be made in any tool. Even so, generating them using few lines of code, it makes possible to easier maintain them along the way the software product is developed. Here is the tool – structurizr.com

Arhitectural Kata

To try this model, I would propose to start from a simple specification, initially designed for an architectural kata sessions. In this article, I would be focused more on the way we graphically represent the system(s), rather to discuss how effectively the system is designed. In some parts, other technology selections would make more sense.

A national sandwich shop wants to enable "fax in your order", but over the Internet instead.

https://www.codeproject.com/Articles/1167140/The-C-Software-Architecture-Model

DeveloperToArchitect.com

go do some architecture!

Architecture by Example

Mark Richards
Independent Consultant
Hands-on Software Architect, Published Author
Founder, DeveloperToArchitect.com
http://www.wmrichards.com
@markrichardssa

Neal Ford
ThoughtWorks
Director / Software Architect / Meme Wrangler
http://www.nealford.com
@neal4d

O'REILLY®