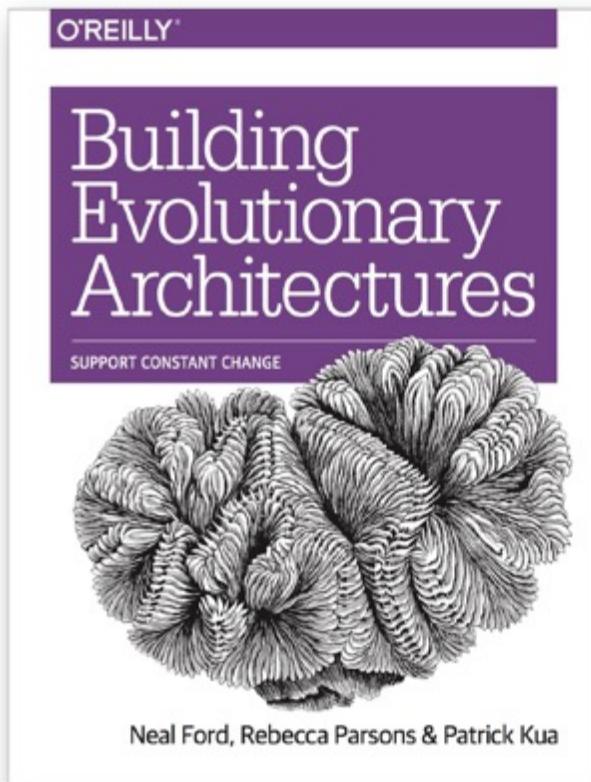


iMpLemENTiNg eVoLuTiONaRy ARcHiEcTuREs



@neal4d
nealford.com



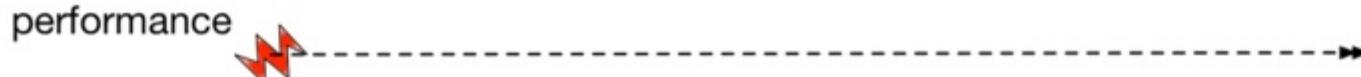
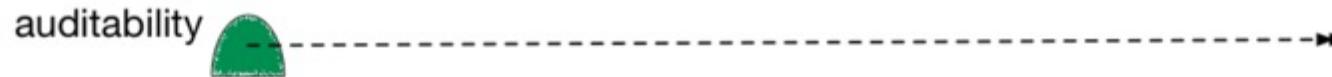
@rebeccaparsons



@patkua







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

mobility
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

productability

provability

recoverability

relevance

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

accessibility	reliability	repeatability
accountability	extensibility	reproducibility
accuracy	failure transparency	resilience
adaptability	fault-tolerance	responsiveness
administrability	fidelity	reusability
affordability	flexibility	robustness
agility	inspectability	safety
auditability	installability	scalability
autonomy	integrity	seamlessness

evolvability

degradability	operability	sustainability
determinability	orthogonality	tailorability
demonstrability	portability	testability
dependability	precision	timeliness
deployability	predictability	traceability
discoverability	process capabilities	transparency
distributability	productability	ubiquity
durability	provability	understandability
effectiveness	recoverability	upgradability
efficiency	relevance	usability

accessibility
accountability
accuracy
adaptability
administrability
affordability
agility
auditability
autonomy

reliability
extensibility
failure transparency
fault-tolerance
fidelity
flexibility
inspectability
installability
integrity

repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness

evolvability

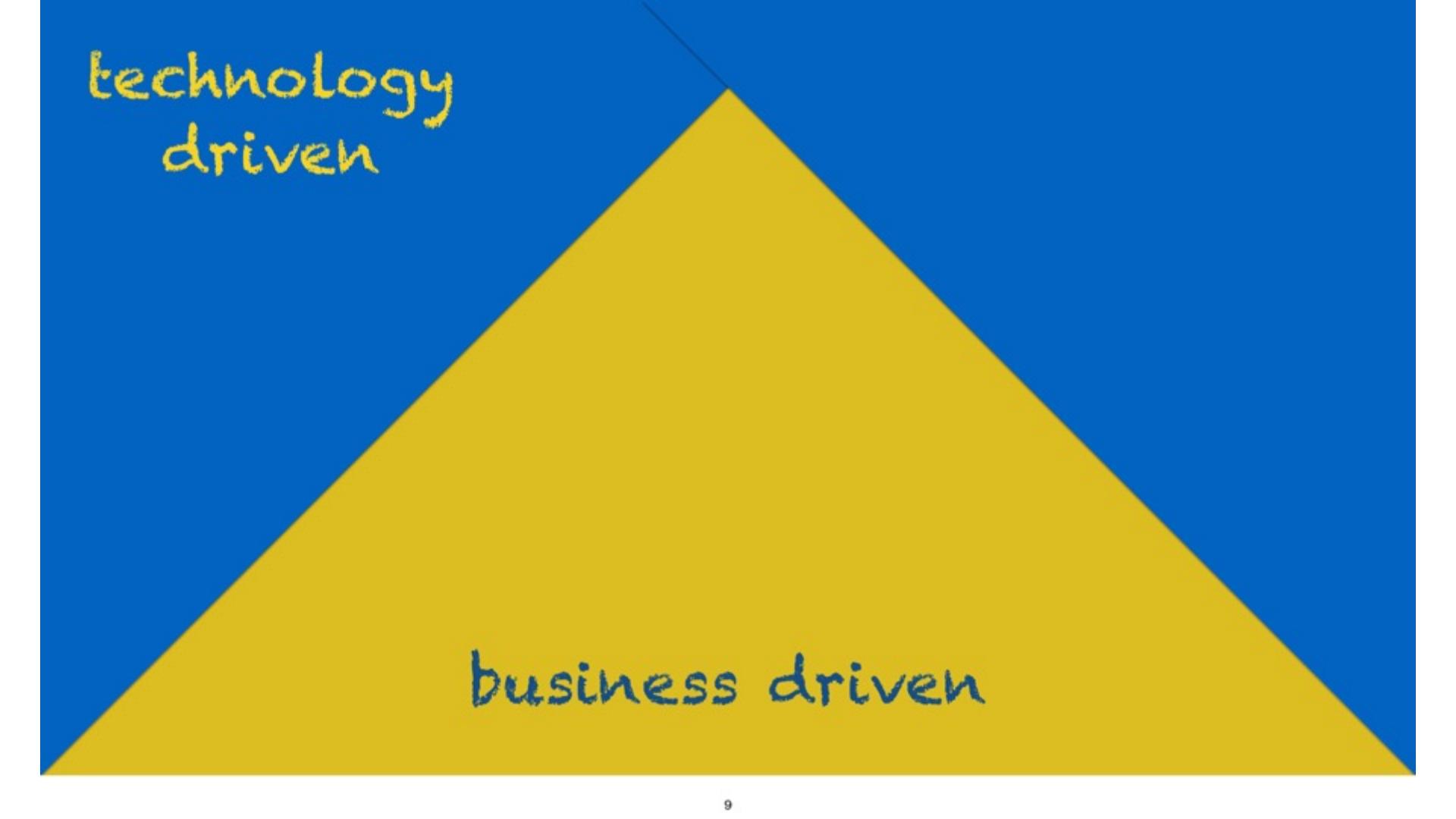
degradability
determinability
demonstrability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency

operability
orthogonality
portability
precision
predictability
process capabilities
productability
provability
recoverability
relevance

sustainability
tailorability
testability
timeliness
traceability
transparency
ubiquity
understandability
upgradability
usability

chAnGe

technology
driven

The image features a large, solid yellow triangle centered against a solid blue background. The triangle's vertices point downwards, creating a sense of depth. It is positioned in the lower half of the frame.

technology
driven

business driven

technology
driven



business driven

technology
driven

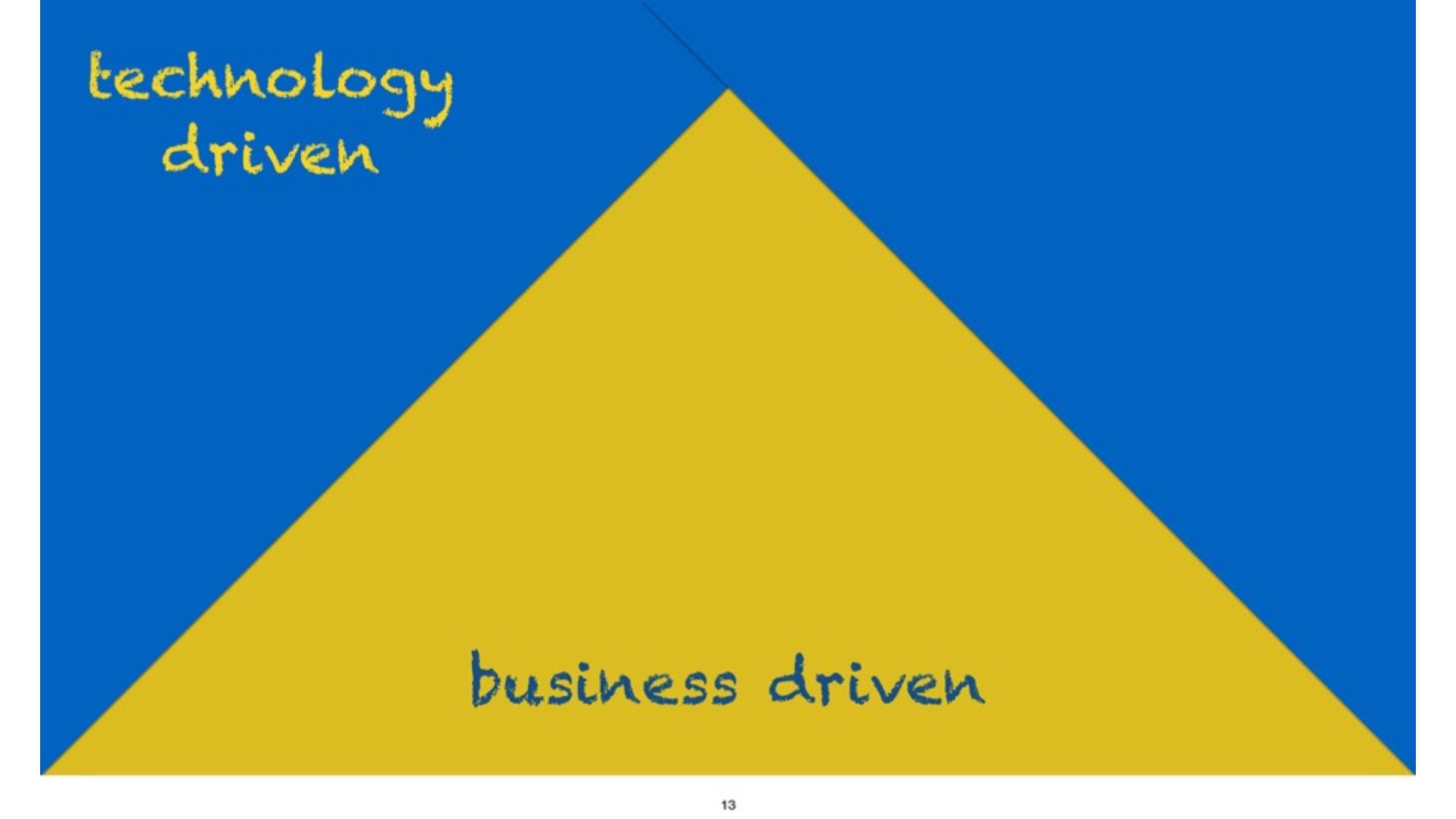


business driven

technology
driven



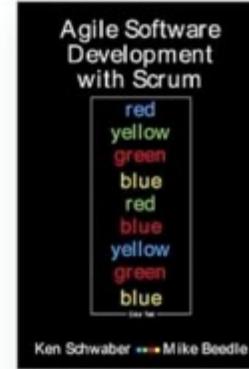
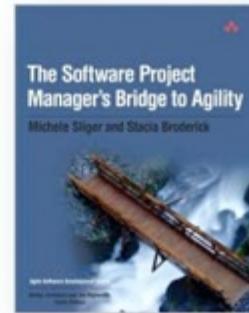
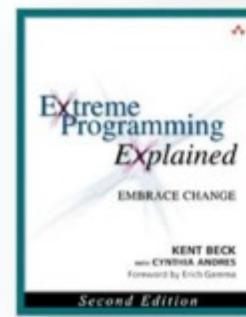
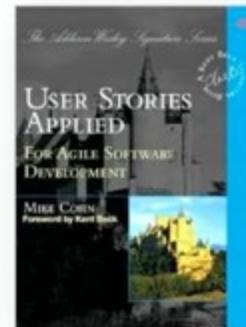
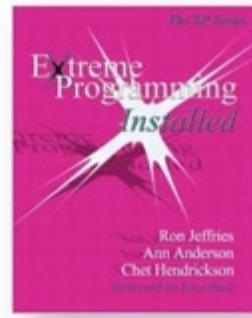
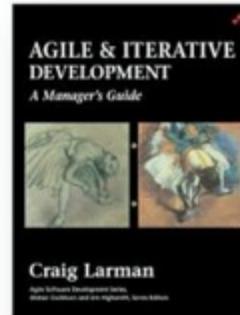
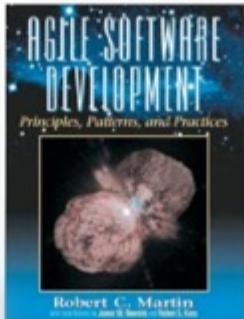
business driven

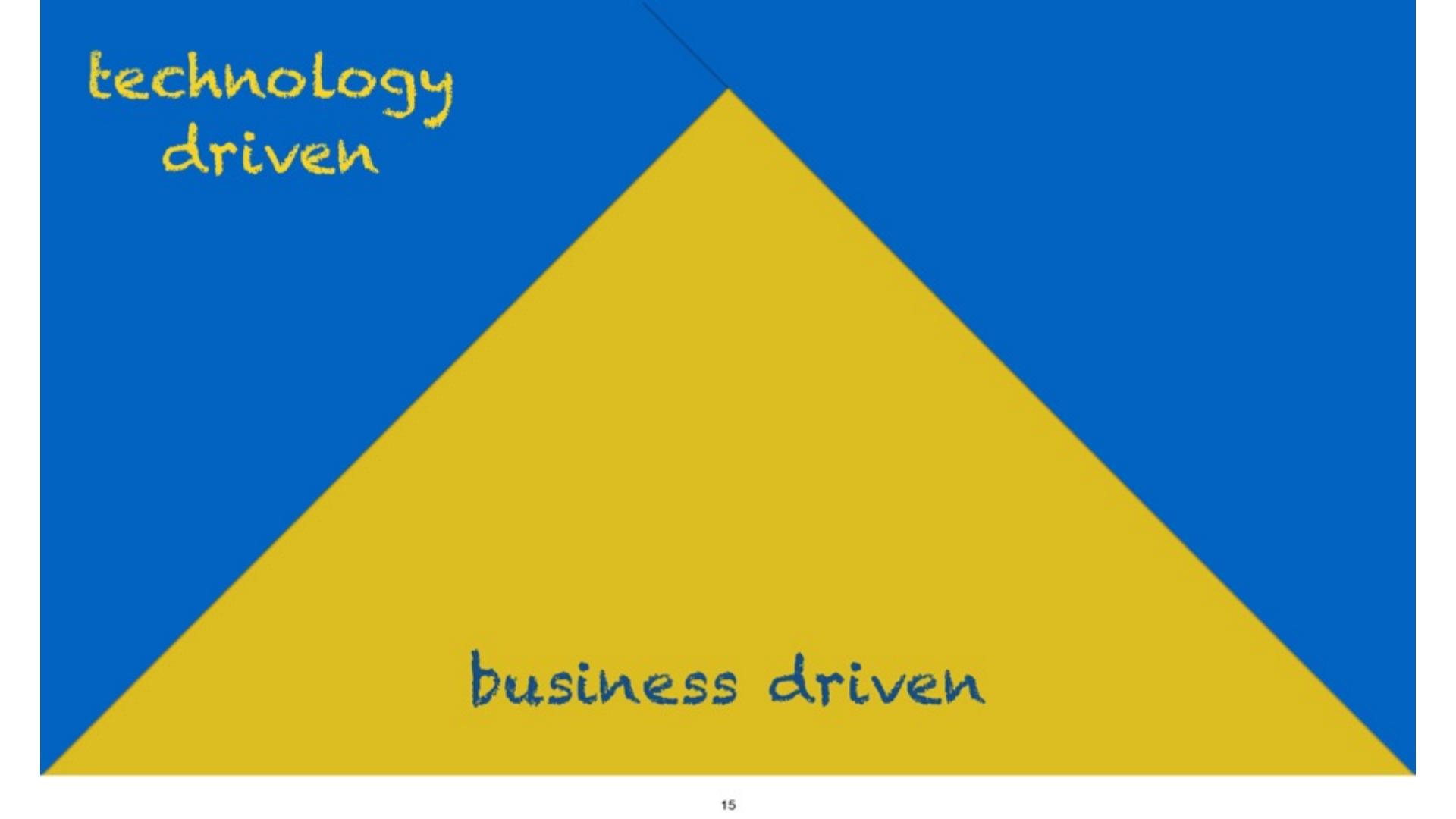
The image features a large, solid yellow triangle centered on a blue background. The blue area is bounded by the triangle's sides and extends beyond them to form a wide base at the bottom.

technology
driven

business driven

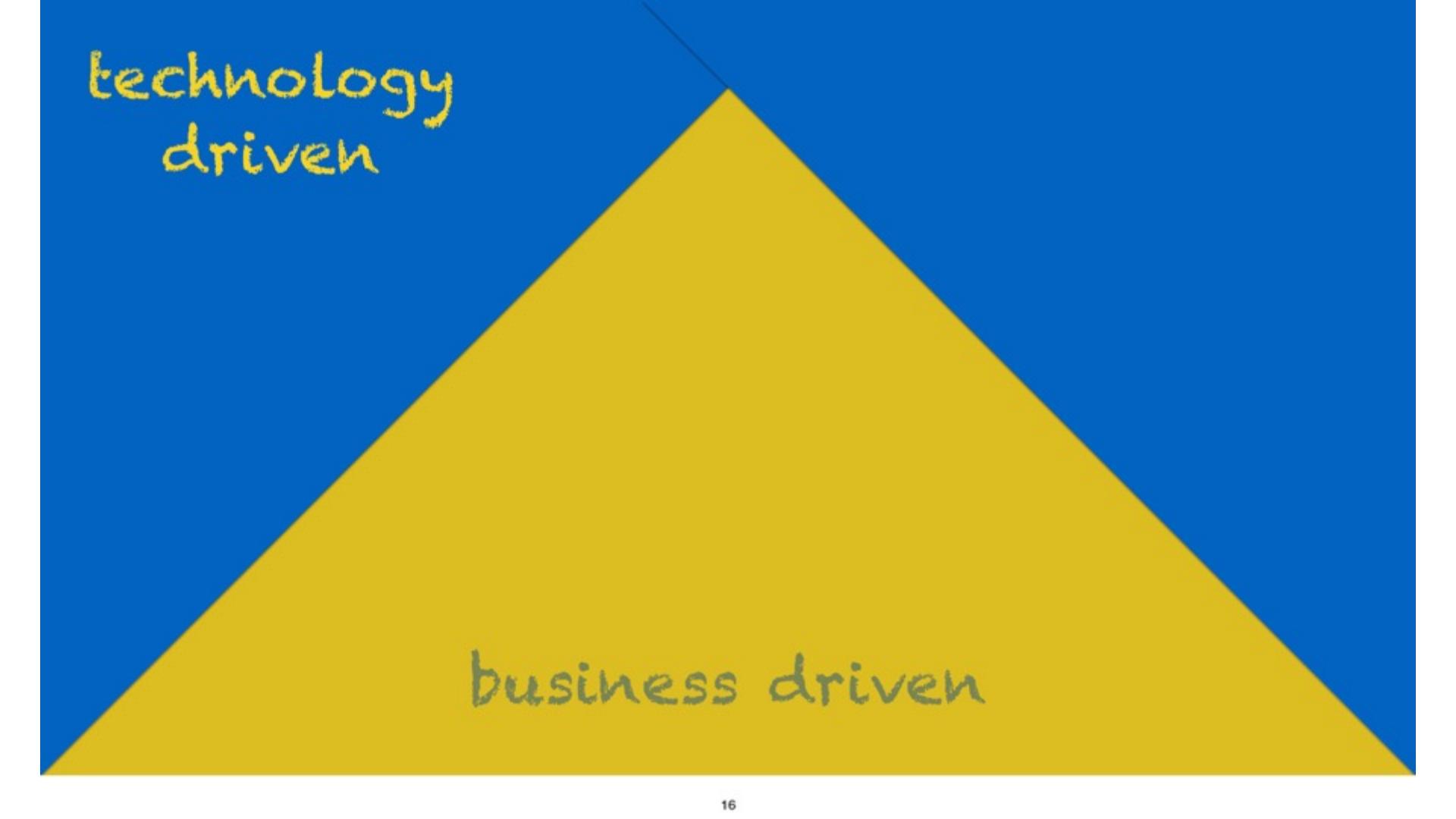
business driven



The image features a large, solid yellow triangle centered against a solid blue background. The triangle's vertices point downwards, creating a sense of depth. It is positioned in the lower half of the frame.

technology
driven

business driven

The image features a large, solid yellow triangle centered against a solid blue background. The yellow triangle is oriented with its apex pointing downwards. In the upper left quadrant of the blue area, the words "technology driven" are written in a yellow, handwritten-style font. In the lower right quadrant of the yellow area, the words "business driven" are written in a grey, handwritten-style font.

technology
driven

business driven

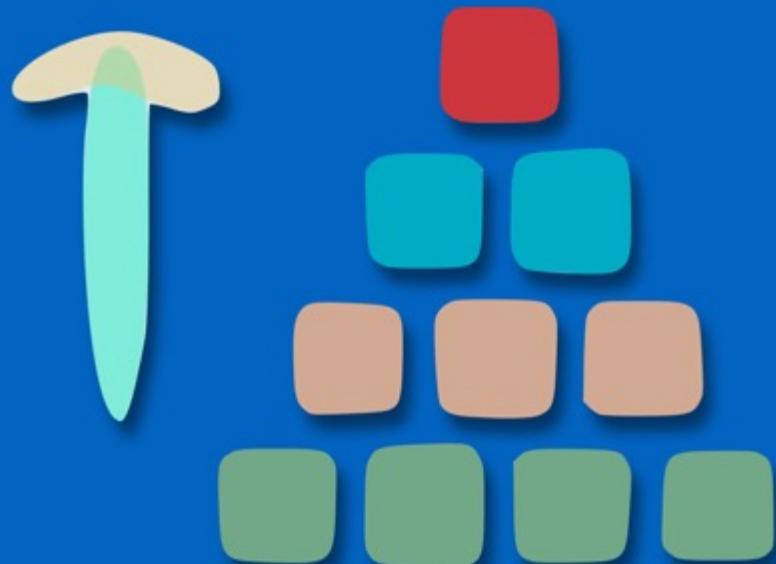
everything chAnGEs
all the time!

dYNaMic eqUiLiBRiuM

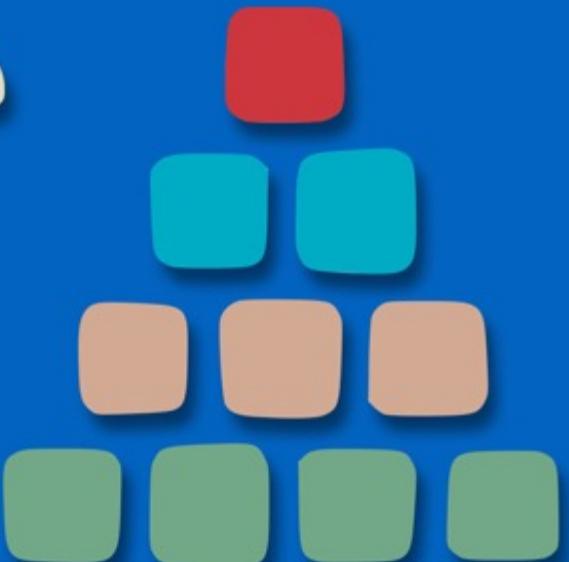
dYNaMic eqUiLiBRiuM



dYNaMic eqUiLiBRiuM



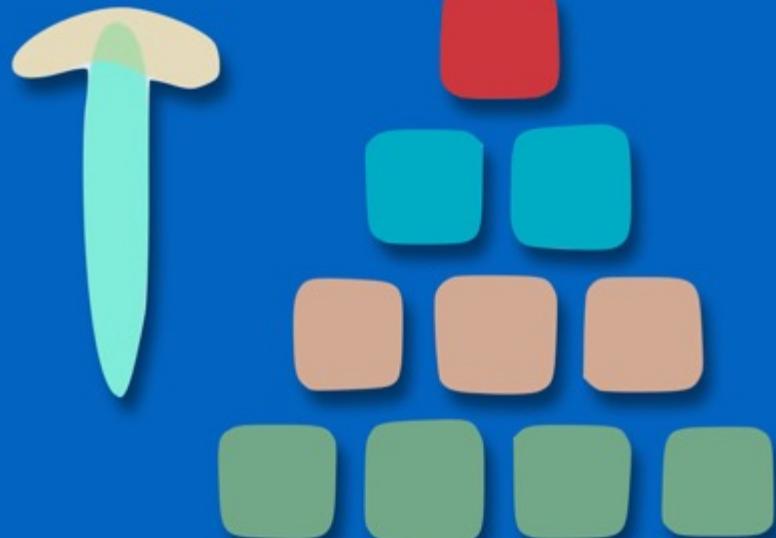
dYNaMic eqUiLiBRiuM



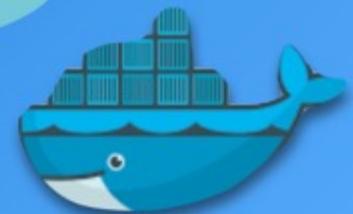
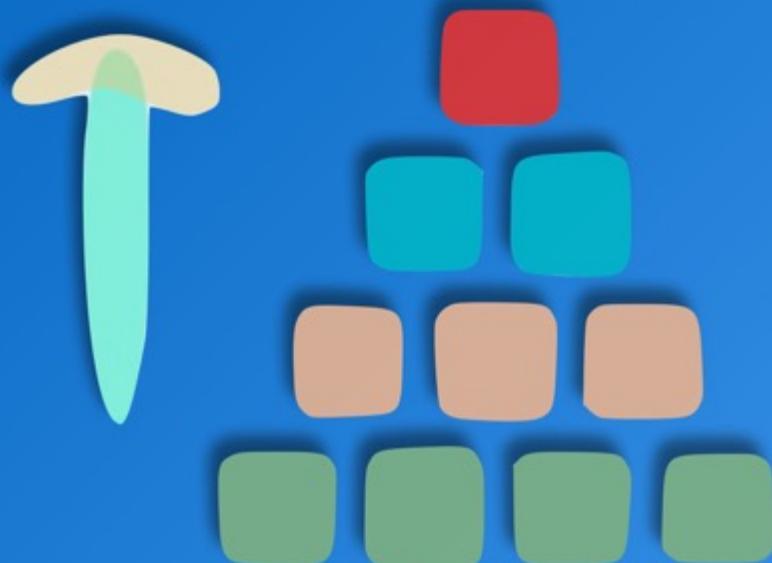
dYNaMic eqUiLiBRiuM



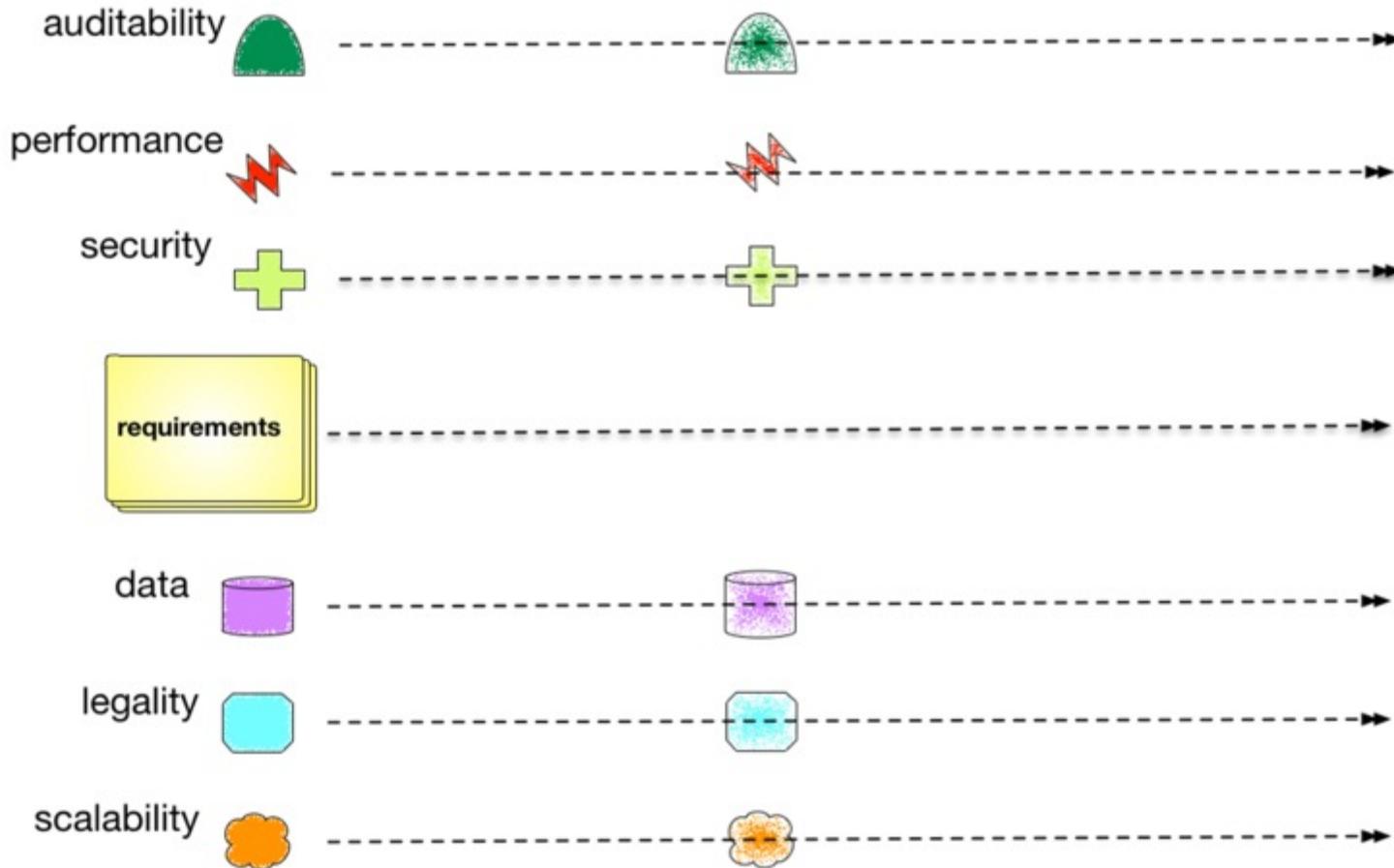
dYNaMic eqUiLiBRiuM



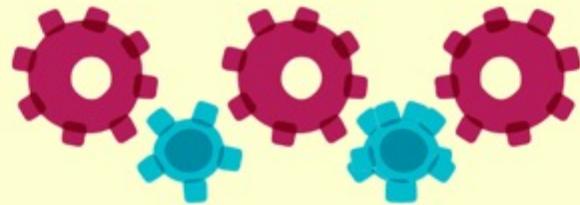
dYNaMic eqUiLiBRiuM



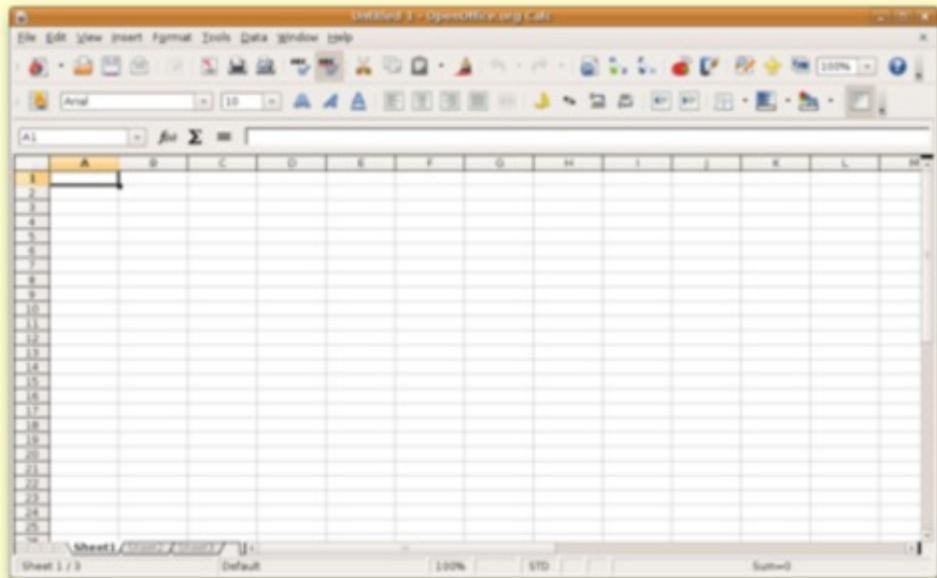
**How is long term planning
possible when things constantly
change in unEXpECtEd ways?**



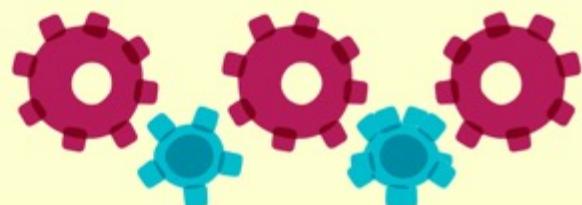
Penultima ↑ e



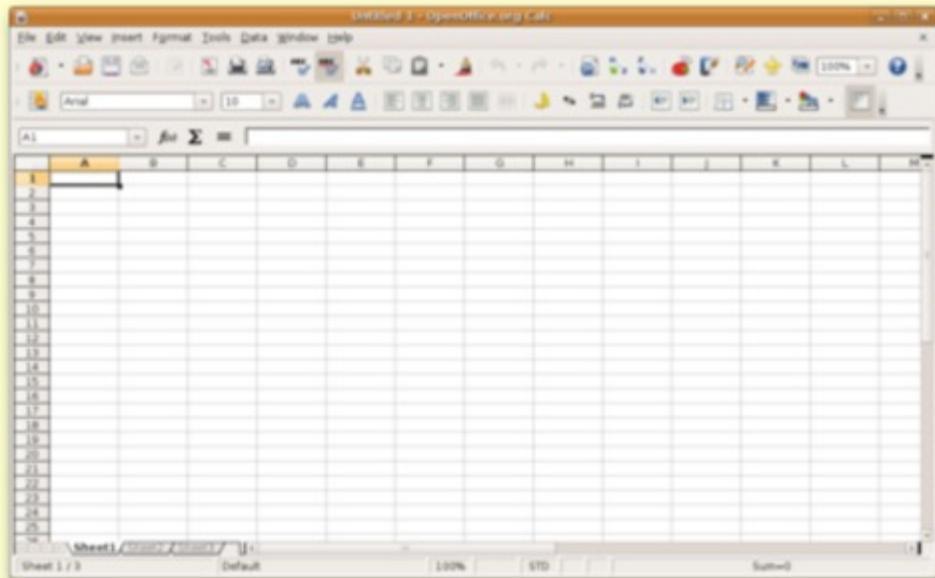
EA Spreadsheet



Penultima ↑ e

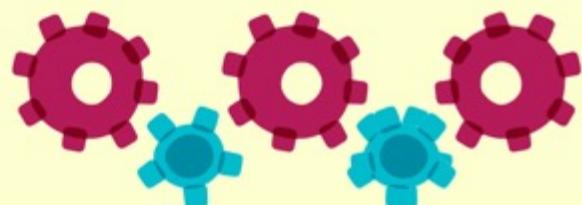


EA Spreadsheet

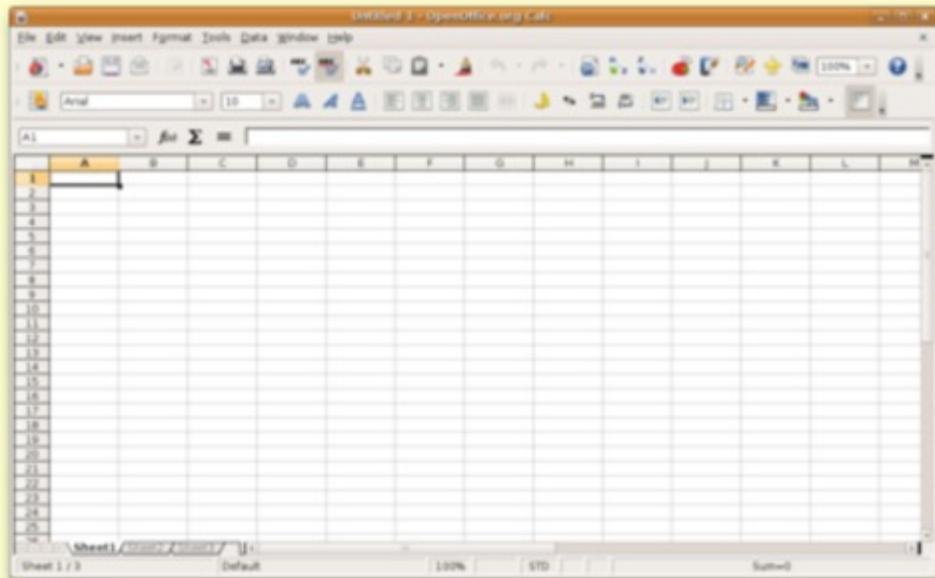


✓ definition

Penultima ↑ e



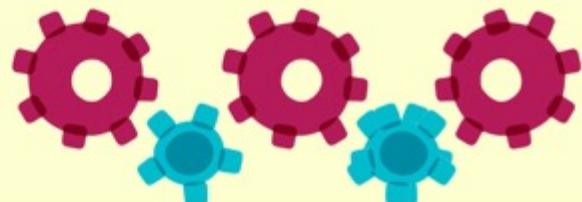
EA Spreadsheet



✓ definition

! verification

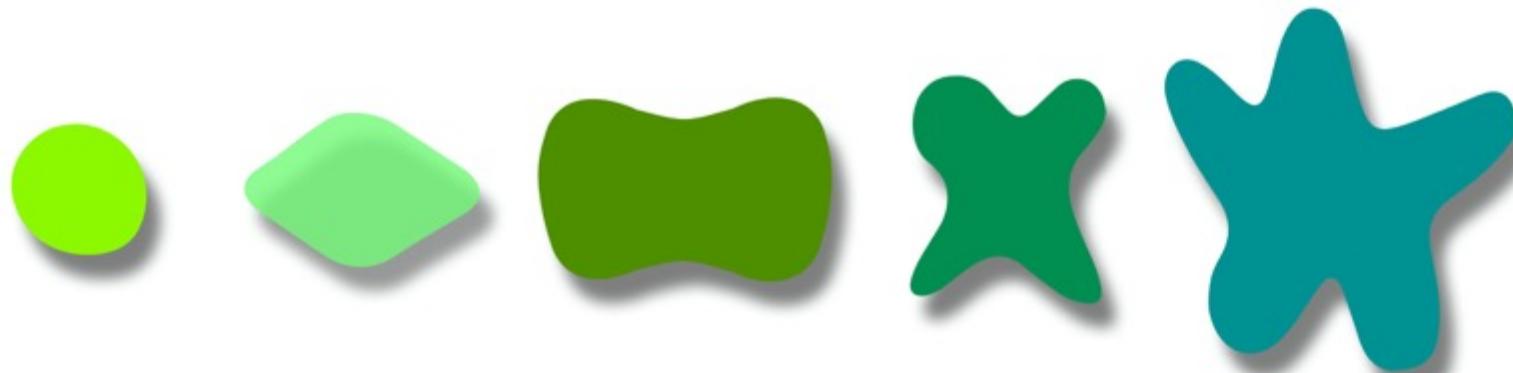
Penultima ↑ e



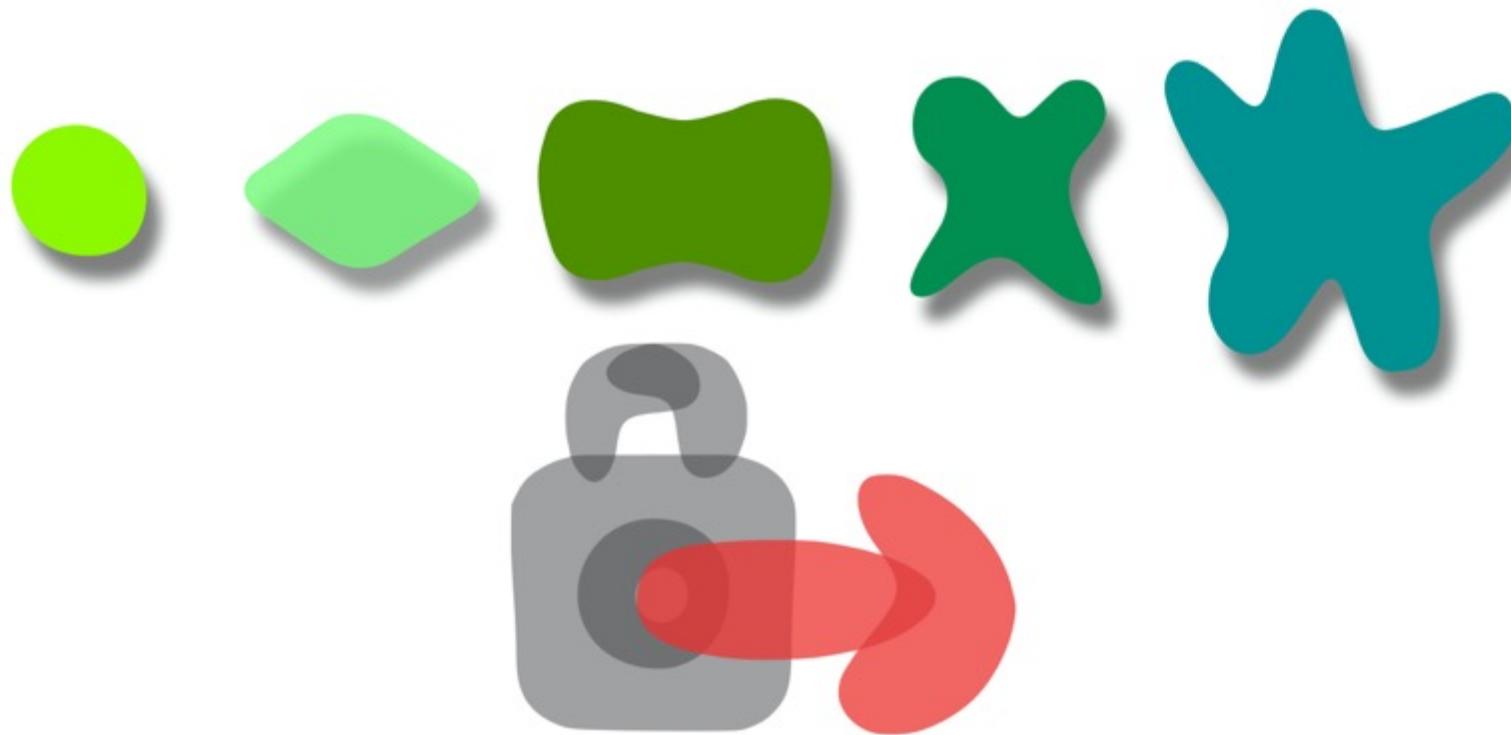
**Once I've built an architecture,
how can I prevent it from
gradually dEgRADiNg over time?**

sEcOND-ORdeR eFfEcT

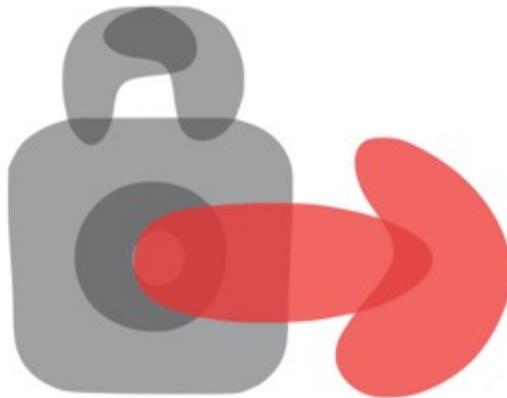
sEcOND-ORdeR eFfEcT



sEcOND-ORdeR eFfEcT



governance



Evolutionary Architecture

Evolutionary Architecture

Continuous Architecture?

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

Adaptable Architecture?

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

Adaptable Architecture?

Evolutionary Architecture

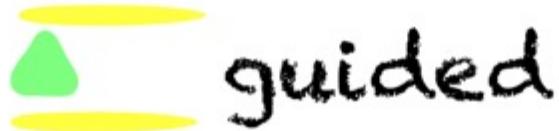
An evolutionary architecture supports
guided,
incremental change
across multiple dimensions.



Evolutionary Architecture

An evolutionary architecture supports
guided
incremental change
across multiple dimensions.





guided

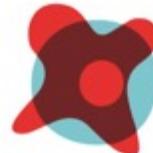
evolutionary computing fitness function:

a particular type of objective function that is used to summarize...how close a given design solution is to achieving the set aims.

Traveling Salesman Problem



Traveling Salesman Problem

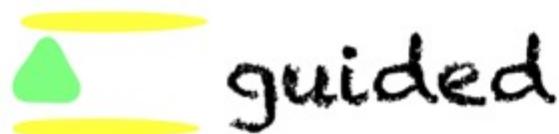


fitness function = length of route



guided

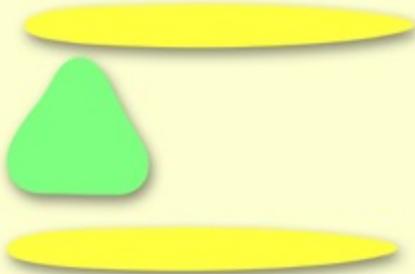




architectural fitness function:

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

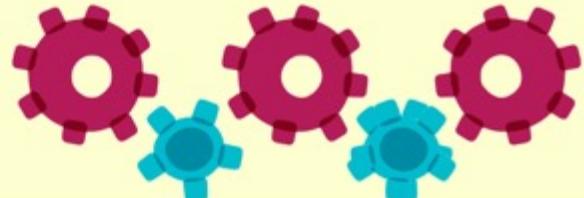
EA Spreadsheet



✓ definition

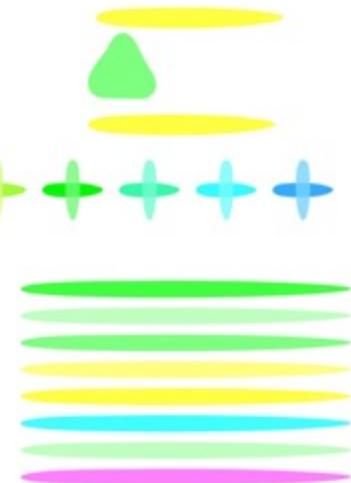
✓ verification

Penultima ↑ e



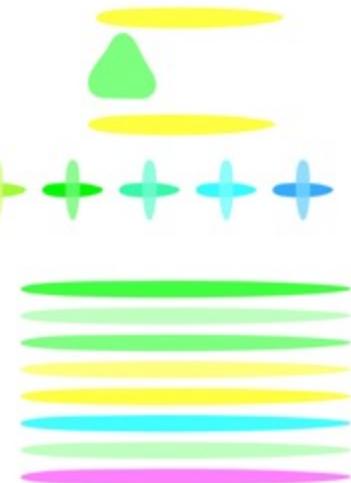
Evolutionary Architecture

An evolutionary architecture supports
guided
incremental change across multiple dimensions.

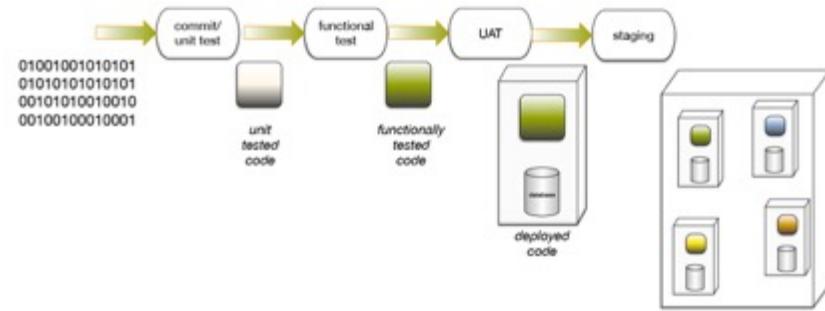
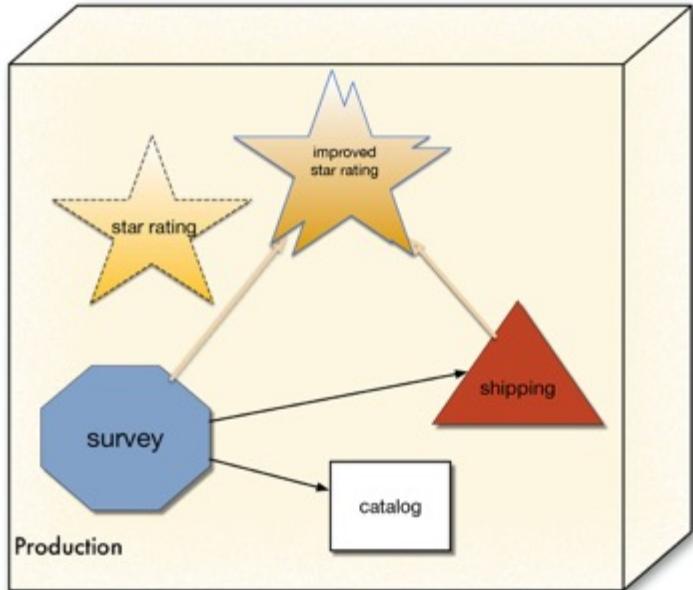


Evolutionary Architecture

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

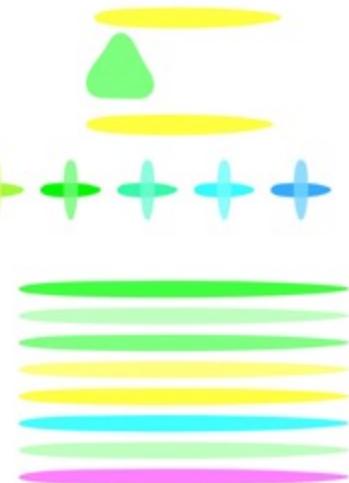


+++++ incremental



Evolutionary Architecture

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



Evolutionary Architecture

An evolutionary architecture supports
guided,

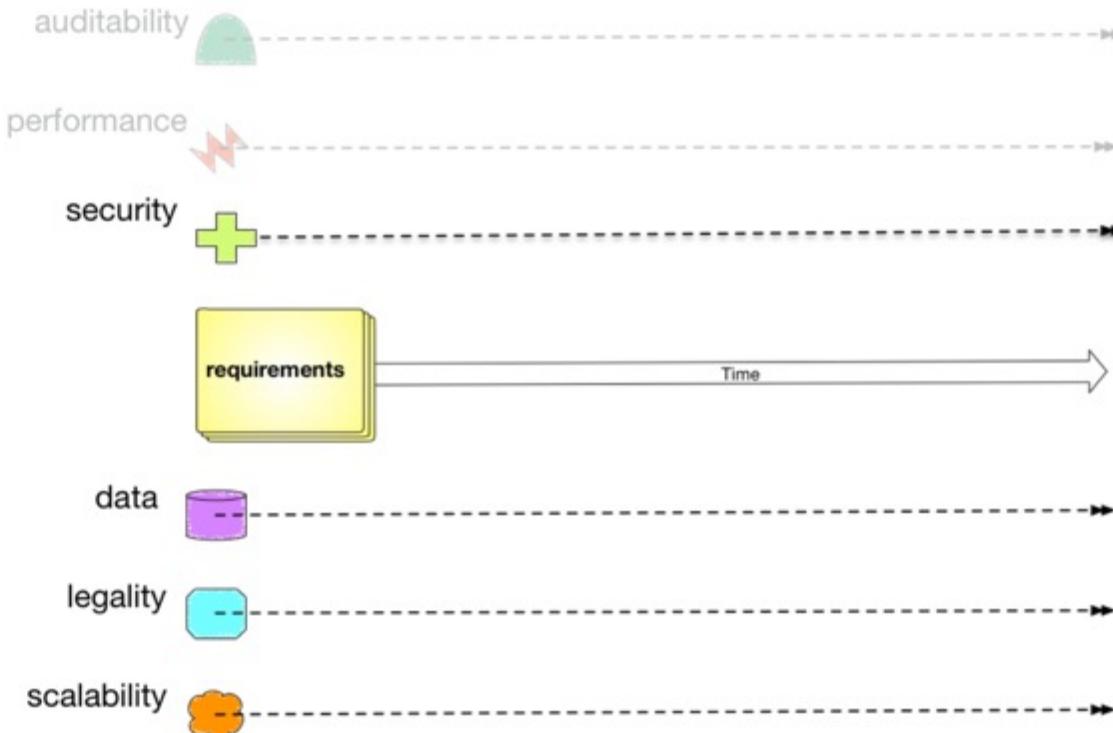
incremental change

across multiple dimensions



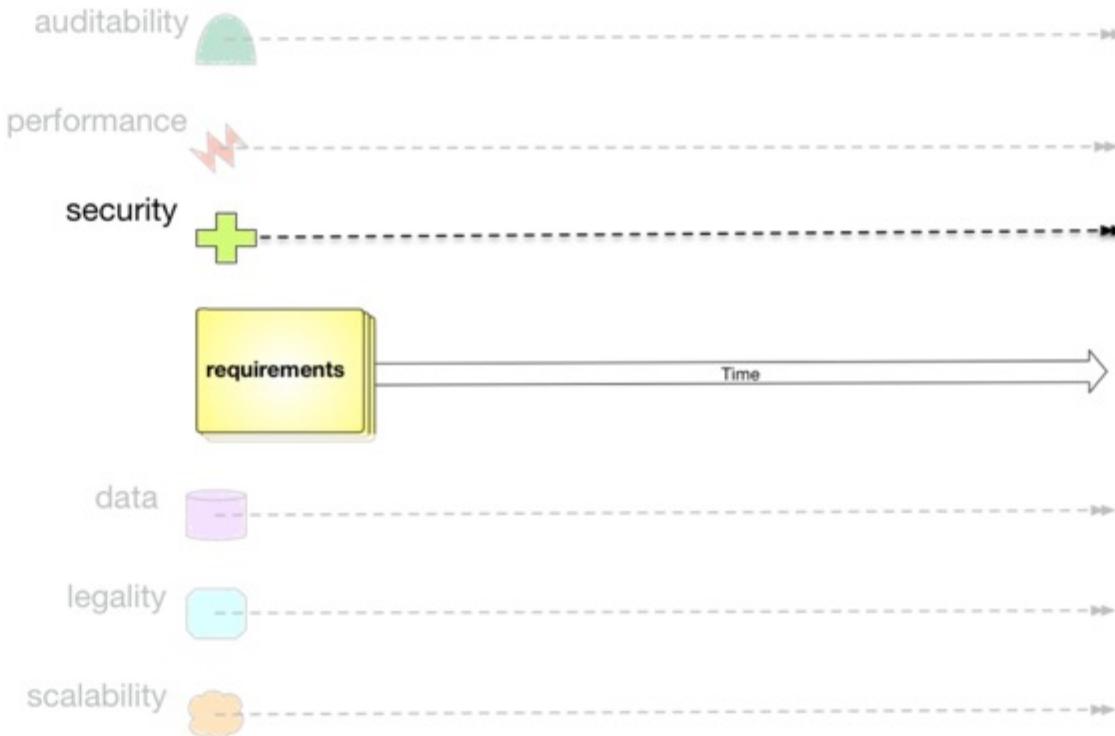


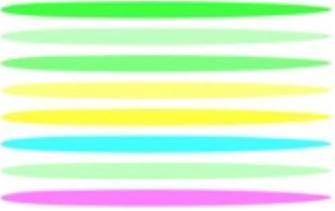
multiple dimensions



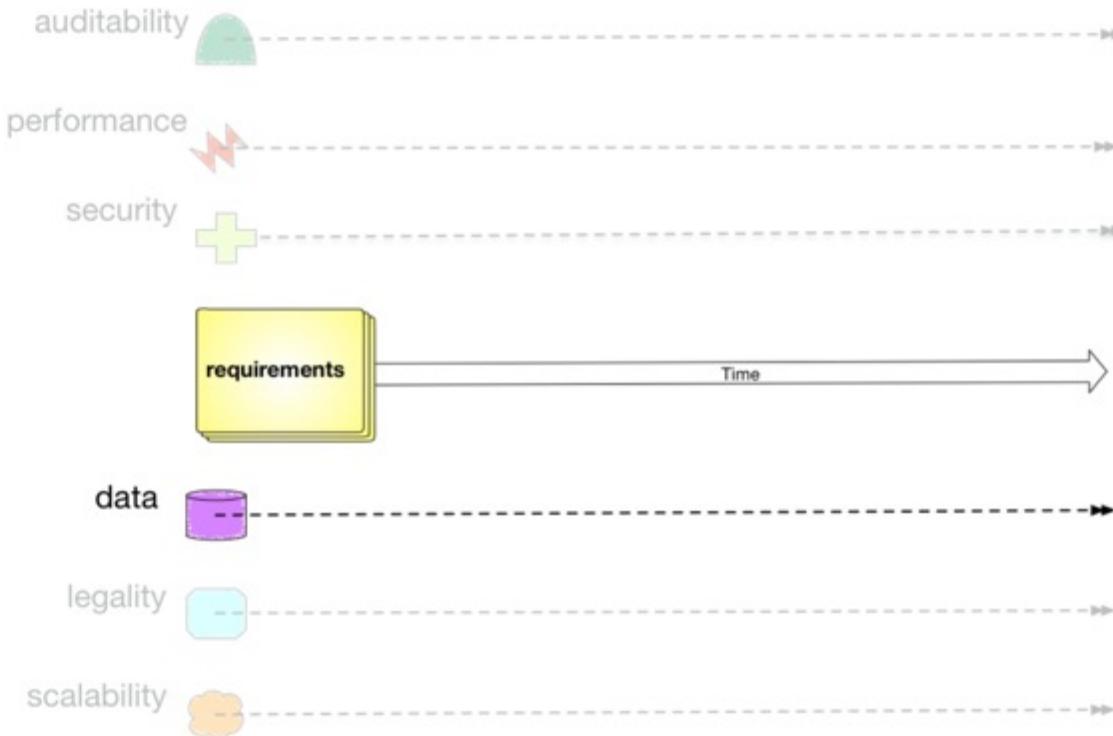


multiple dimensions





multiple dimensions



Evolutionary Architecture

An evolutionary architecture supports
guided,

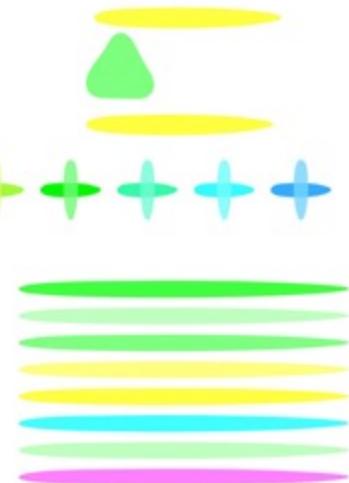
incremental change

across multiple dimensions



Evolutionary Architecture

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

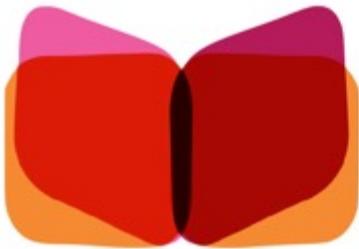




Agenda Part 1

Definition

Agenda Part 1



Definition



Guided Change via Fitness Functions



Agenda Part 1



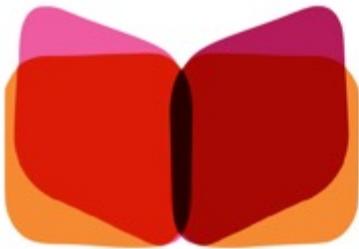
Definition



Guided Change via Fitness Functions



Agenda Part 1



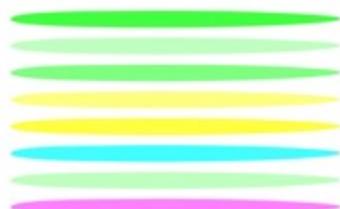
Definition



Guided Change via Fitness Functions



Structuring architecture for evolution



Agenda Part 1



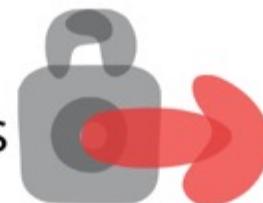
Definition



Guided Change via Fitness Functions



Incremental Change

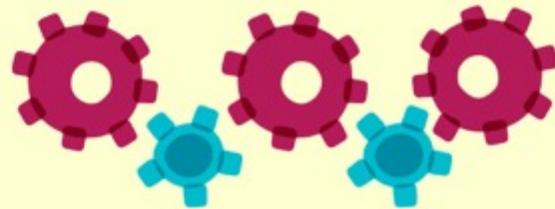


Structuring architecture for evolution



Agenda Part 2

The Evolution of
Penultima↑e



Agenda Part 2

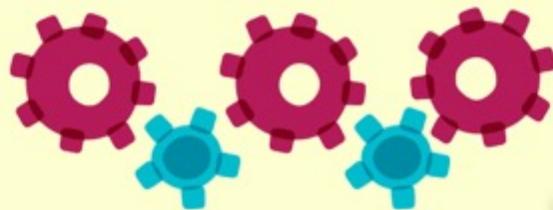
fitness functions

architecture migration

experimentation

automating
governance

The Evolution of Penultima ↑e

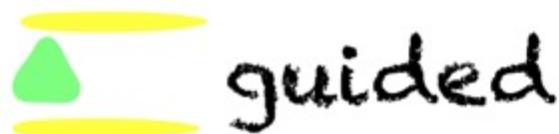


hypothesis/data driven
development

Evolutionary Architecture

An evolutionary architecture supports
guided
incremental change
across multiple dimensions.

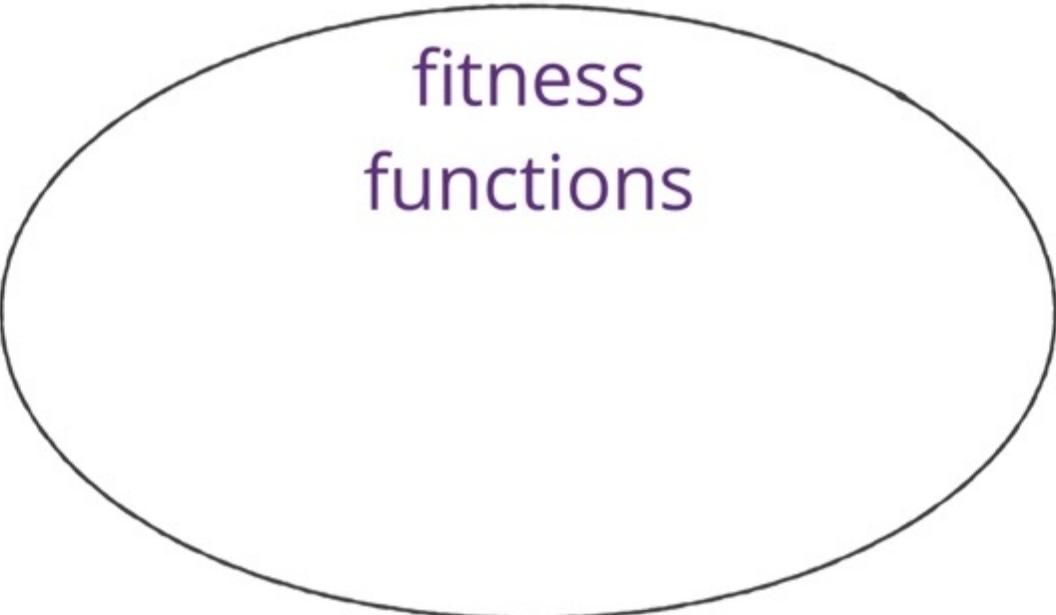




architectural fitness function:

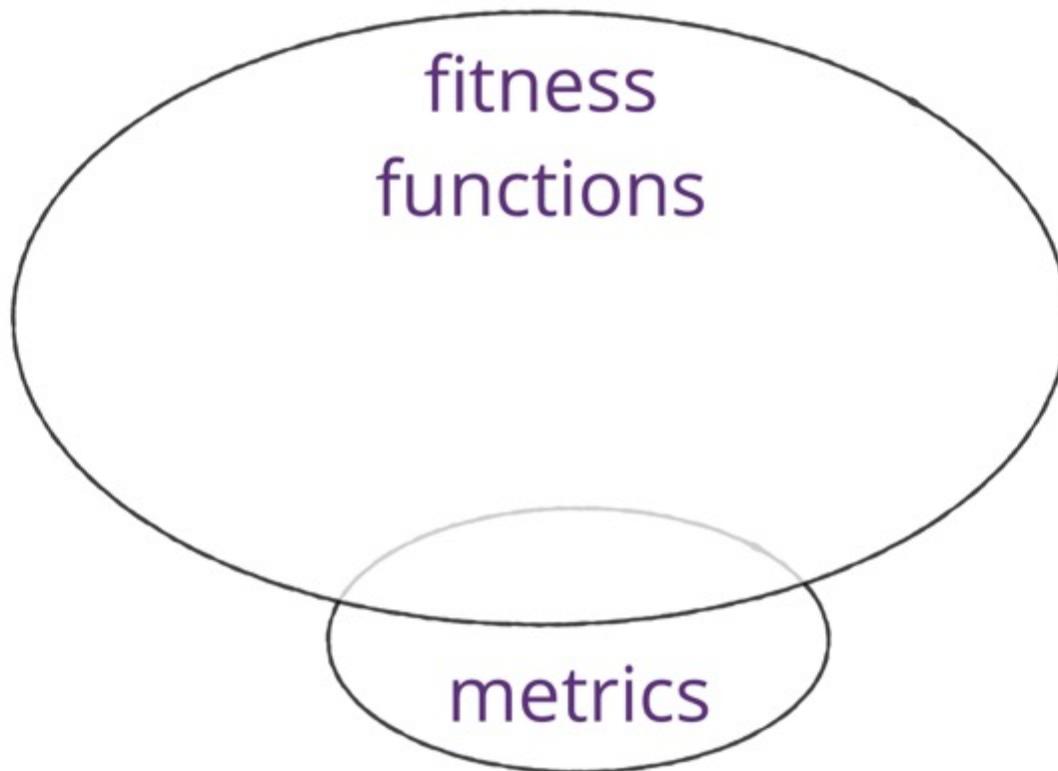
An architectural fitness function provides an objective integrity assessment of some architectural characteristic(s).

Fitness Functions

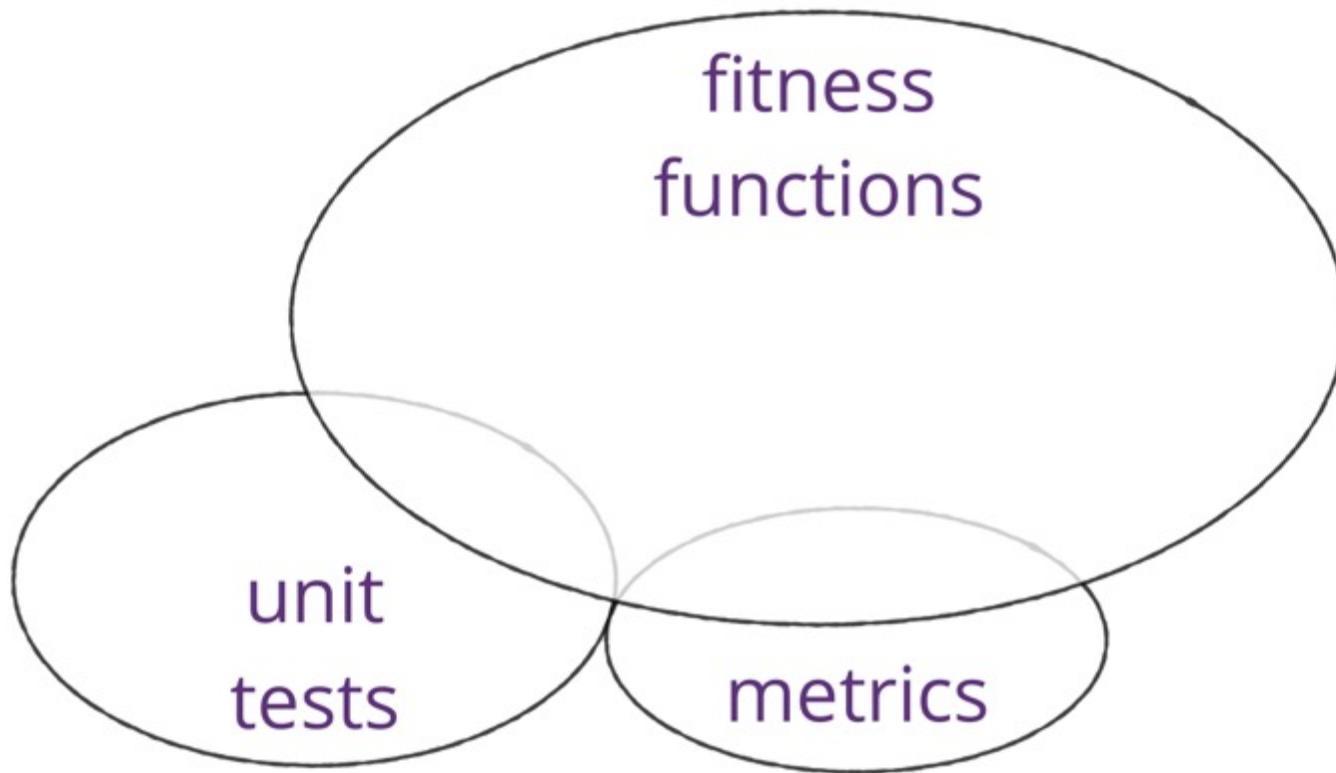


fitness
functions

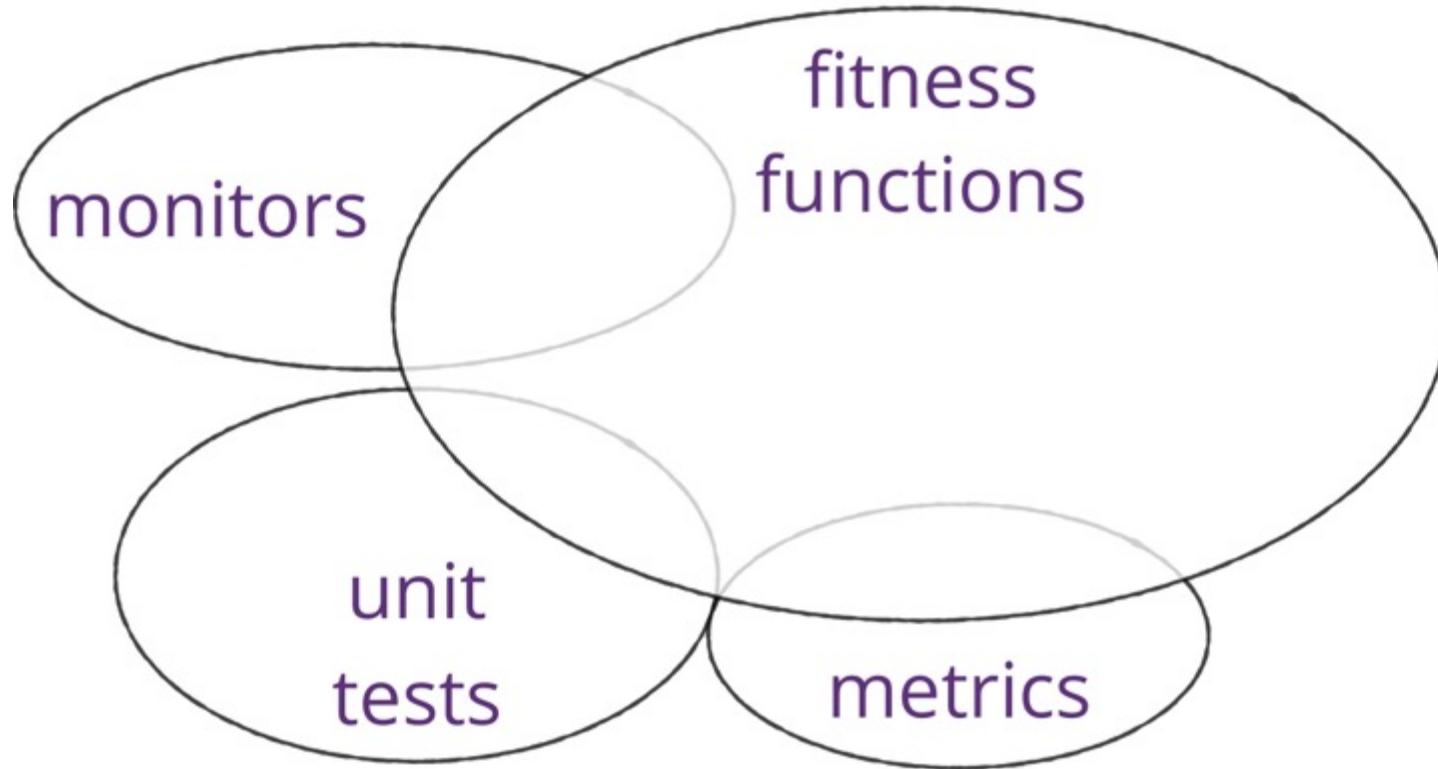
Fitness Functions



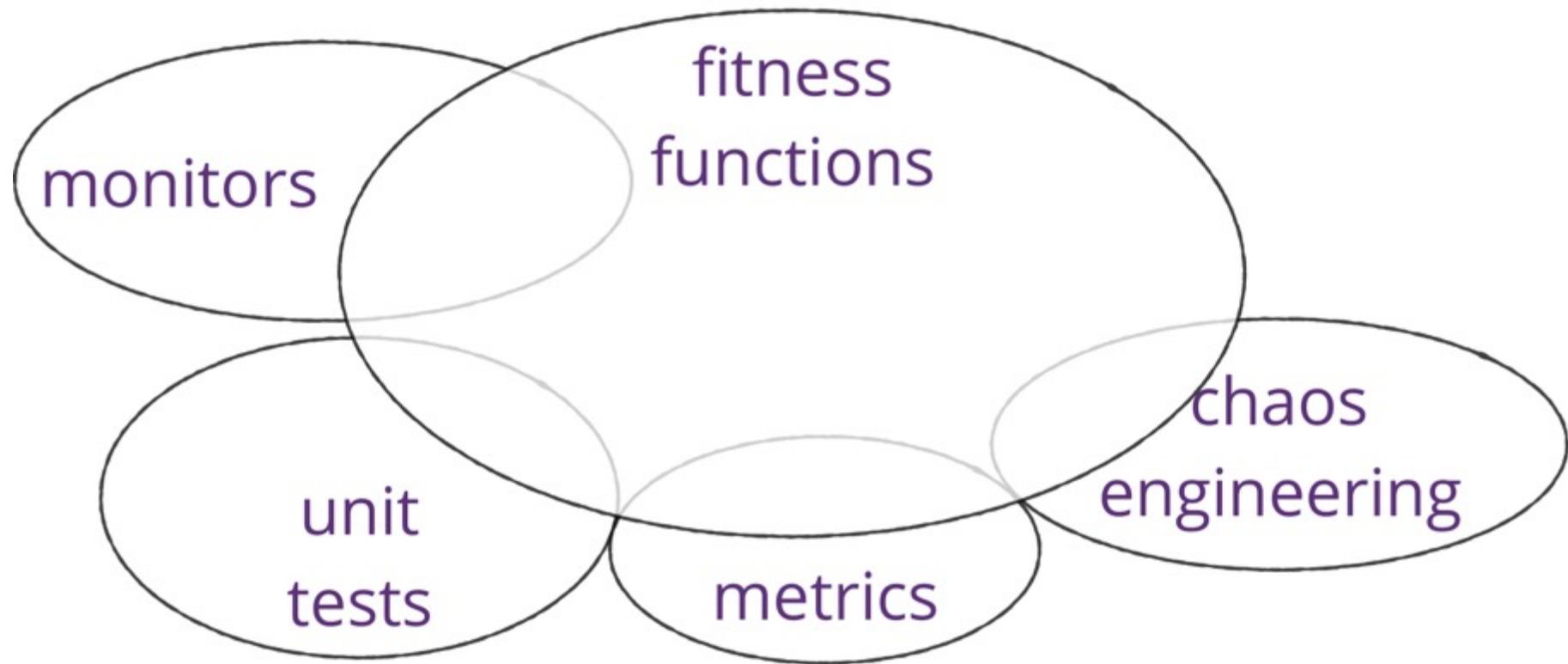
Fitness Functions



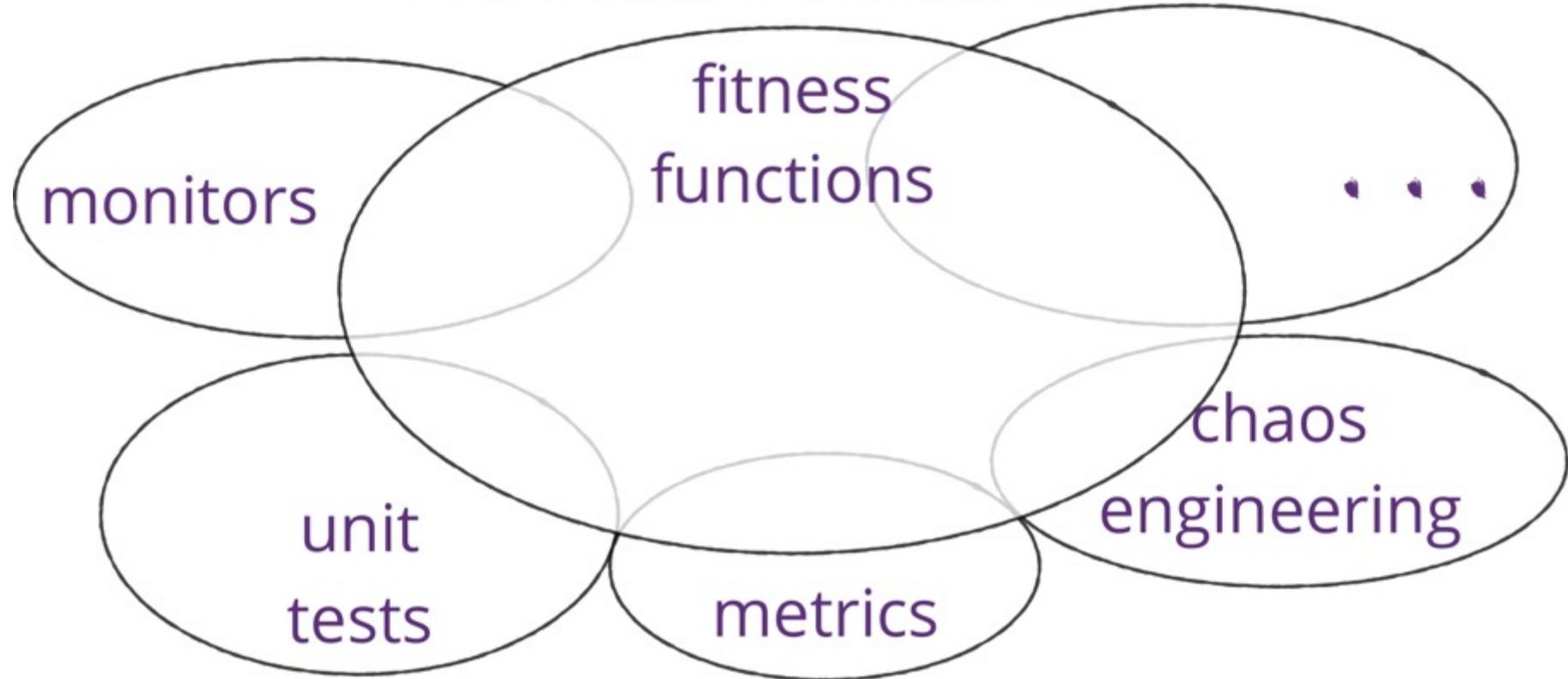
Fitness Functions



Fitness Functions

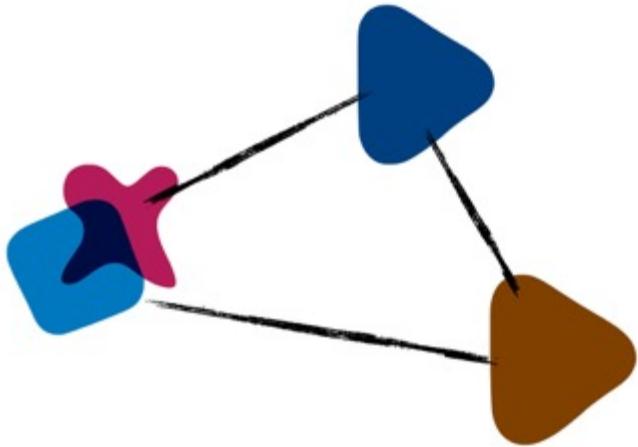


Fitness Functions

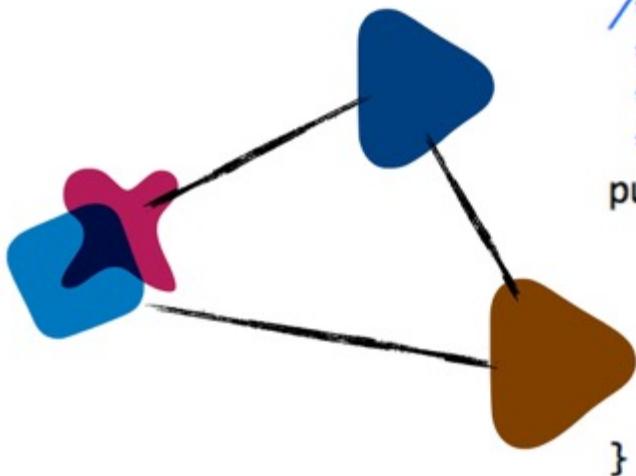


Cyclic Dependency Function

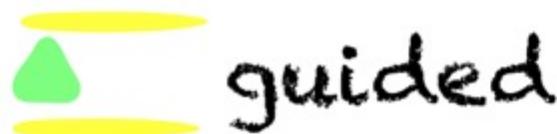
Cyclic Dependency Function



Cyclic Dependency Function



```
/**  
 * 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());  
}
```



architectural fitness function:

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

Categories of Fitness Functions



run against a singular context and exercise one particular aspect of the architecture.

Categories of Fitness Functions



run against a singular context and exercise one particular aspect of the architecture.



run against a shared context and exercise a combination of architectural aspects such as security and scalability

Categories of Fitness Functions



run based on a particular event:
— developer executing a unit test

Categories of Fitness Functions



- run based on a particular event:
 - developer executing a unit test
 - deployment pipeline running tests

Categories of Fitness Functions



- run based on a particular event:
 - developer executing a unit test
 - deployment pipeline running tests
 - timed task

Categories of Fitness Functions



- run based on a particular event:
 - developer executing a unit test
 - deployment pipeline running tests
 - timed task



- executes constant verification of architectural aspect(s)

Categories of Fitness Functions

static



have a fixed result, such as the binary pass/fail of a unit test.

Categories of Fitness Functions

static



have a fixed result, such as the binary pass/fail of a unit test.

dynamic



rely on a shifting definition based on extra context.

Categories of Fitness Functions



tests and other verification mechanism that run without human interaction.

Categories of Fitness Functions

automated



tests and other verification mechanism that run without human interaction.

manual



must involve at least one human.

Categories of Fitness Functions

temporal



architects may want to build a time component into assessing fitness

Categories of Fitness Functions

temporal



architects may want to build a time component into assessing fitness

break on upgrade

Categories of Fitness Functions

temporal



architects may want to build a time component into assessing fitness

break on upgrade

overdue library update

Categories of Fitness Functions

domain-specific



Some architectures have specific concerns,
such as special security or regulatory
requirements

Categories of Fitness Functions

architectural
characteristic

domain-specific



Categories of Fitness Functions

architectural
characteristic

domain-specific



problem domain

unit



UAT



functional



Categories of Fitness Functions



architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...

Categories of Fitness Functions



architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...

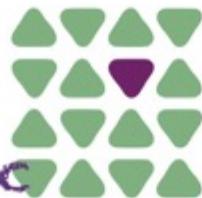
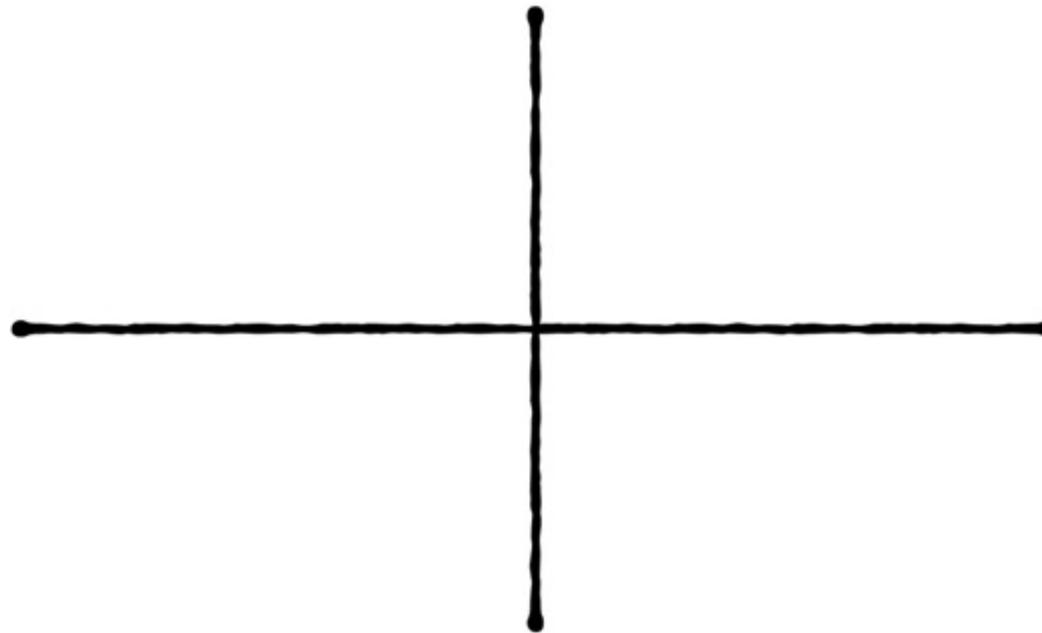


...some fitness functions will emerge during development of the system

Fitness Function



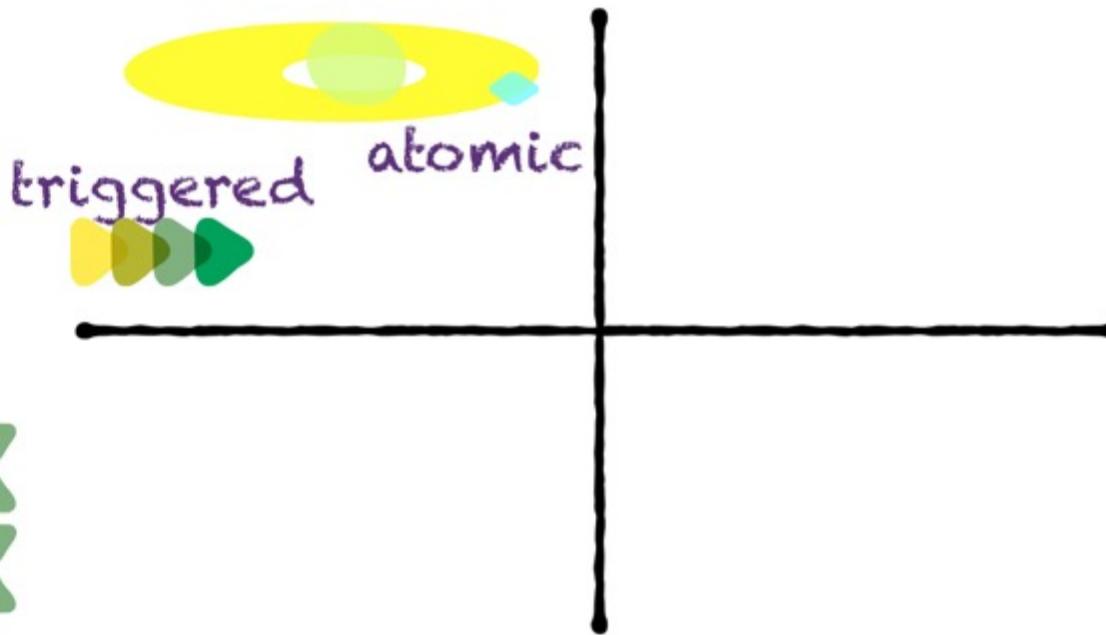
atomic



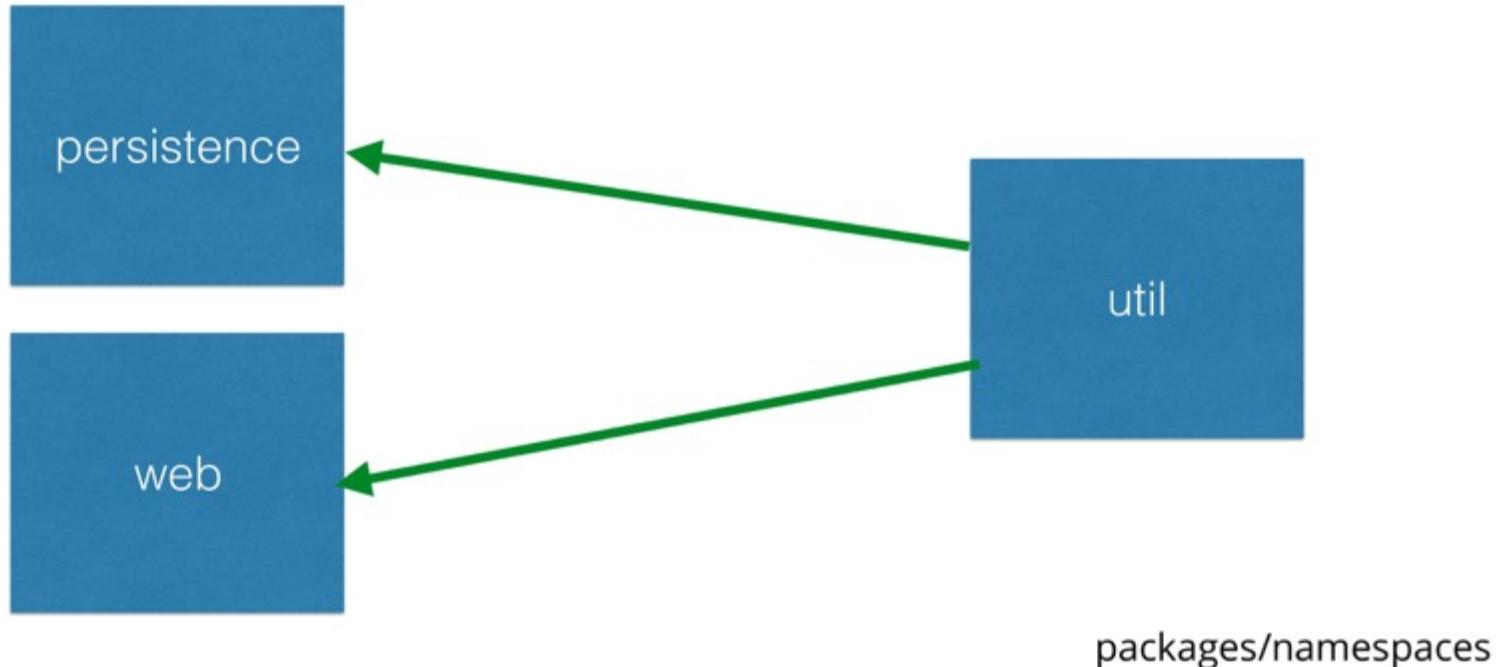
holistic



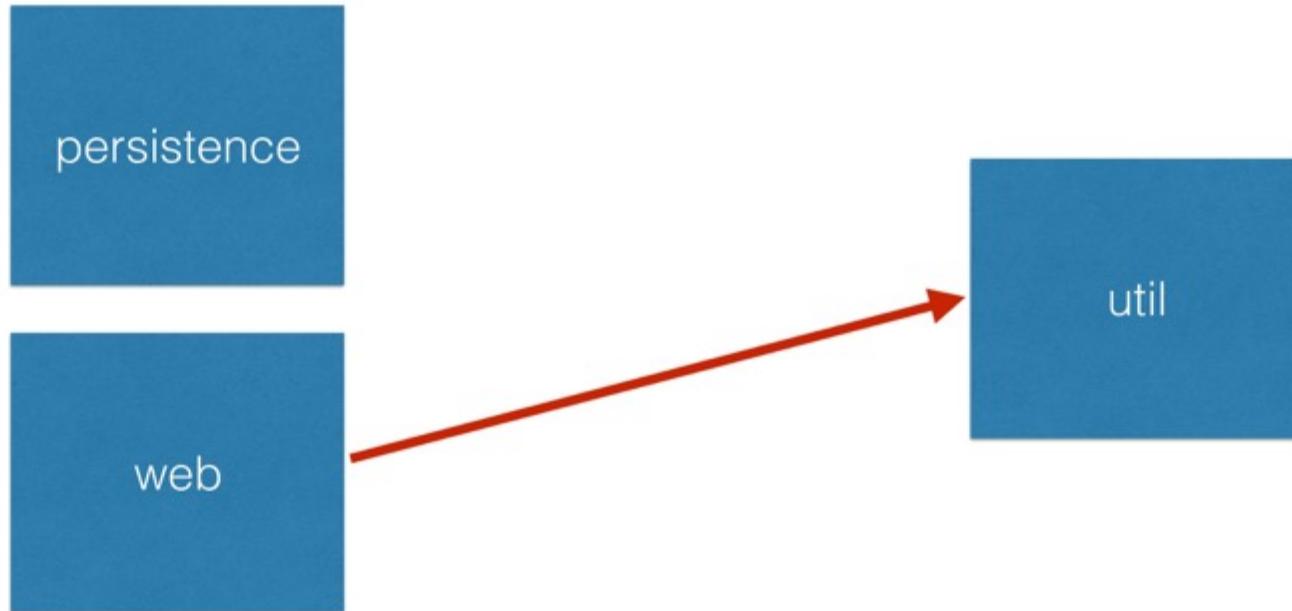
Fitness Function



Directionality of Imports



Directionality of Imports

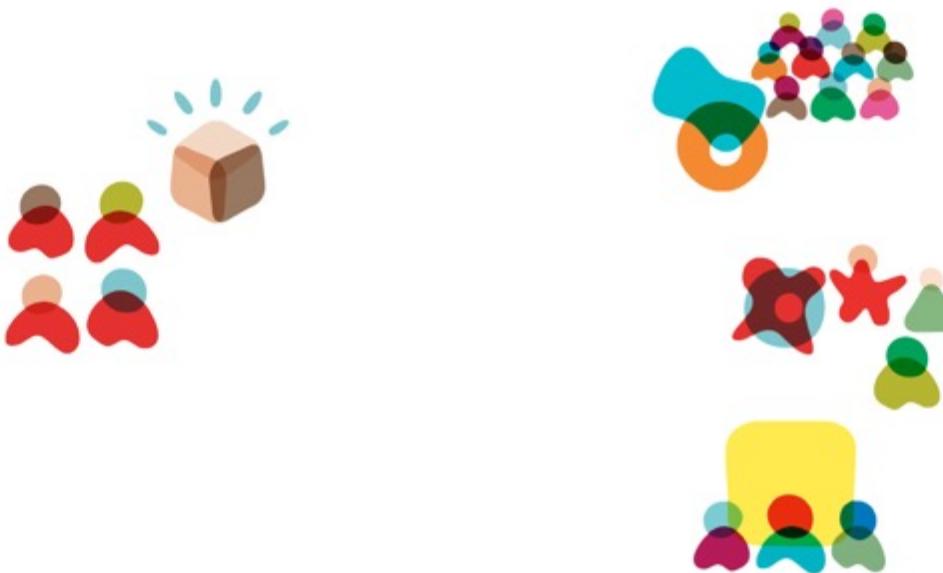


packages/namespaces

Coupling Fitness Function

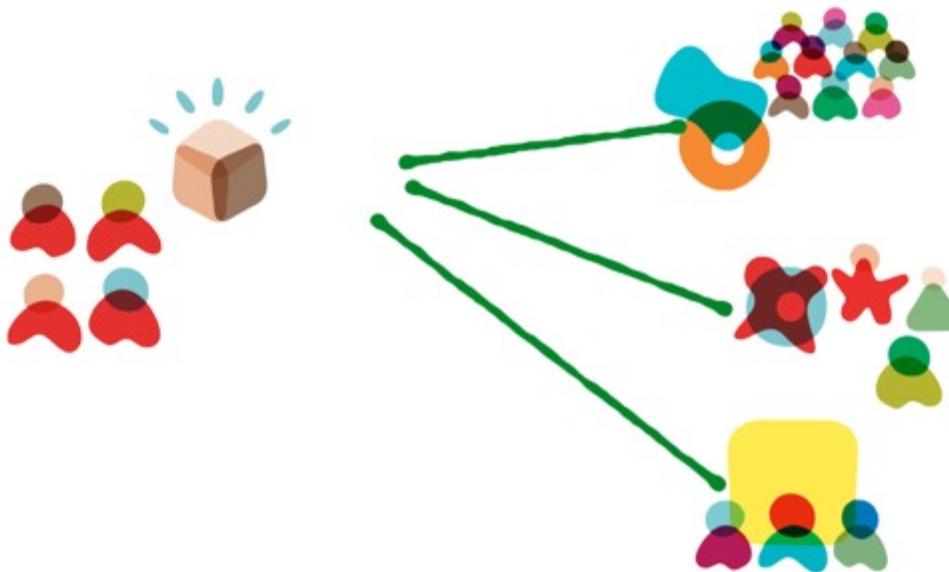
```
public void testMatch() {  
    DependencyConstraint constraint = new DependencyConstraint();  
  
    JavaPackage persistence = constraint.addPackage("com.xyz.persistence");  
    JavaPackage web = constraint.addPackage("com.xyz.web");  
    JavaPackage util = constraint.addPackage("com.xyz.util");  
  
    persistence.dependsUpon(util);  
    web.dependsUpon(util);  
  
    jdepend.analyze();  
  
    assertEquals("Dependency mismatch",  
                true, jdepend.dependencyMatch(constraint));  
}
```

Consumer Driven Contracts



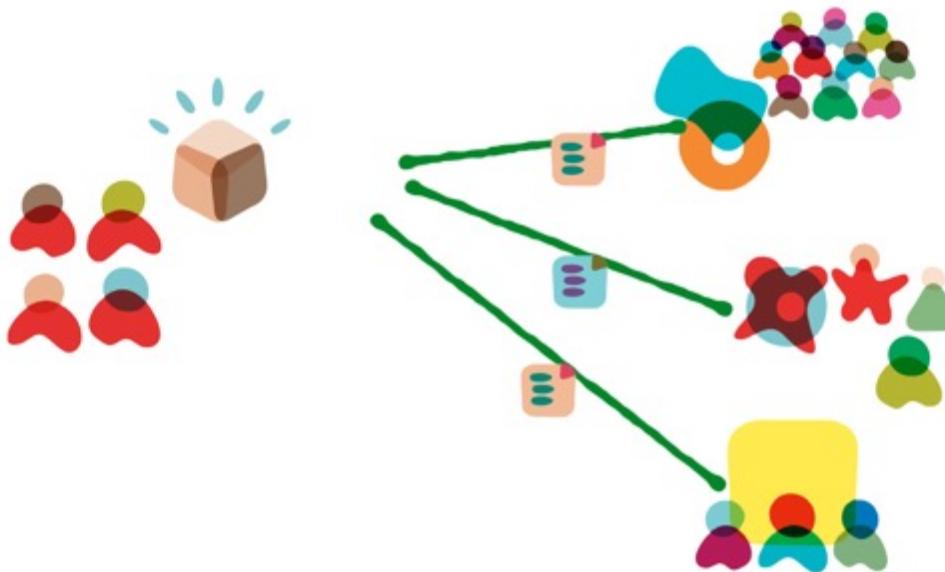
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



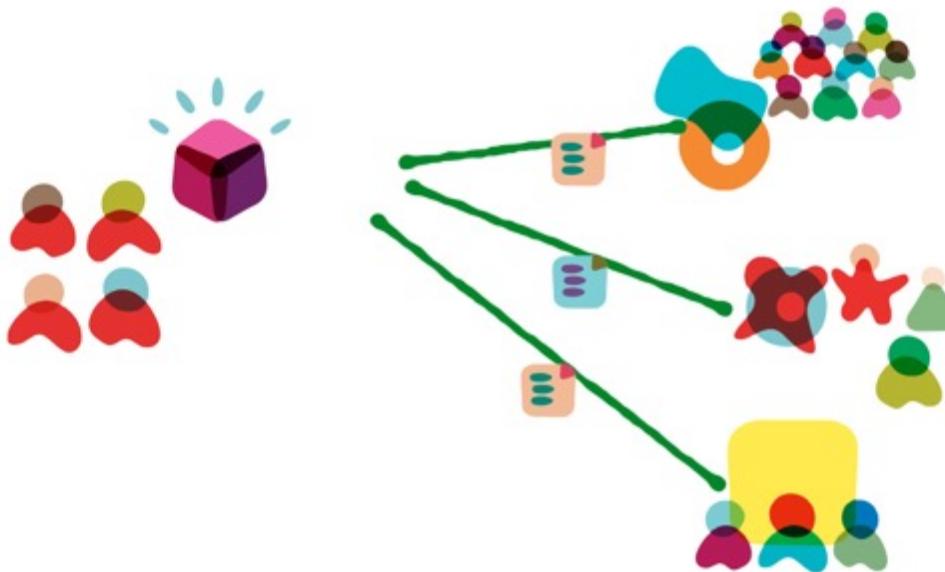
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



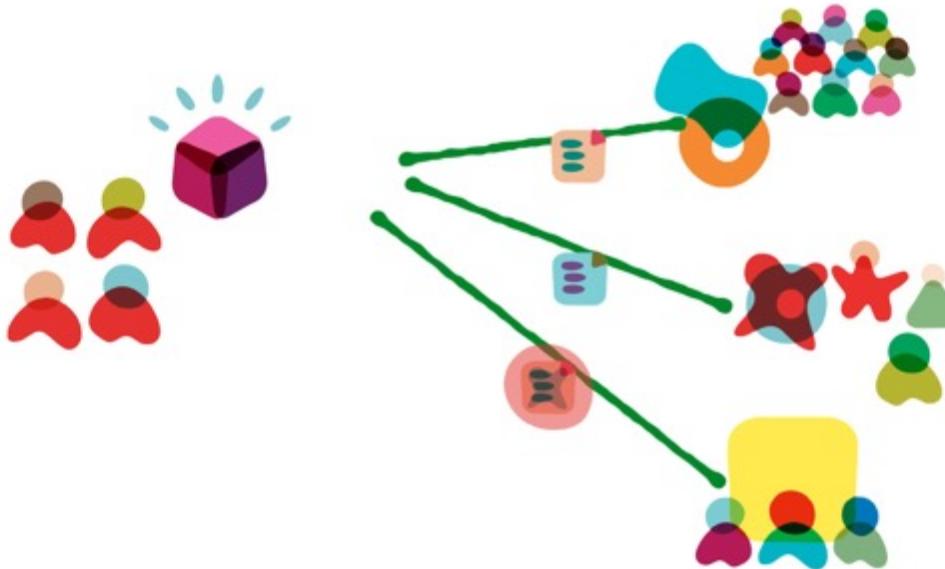
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



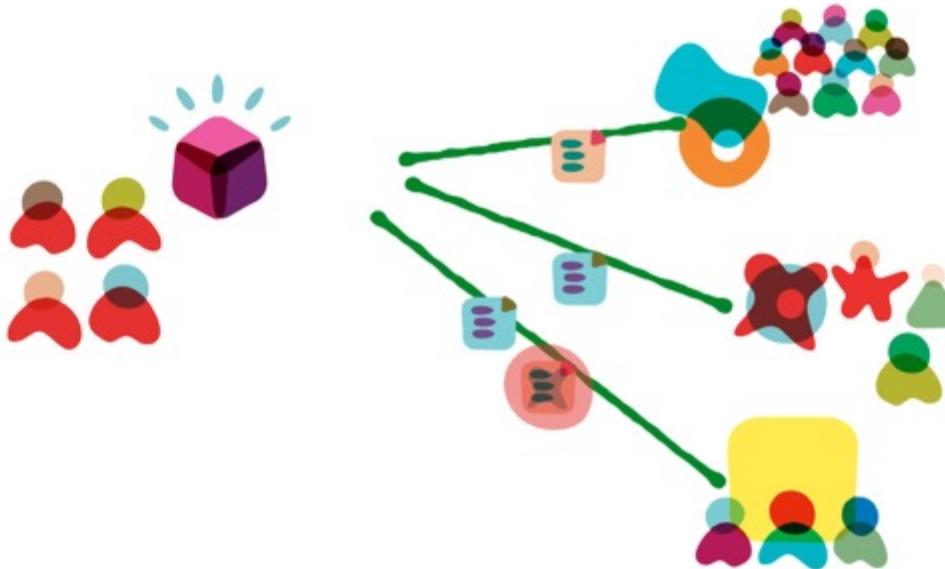
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



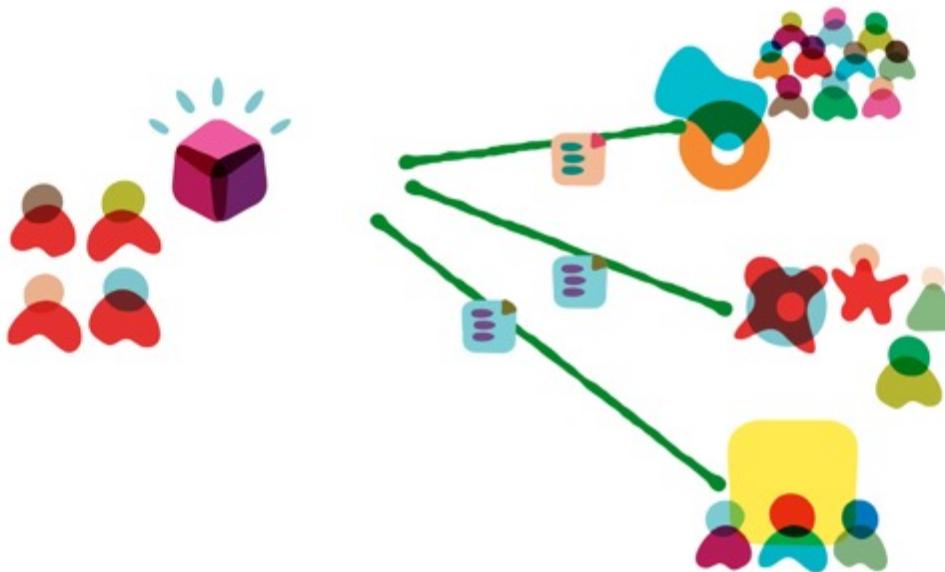
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



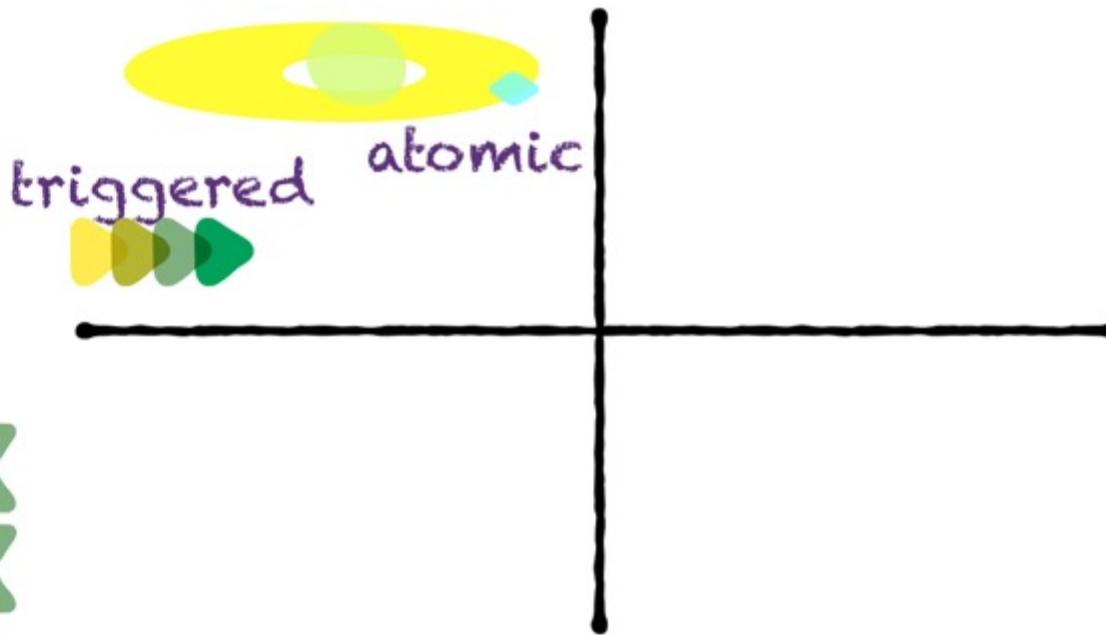
martinfowler.com/articles/consumerDrivenContracts.html

Consumer Driven Contracts



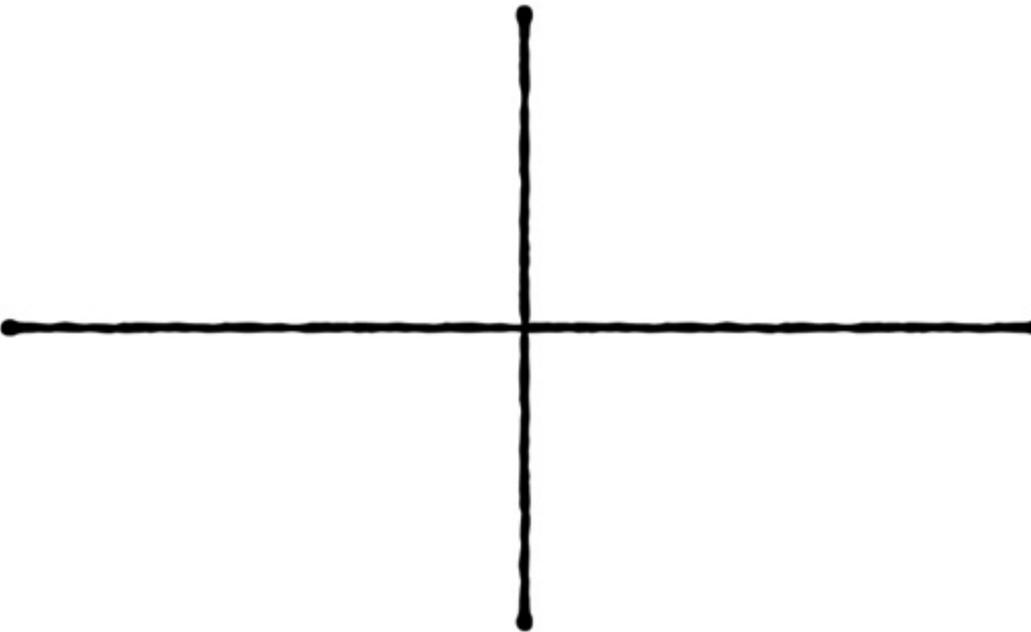
martinfowler.com/articles/consumerDrivenContracts.html

Fitness Function



continuous

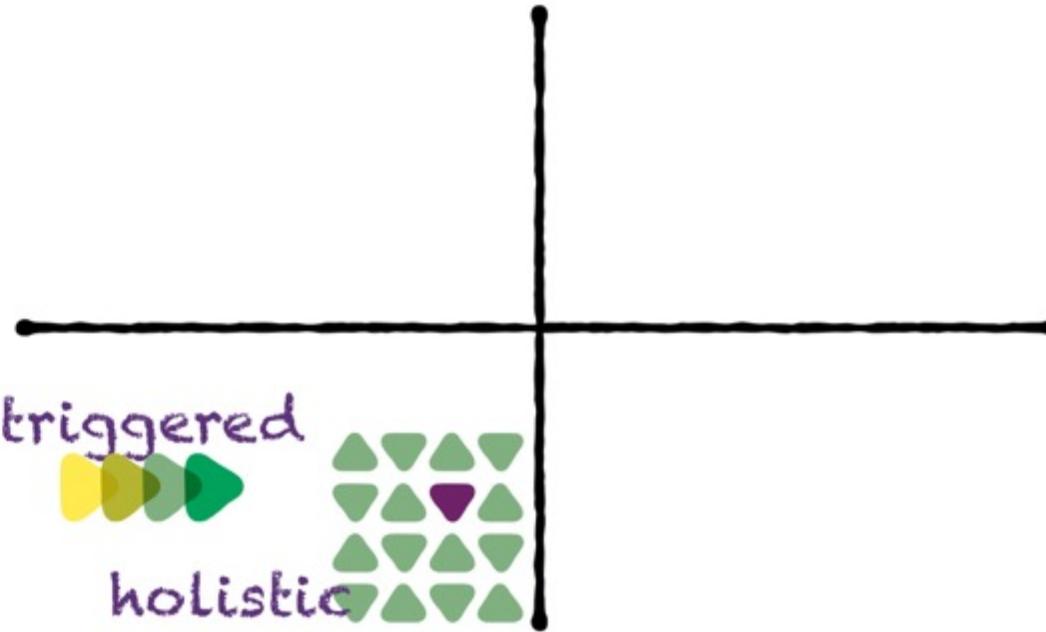
Fitness Function



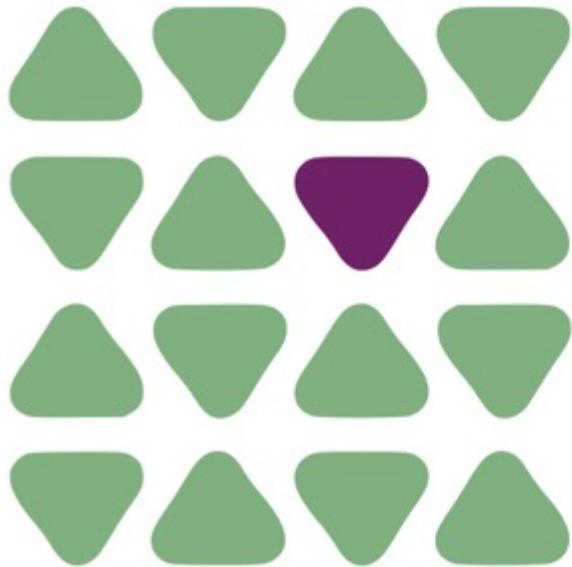
Fitness Function



atomic



triggered

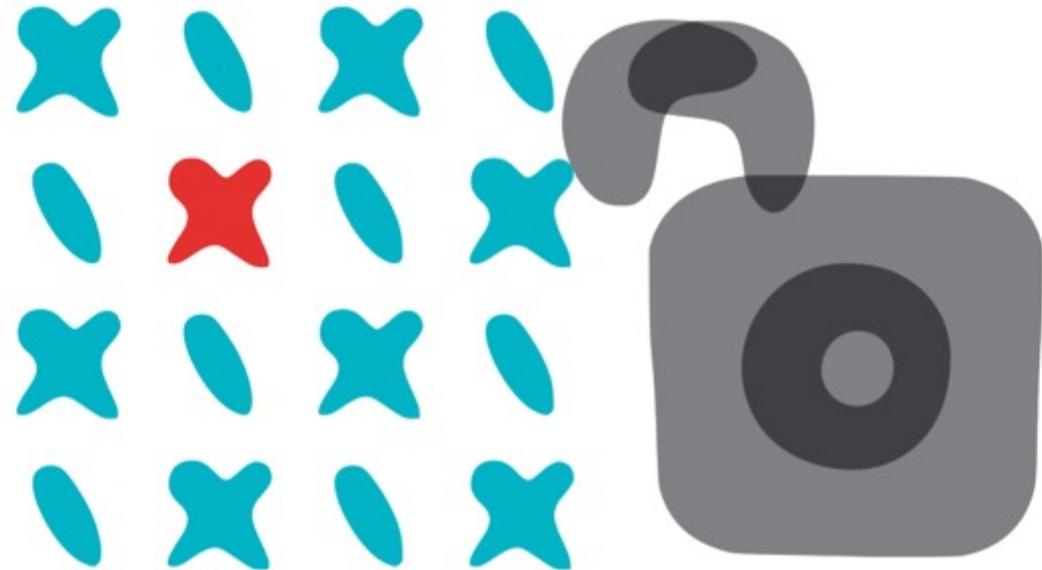




triggered



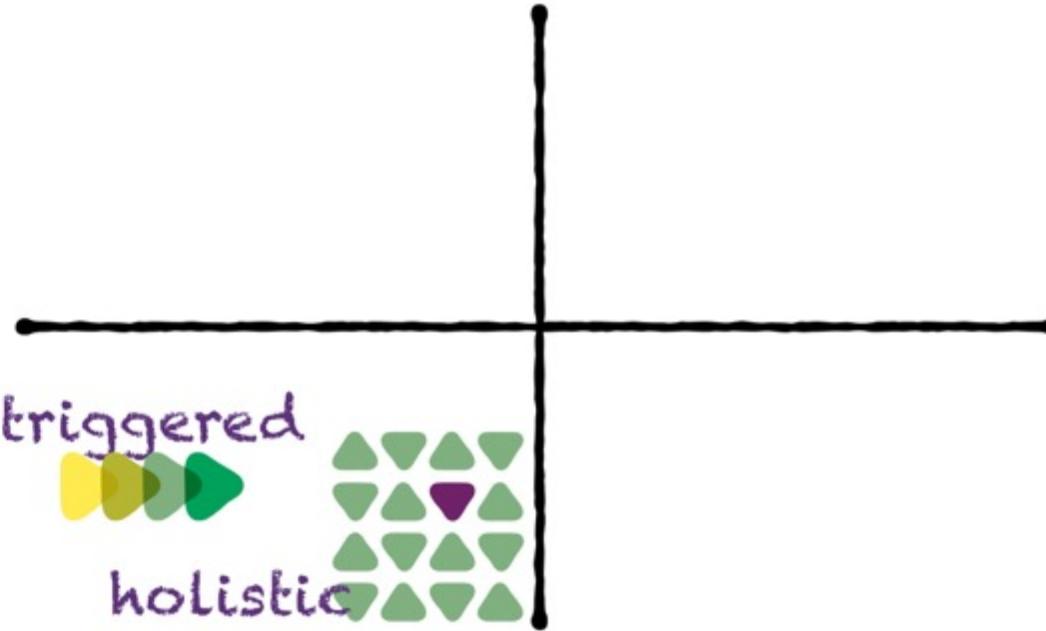
holistic



Fitness Function



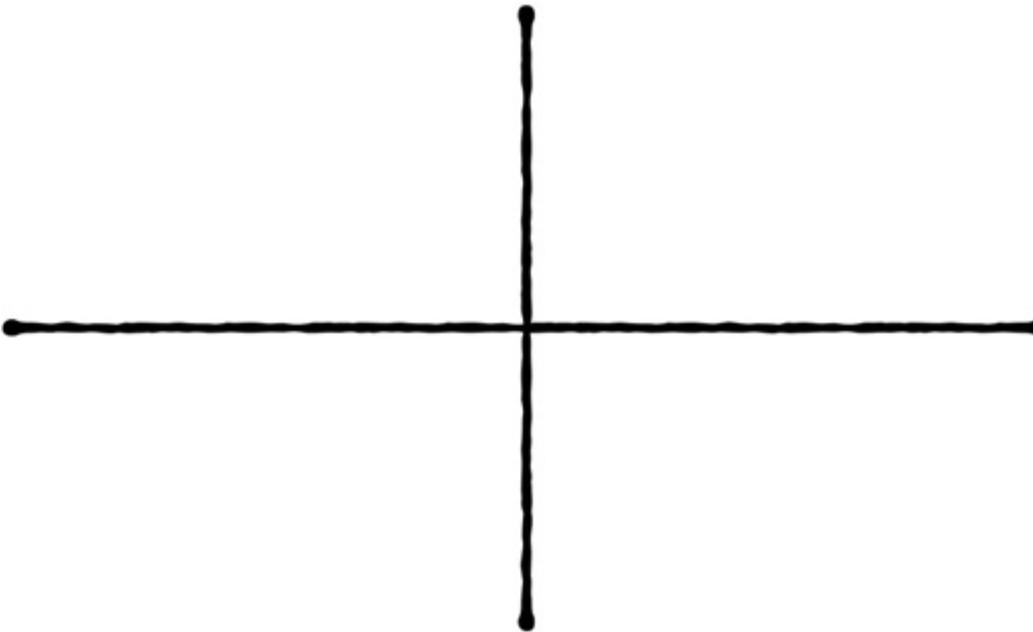
atomic



Fitness Function



atomic



holistic



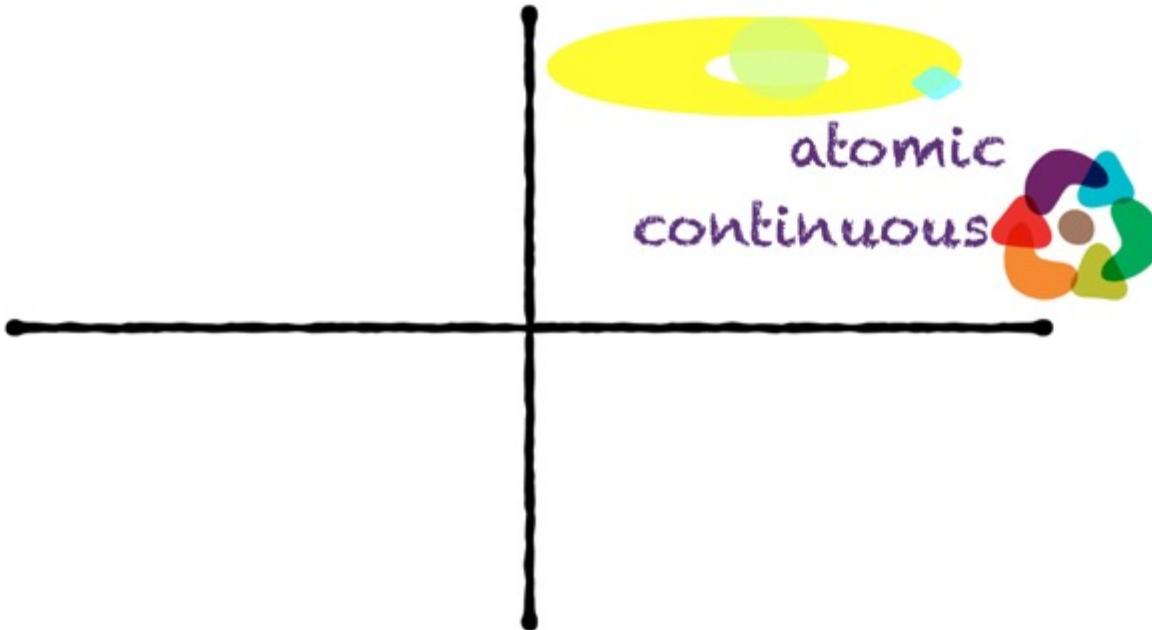
Fitness Function



holistic



triggered





monitoring



logging



monitoring

A screenshot of the Nagios XI web interface showing various monitoring dashboards. The main dashboard includes sections for Host Status Summary, Service Status Summary, and a bar chart titled "Top Alert Producers Last 24 Hours". Another dashboard on the right shows "Hostgroup Status Summary" for various host groups like "All EMC SAN Hosts" and "Linux Servers". A third dashboard at the bottom shows "Disk Usage" for hosts like "localhost" and "vt1.nagios.com".

Nagios XI

Home Views Dashboards Reports Configure Tools Help Admin

Quick View

- Home Dashboard
- Technical Overview
- BirdsEye
- Operations Center
- Operations Screen
- Open Service Problems
- Open Host Problems
- All Service Problems
- All Host Problems
- Network Outages

Details

- Service Detail
- Host Detail
- Hostgroup Summary
- Hostgroup Overview
- Hostgroup Grid
- Services Group Summary
- Servicegroup Services
- Servicegroup Grid
- BPI
- Metrica

Graphs

- Performance Graphs
- Graph Explorer

Maps

- BirdsEye
- Google Map
- Hypermap
- Network Map
- Legacy Network Status Map

Incident Management

- Last Alerts
- Action Requirements
- Scheduled Downtime
- Mass Acknowledgement
- Recurring Downtime
- Notifications

Monitoring Process

- Process Info
- Performance
- Event Log

Host Status Summary

Up	Down	Unreachable	Pending
13	2	2	0
Unhandled	Problems	All	
14	0	2	117

Service Status Summary

OK	Warning	Unknown	Critical	Pending
12	12	24	2	2
Unhandled	Problems	All		
204	262	309	309	

Hostgroup Status Summary

Status Summary For All Host Groups

Host Group	Hosts	Services
All EMC SAN Hosts (el_eme_sans)	2 OK	2 OK
Firewalls (firewalls)	2 OK	2 OK
Host Deadpool (host-deadpool)	2 OK	2 OK
Linux Servers (linux-servers)	2 OK	2 OK
new group (new group)	2 OK	2 OK
Printers (printers)	2 OK	2 OK
Websites (websites)	2 OK	2 OK
Windows Servers (windows-servers)	2 OK	2 OK

Top Alert Producers Last 24 Hours

Alert Producer	Count
Port-14-Capable—Lower Bandwidth	21
Port-1-Capable—Lower Bandwidth	19
Port-23-Bandwidth	18
vt1.nagios.com	17
Users—Sensors	16
Port-19-Capable—Lower Bandwidth	15
Port-1-2-3-4-5-41	13
Port-15-Capable—Lower Bandwidth	10
Switch 1	10
exchange.nagios.org	8
Memory Usage	8
exchange.nagios.org	7
Total Processes	7

Metrica Overview

Disk Usage

Host	Service	% Utilization	Details
localhost	Root Partition	78.67%	DISK WARNING - free space: / 1207 MB (17% inode=68%)
vt1.nagios.com	/ Disk Usage	37.30%	DISK OK - free space: / 117214 MB (61% inode=99%)
exchange.nagios.org	/ Disk Usage	13.22%	DISK OK - free space: / 68057 MB (86% inode=97%)

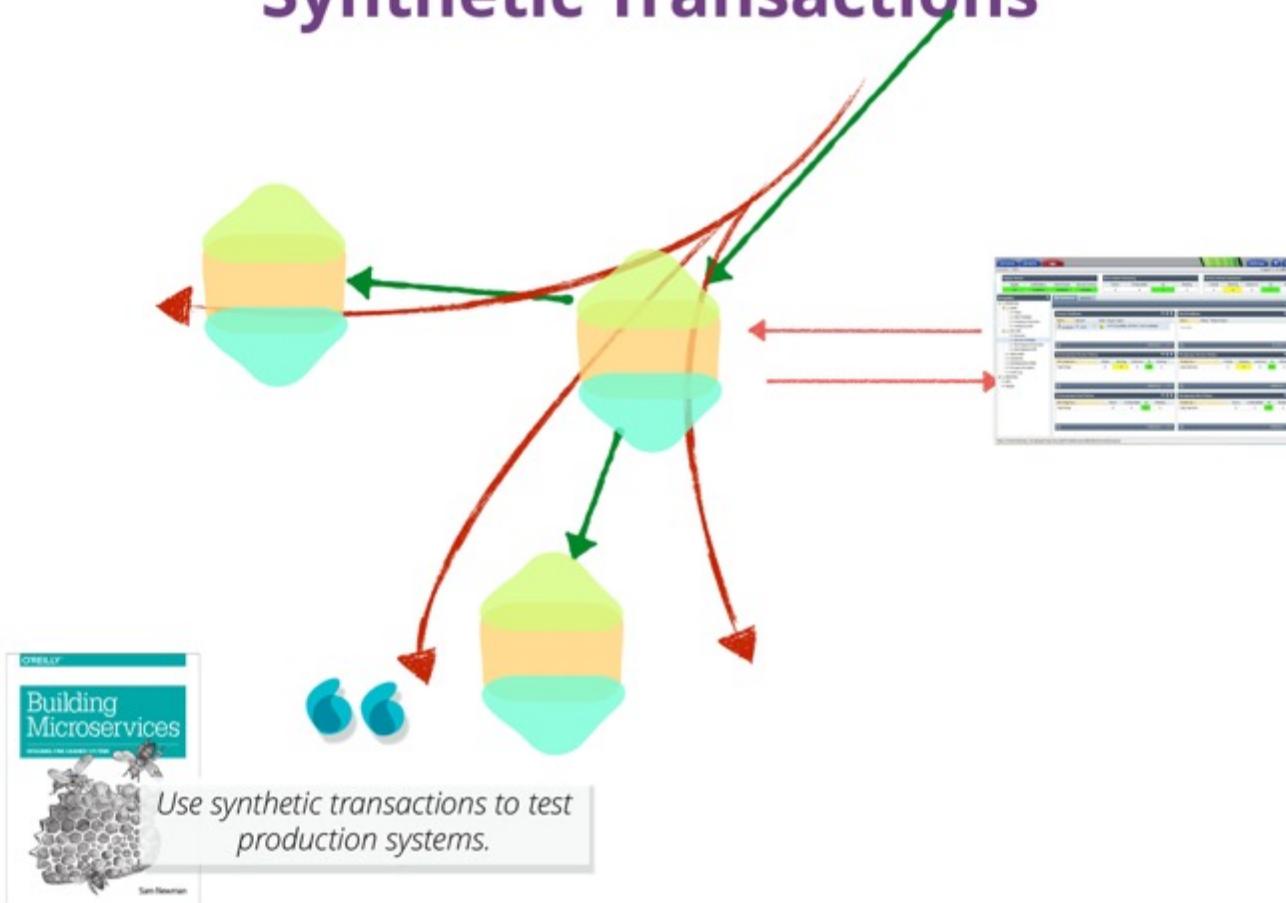
Nagios XI 5.4.10 • Check for Updates

About | Legal | Copyright © 2004-2017 Nagios Enterprises, LLC



logging

Synthetic Transactions



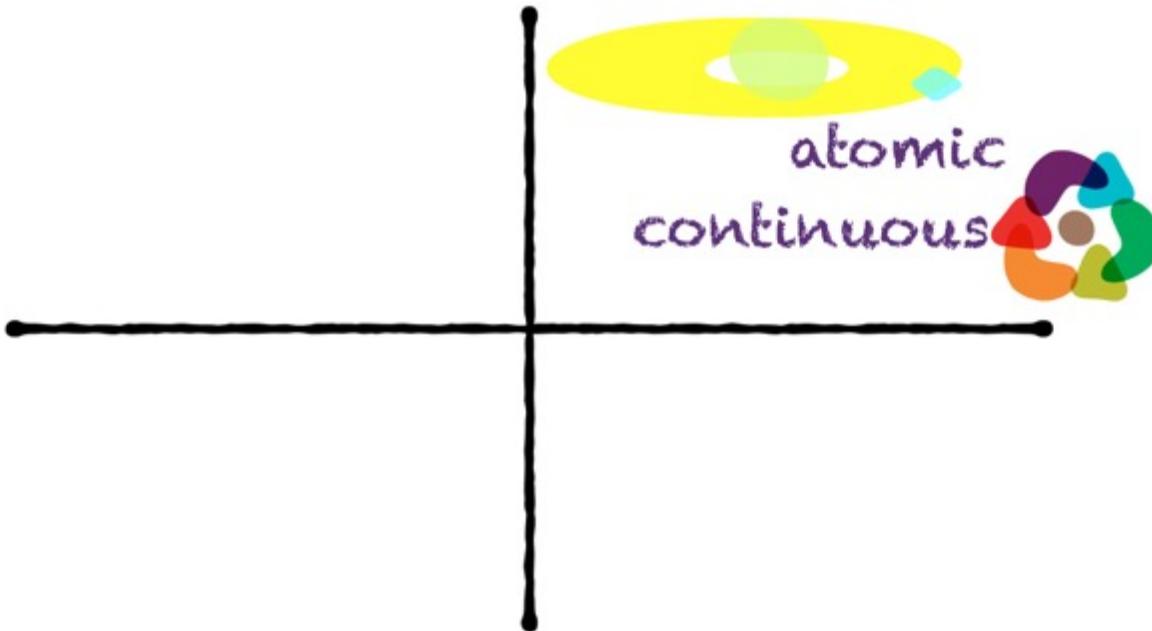
Fitness Function



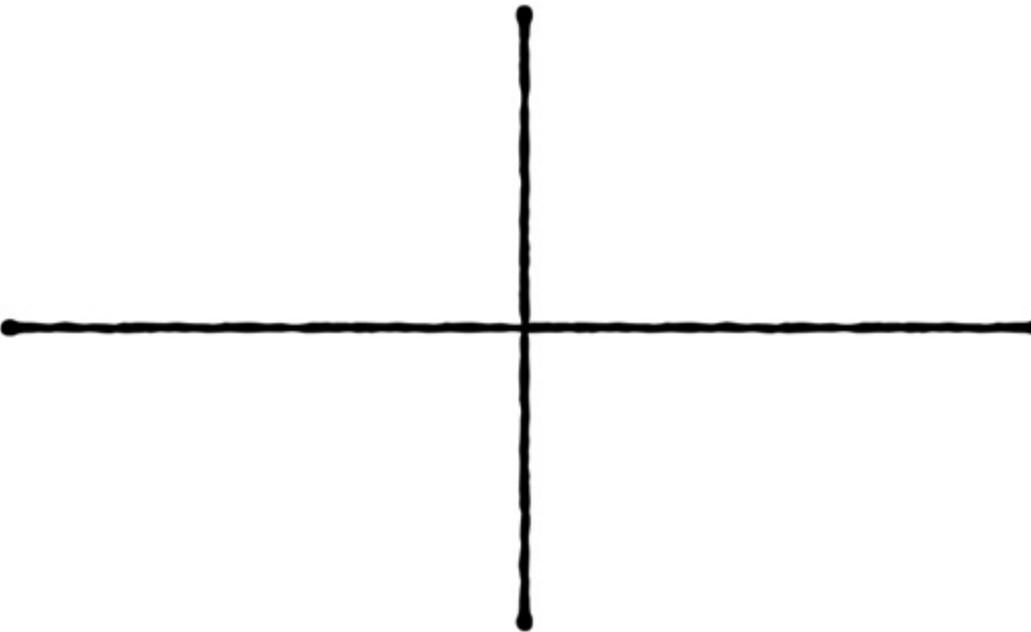
holistic



triggered



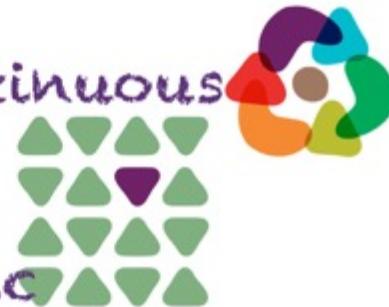
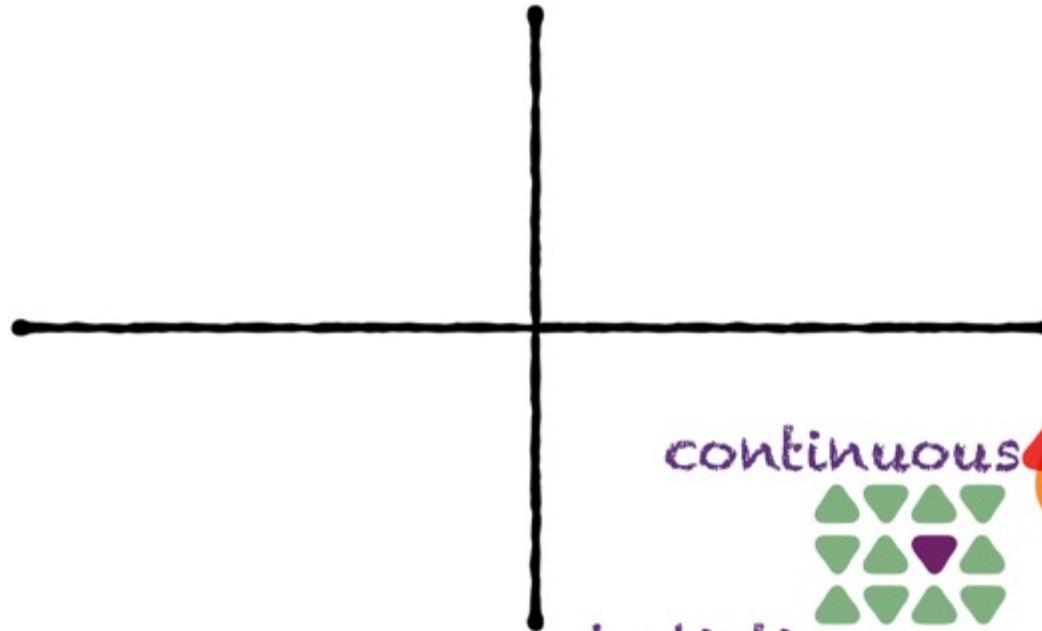
Fitness Function



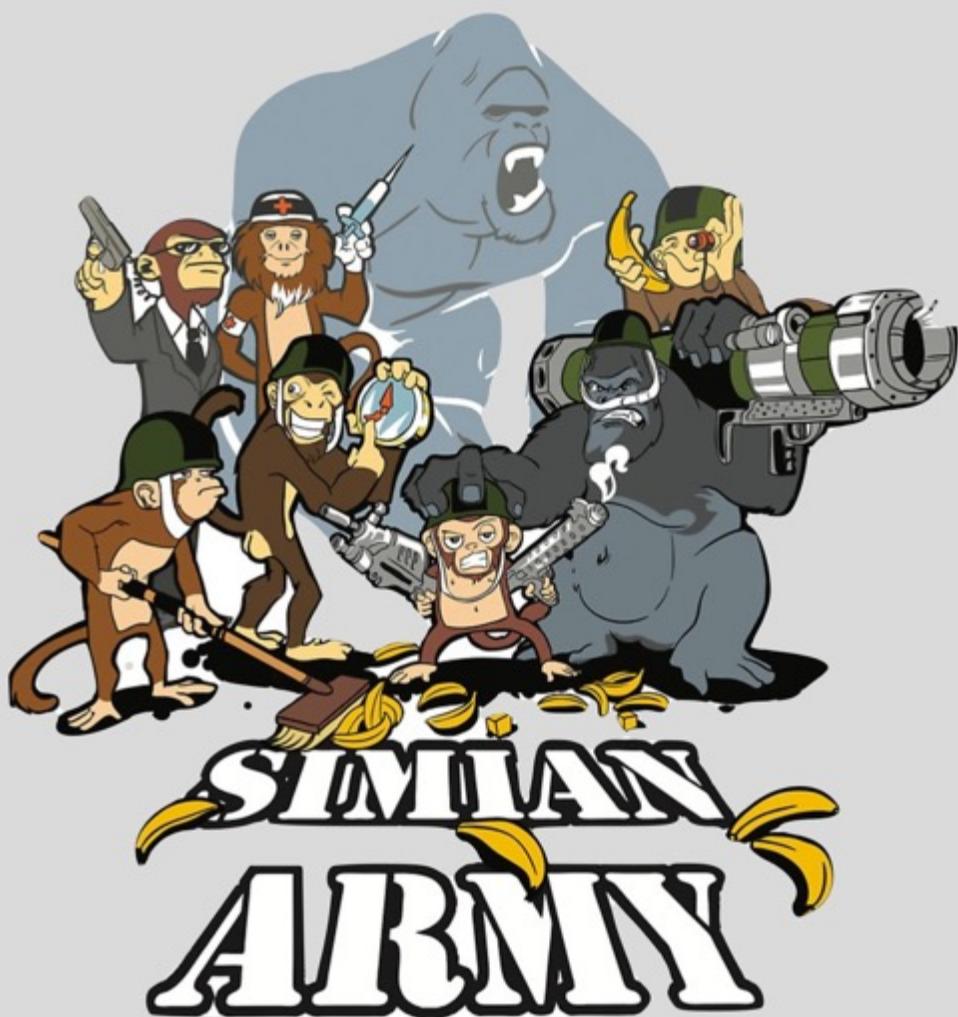
Fitness Function

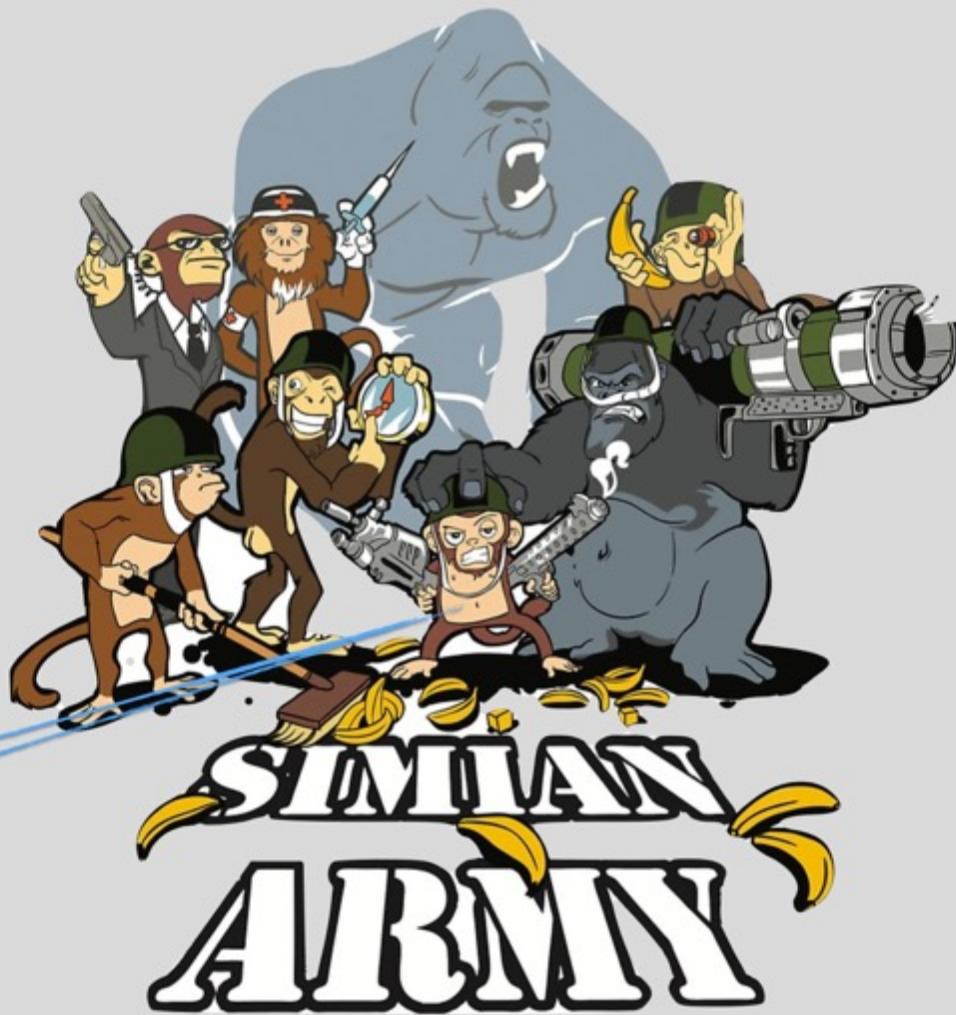


atomic







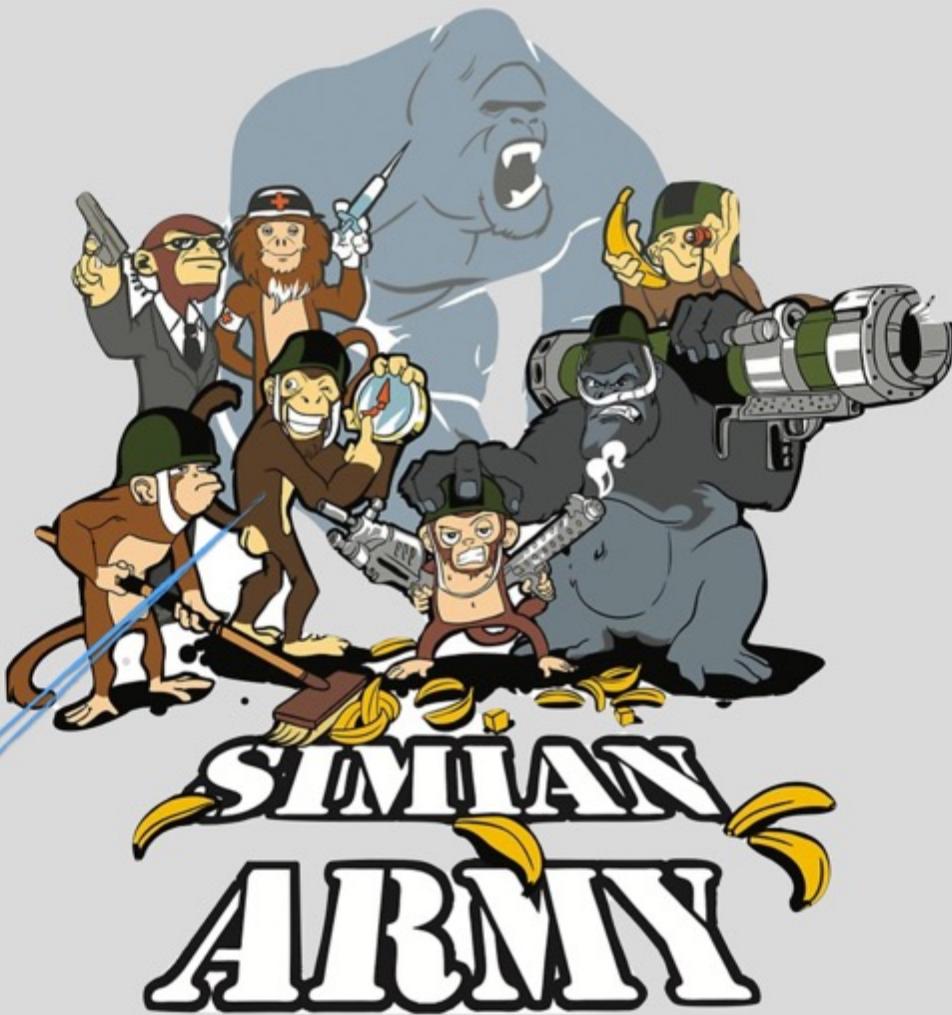


chaos monkey



chaos gorilla

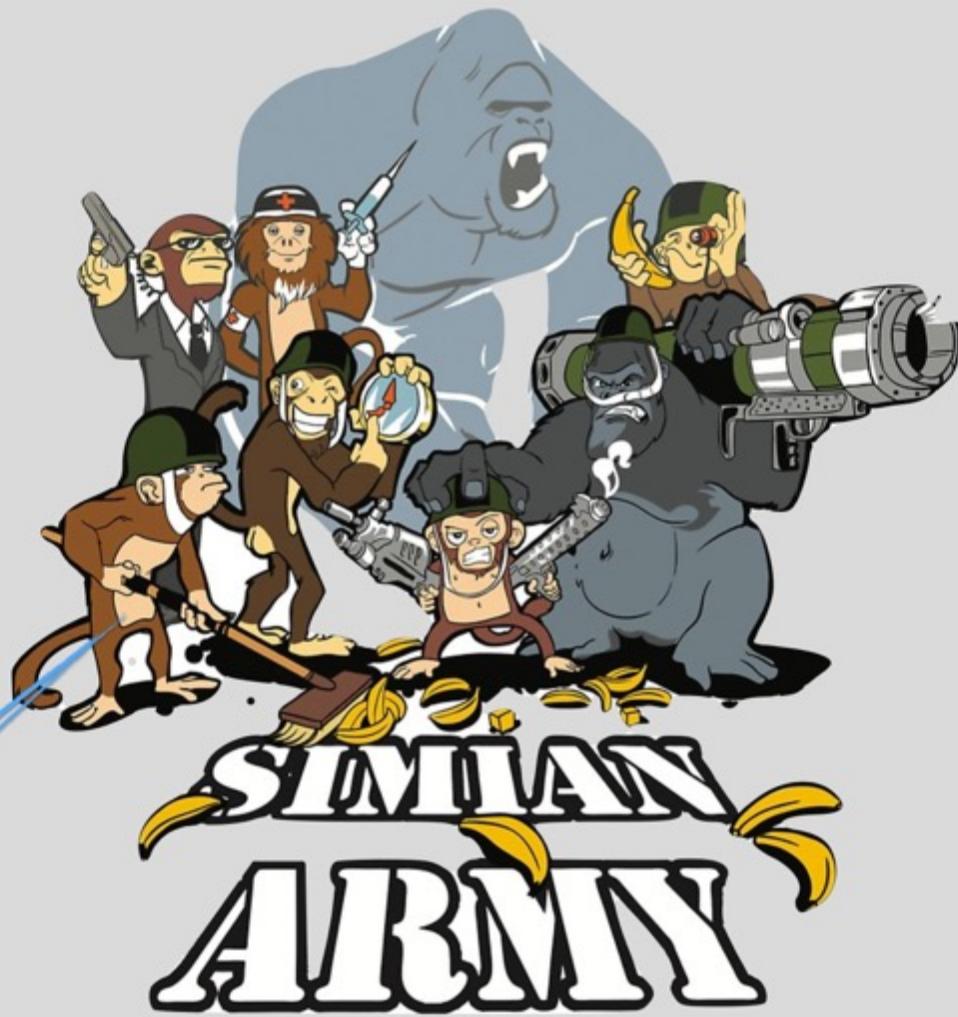
SIMIAN ARMY



Latency monkey

doctor monkey







security monkey

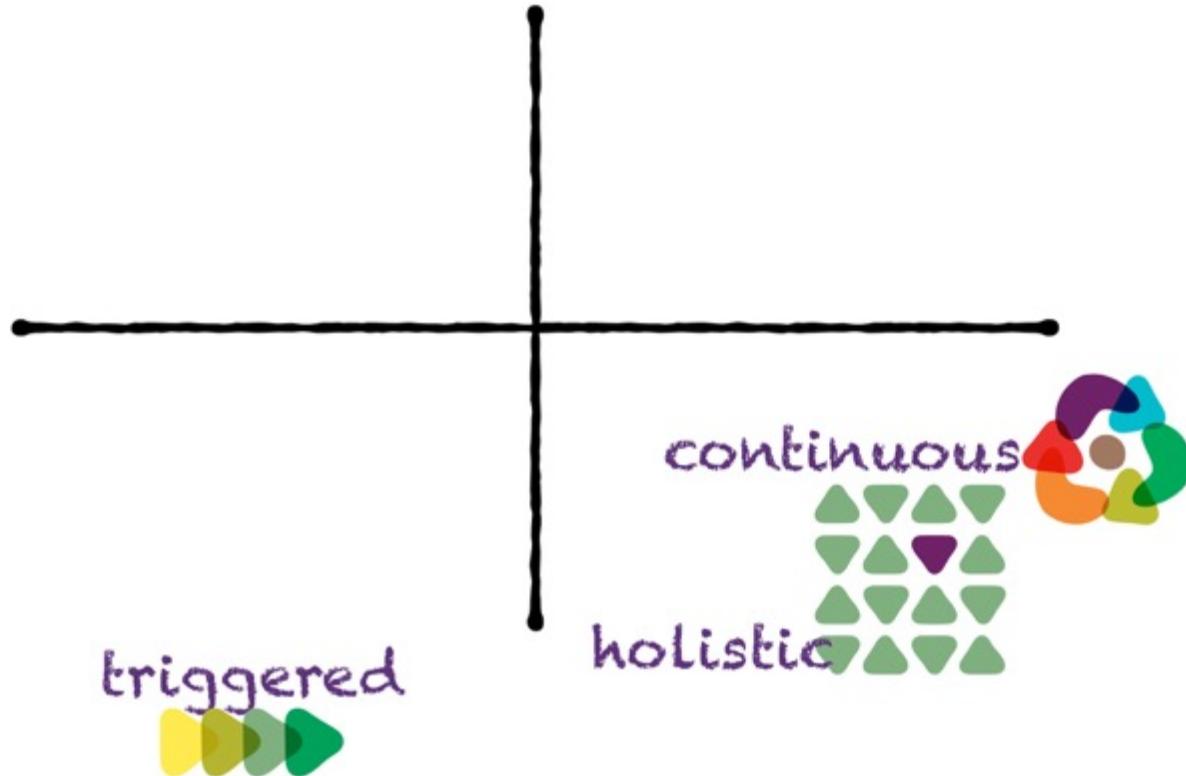


SIMIAN ARMY

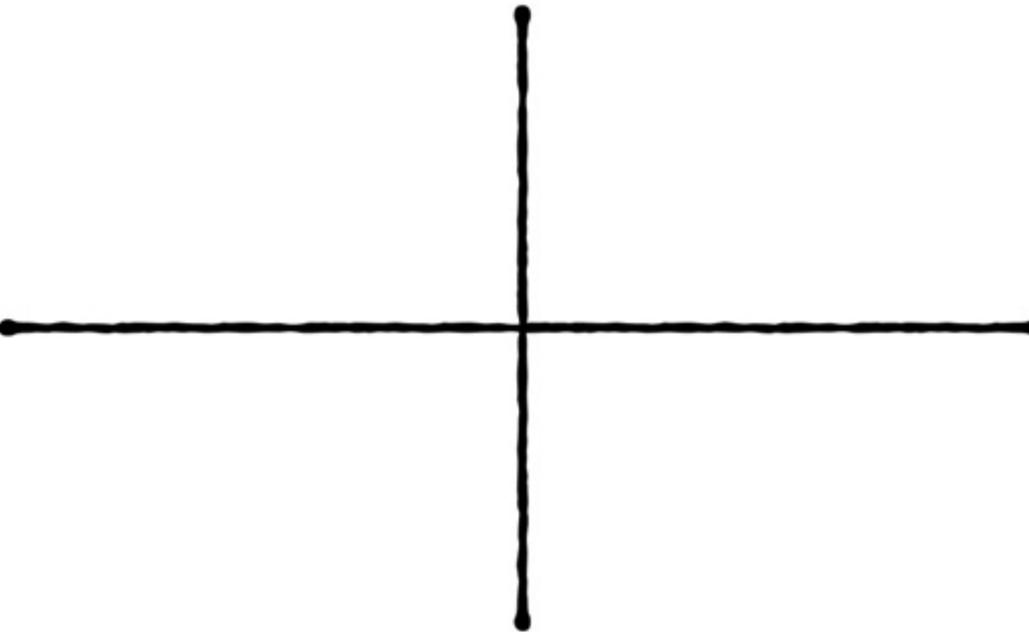
Fitness Function



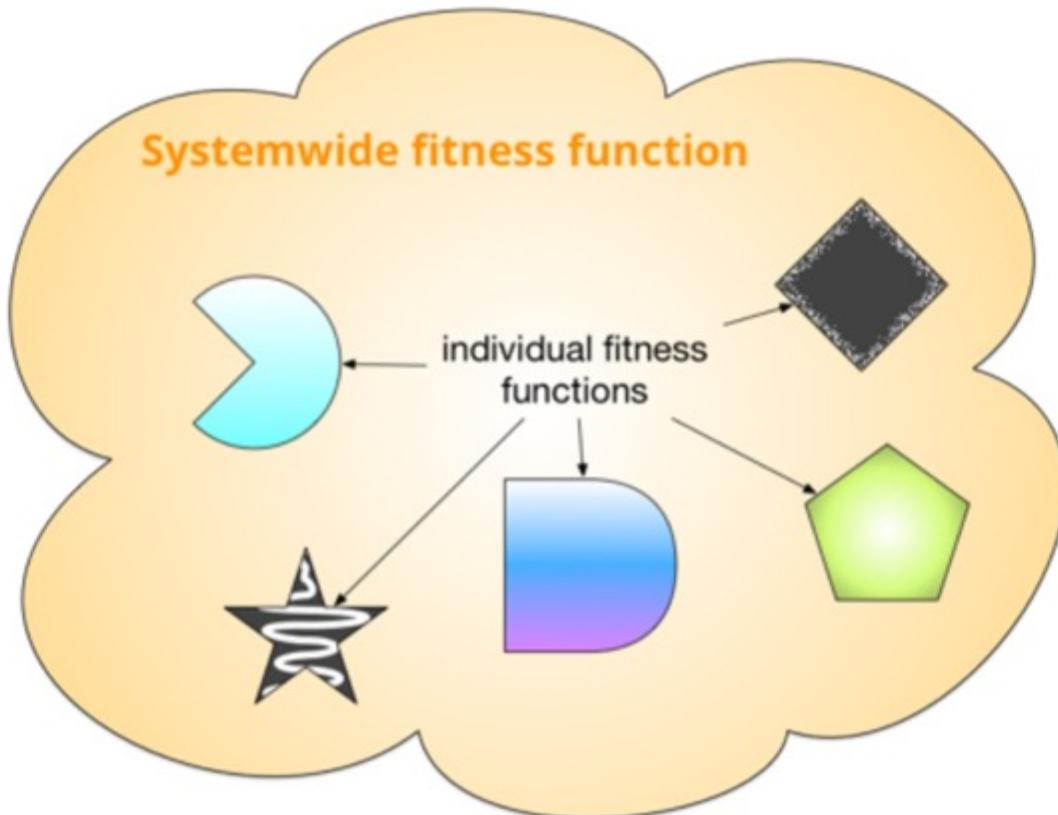
atomic



Fitness Function

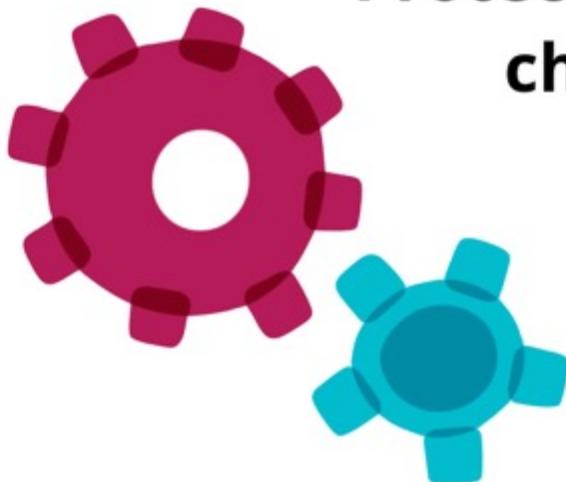


System-wide Fitness Function



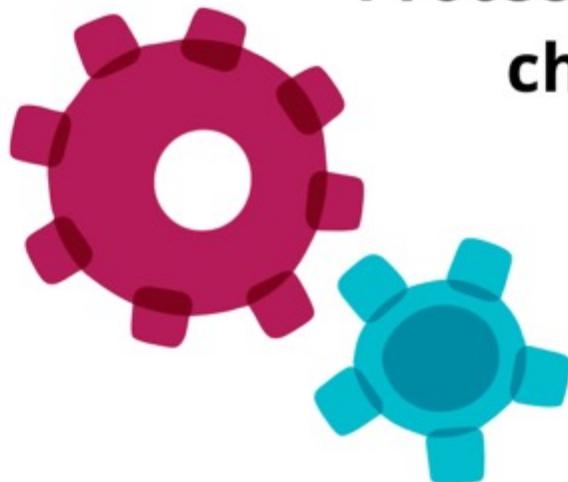
Implementing Fitness Functions

Protecting architectural
characteristics

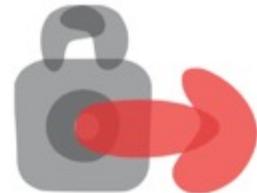


Implementing Fitness Functions

Protecting architectural
characteristics



Automating governance



maintainable?

maintainable?

Cyclomatic complexity < 50 for all projects

maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

(incoming/outgoing)
Controlled afferent/efferent coupling

Cyclomatic complexity < 50 for all projects

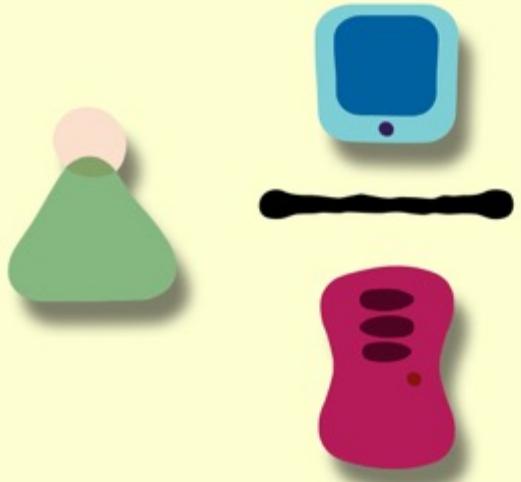
Naming conventions

immutability

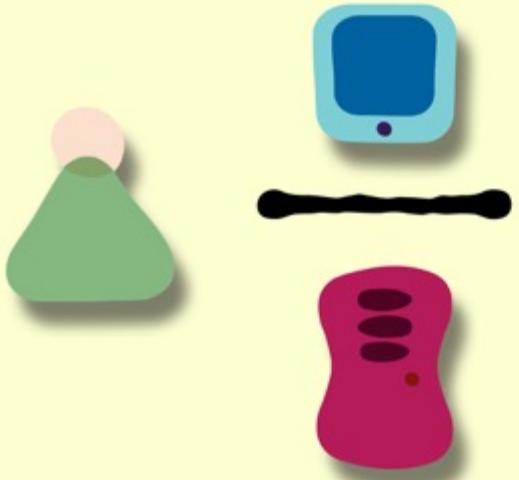
maintainable?

(incoming/outgoing)
Controlled afferent/efferent coupling

Governing Code Quality

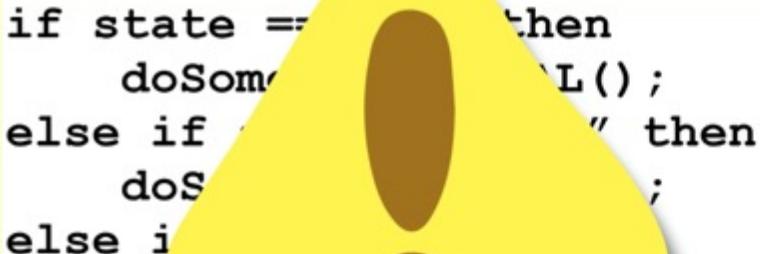
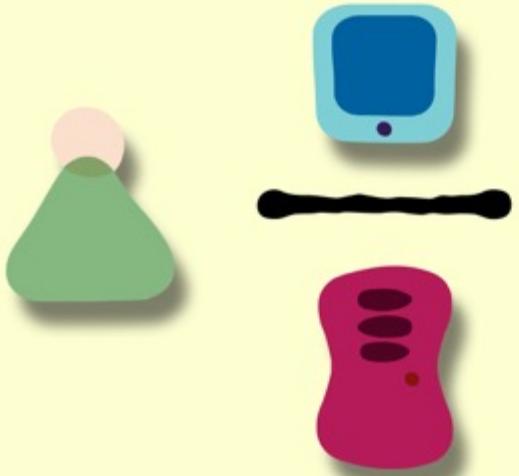


Governing Code Quality



```
if state == "AL" then
    doSomethingForAL();
else if state == "GA" then
    doSomethingForGA();
else if ...
```

Governing Code Quality



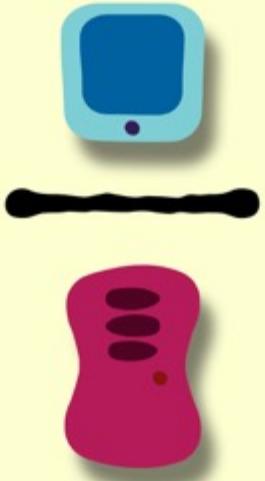
```
if state == "A" then
    doSomethingA()
else if state == "B" then
    doSomethingB()
else if state == "C" then
    doSomethingC()
```

Governing Code Quality

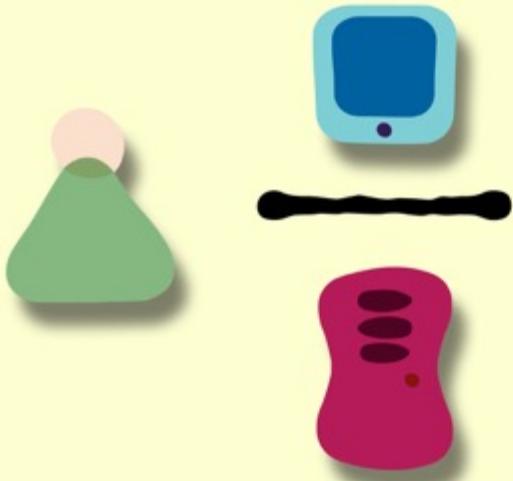


Governing Code Quality

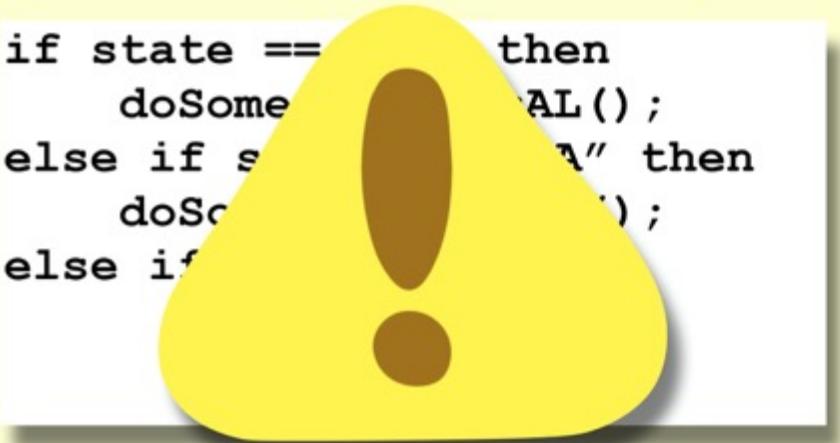
```
if state == "AL" then
    doSomethingForAL();
else if state == "GA" then
    doSomethingForGA();
else if ...
```



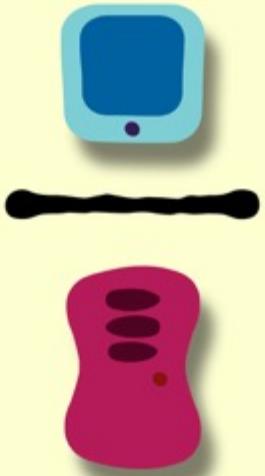
Governing Code Quality



```
if state == "A" then
    doSomethingA();
else if state == "B" then
    doSomethingB();
else if state == "C" then
    doSomethingC();
```



Governing Code Quality



Strategy Design
Pattern



Safari File Edit View History Bookmarks Develop Window Help blog.jdriven.com Sun 10:30

Apple iCloud Yahoo Bing Google Wikipedia Facebook Twitter LinkedIn The Weather Channel Yelp TripAdvisor



← Previous Next →

Search

Implementing architectural fitness functions using Gradle, JUnit and code-assert

Posted on October 6, 2017 by Rob Brinkman [Twitter](#)

Architectural fitness functions

Inspired by Neal Ford's presentation at our [Change is the Only constant event](#) I started experimenting with architectural fitness functions. An architectural fitness function provides an objective integrity assessment of some architectural characteristic(s).

If you want to take a deeper dive into evolutionary architectures including fitness functions take look at Neals book: [Building Evolutionary Architectures: Support Constant Change](#).

Neal's [slides](#) contained an example of verifying package dependencies from a Unit Test using [JDepend](#).

Verifying code modularity

In this blog post we'll elaborate on that approach and create a Unit Test that verifies that our code complies to the chosen packaging intention using an

JDriven
[blog.jdriven.com](#)
[www.jdriven.com](#)

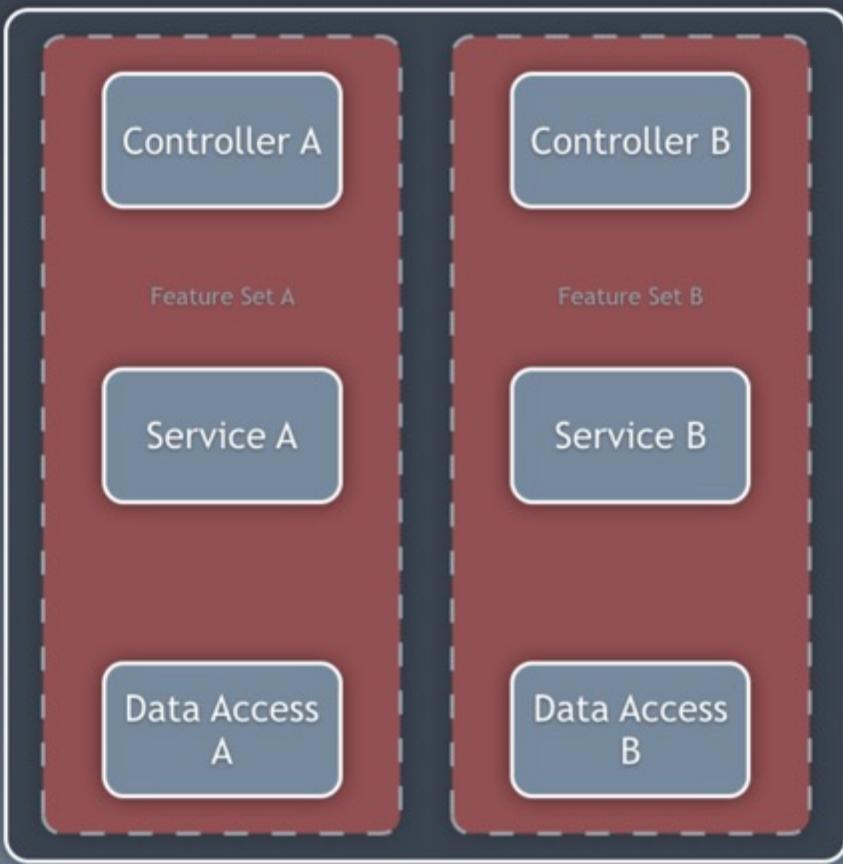
Featured Posts

- Spring Sweets: Add (Extra)
- Build Information To Info
- Endpoint
- Angular2 and Spring Boot: Getting Started
- Het ontstaan van de passie voor het moderne maken
- Securing your application
- landscape with Spring Cloud
- Security – Part 1
- Spicy Spring : Dynamically create your own BeanDefinition

Recent Posts

- Awesome Asciidocitor:
- Grouping Floating Images
- Awesome Asciidocitor: Using Tab Separated Data In A Table
- Implementing architectural fitness functions using Gradle, JUnit and code-assert
- 6 Steps to help you debug your application

<https://blog.jdriven.com/2017/10/implementing-architectural-fitness-functions-using-gradle-junit-code-assert/>



Package by feature (vertical slicing)

```
public class VerifyPackageByFeatureTest {  
  
    @Test  
    public void verifyPackageByFeature() {  
  
        /// Create an analyzer config for the package we'd like to verify  
        AnalyzerConfig analyzerConfig = GradleAnalyzerConfig.gradle().main("com.jdriven.fitness.packaging.by.feature");  
  
        // Dependency Rules for Packaging By Feature  
        // NOTE: the classname should match the packagename  
        class ComJdrivenFitnessPackagingByFeature extends DependencyRuler {  
  
            // Rules for feature child packages  
            // NOTE: they should match the name of the sub packages  
            DependencyRule a, b;  
  
            @Override  
            public void defineRules() {  
                // Our App classes depends on all subpackages because it constructs all of them  
                base().mayUse(base().allSub());  
            }  
        }  
  
        // All dependencies are forbidden, except the ones defined in ComJdrivenFitnessPackagingByFeature  
        // java, org, net packages may be used freely  
        DependencyRules rules = DependencyRules.denyAll()  
            .withRelativeRules(new ComJdrivenFitnessPackagingByFeature())  
            .withExternals("java.*", "org.*", "net.*");  
  
        DependencyResult result = new DependencyAnalyzer(analyzerConfig).rules(rules).analyze();  
        assertThat(result, matchesRulesExactly());  
    }  
}
```

```
public class ControllerA {  
    private final ServiceA serviceA;  
    private final ServiceB serviceB;  
  
    public ControllerA(ServiceA serviceA, ServiceB serviceB) {  
        this.serviceA = serviceA;  
        this.serviceB = serviceB;  
    }  
}
```

```
java.lang.AssertionError:  
Expected: Comply with rules  
but: DENIED com.jdriven.fitness.packaging.by.feature.a -&gt;  
com.jdriven.fitness.packaging.by.feature.b (by com.jdriven.fitness.packaging.by.feature.a.ControllerA)
```

```
public class ControllerA {  
    private final ServiceA serviceA;  
    private final ServiceB serviceB;  
  
    public ControllerA(ServiceA serviceA, ServiceB serviceB) {  
        this.serviceA = serviceA;  
        this.serviceB = serviceB;  
    }  
}
```

```
java.lang.AssertionError:  
Expected: Comply with rules  
but: DENIED com.jdriven.fitness.packaging.by.feature.a -&gt;  
com.jdriven.fitness.packaging.by.feature.b (by com.jdriven.fitness.packaging.by.feature.a.ControllerA)
```

// Allow package / module a to access b
a.mayUse(b);

ArchUnit

<https://www.archunit.org/>

The screenshot shows the homepage of the ArchUnit website. At the top, there is a navigation bar with links for "Getting Started", "Motivation", "News", "User Guide", "API", and "About". Below the navigation bar, the main heading is "Unit test your Java architecture". A sub-headline below it reads: "Start enforcing your architecture within 30 minutes using the test setup you already have." A prominent "Start Now" button is located below this text. The background features a blue gradient with faint icons of code and files. At the bottom of the page, there is a section titled "News" which contains a brief description of what ArchUnit is and how it can be used.

ArchUnit is a free, simple and extensible library for checking the architecture of your Java code using any plain Java unit test framework. That is, ArchUnit can check dependencies between packages and classes, layers and slices, check for cyclic dependencies and more. It does so by analyzing given Java bytecode, importing all classes into a Java code structure. You can find examples for the current release at [ArchUnit Examples](#) and the sources on [GitHub](#).

News

ArchUnit

<https://www.archunit.org/>

coding rules

```
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

@Test

<https://www.archunit.org/>

```
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);  
}
```

coding rules

ArchUnit

<https://www.archunit.org/>

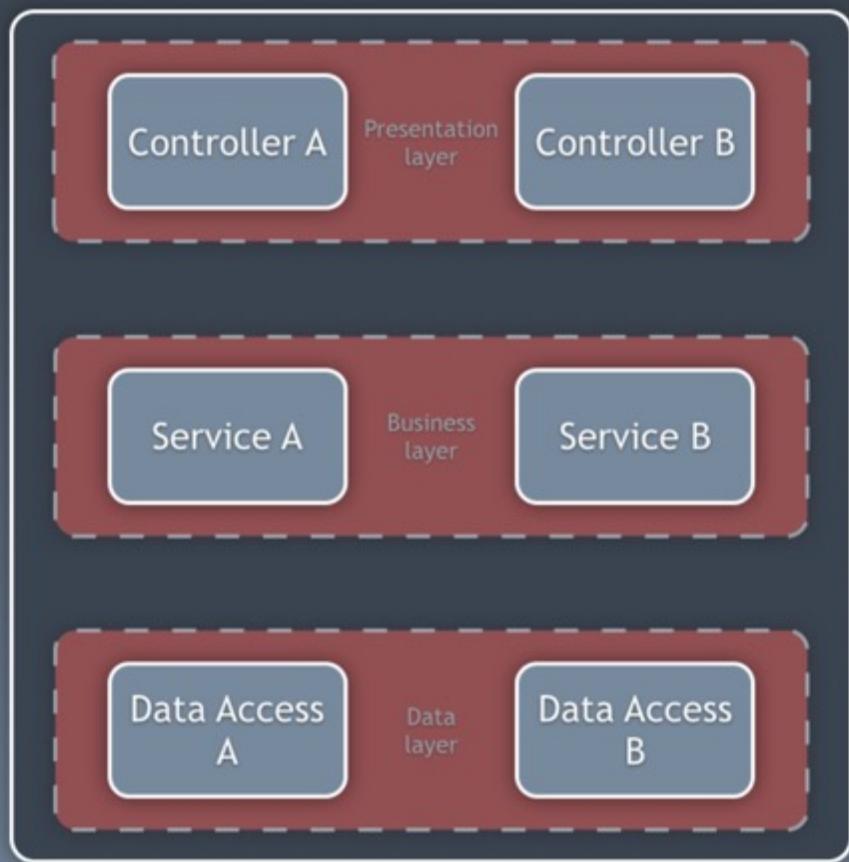
```
public class InterfaceRules {  
  
    @Test  
    public void interfaces_should_not_have_names_ending_with_the_word_interface() {  
        JavaClasses classes = new ClassFileImporter().importClasses(  
            SomeBusinessInterface.class,  
            SomeDao.class  
        );  
  
     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

ArchUnit

<https://www.archunit.org/>

```
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")  
    }  
  
    @Test  
    public void interfaces_should_not_have_simple_class_names_ending_with_the_word
```



Package by layer (horizontal slicing)

ArchUnit

<https://www.archunit.org/>

```
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

```
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);
}
```

ArchUnit

<https://www.archunit.org/>

```
@Test
public void third_party_class_should_only_be_instantiated_via_workaround() {
    classes().should(notCreateProblematicClassesOutsideOfWorkaroundFactory()
        .as(THIRD_PARTY_CLASS_RULE_TEXT))
        .check(classes);
}

private ArchCondition<JavaClass> notCreateProblematicClassesOutsideOfWorkaroundFactory() {
    DescribedPredicate<JavaCall<?>> constructorCallOfThirdPartyClass =
        target(is(constructor())).and(targetOwner(is(assignableTo(ThirdPartyClassWithProblem.class))));

    DescribedPredicate<JavaCall<?>> notFromWithinThirdPartyClass =
        originOwner(is(not(assignableTo(ThirdPartyClassWithProblem.class)))).forSubType();

    DescribedPredicate<JavaCall<?>> notFromWorkaroundFactory =
        originOwner(is(not(equivalentTo(ThirdPartyClassWorkaroundFactory.class)))).forSubType();

    DescribedPredicate<JavaCall<?>> targetIsIllegalConstructorOfThirdPartyClass =
        constructorCallOfThirdPartyClass.
            and(notFromWithinThirdPartyClass).
            and(notFromWorkaroundFactory);

    return never(callCodeUnitWhere(targetIsIllegalConstructorOfThirdPartyClass));
}
```

governance

Legality of Open Source Libraries



Penultima ↑e


Legality of Open Source Libraries



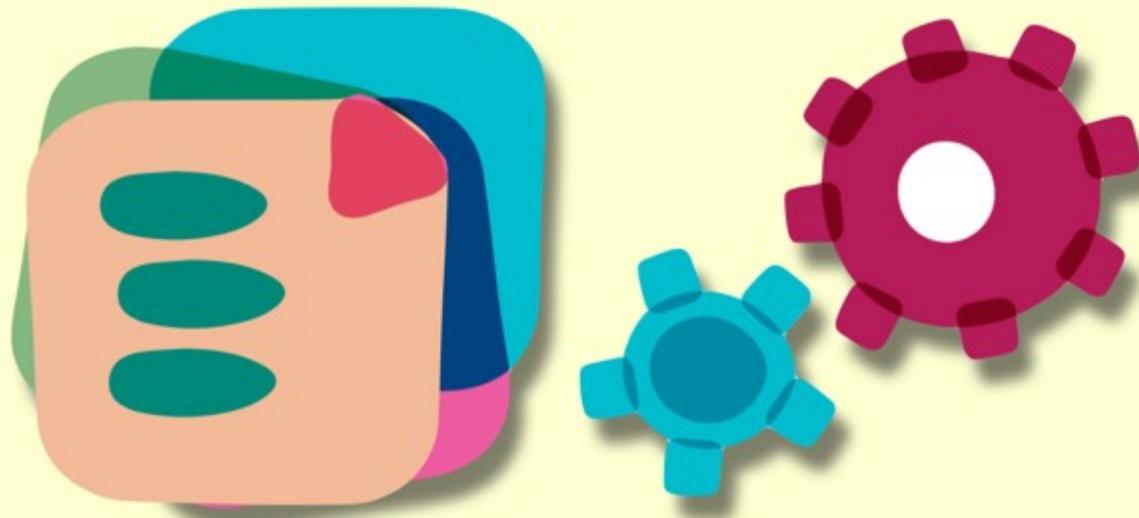
Penultima ↑e
 

Legality of Open Source Libraries



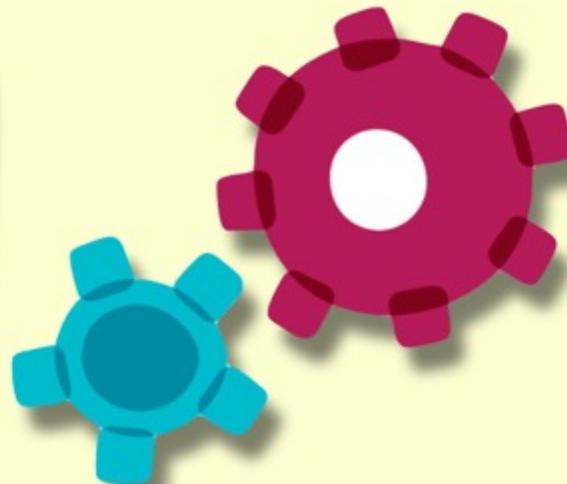
Penultima ↑e


Legality of Open Source Libraries



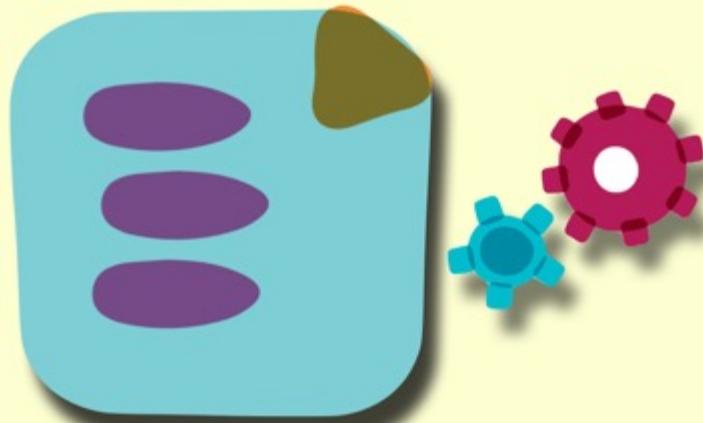
Penultima ↑e
A small cluster of three stylized gears in orange, blue, and pink, with an upward-pointing arrow above the text.

Legality of Open Source Libraries



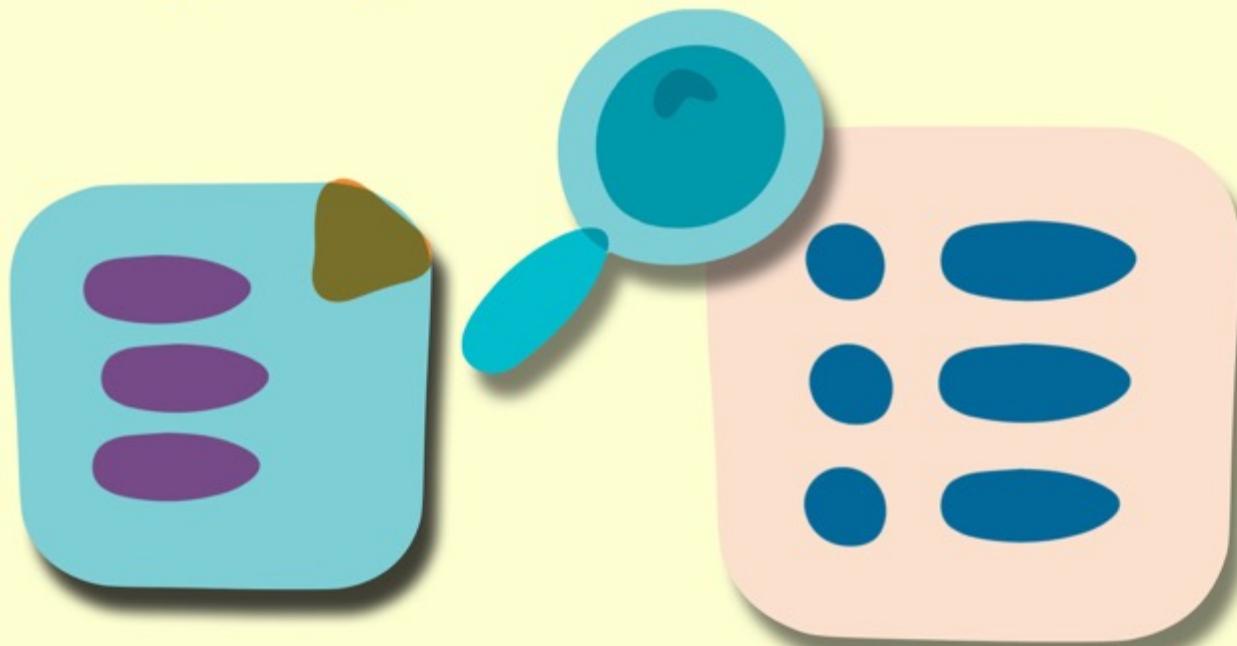
Penultima ↑e
↑e

Legality of Open Source Libraries



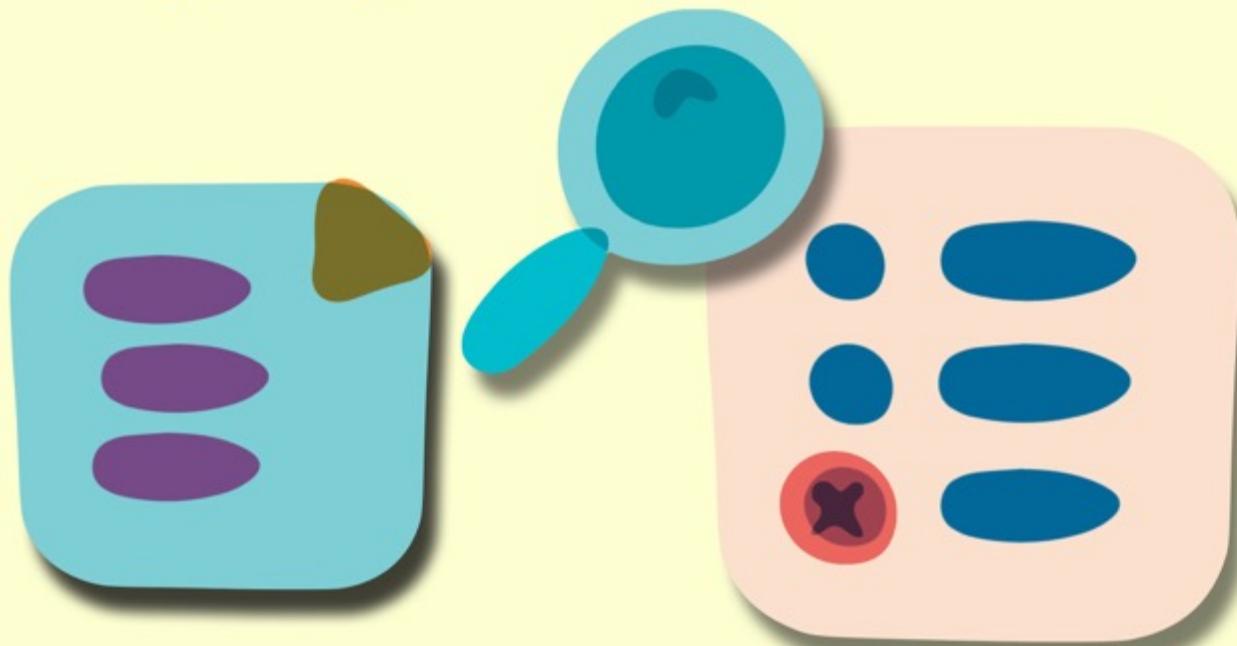
Penultima ↑
A decorative footer element located in the bottom right corner. It features three small, semi-transparent gears in teal, pink, and purple, arranged horizontally. Above the gears is a thin, black, upward-pointing arrow.

Legality of Open Source Libraries



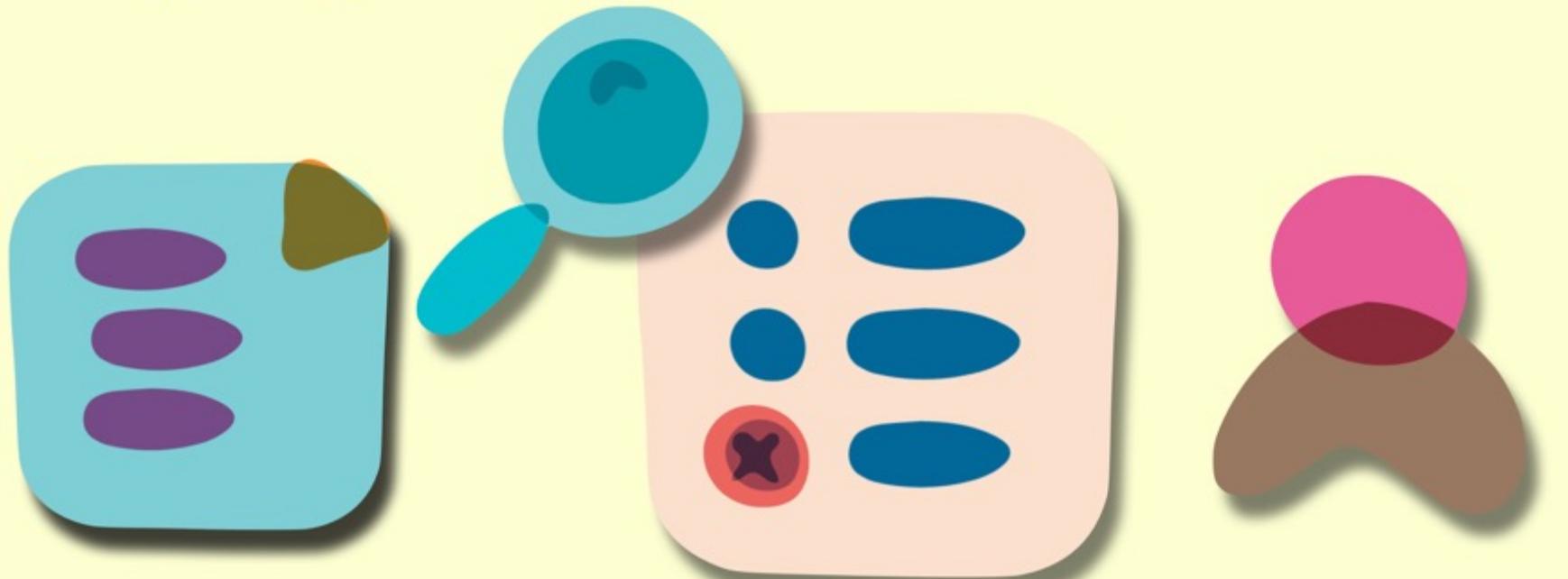
Penultima ↑e
A set of small, colorful icons located in the bottom right corner. It includes three interlocking gears in red, green, and blue, and two blue arrows pointing upwards.

Legality of Open Source Libraries



Penultima ↑e

Legality of Open Source Libraries



Penultima ↑
e
gears

Agenda Part 1



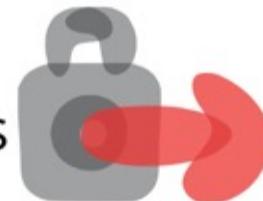
Definition



Guided Change via Fitness Functions



Incremental Change

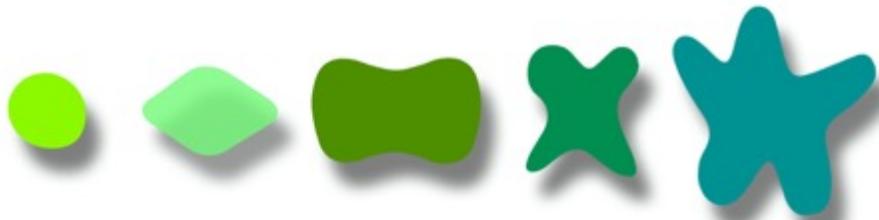


Structuring architecture for evolution



Two Big Ideas

Two Big Ideas



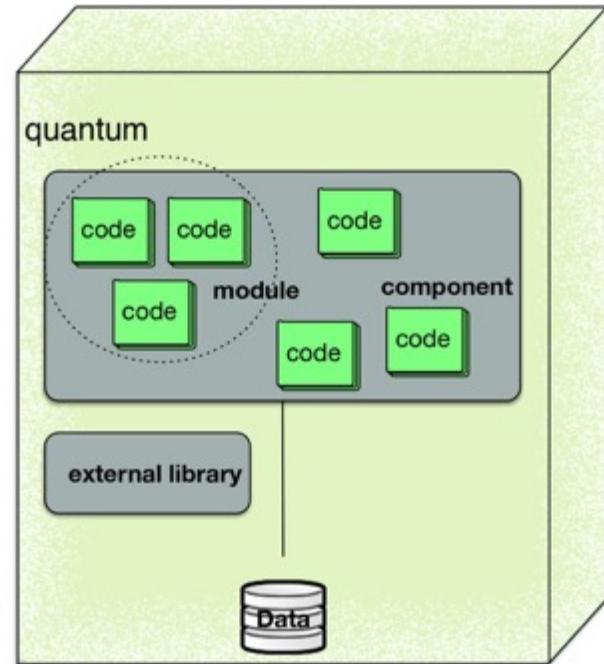
fitness functions for
evolvability

Two Big Ideas

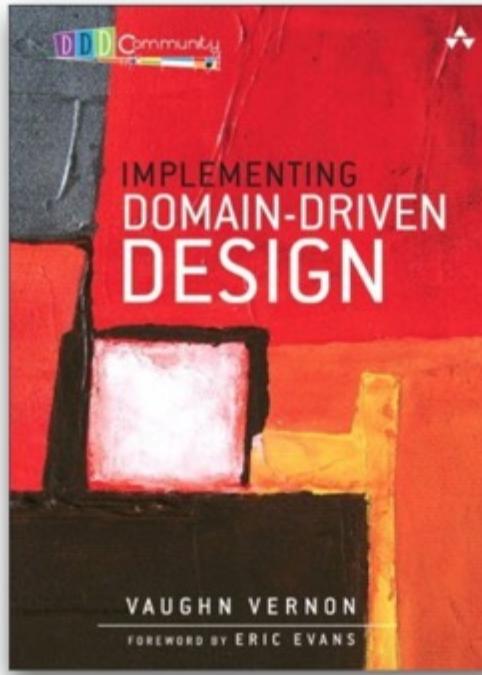
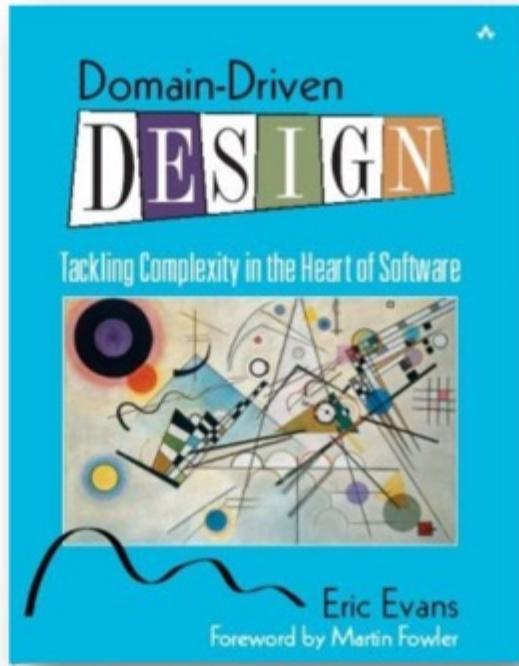


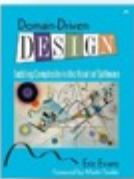
fitness functions for
evolvability

architectural
quantum



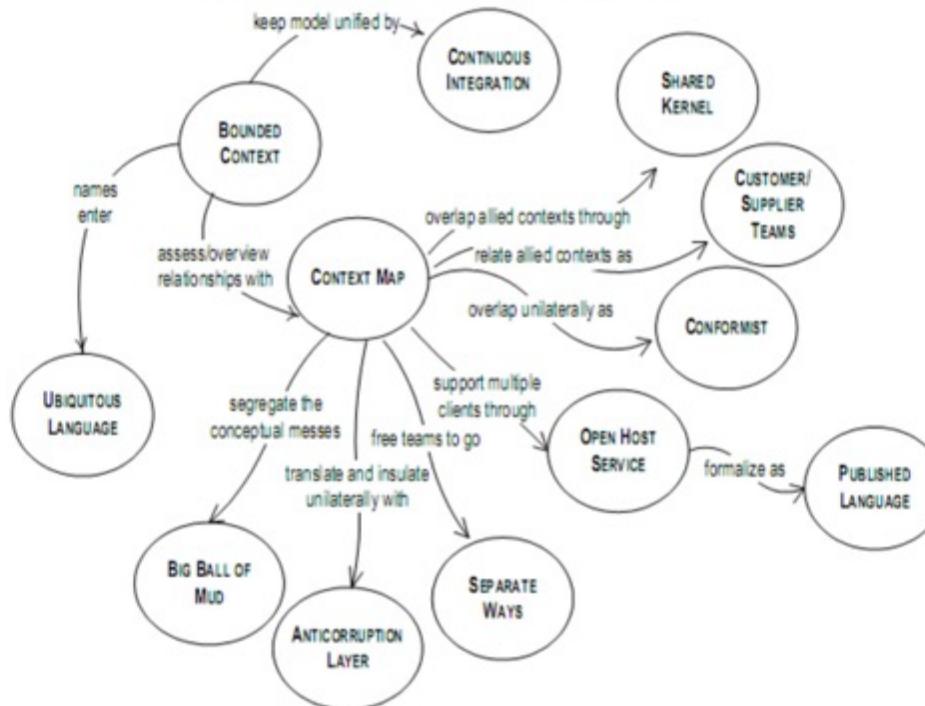
Domain Driven Design





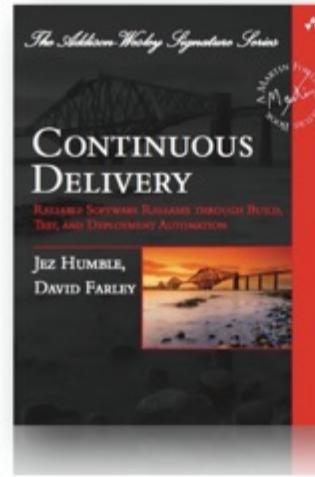
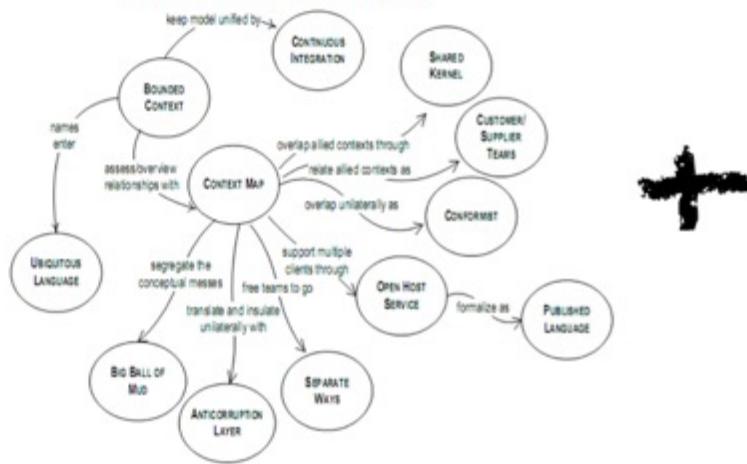
Bounded Context

Maintaining Model Integrity



Bounded Context + Continuous Delivery = microservices

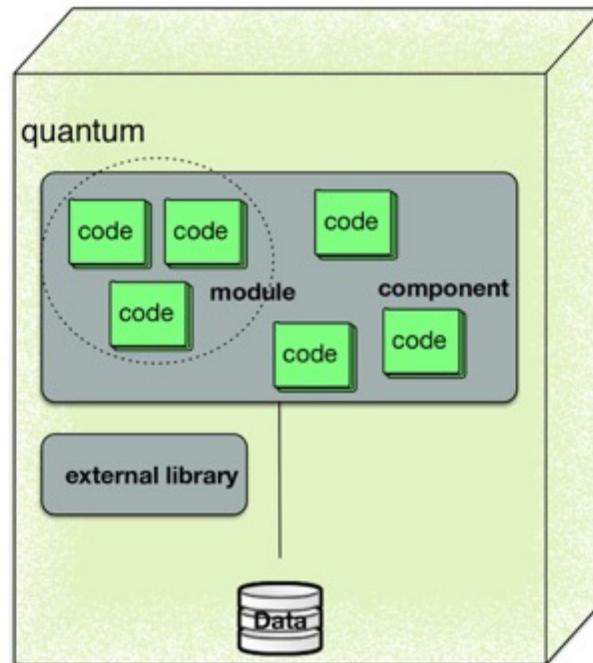
Maintaining Model Integrity



Architectural Quantum

An architectural quantum is an independently deployable component with high functional cohesion, which includes all the structural elements required for the system to function properly.

Architectural Quantum



Why Quantum?

Why Quantum?

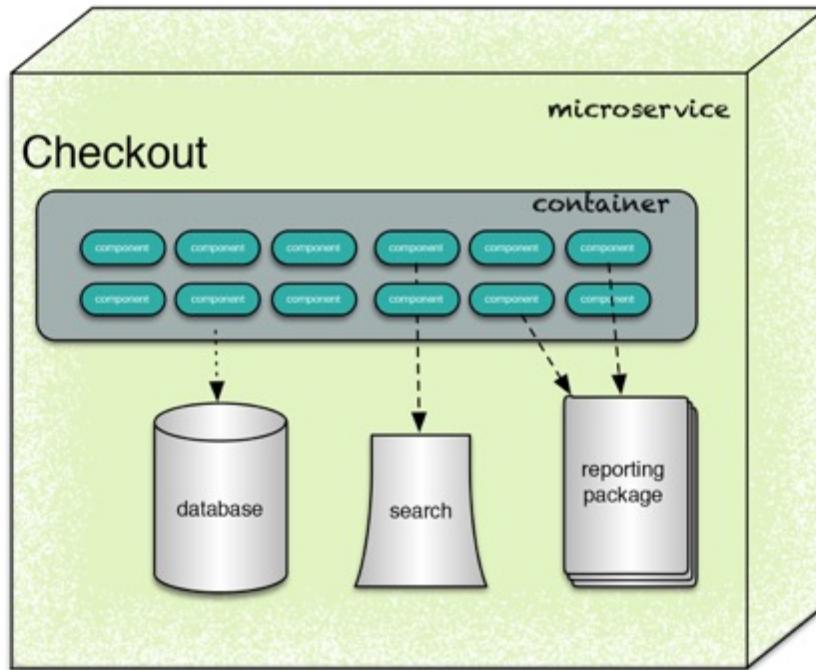


operational view
of architecture

Why Quantum?



operational view
of architecture



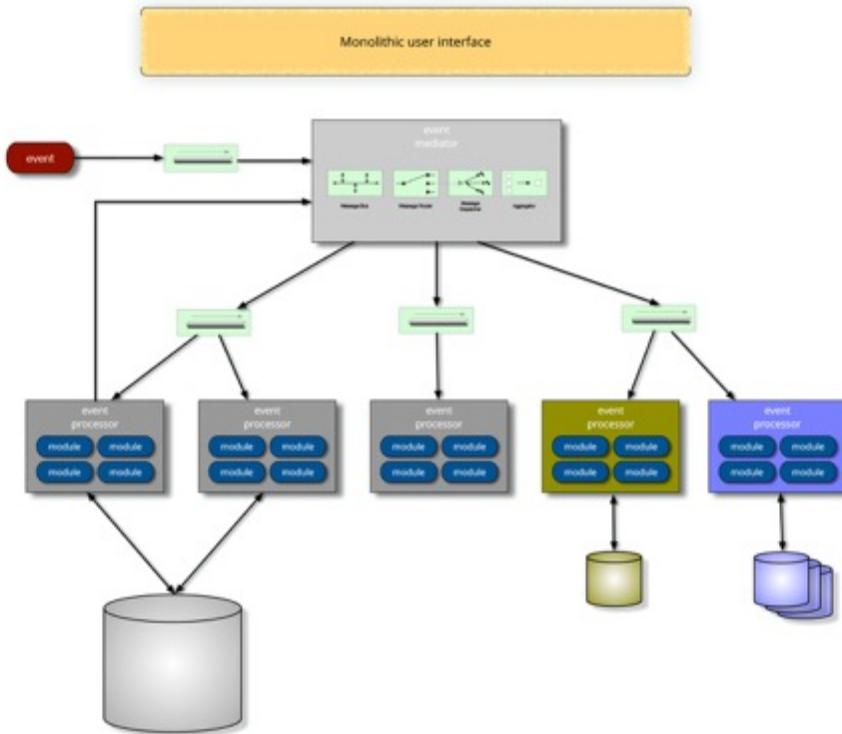
holistic



Why Quantum?

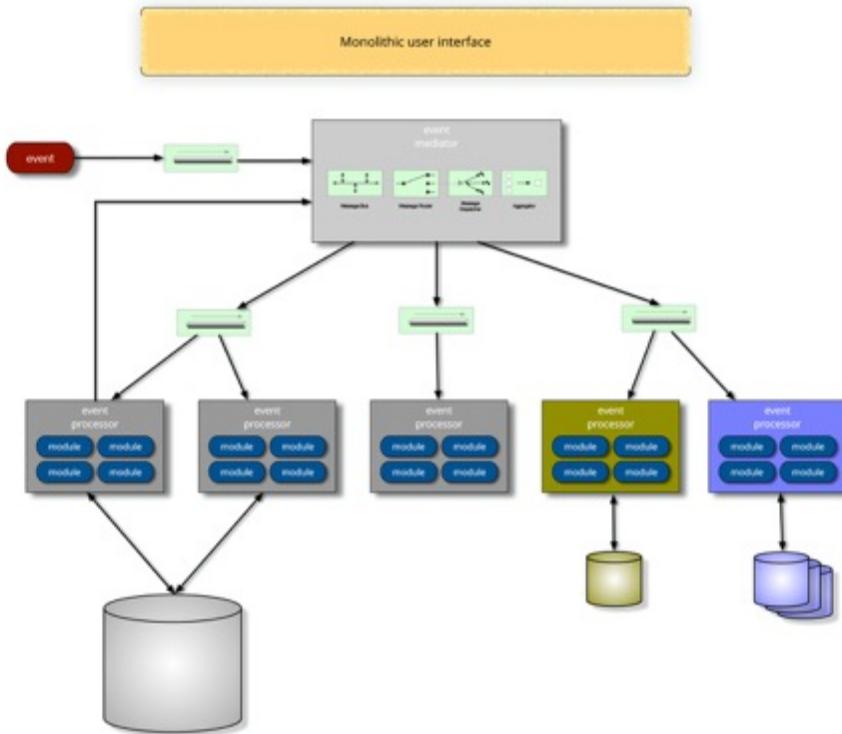
holistic

Why Quantum?



holistic

Why Quantum?



"multiple dimensions"



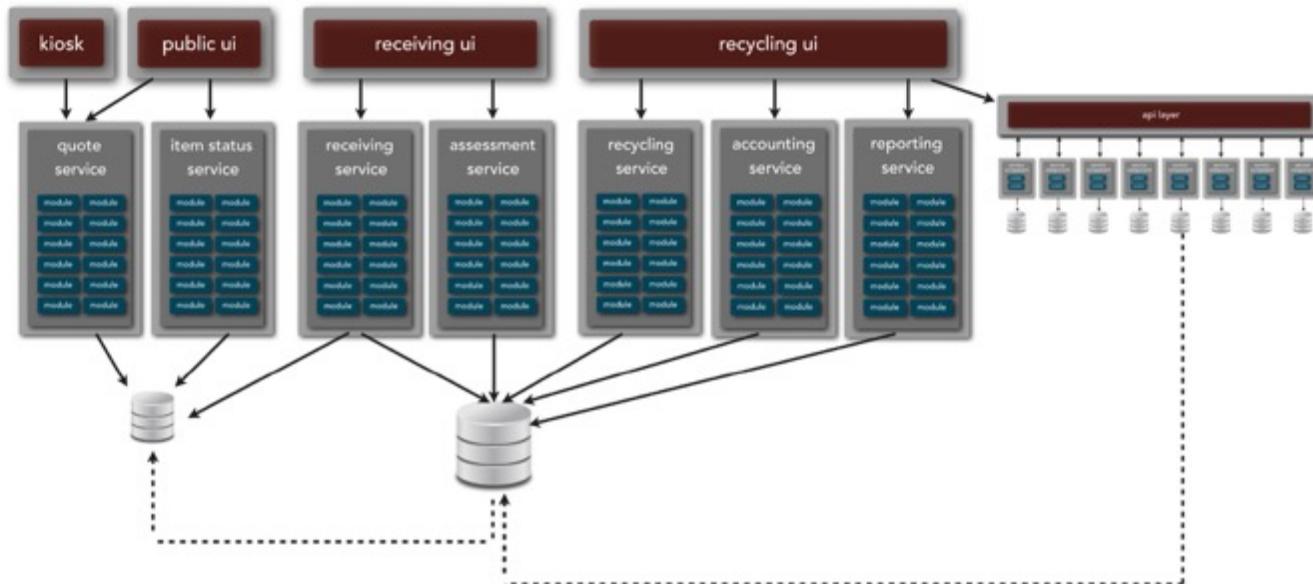
Why Quantum?

useful for
architectural analysis



Why Quantum?

useful for
architectural analysis



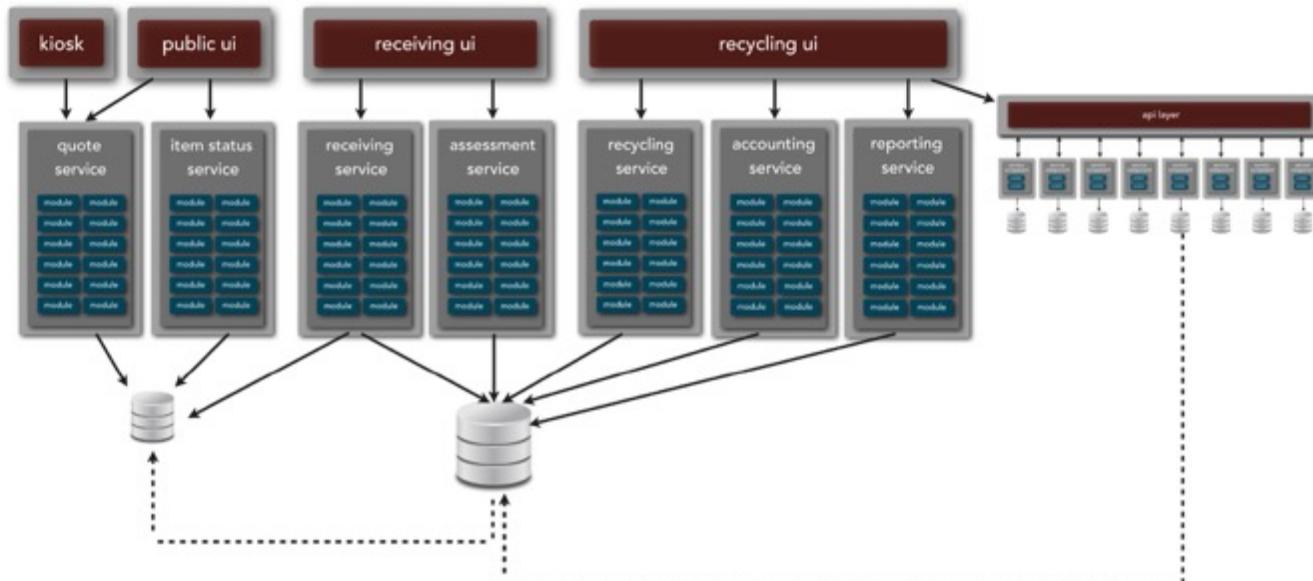


Why Quantum?

useful for
architectural analysis

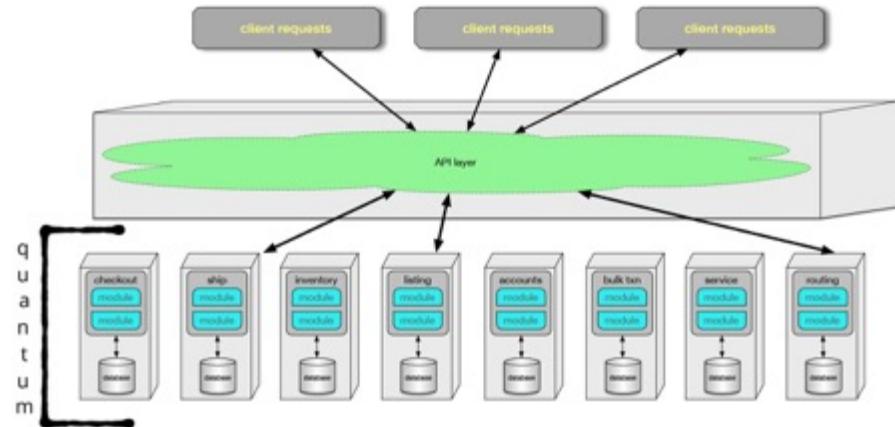
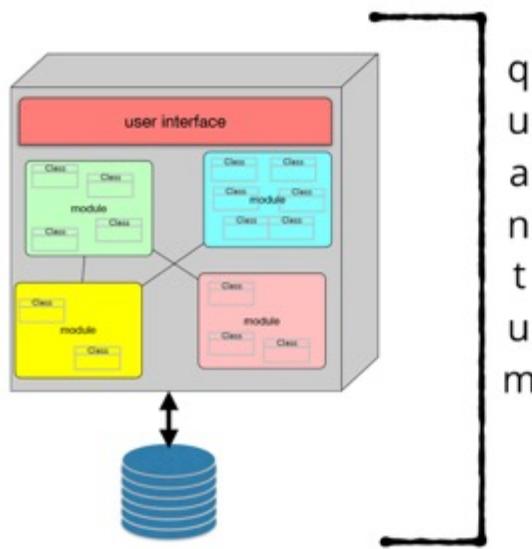


helps analyze
coupling



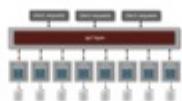
Why Quantum?

The quantum is where architectural characteristics live.

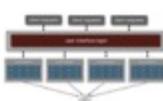


Architectural Patterns

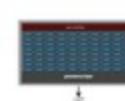
Architectural Patterns



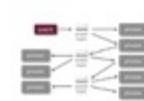
microservices
architecture



service-based
architecture



layered
architecture



event-driven
architecture



pipeline
architecture

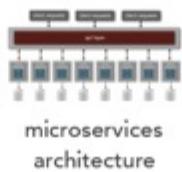


microkernel
architecture

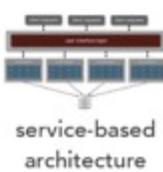


space-based
architecture

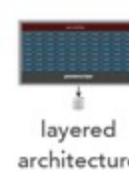
Architectural Patterns



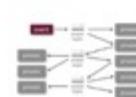
microservices
architecture



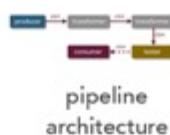
service-based
architecture



layered
architecture



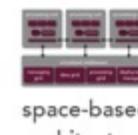
event-driven
architecture



pipeline
architecture



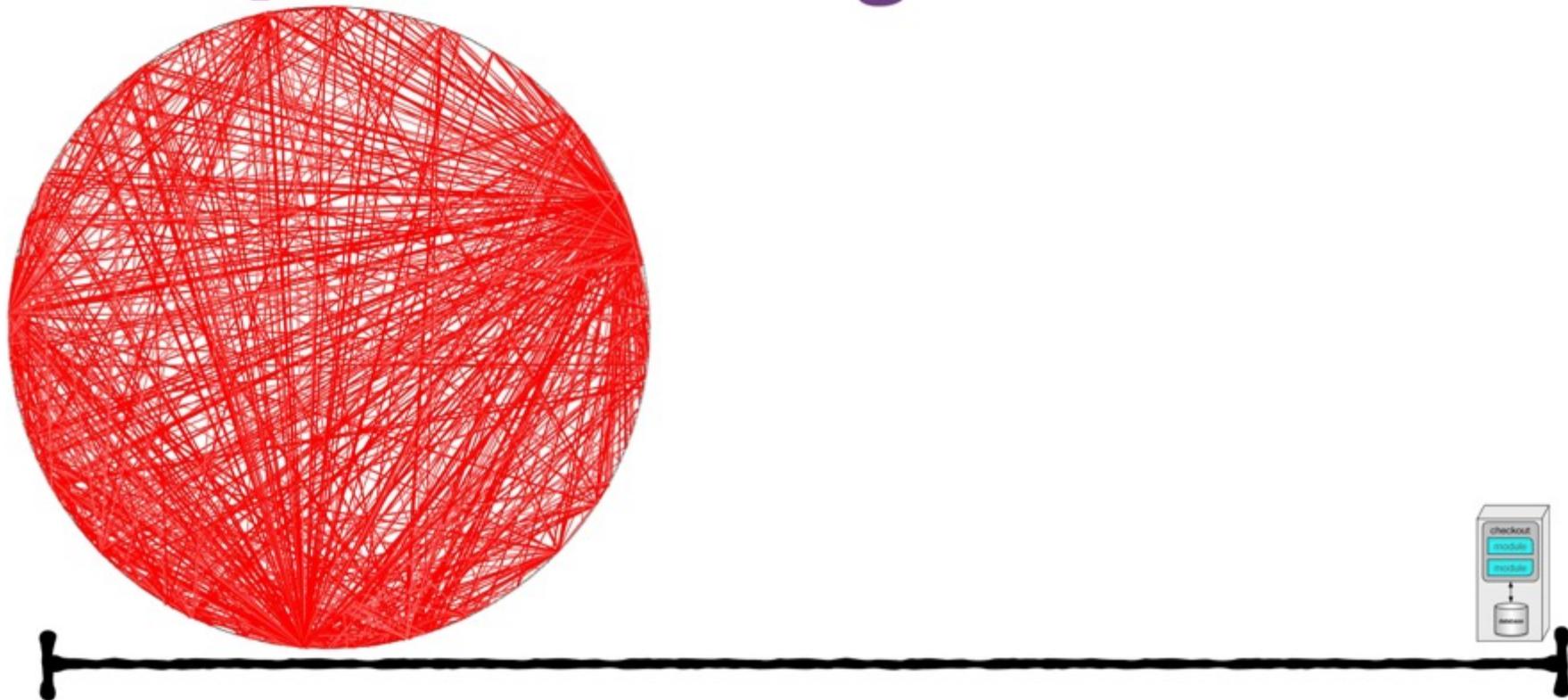
microkernel
architecture



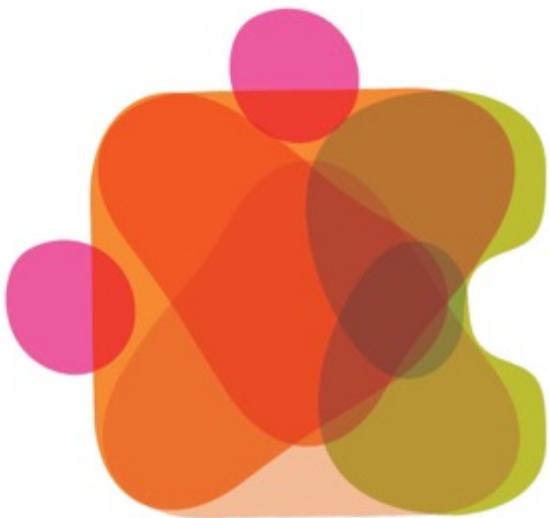
space-based
architecture

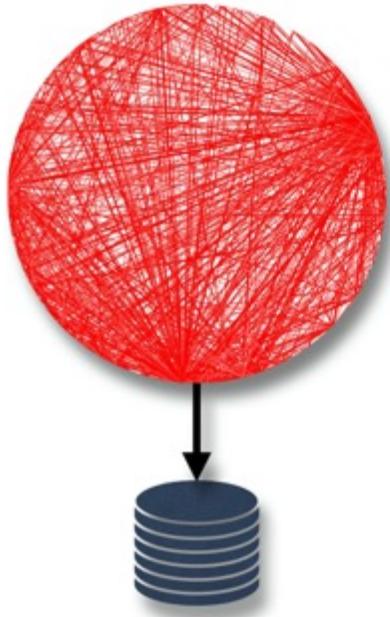
Patterns encapsulate a well-known set of architectural characteristics.

Quantum: Large to Small



Monoliths

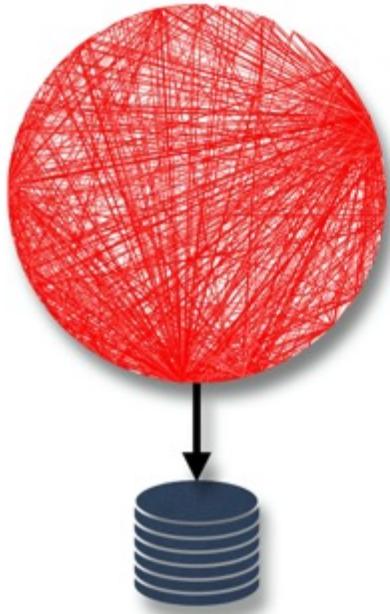




Monoliths

"Big Ball of Mud"

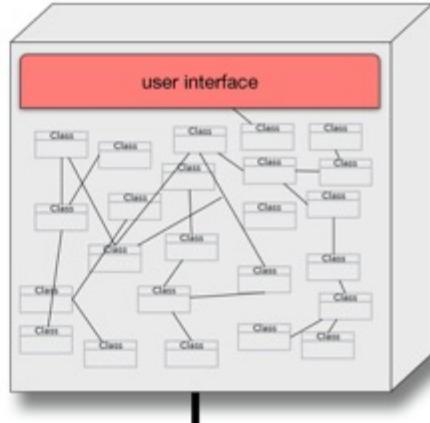
https://en.wikipedia.org/wiki/Big_ball_of_mud



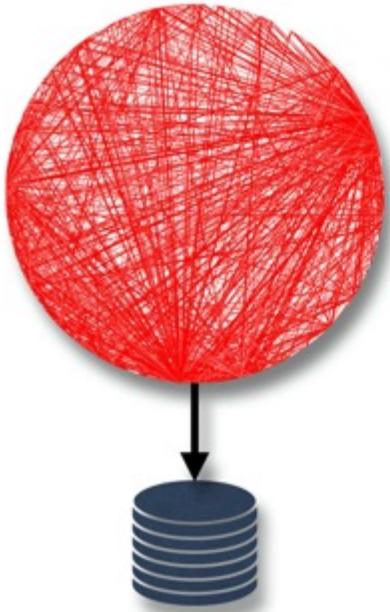
"Big Ball of Mud"

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

Monoliths

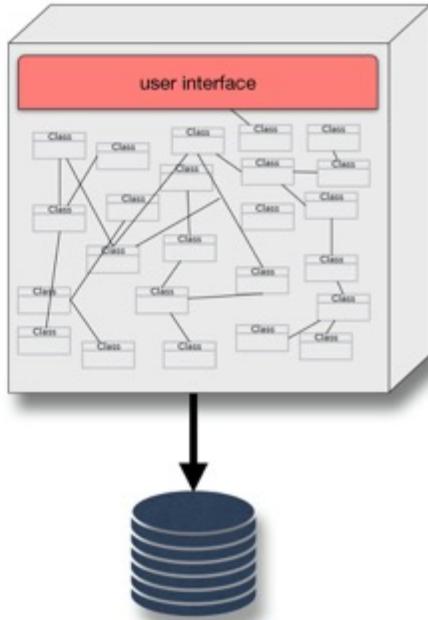


unstructured
monolith

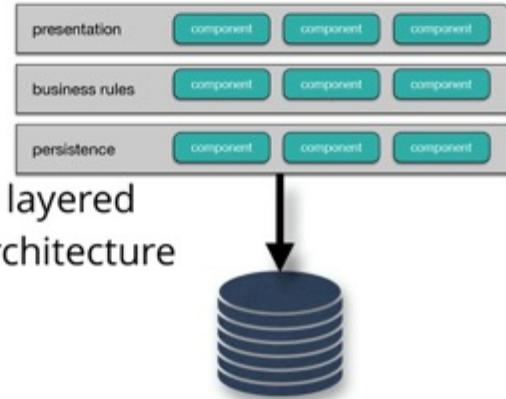


“Big Ball of Mud”

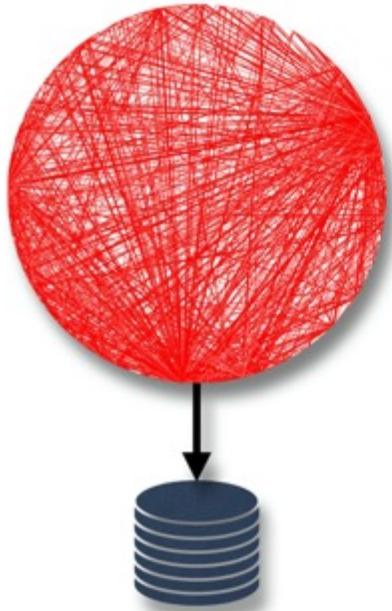
https://en.wikipedia.org/wiki/Big_ball_of_mud



unstructured
monolith

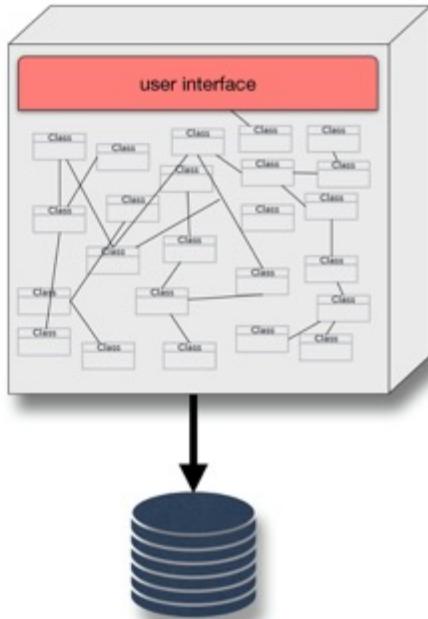


layered
architecture



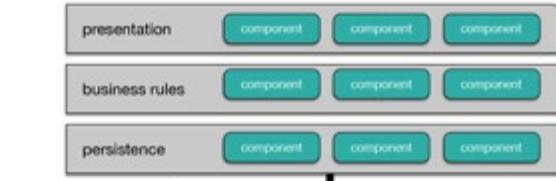
“Big Ball of Mud”

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

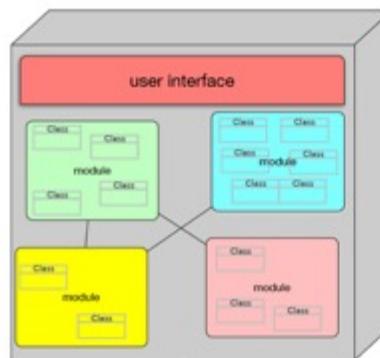


unstructured
monolith

Monoliths

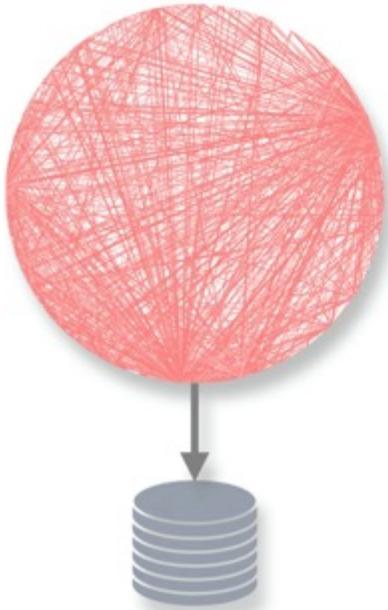


layered
architecture



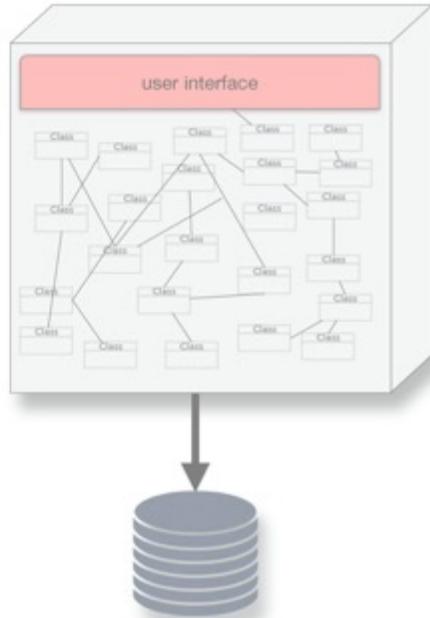
modular monolith

<http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths>



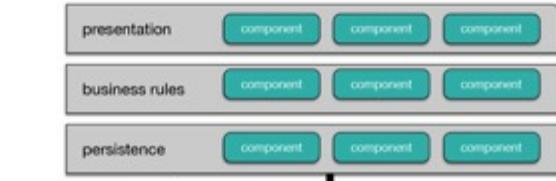
“Big Ball of Mud”

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

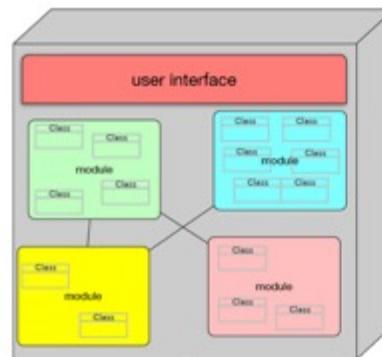


unstructured
monolith

Monoliths



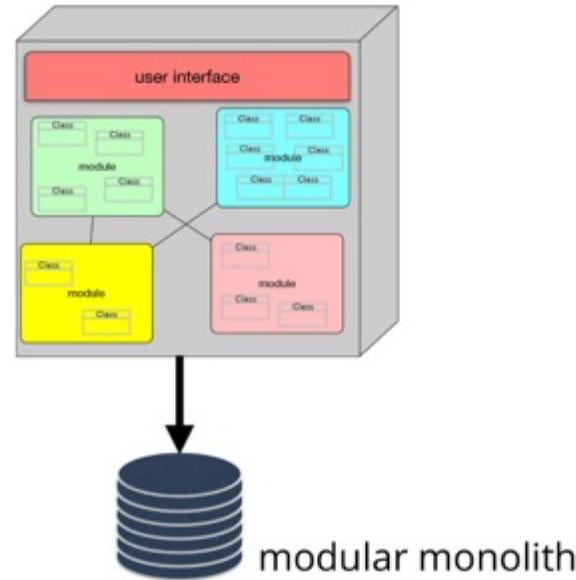
layered
architecture



modular monolith

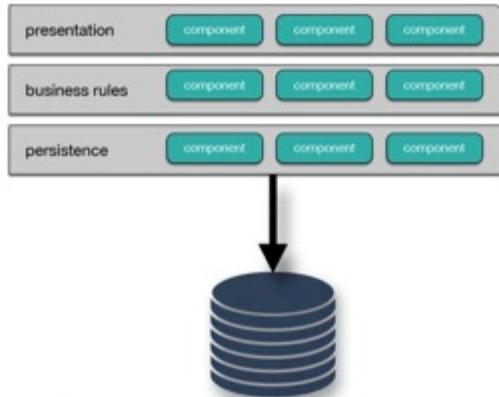
<http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths>

Monoliths

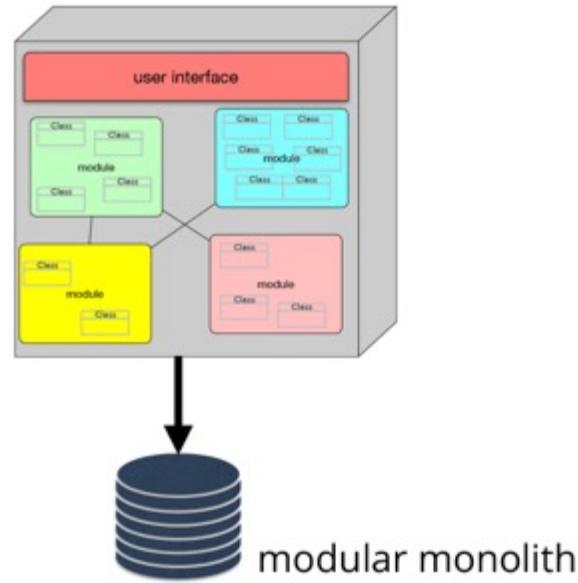


Monoliths

technical partitioning



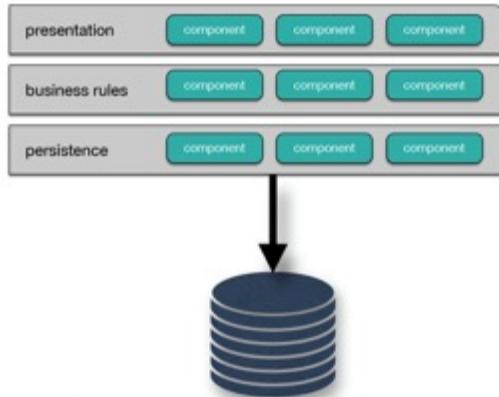
layered
architecture



modular monolith

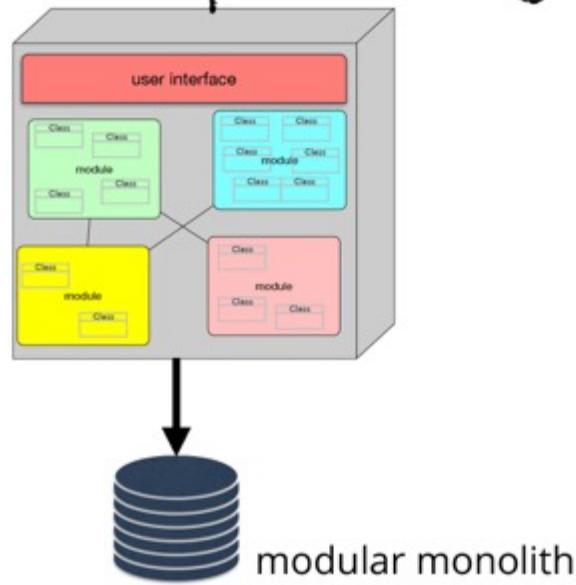
Monoliths

technical partitioning

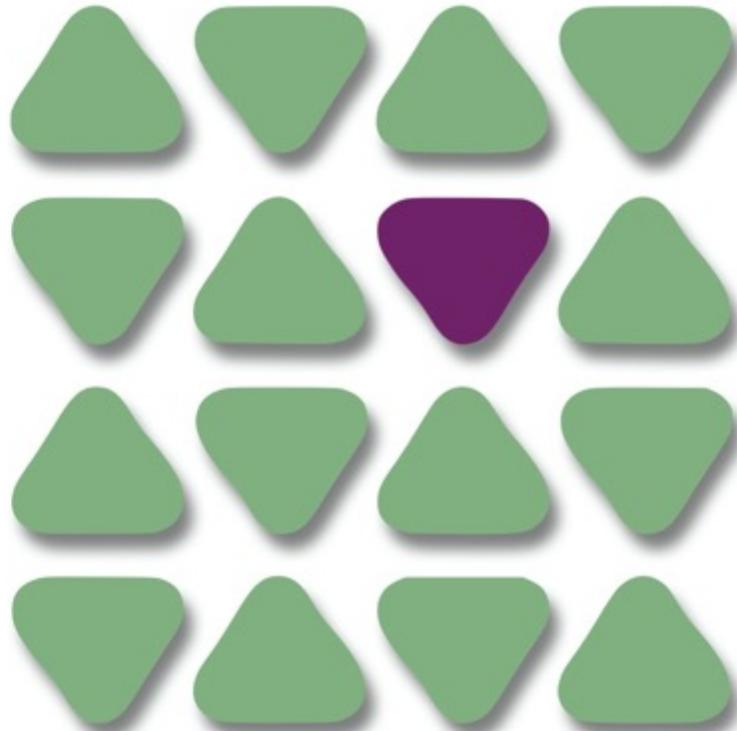


layered
architecture

domain partitioning

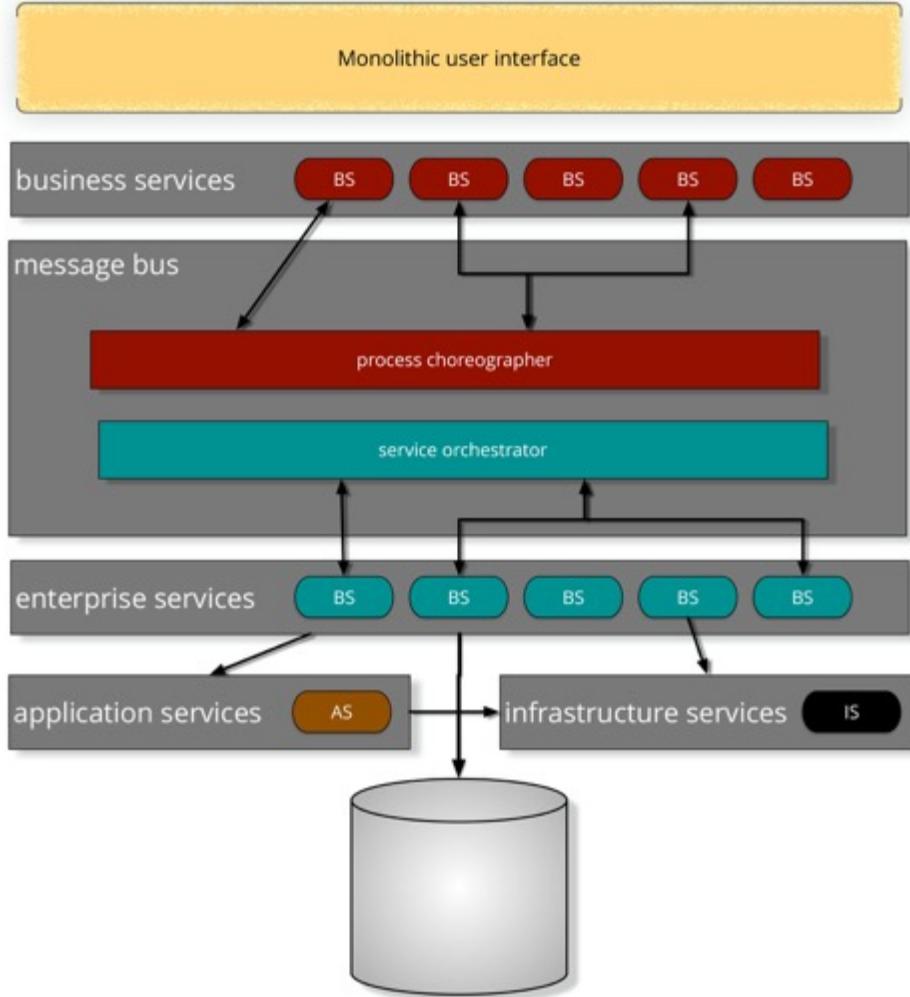


modular monolith

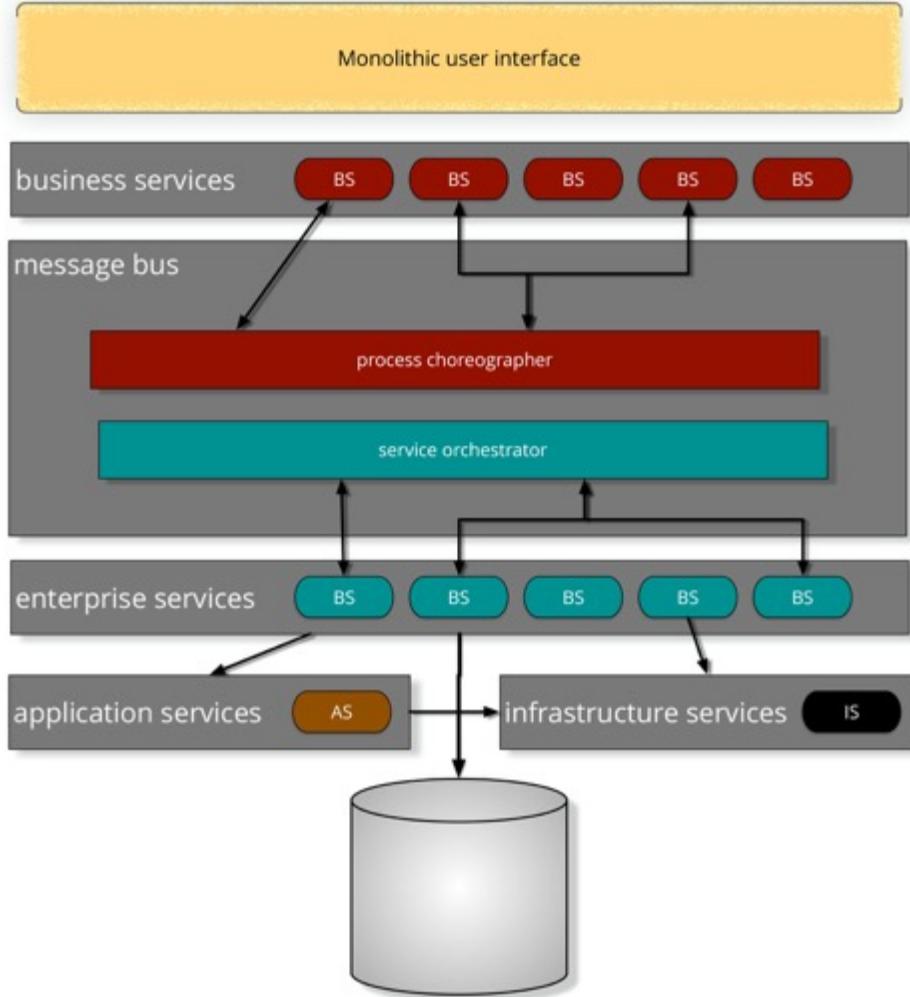


distributed systems

ESB-driven SOA

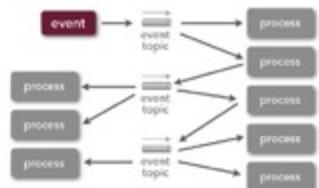


ESB-driven SOA

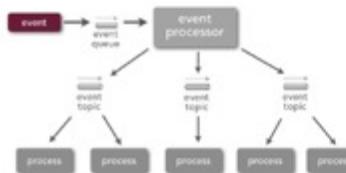


technical partitioning

Event-driven Architectures

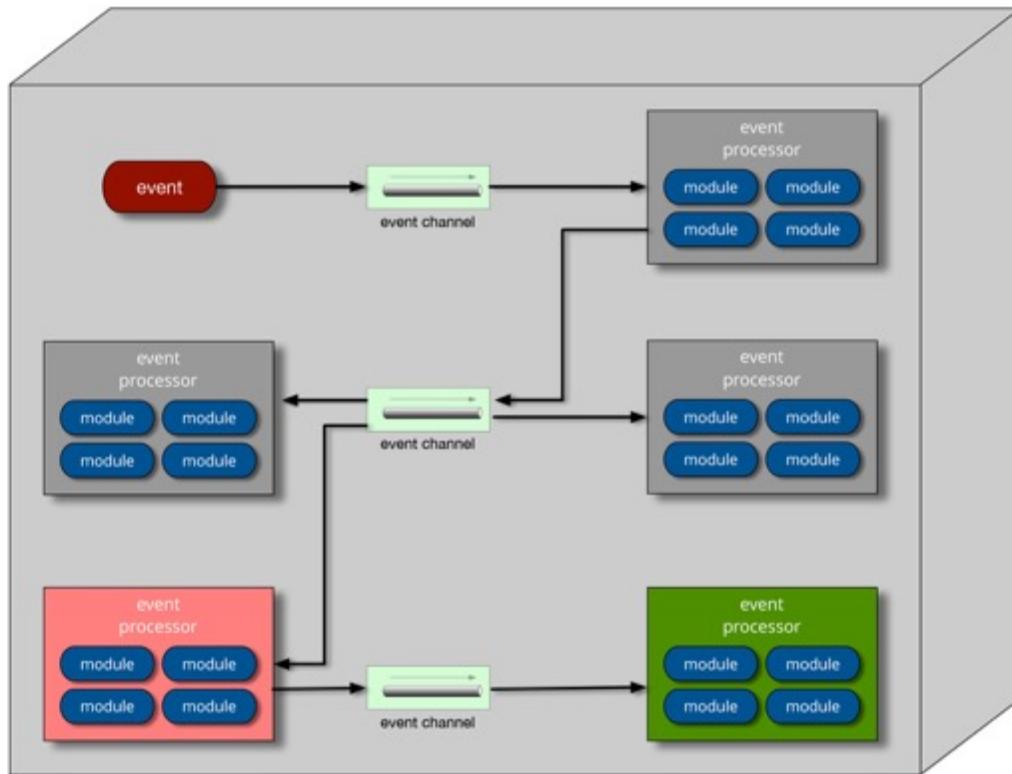


broker topology

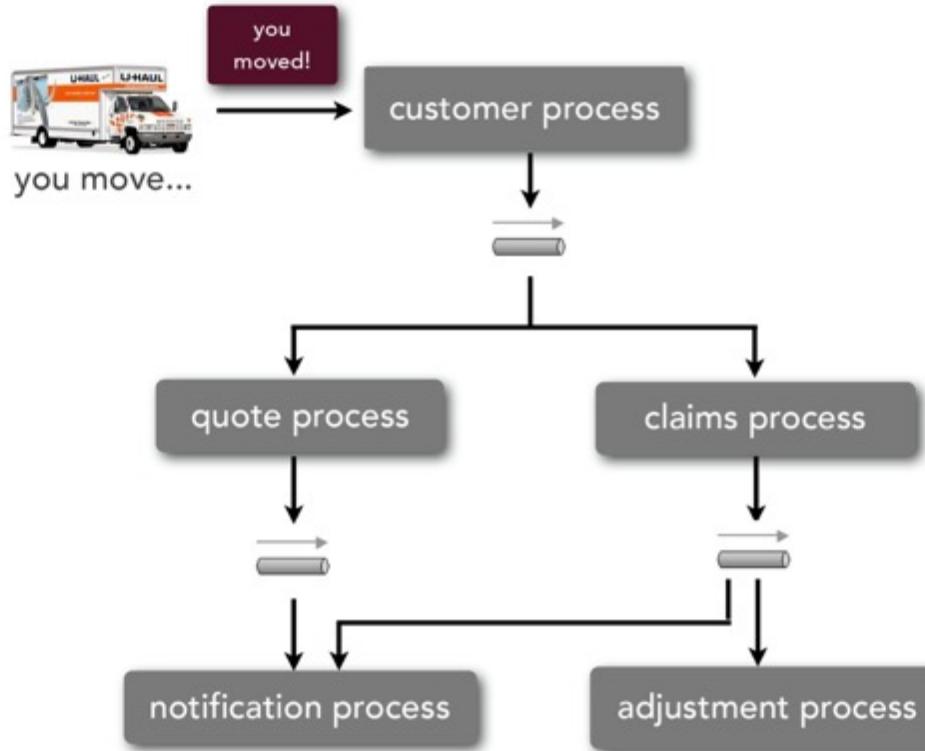


mediator topology

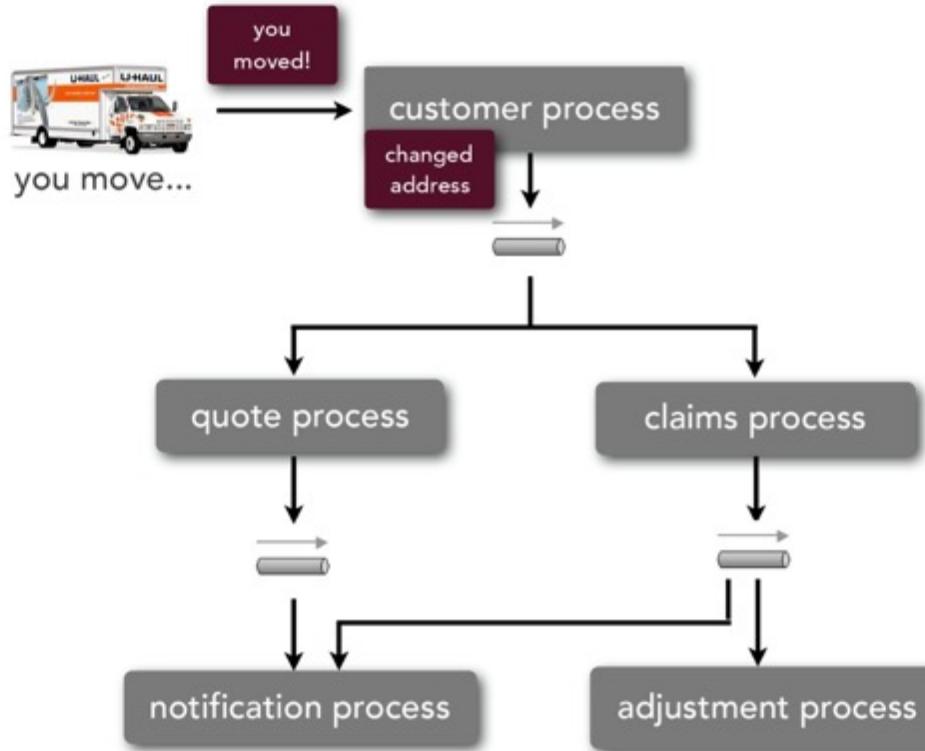
Broker Base Architecture



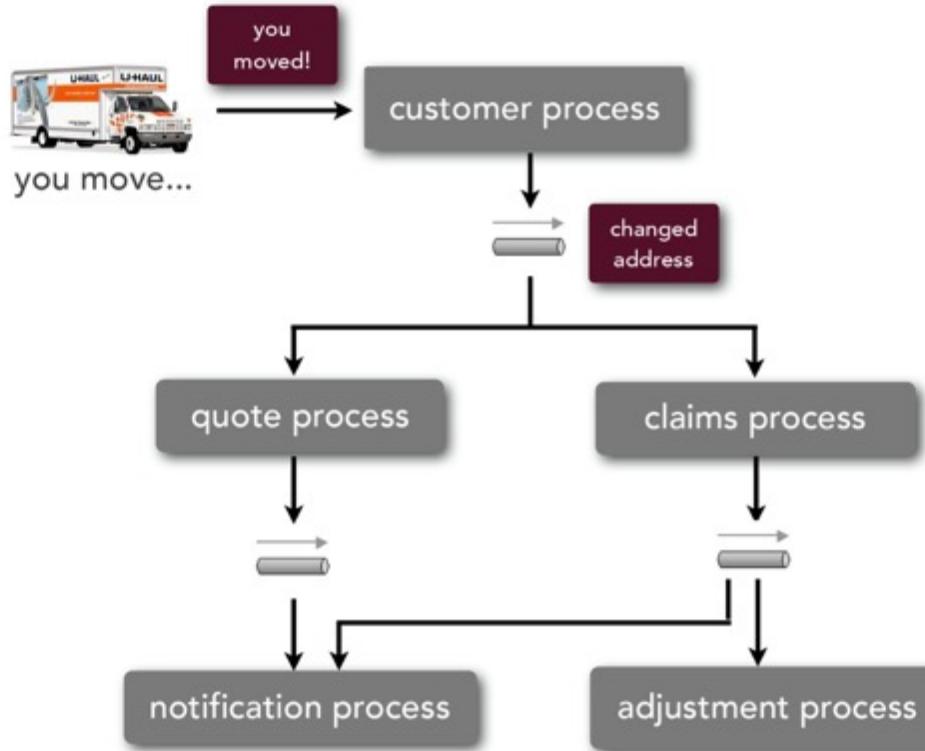
Broker Message Passing



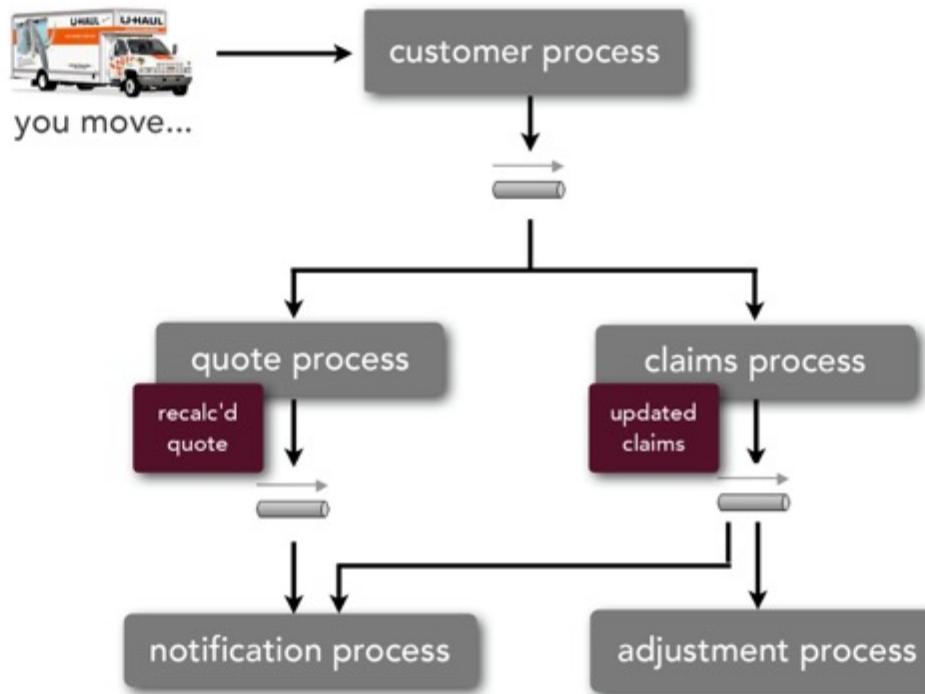
Broker Message Passing



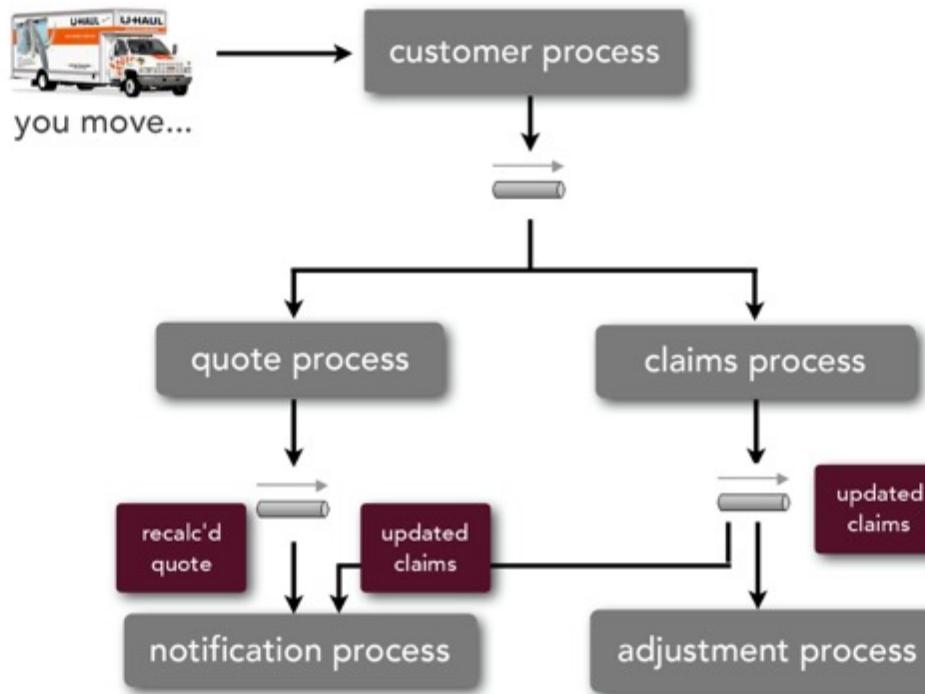
Broker Message Passing

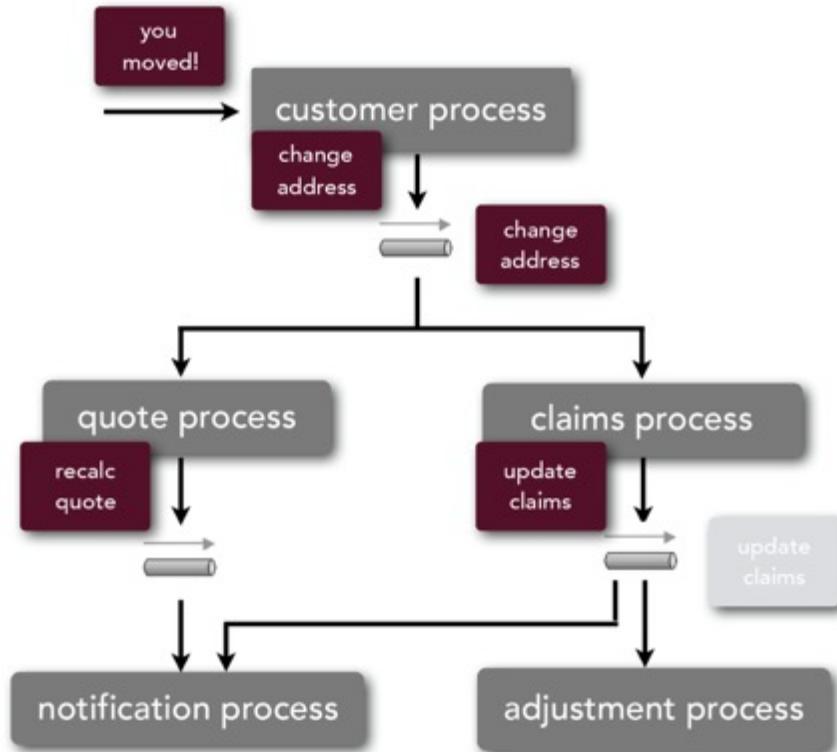


Broker Message Passing



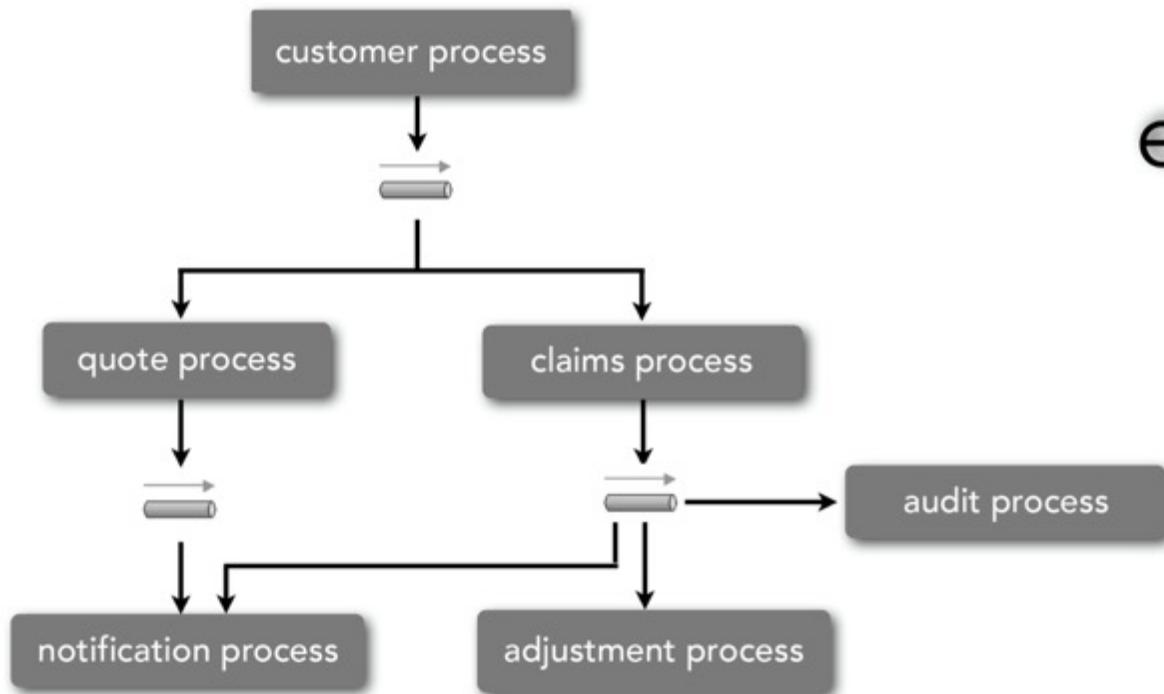
Broker Message Passing



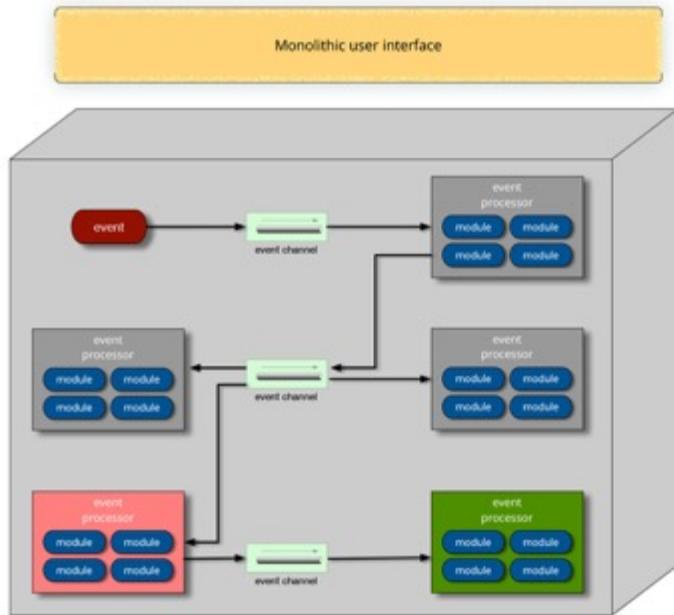


Error handling?

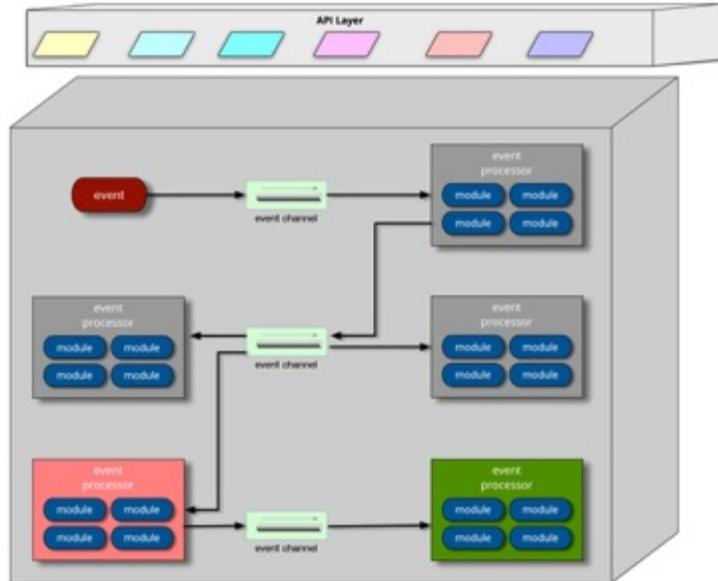
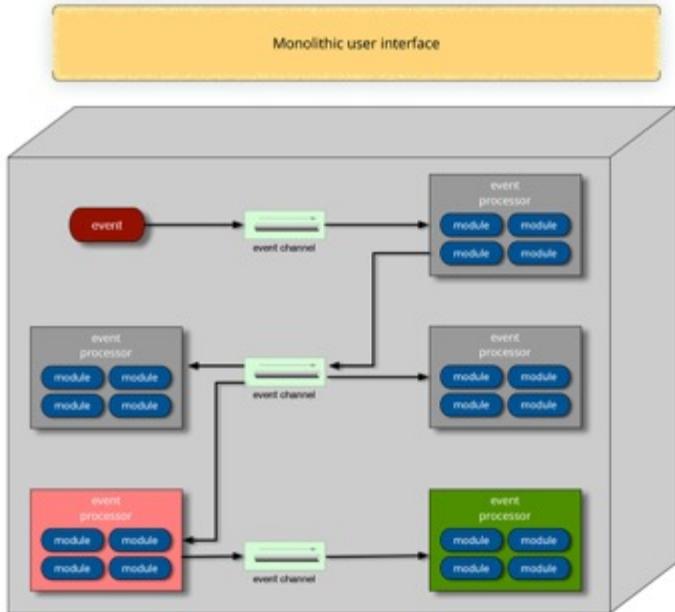
evolution



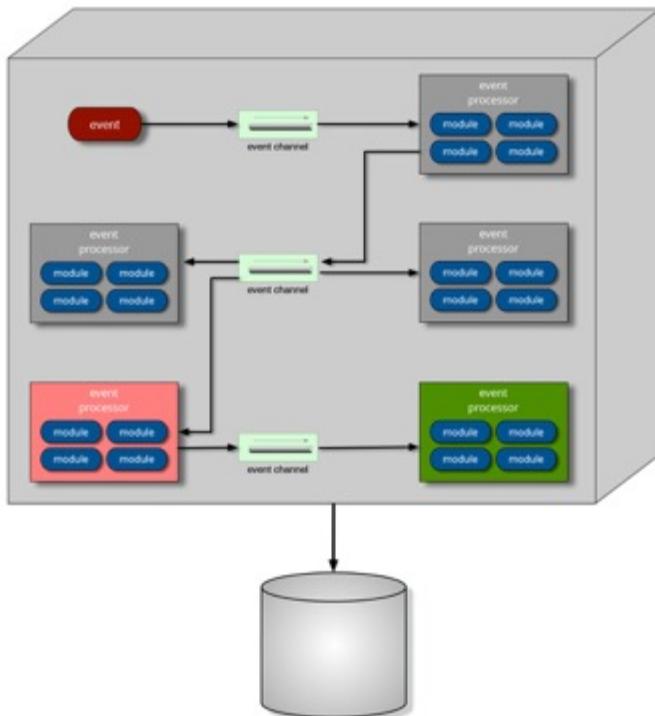
Broker: UI



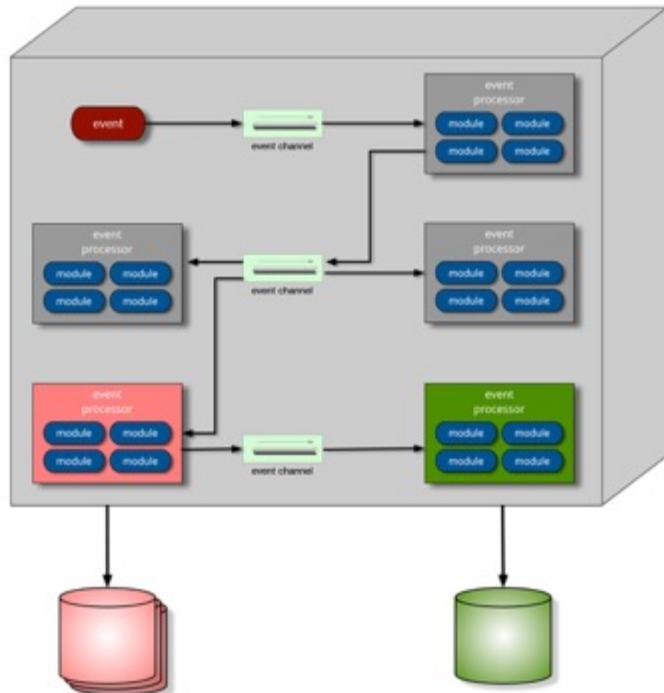
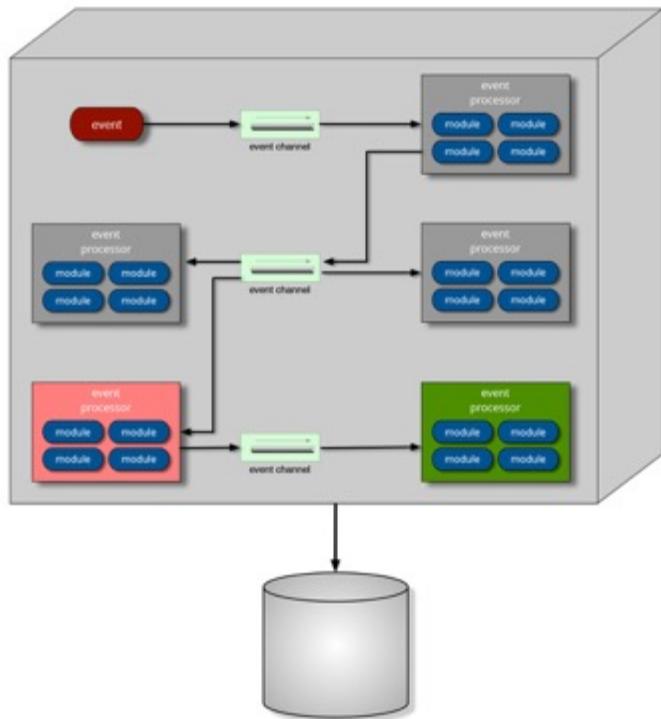
Broker: UI



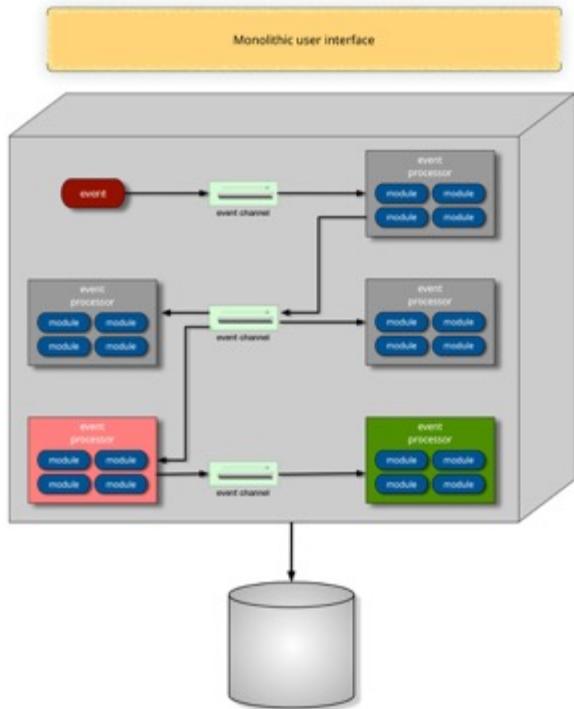
Broker: Data



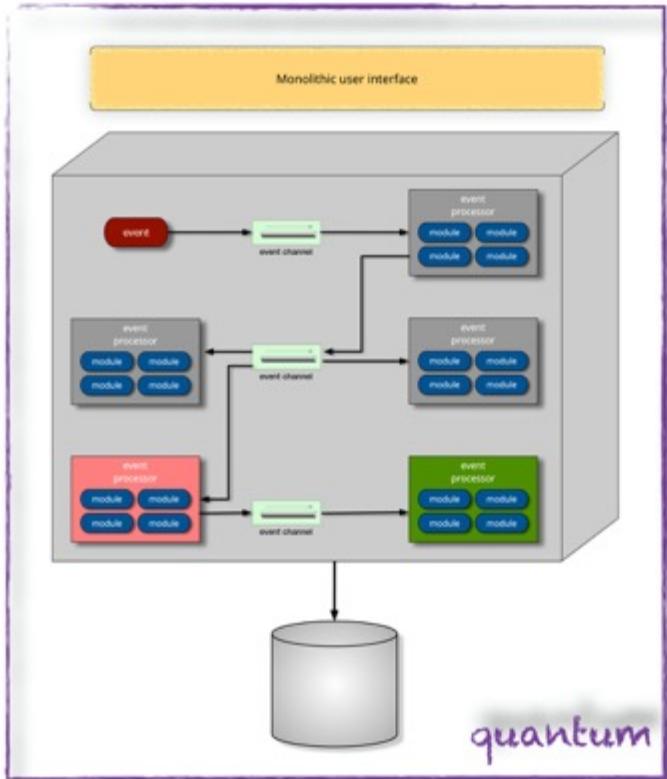
Broker: Data



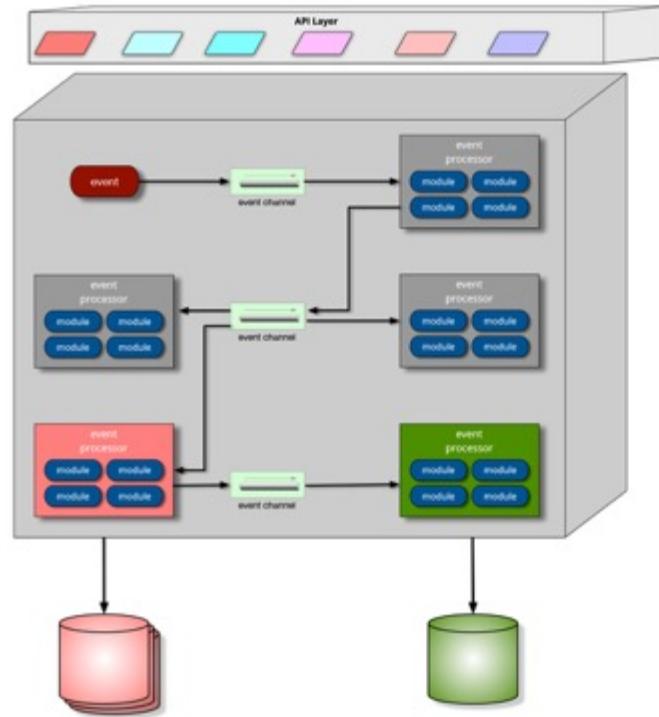
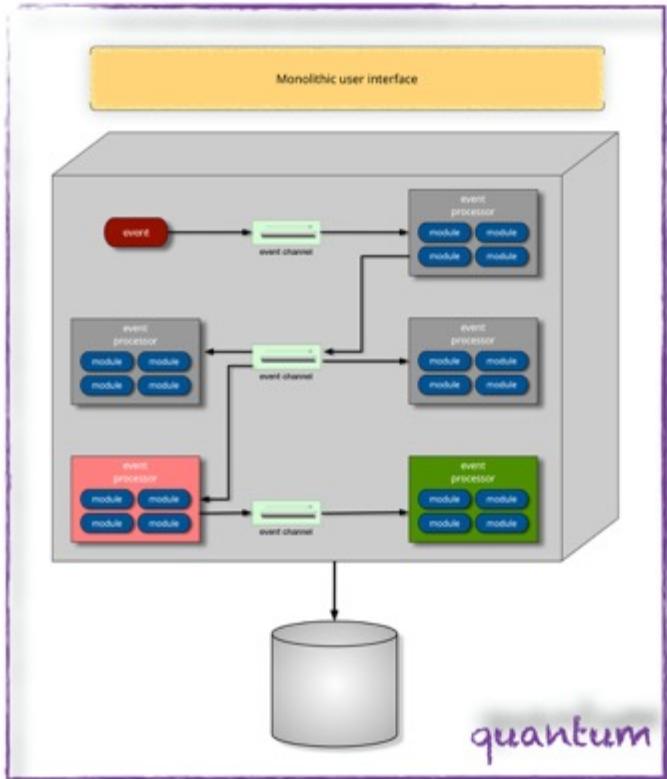
Broker: Quanta



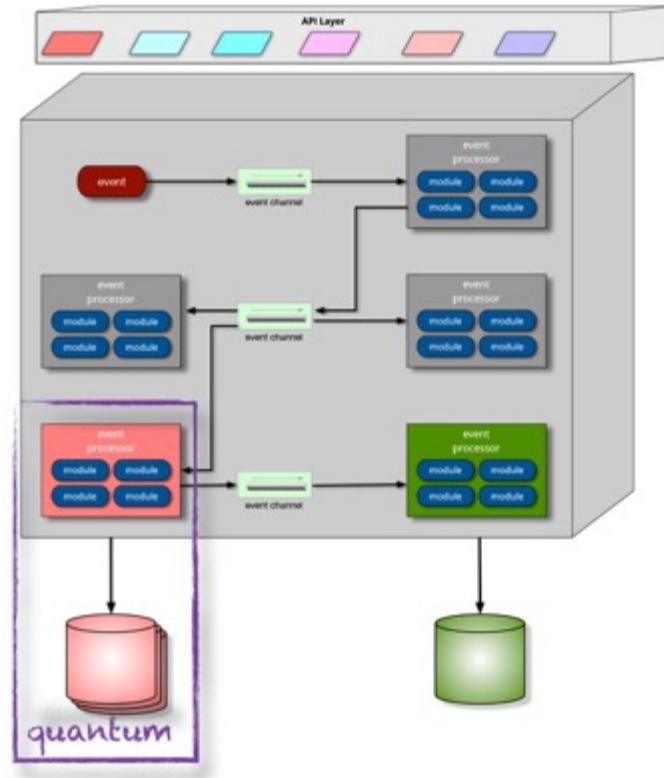
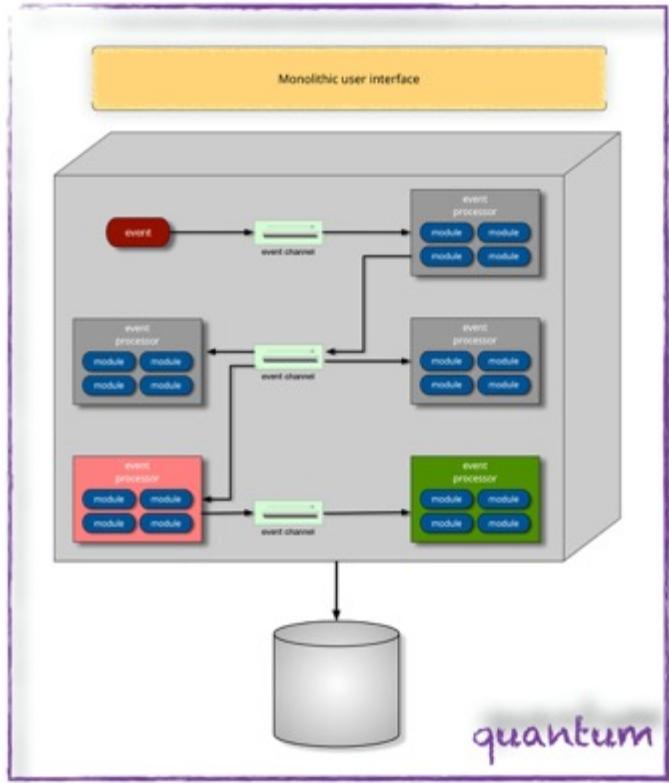
Broker: Quanta



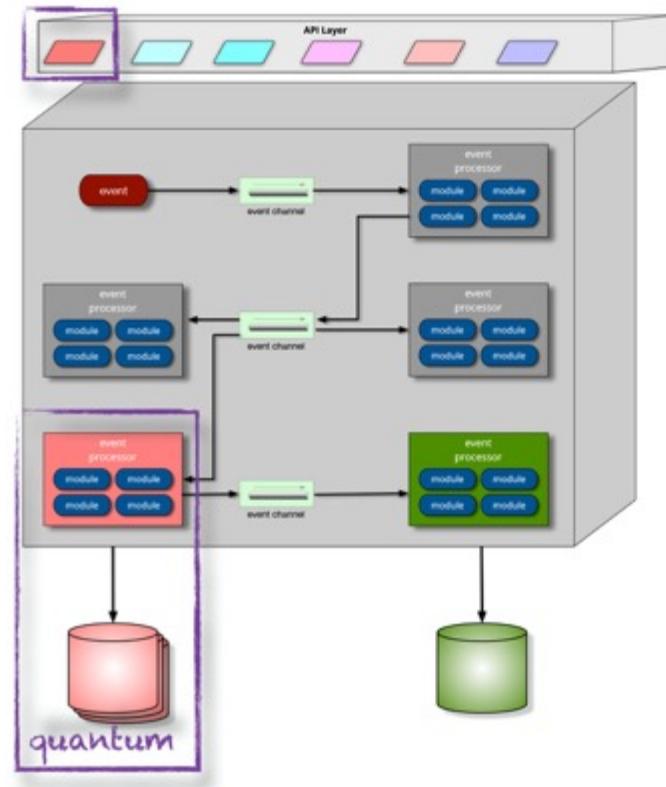
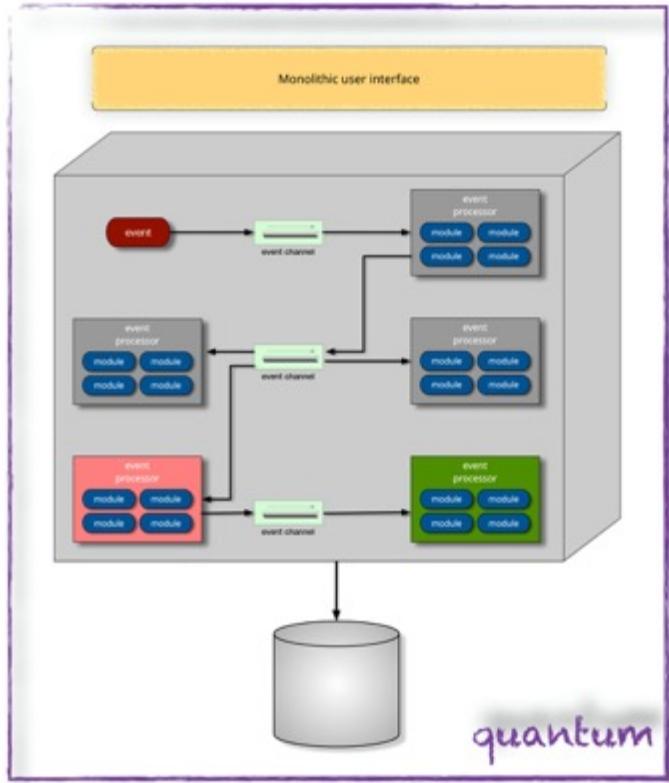
Broker: Quanta



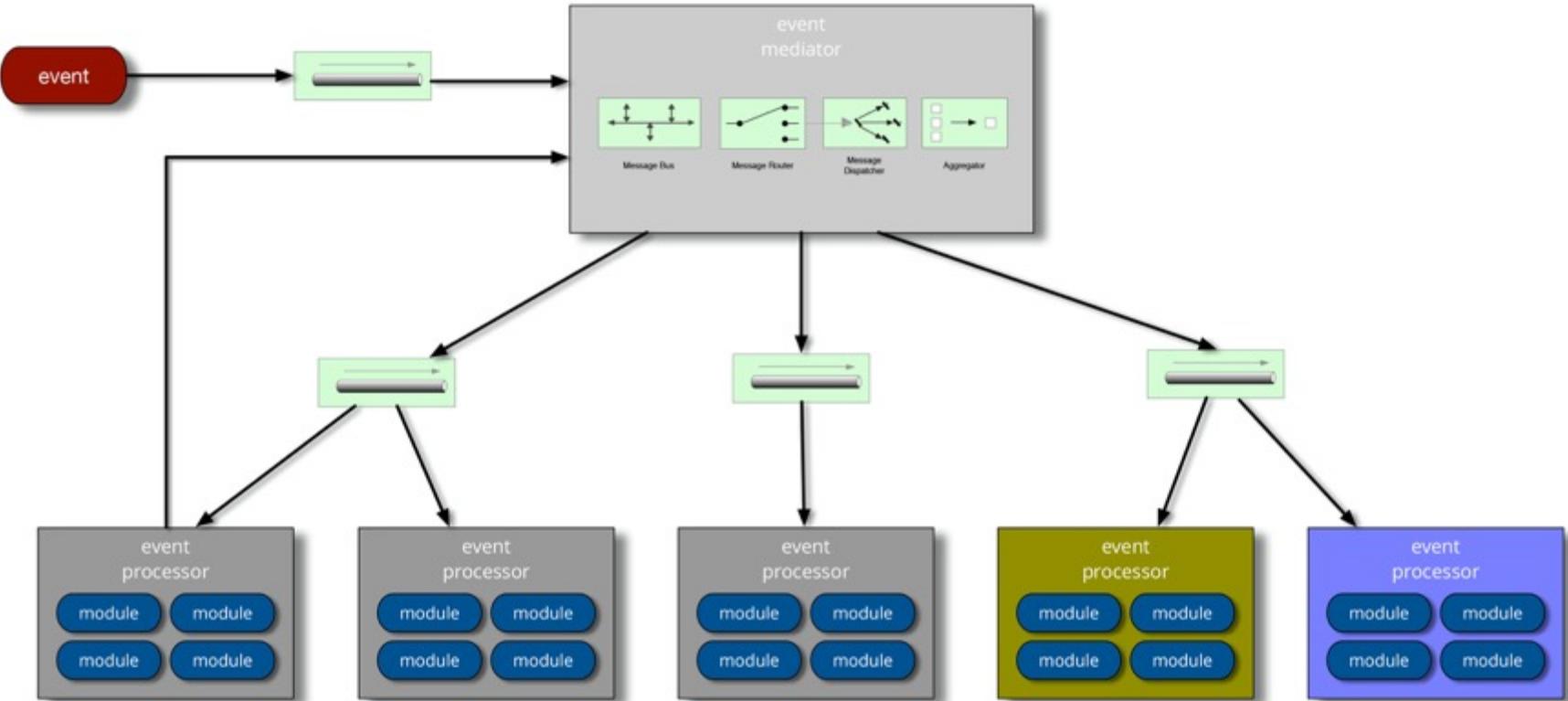
Broker: Quanta



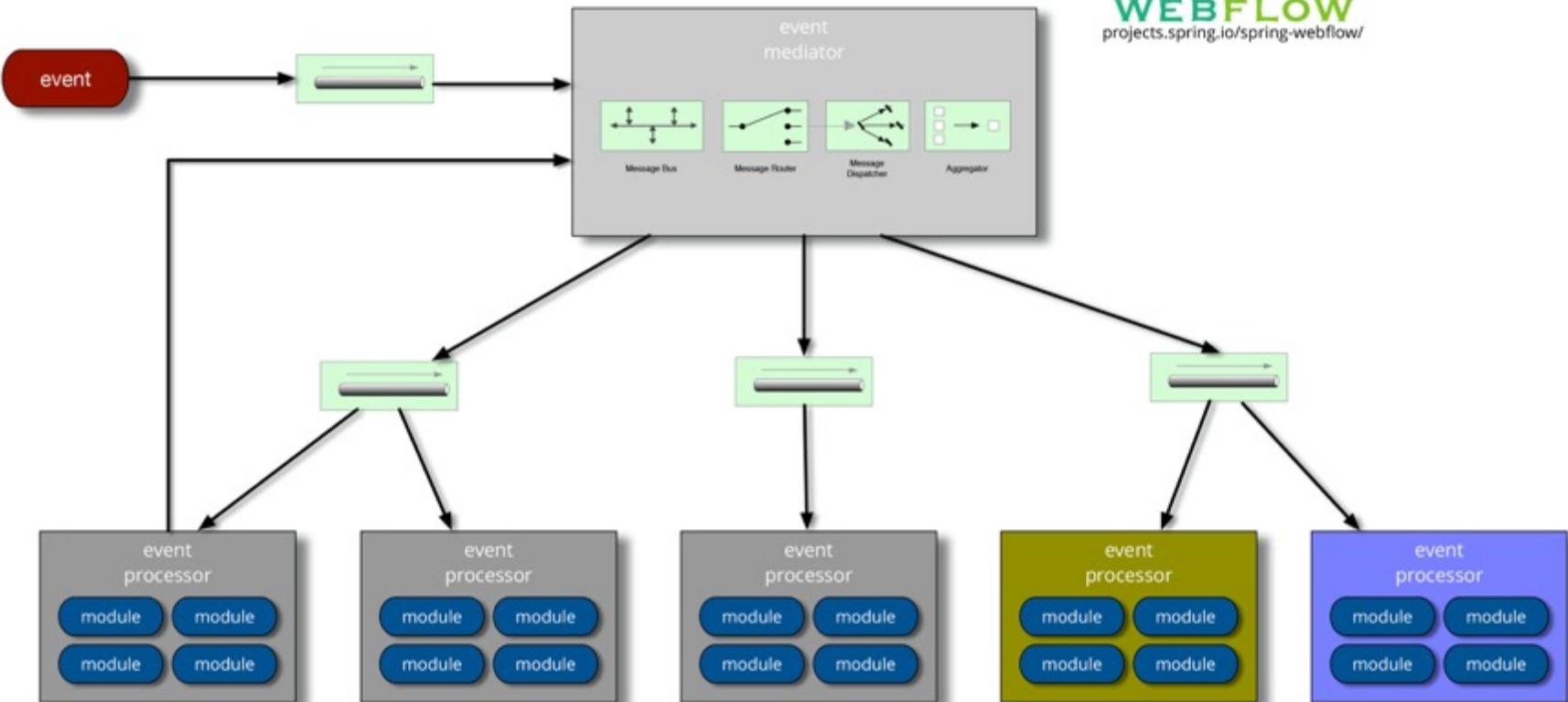
Broker: Quanta



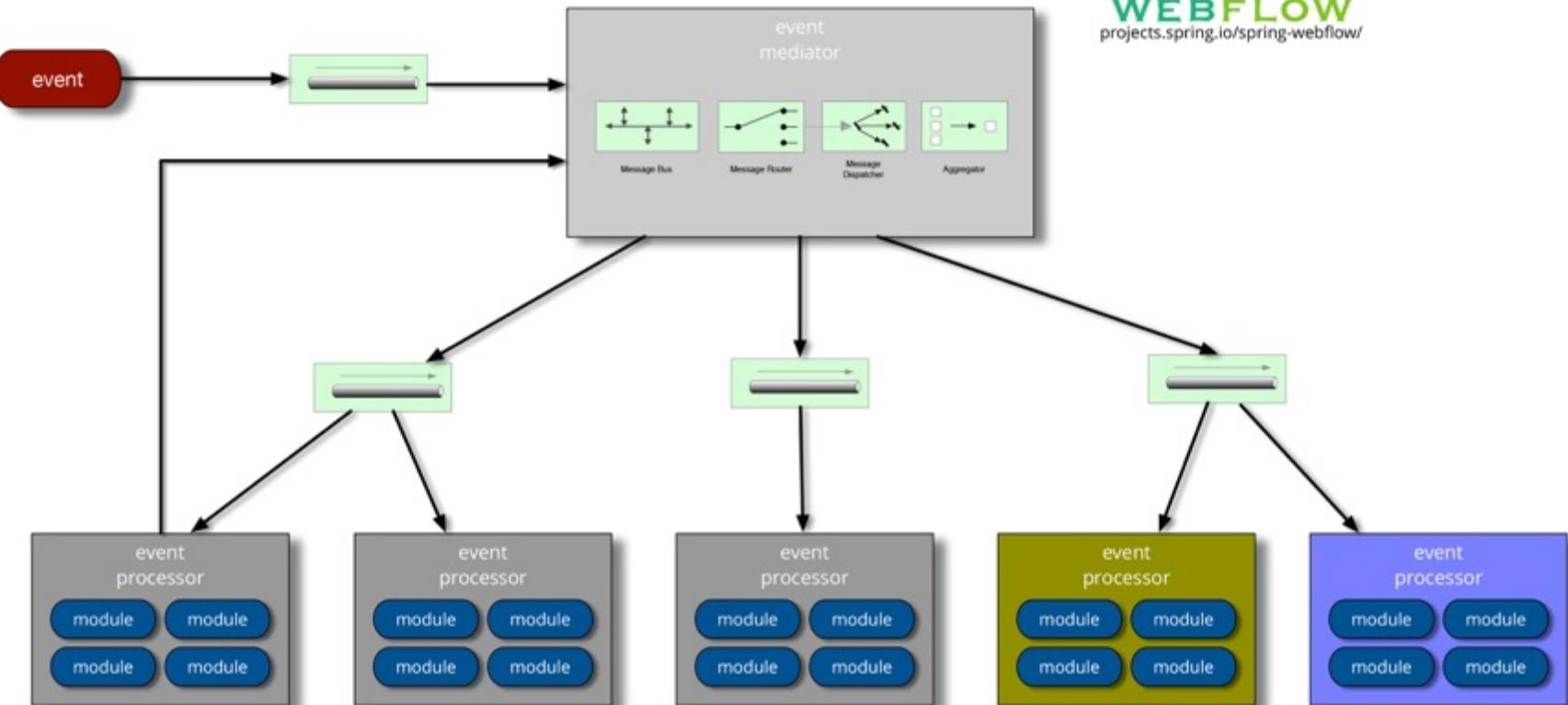
Mediator EDA



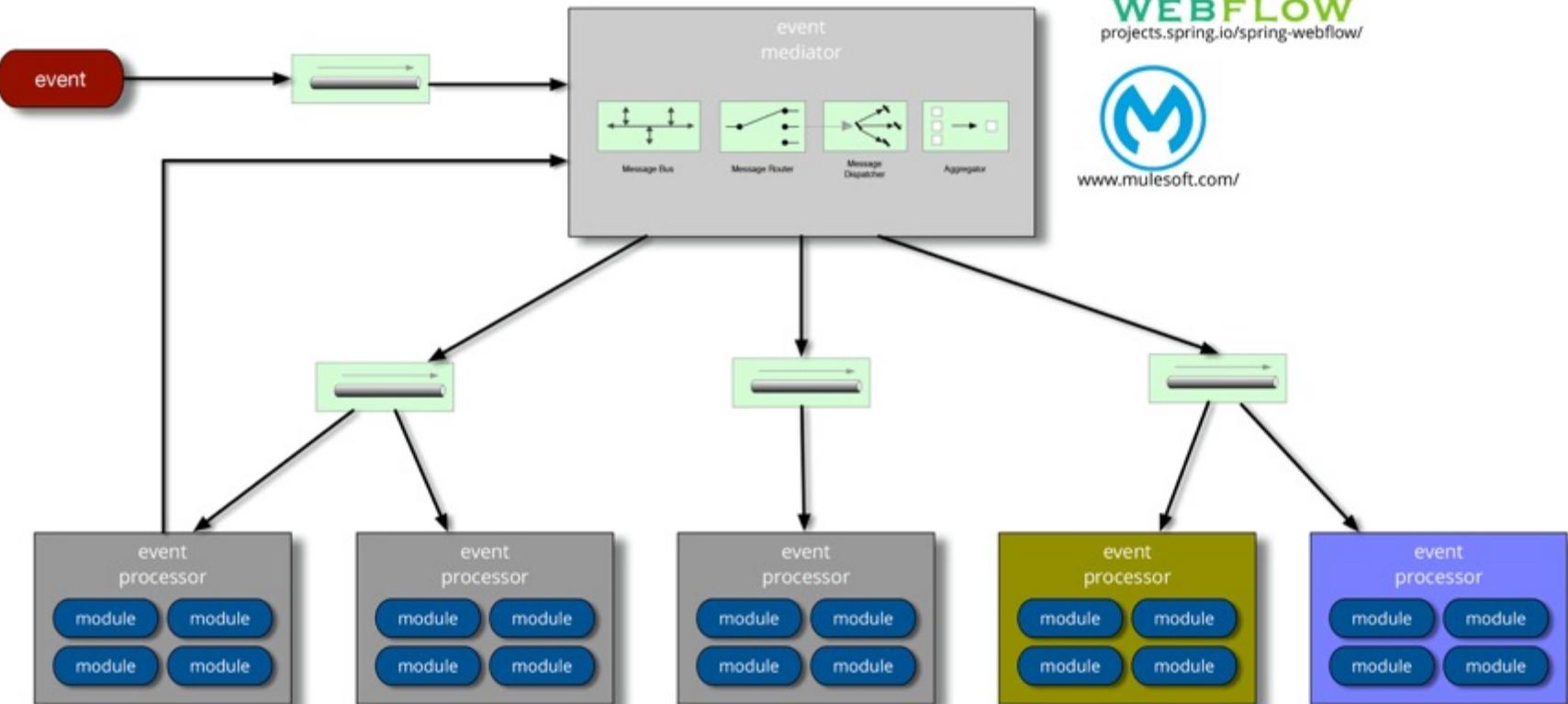
Mediator EDA



Mediator EDA



Mediator EDA

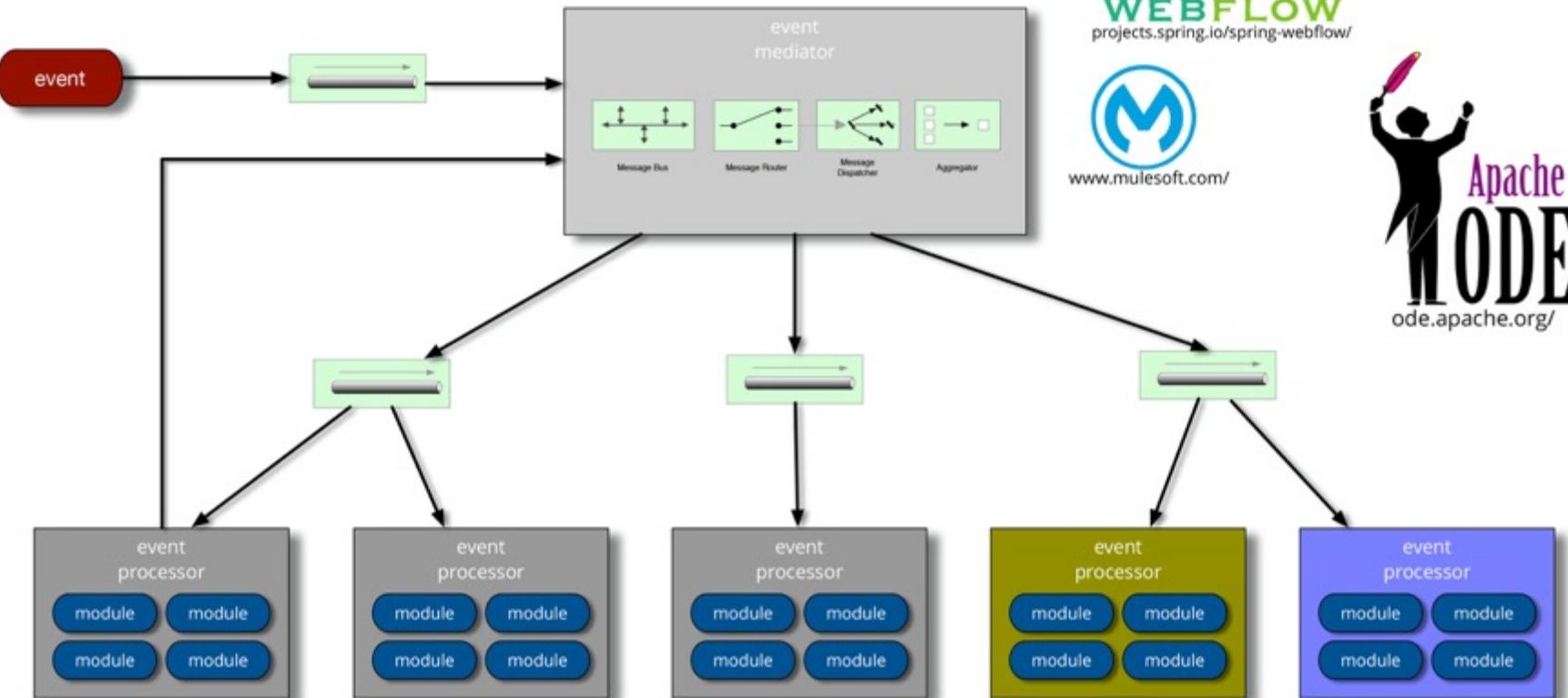


WEBFLOW
projects.spring.io/spring-webflow/



www.mulesoft.com/

Mediator EDA



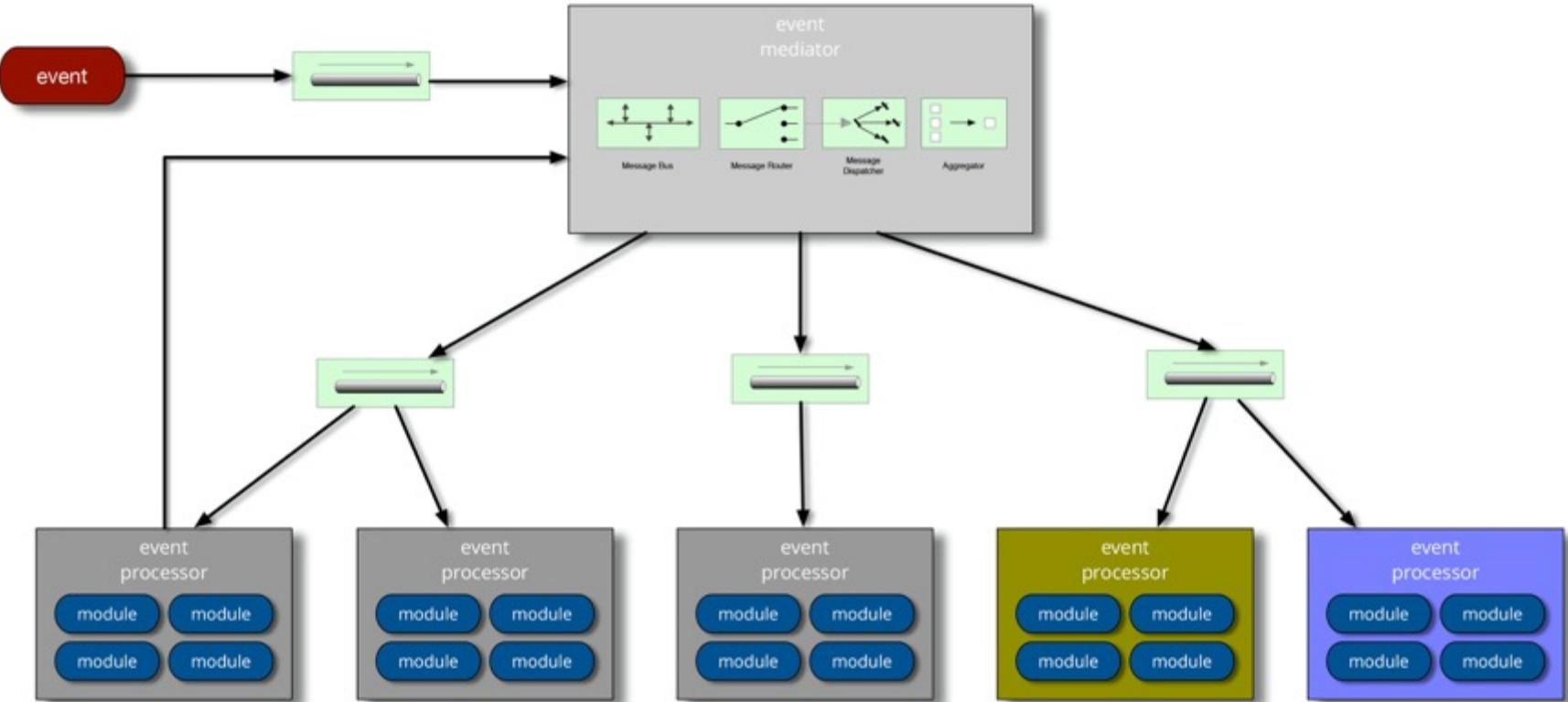
WEBFLOW
projects.spring.io/spring-webflow/



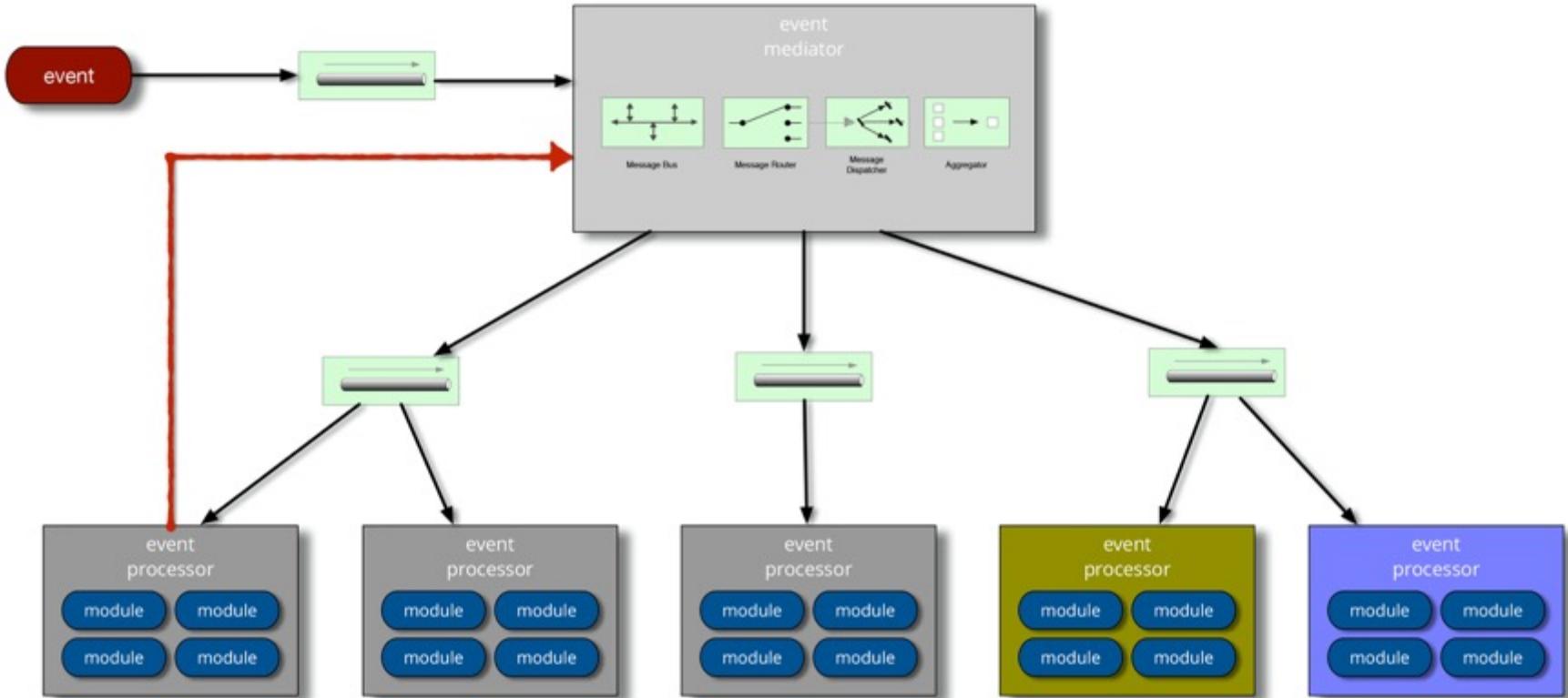
www.mulesoft.com/



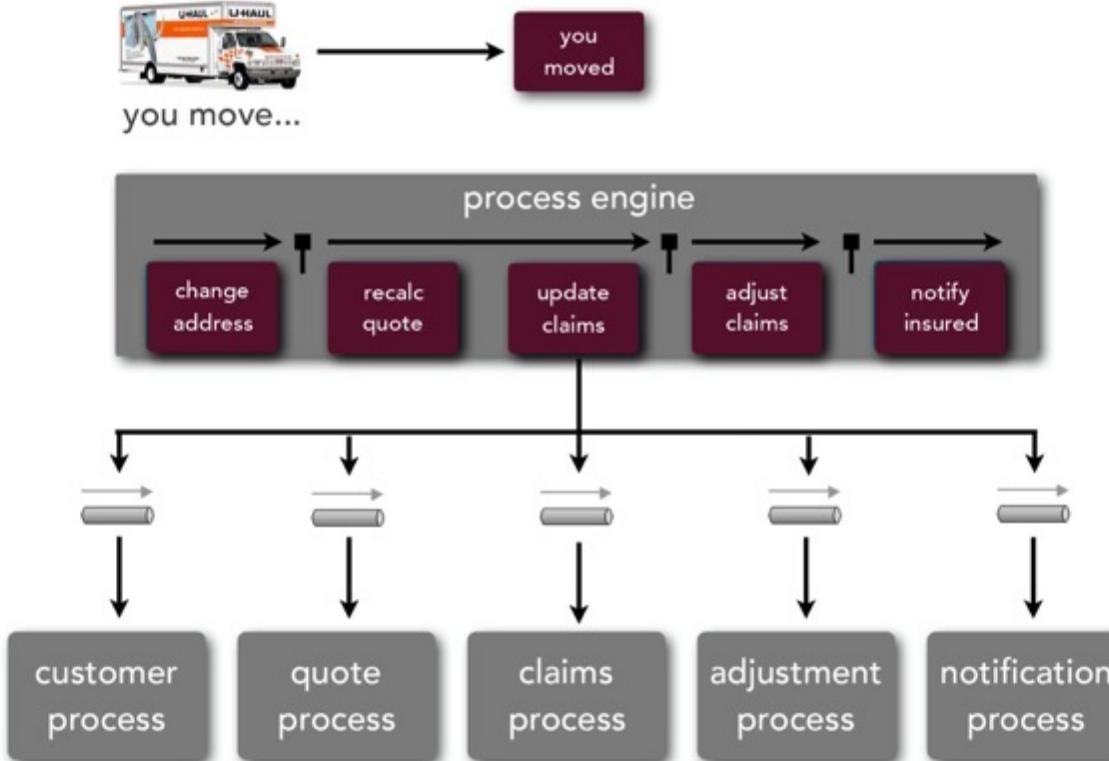
Mediator EDA



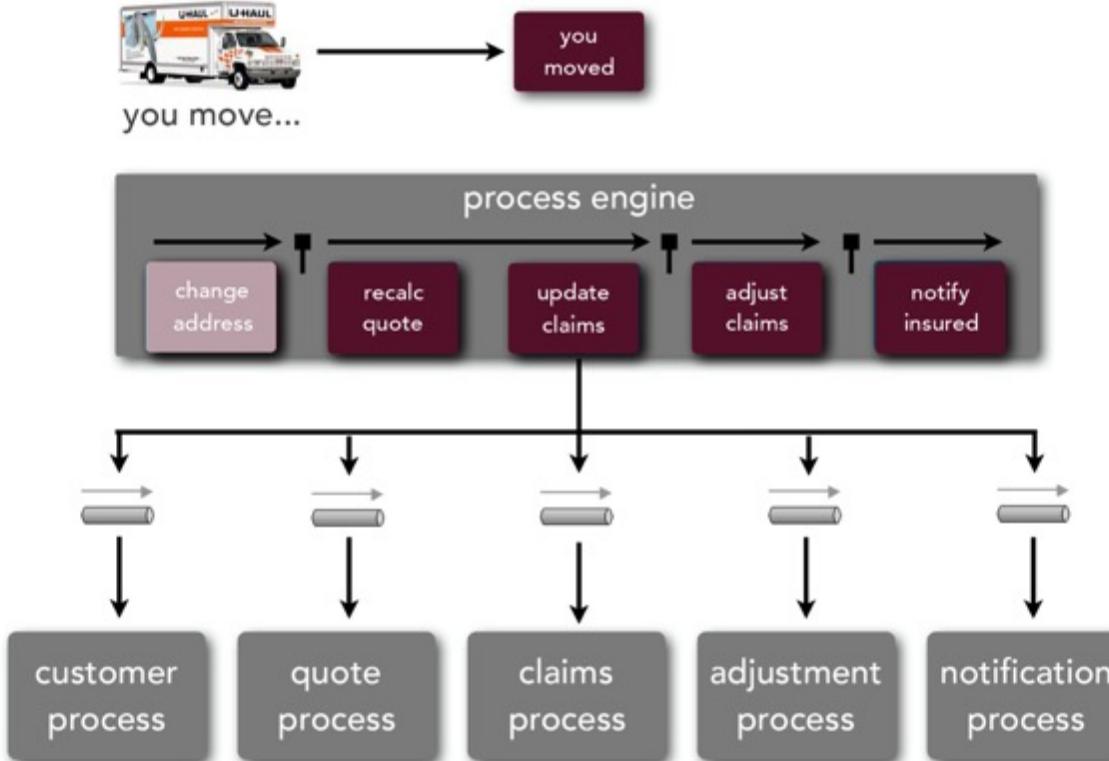
Mediator EDA



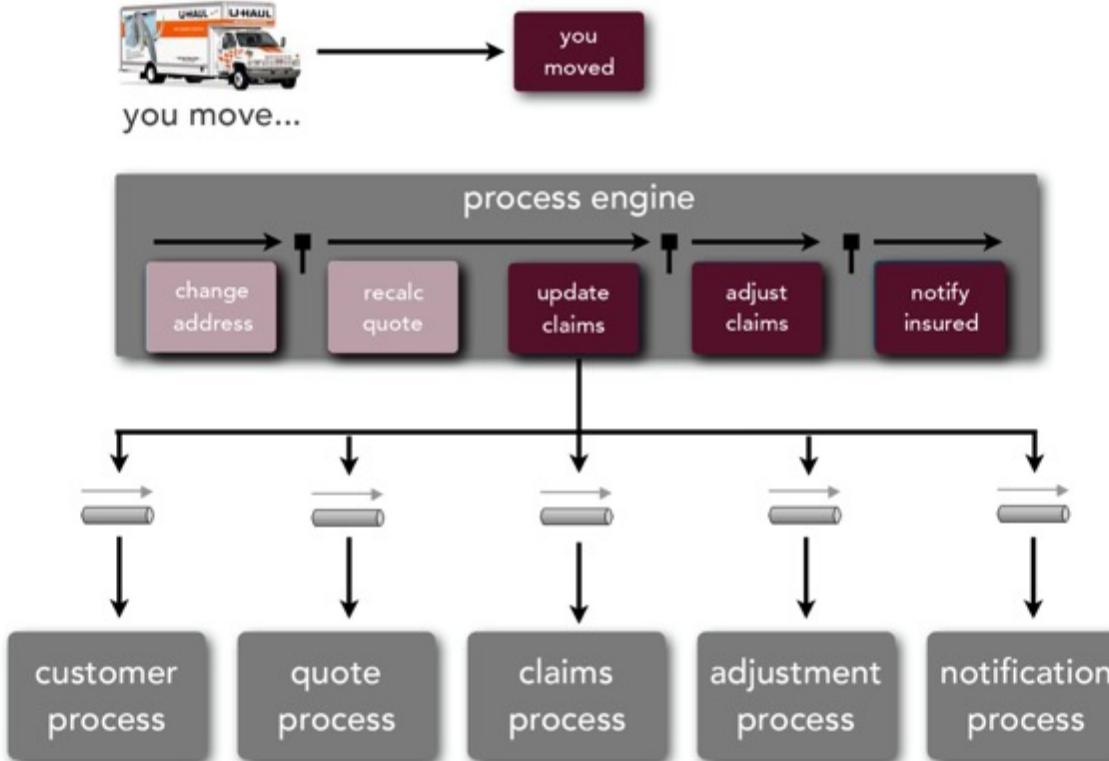
Mediator Message Flow



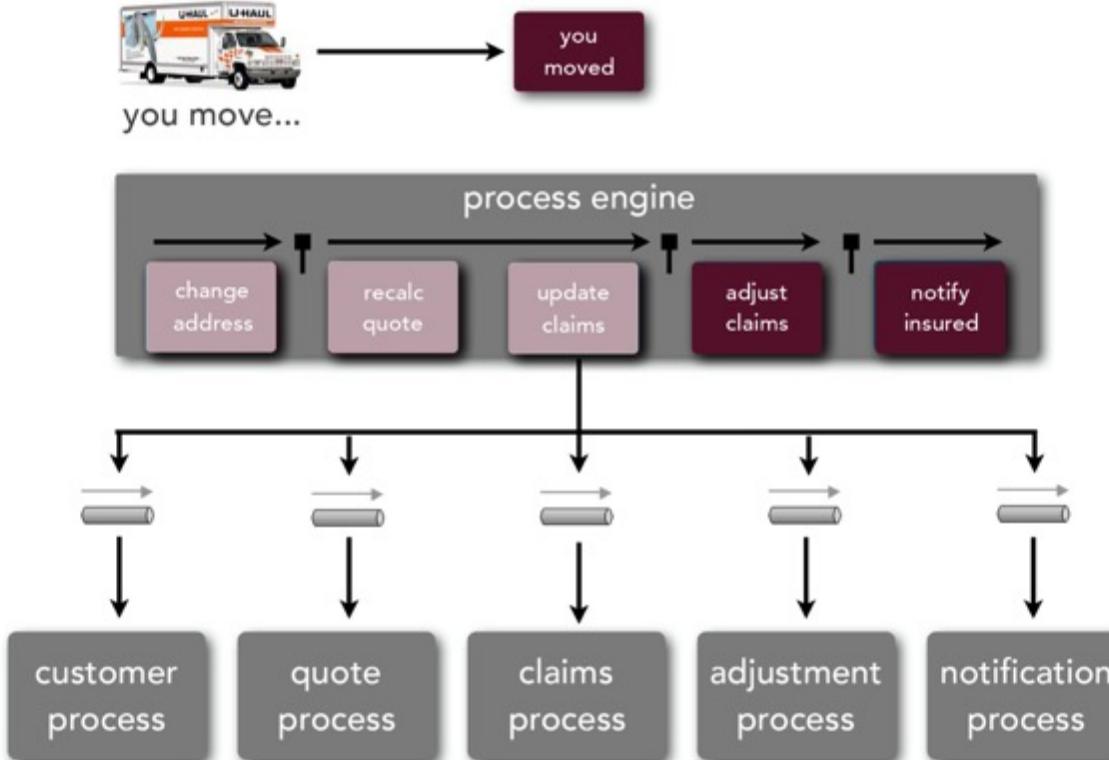
Mediator Message Flow



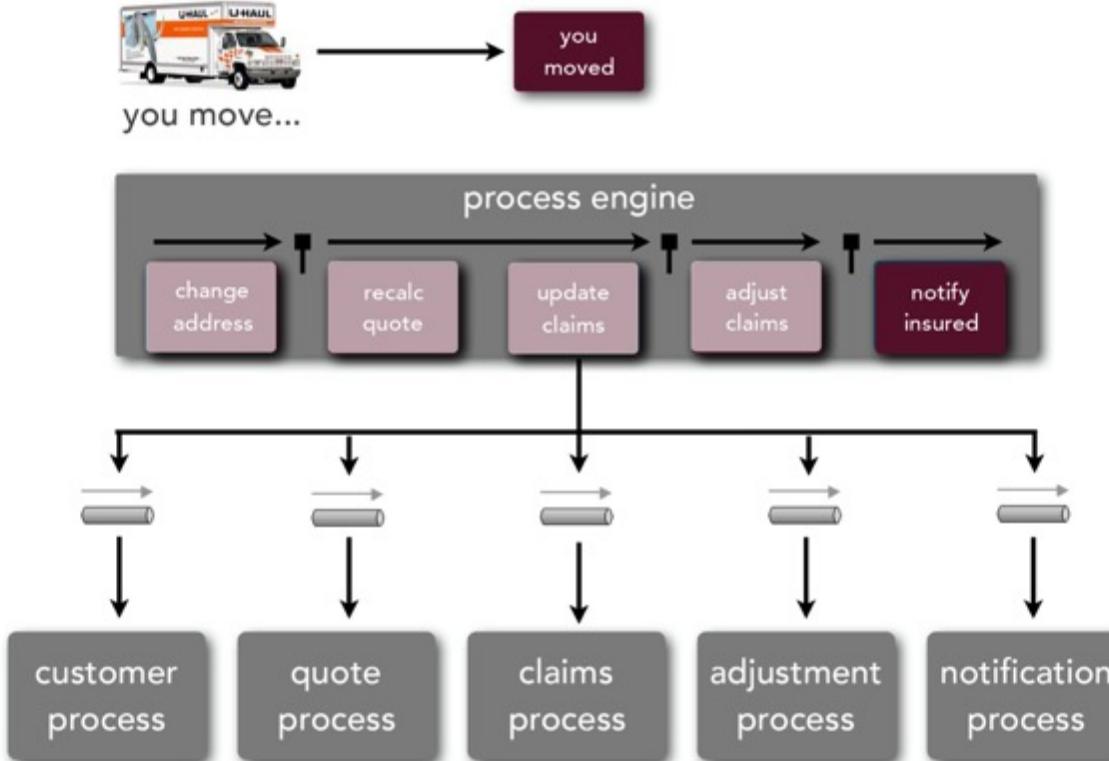
Mediator Message Flow



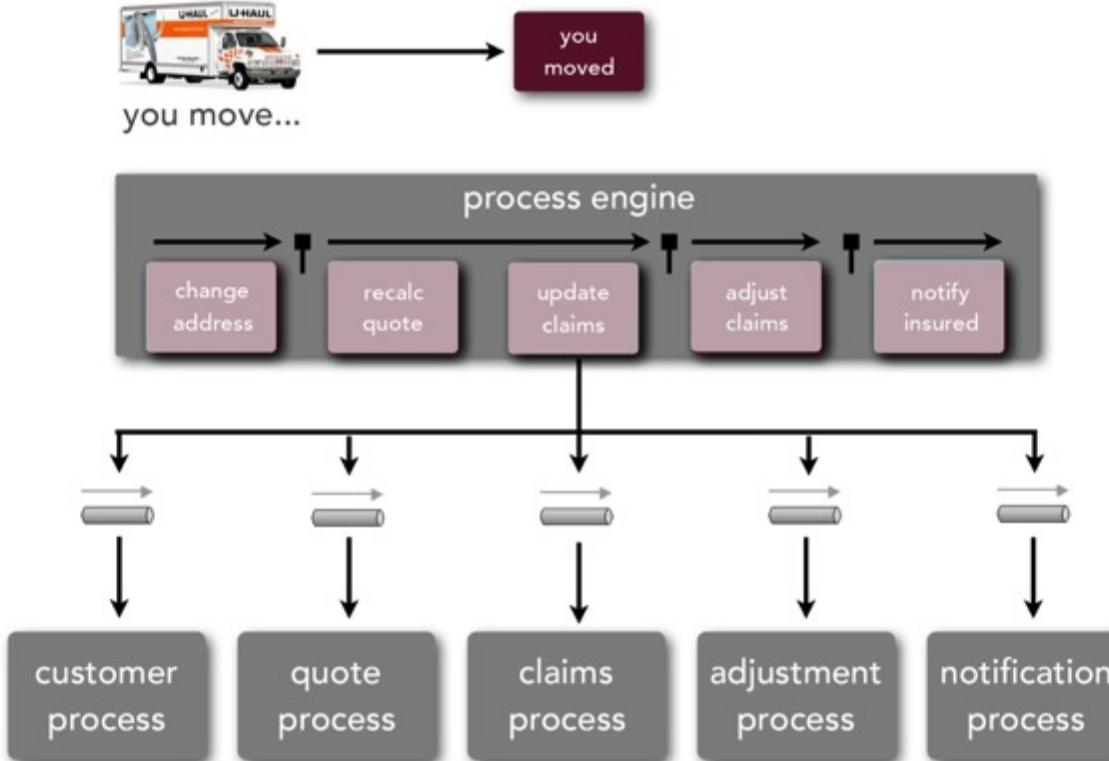
Mediator Message Flow



Mediator Message Flow

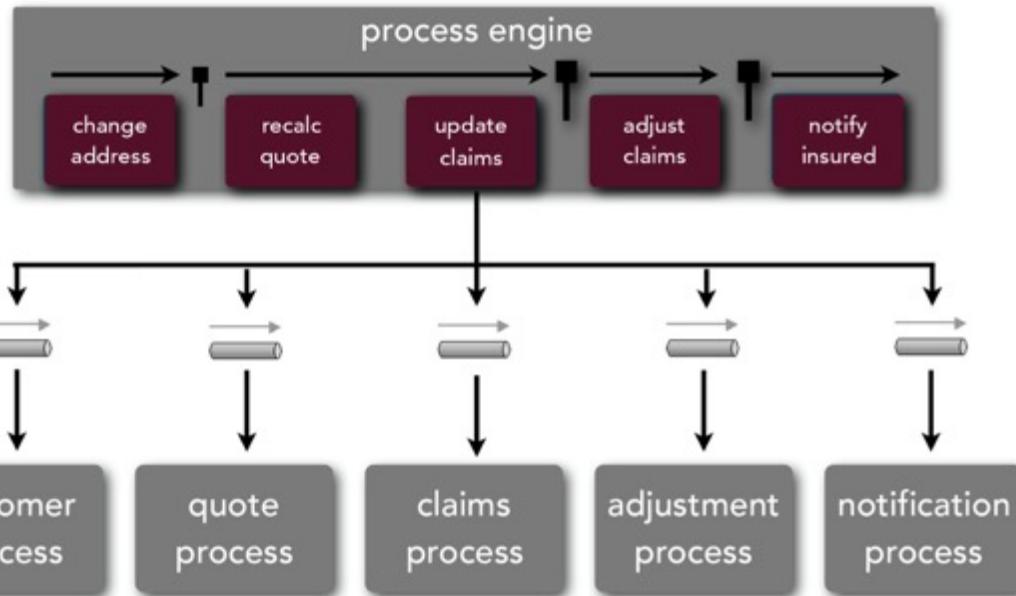


Mediator Message Flow





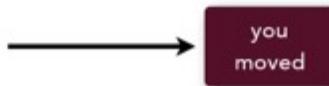
you move...



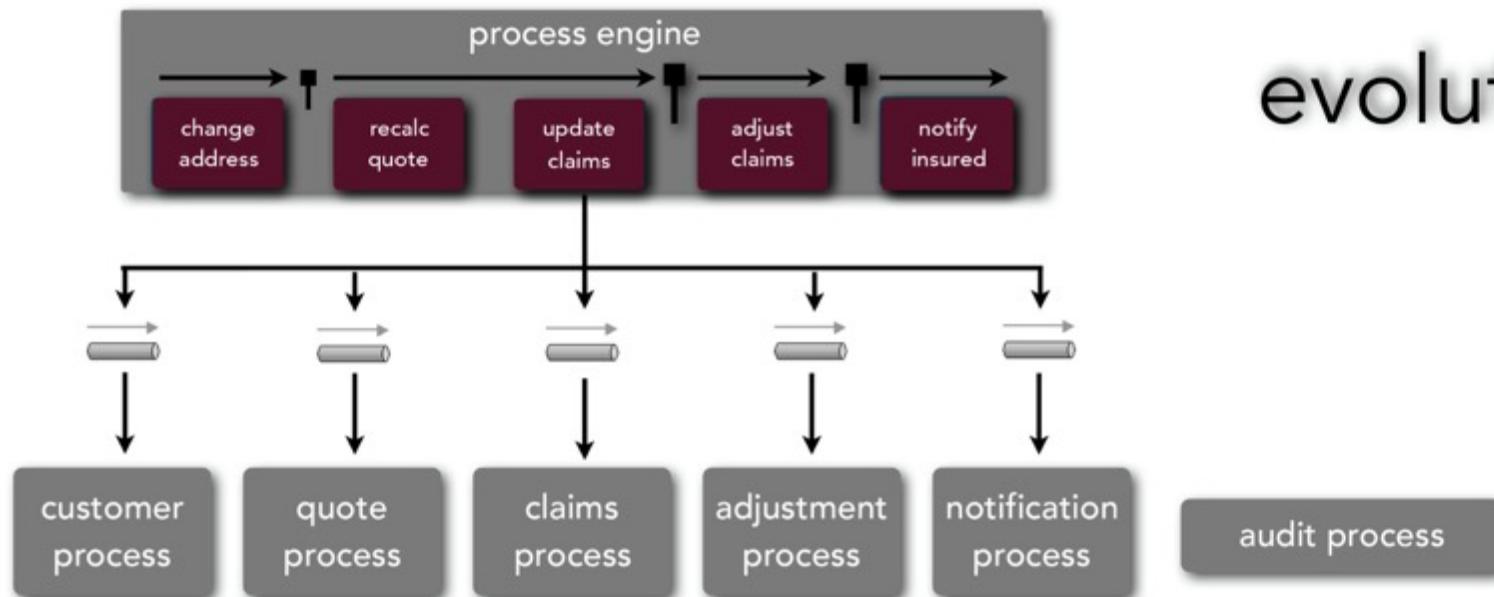
Error handling?



you move...



evolution

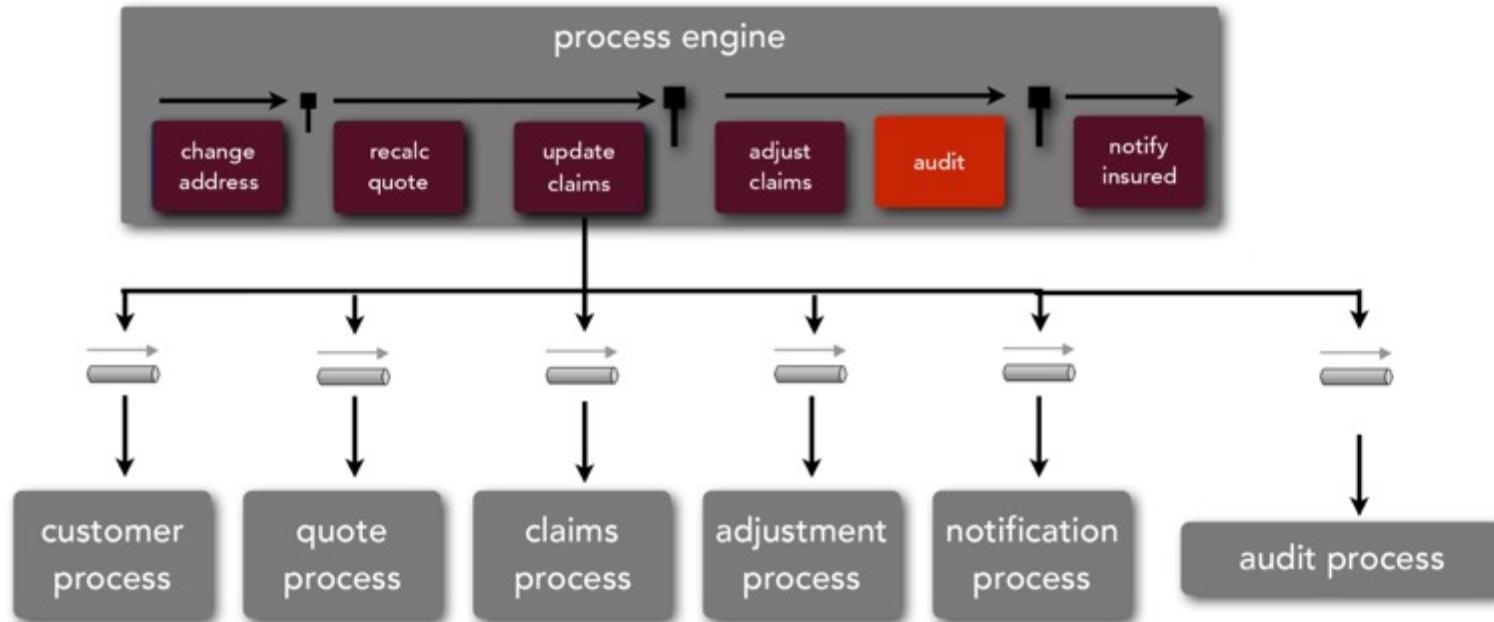




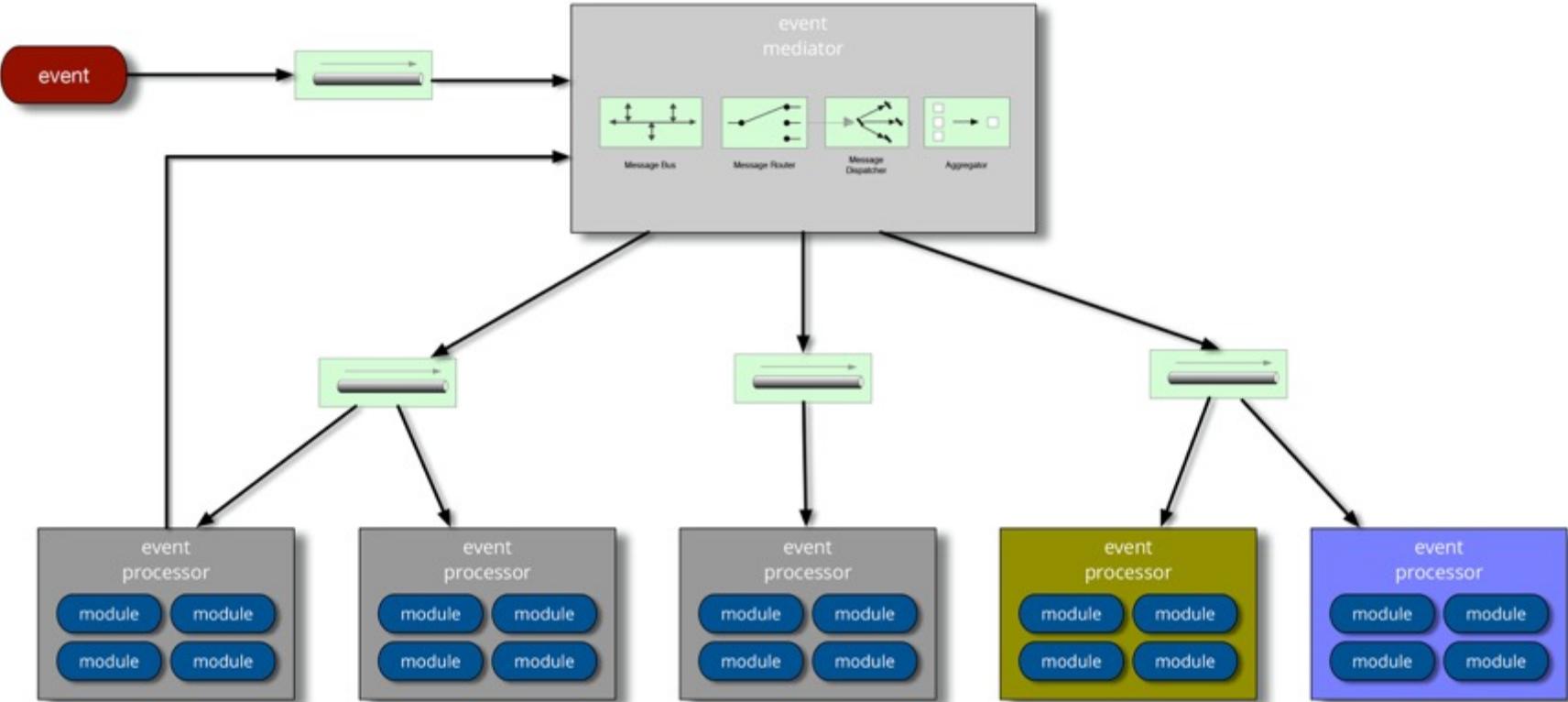
you moved

you move...

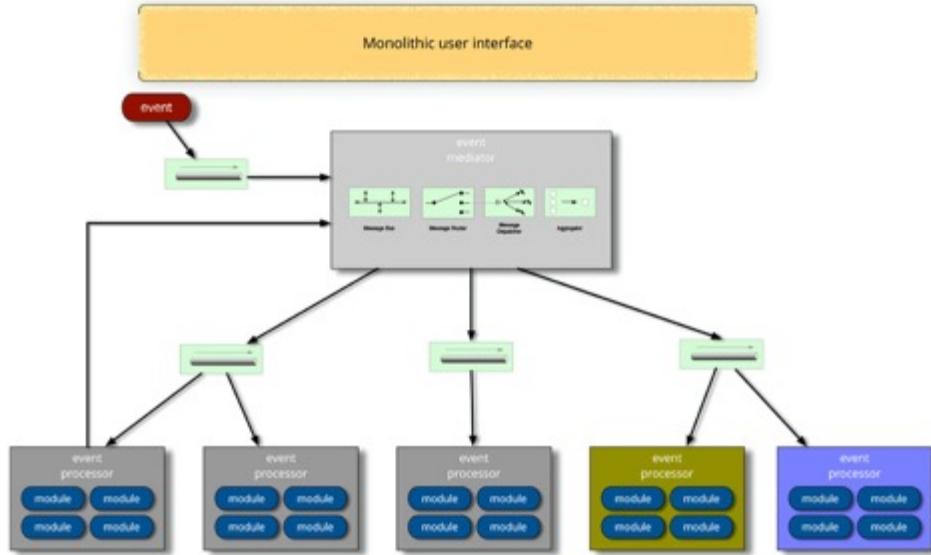
evolution



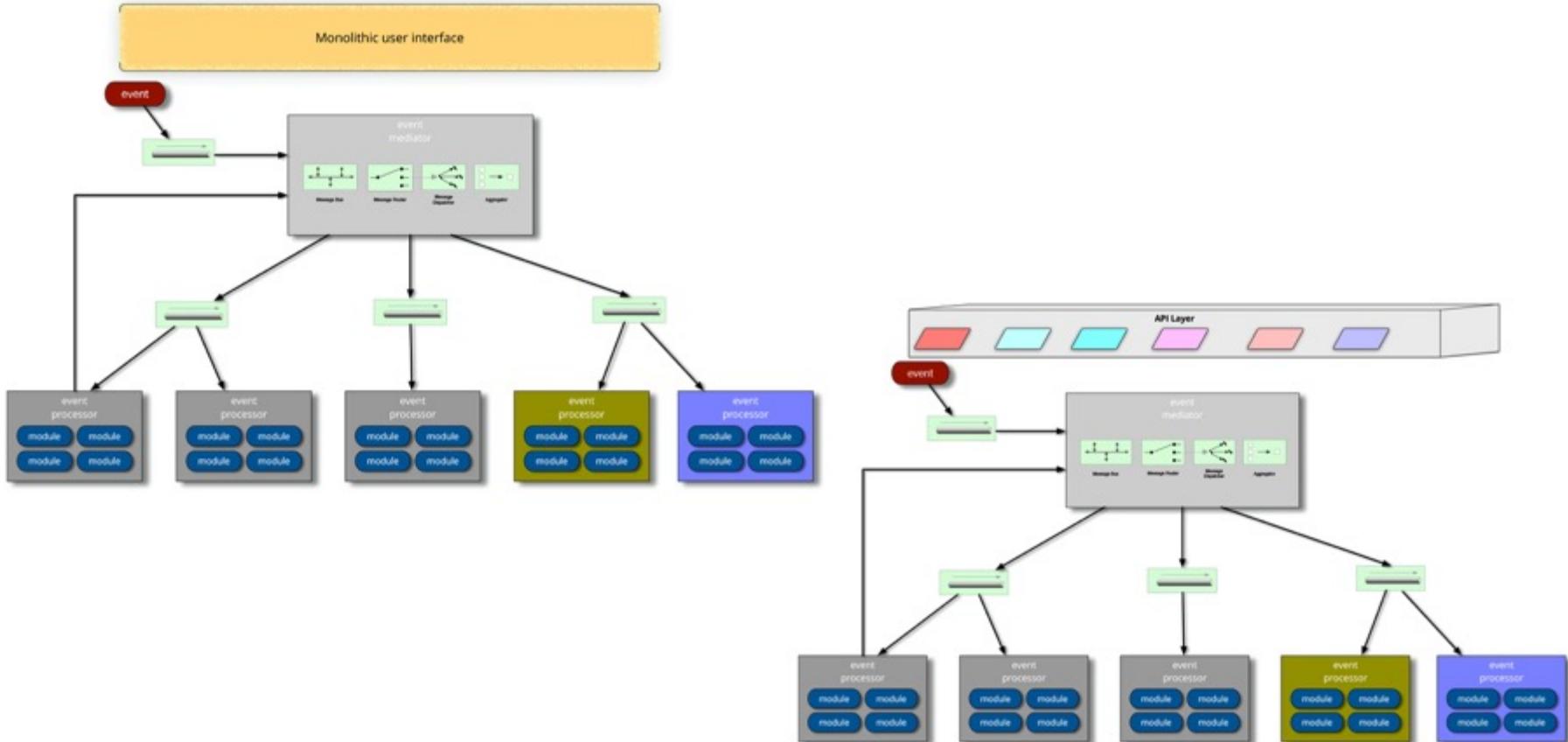
Mediator Quanta



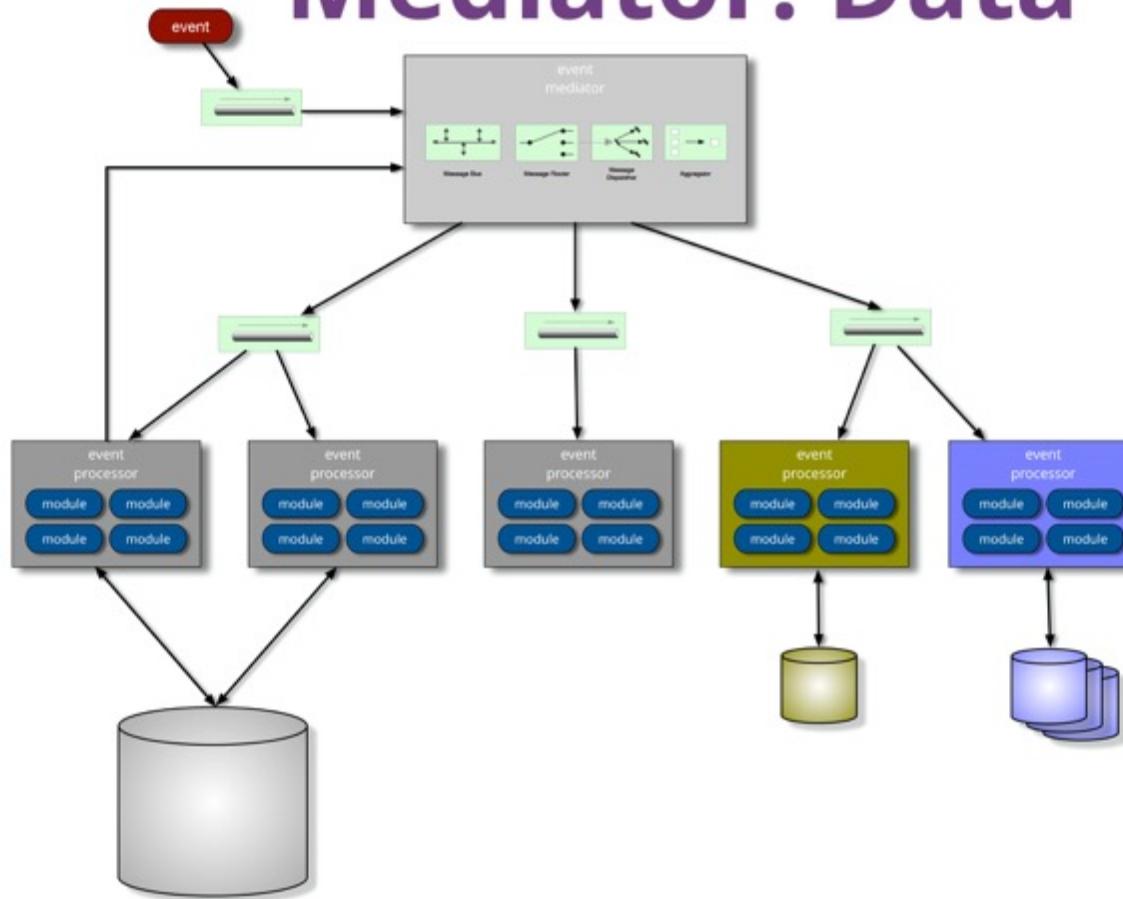
Mediator: UI



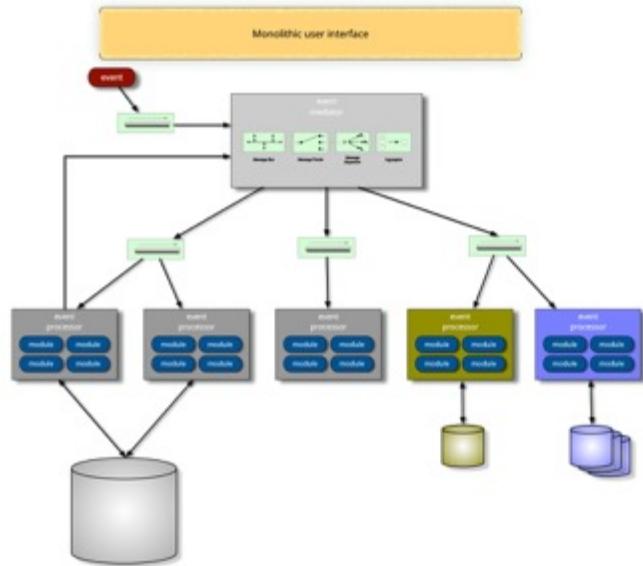
Mediator: UI



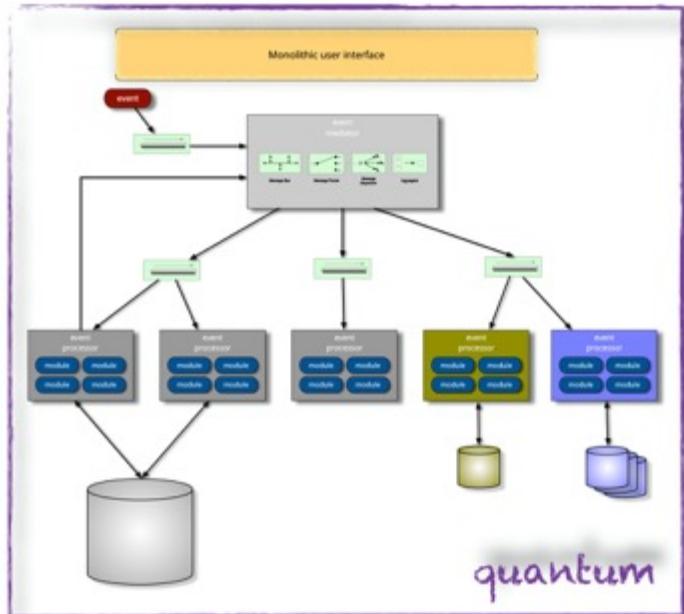
Mediator: Data



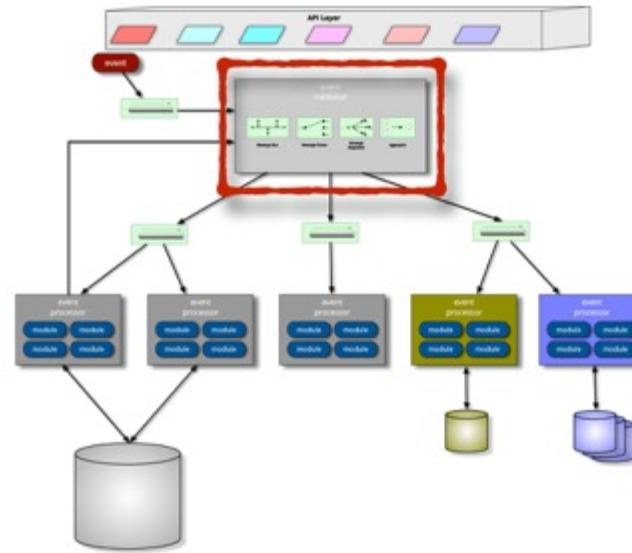
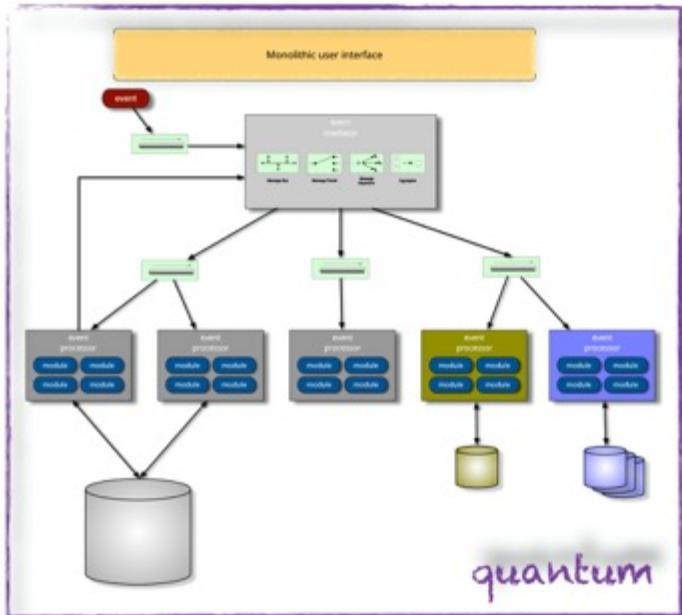
Mediator Quanta



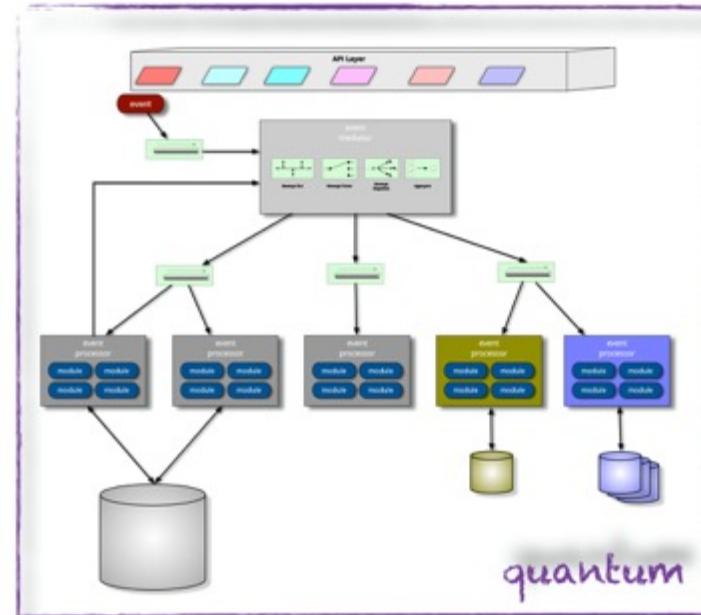
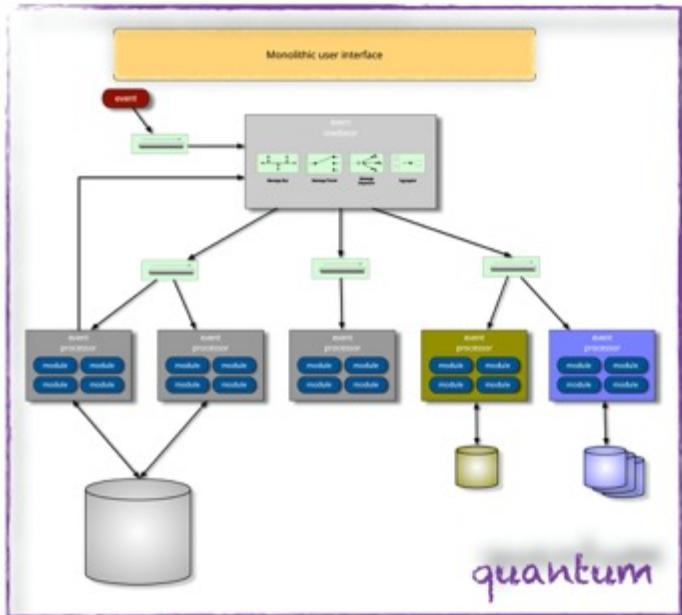
Mediator Quanta



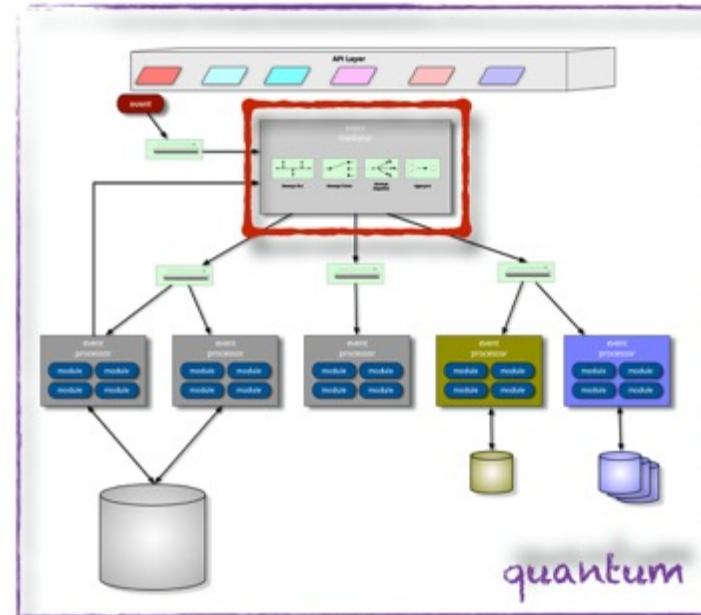
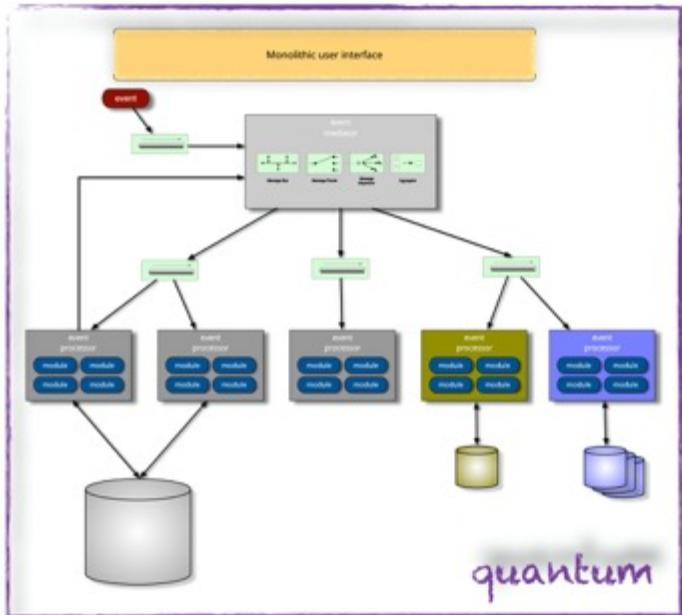
Mediator Quanta



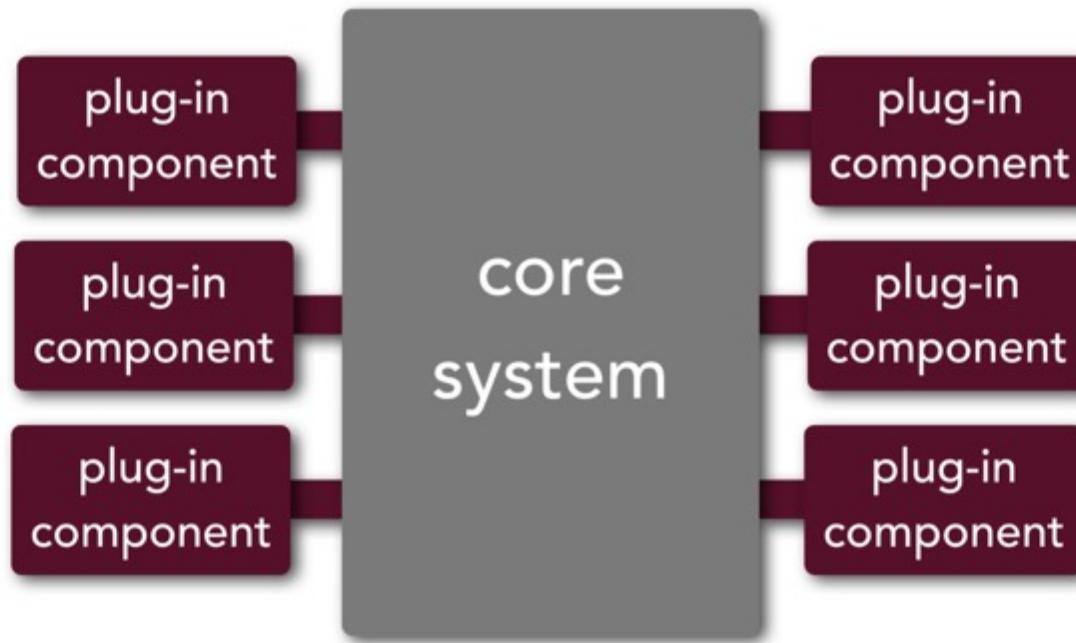
Mediator Quanta



Mediator Quanta



Microkernel

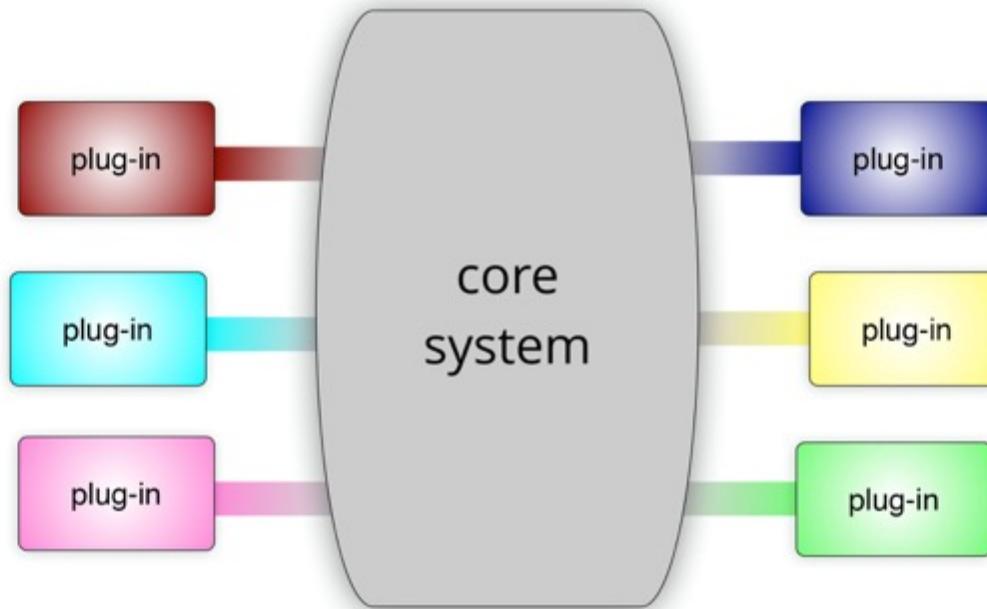


Microkernel

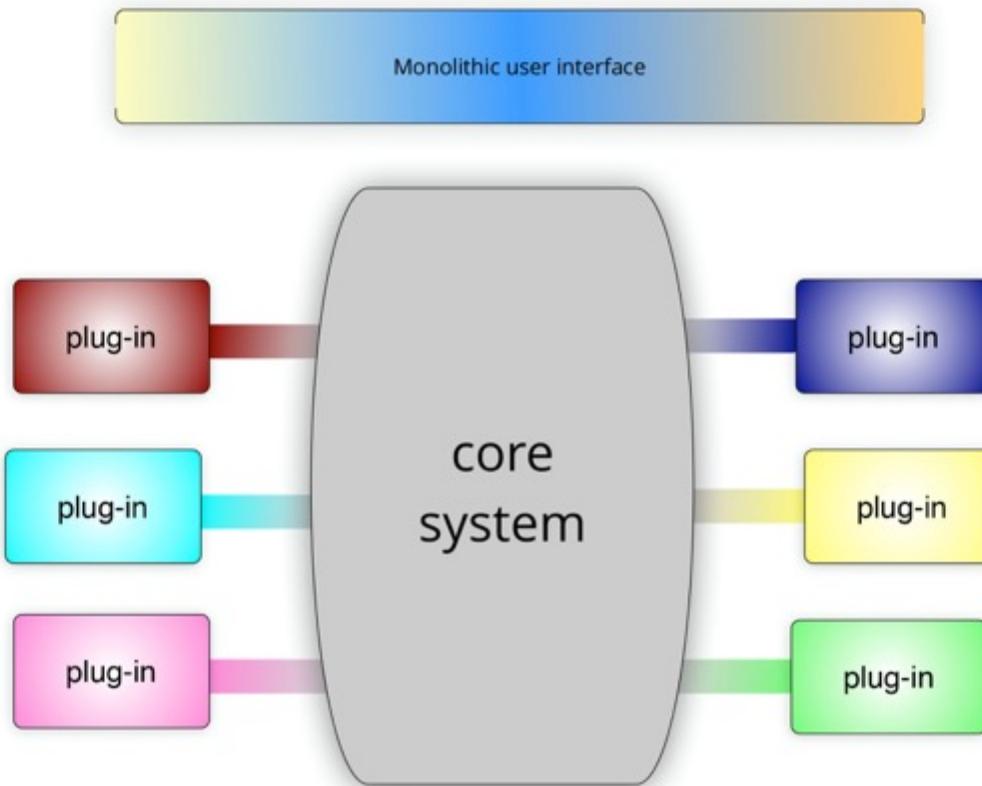
claims processing



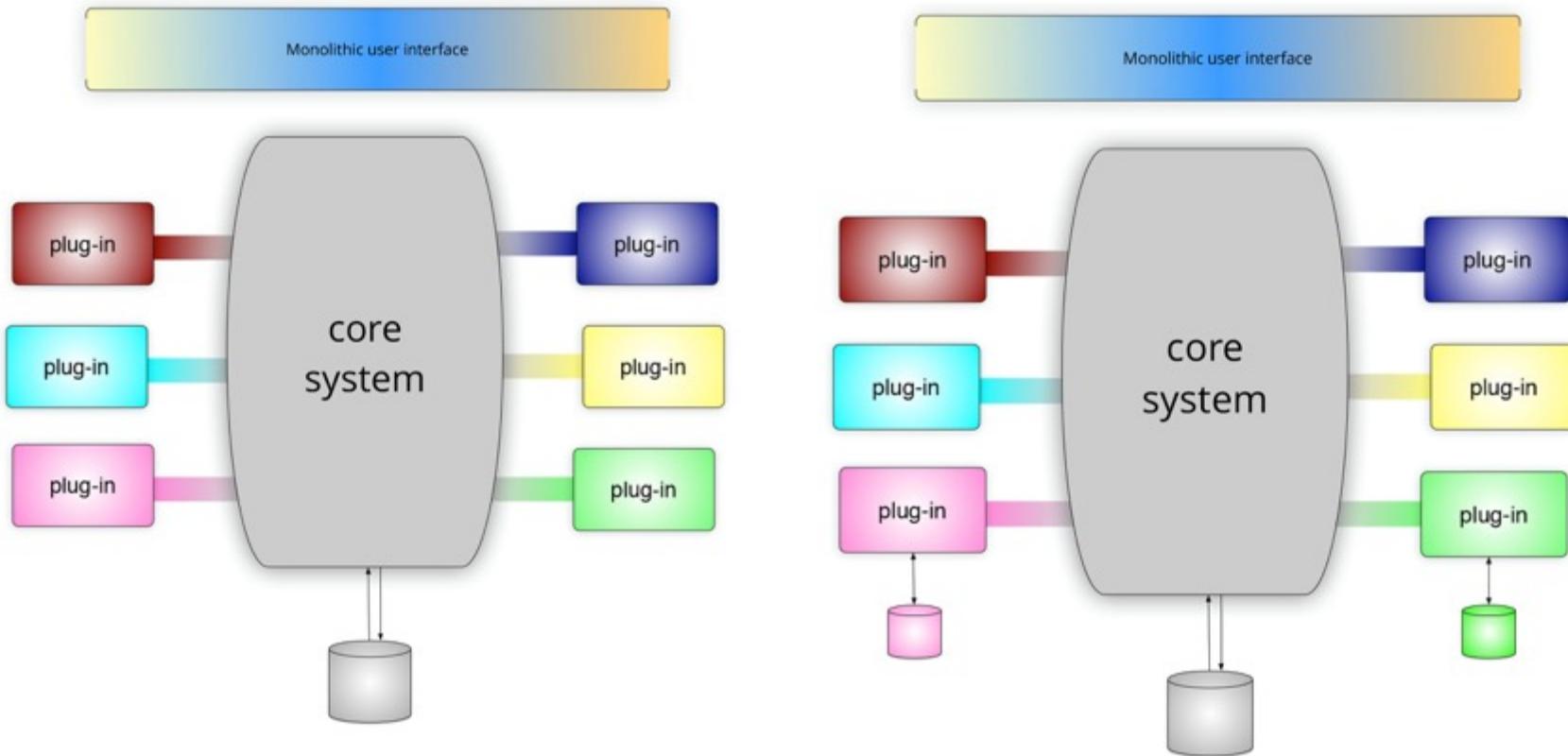
Microkernel Quanta



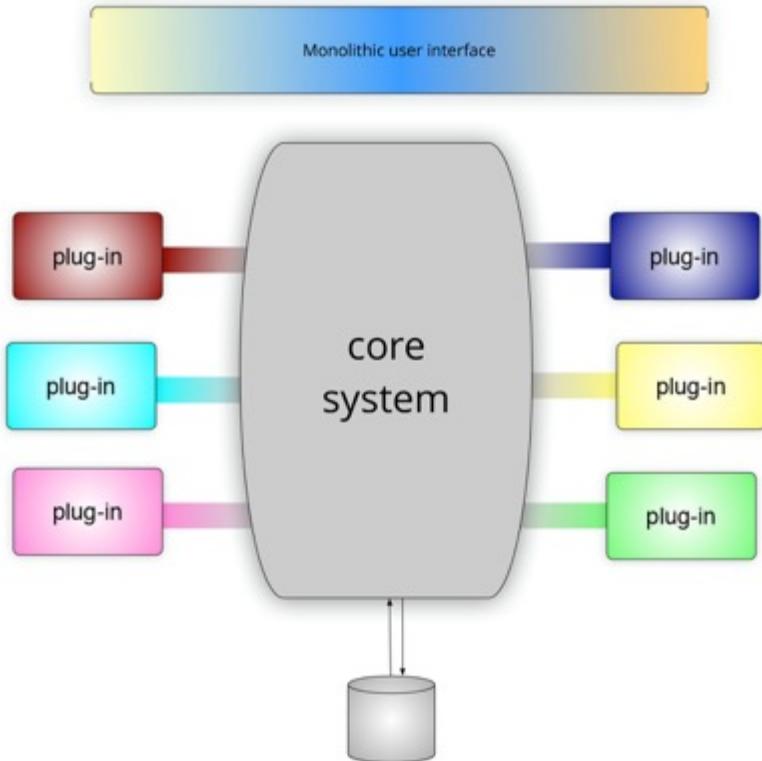
Microkernel: UI



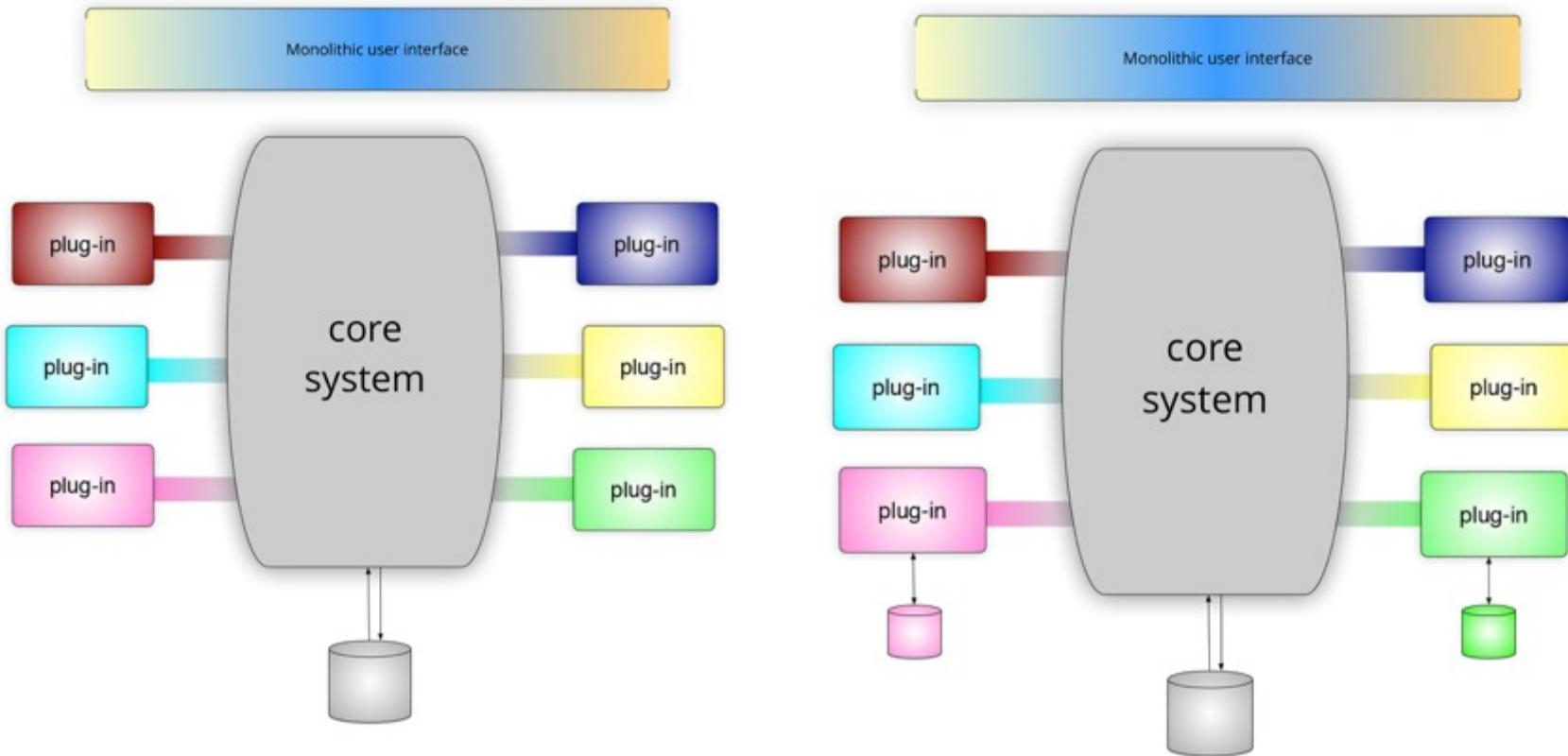
Microkernel: Data



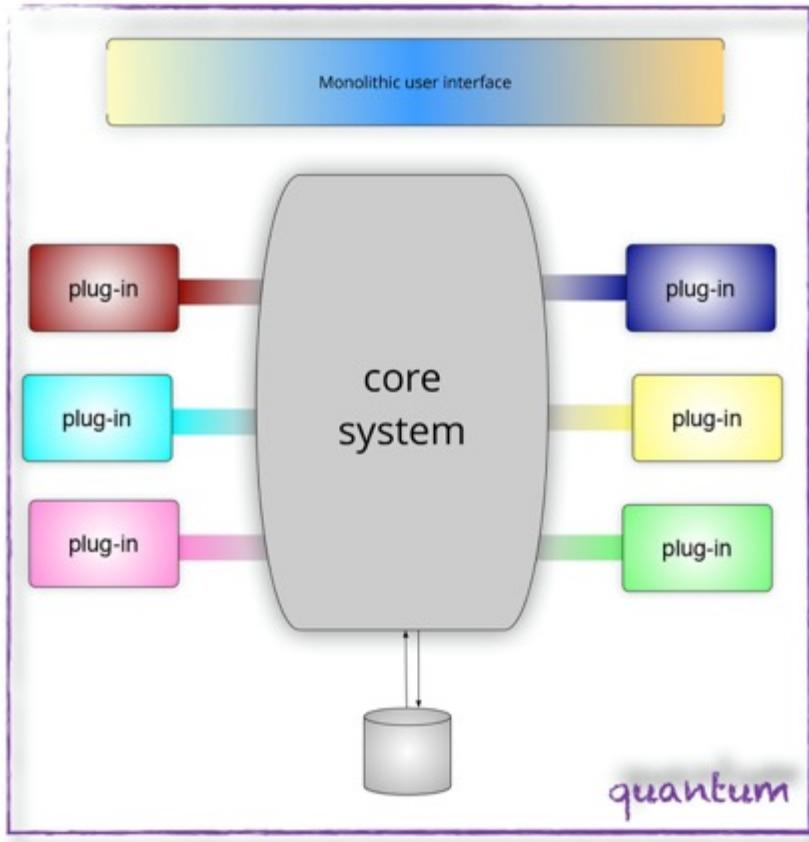
Microkernel: Data



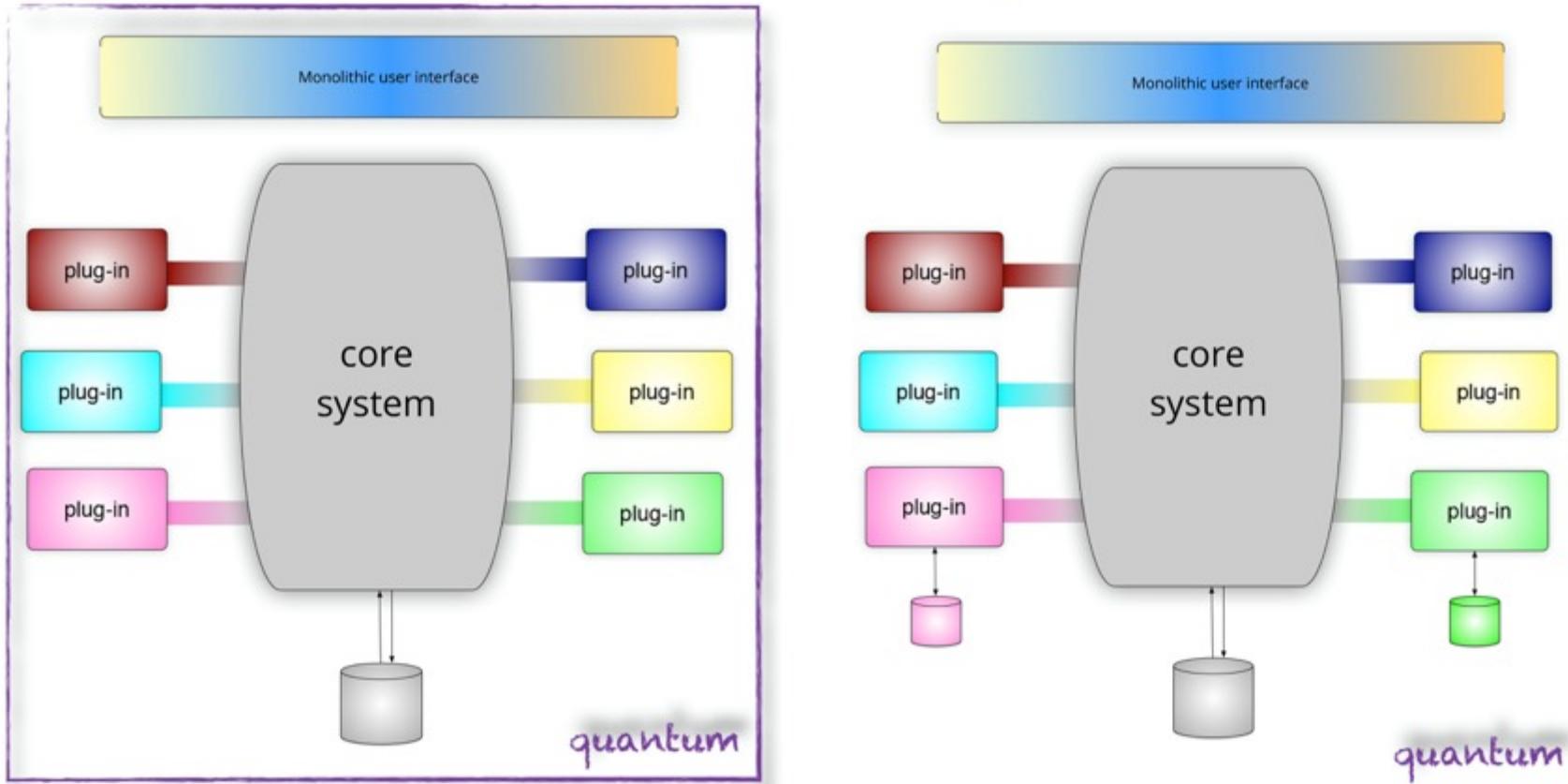
Microkernel: Data



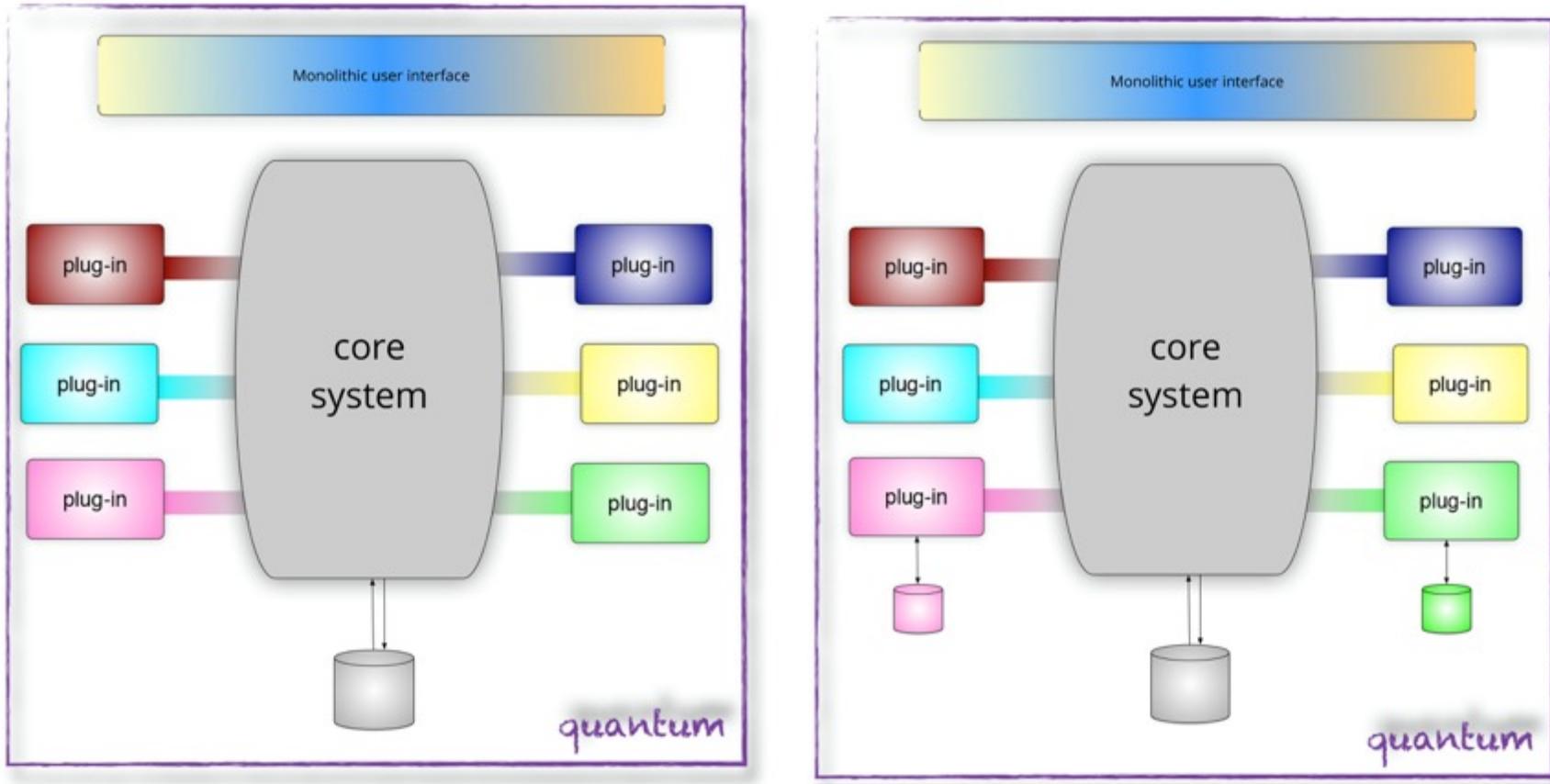
Microkernel: Quanta



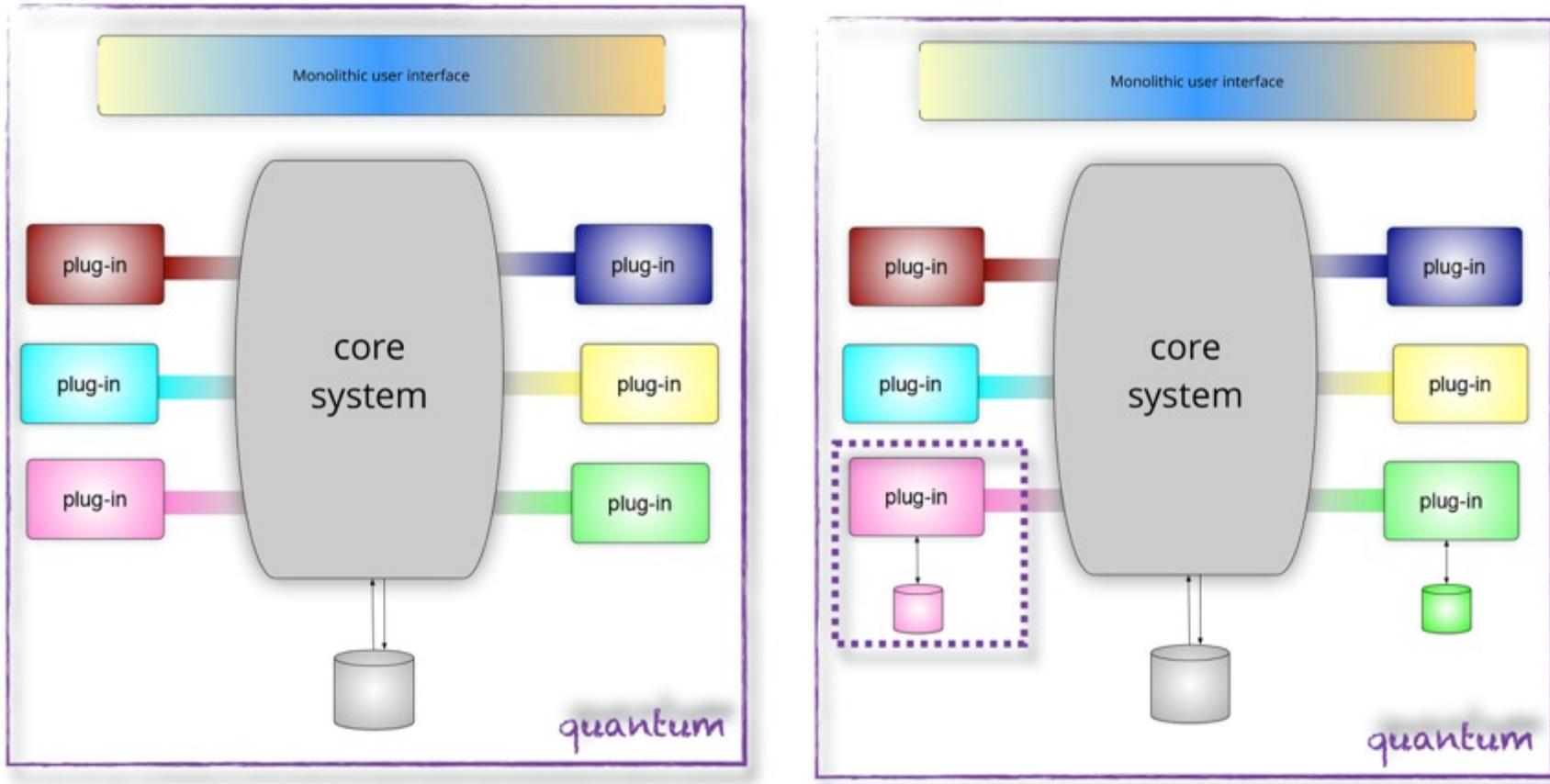
Microkernel: Quanta



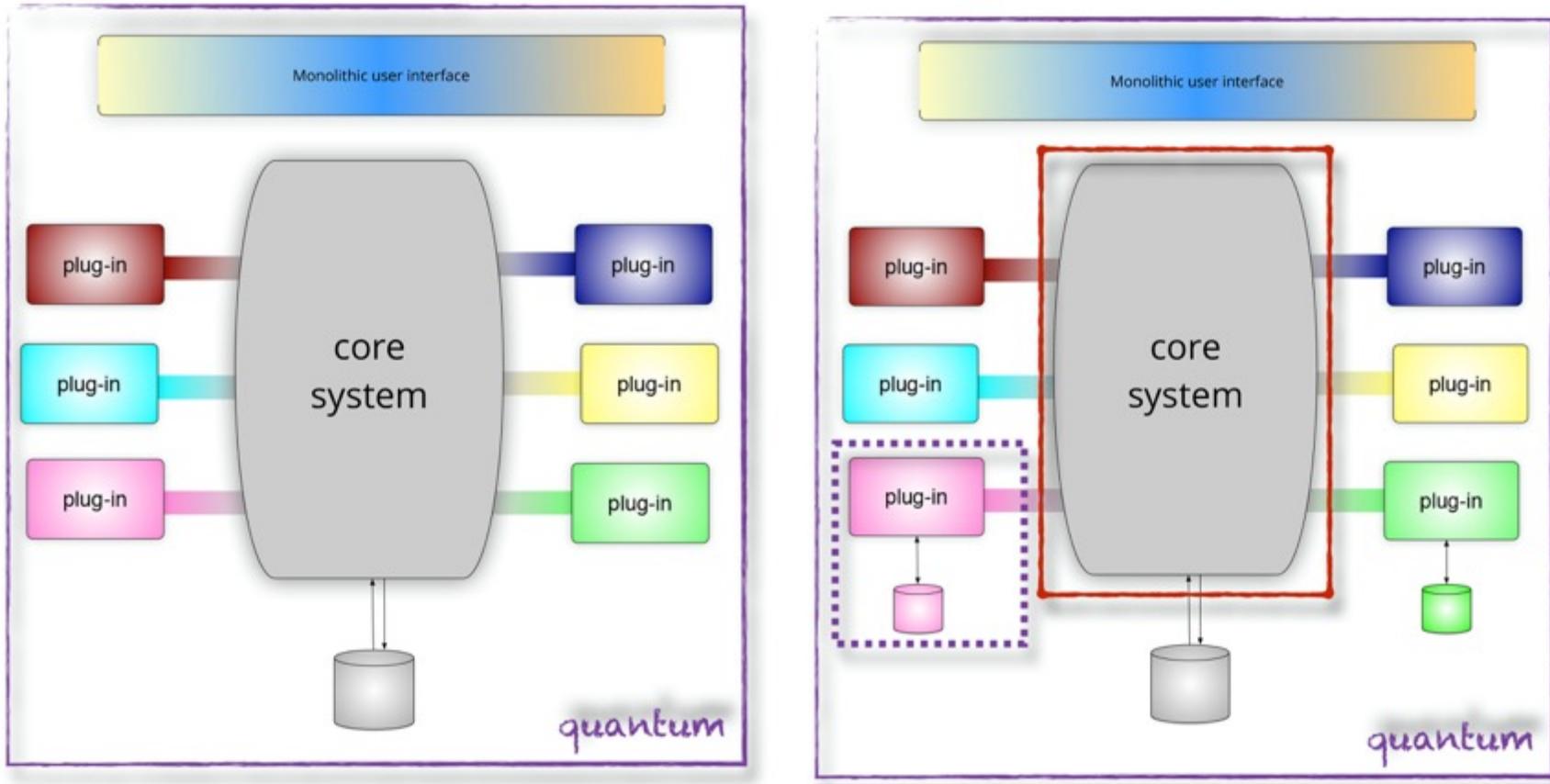
Microkernel: Quanta



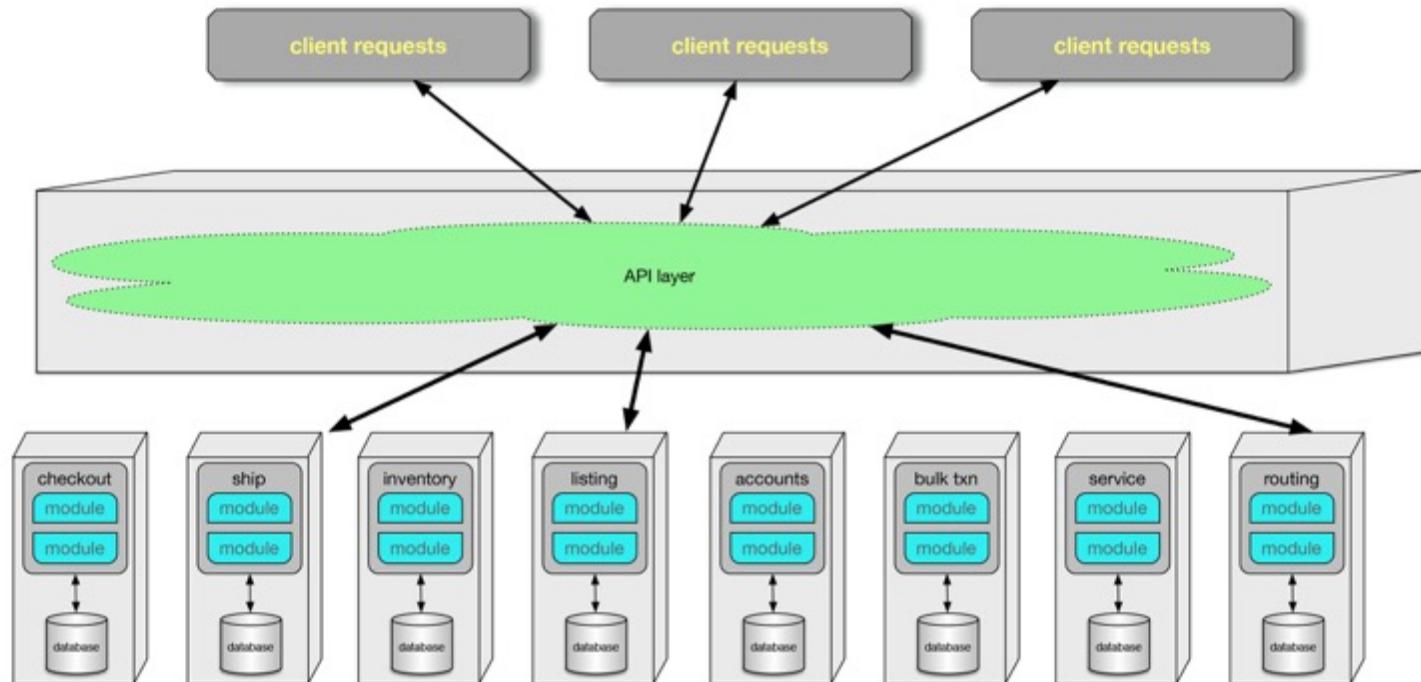
Microkernel: Quanta



Microkernel: Quanta



Microservices



What makes microservices so evolvable?

extremely loose coupling

What makes microservices so evolvable?

extremely loose coupling



What makes microservices so evolvable?

extremely loose coupling



eventual consistency

What makes microservices so evolvable?

extremely loose coupling

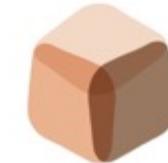


What makes microservices so evolvable?

extremely loose coupling

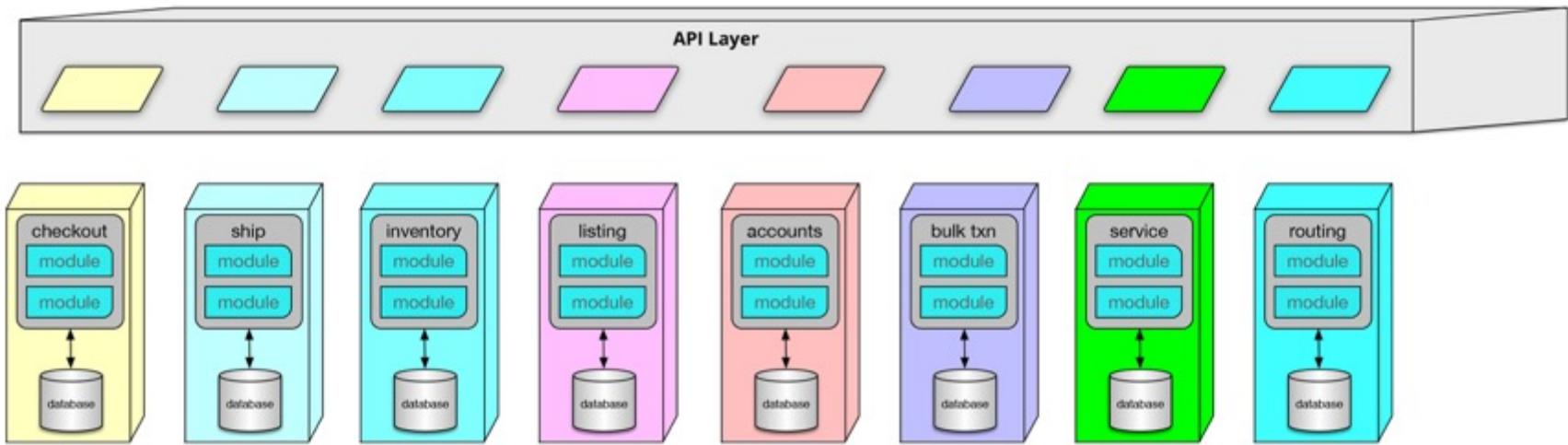


eventual consistency

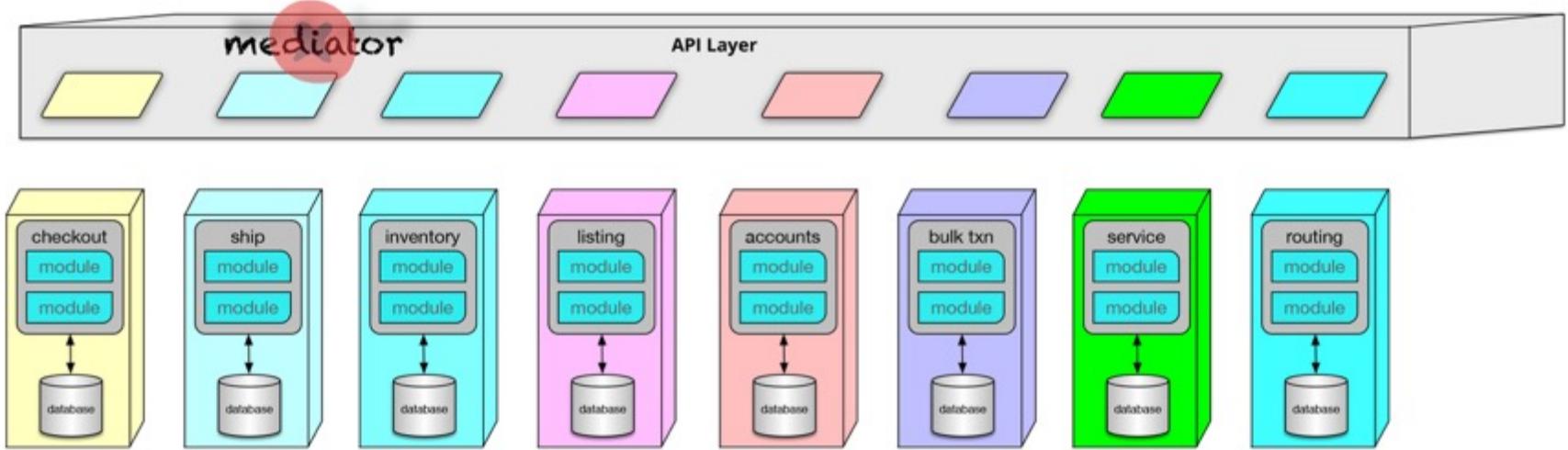


small quantum

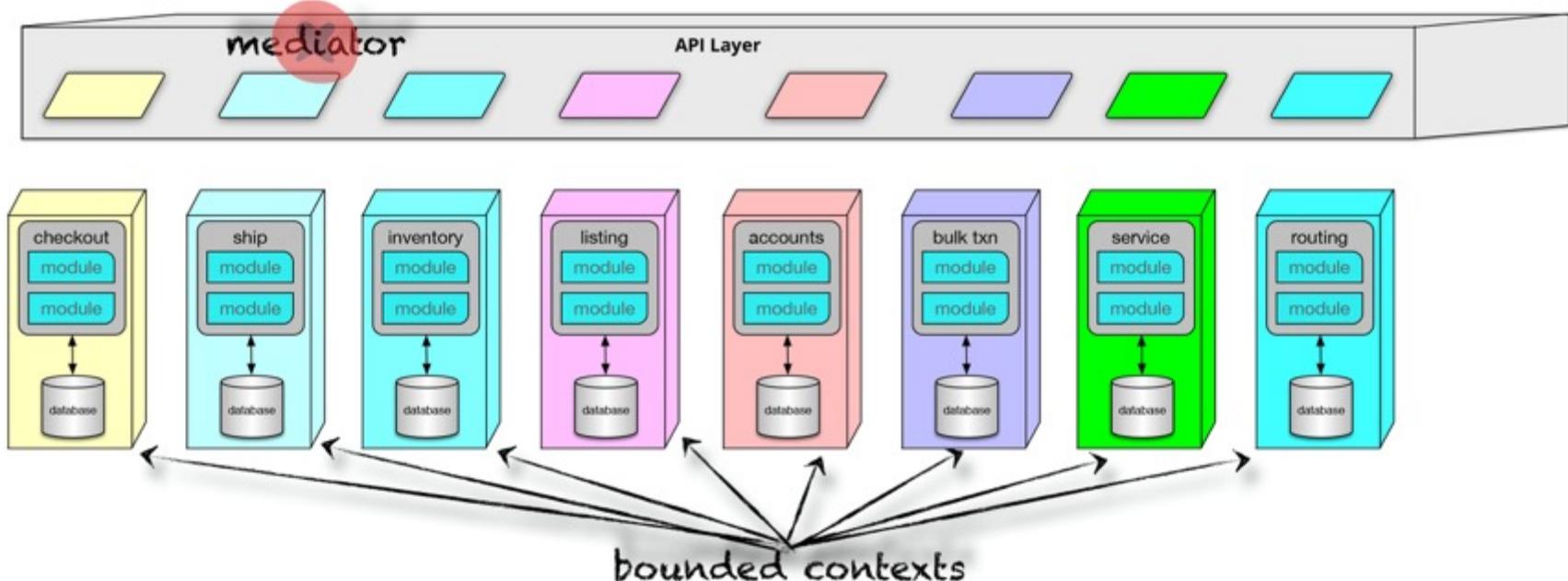
Microservices



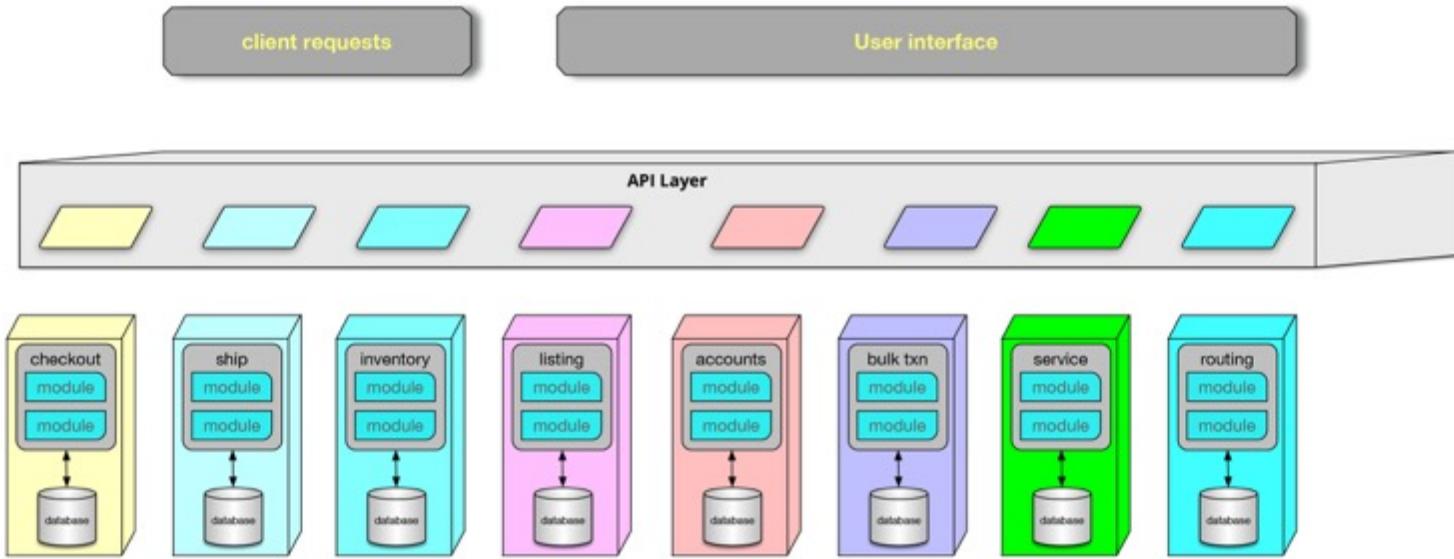
Microservices



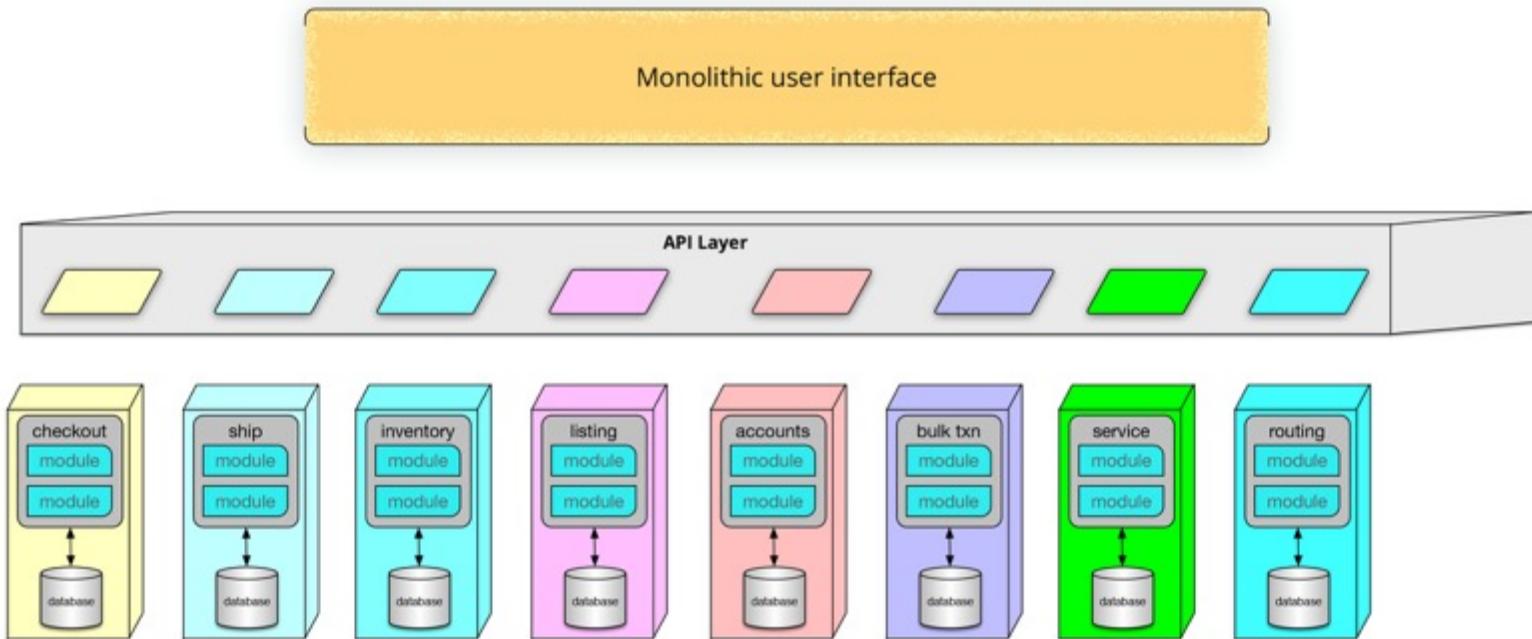
Microservices



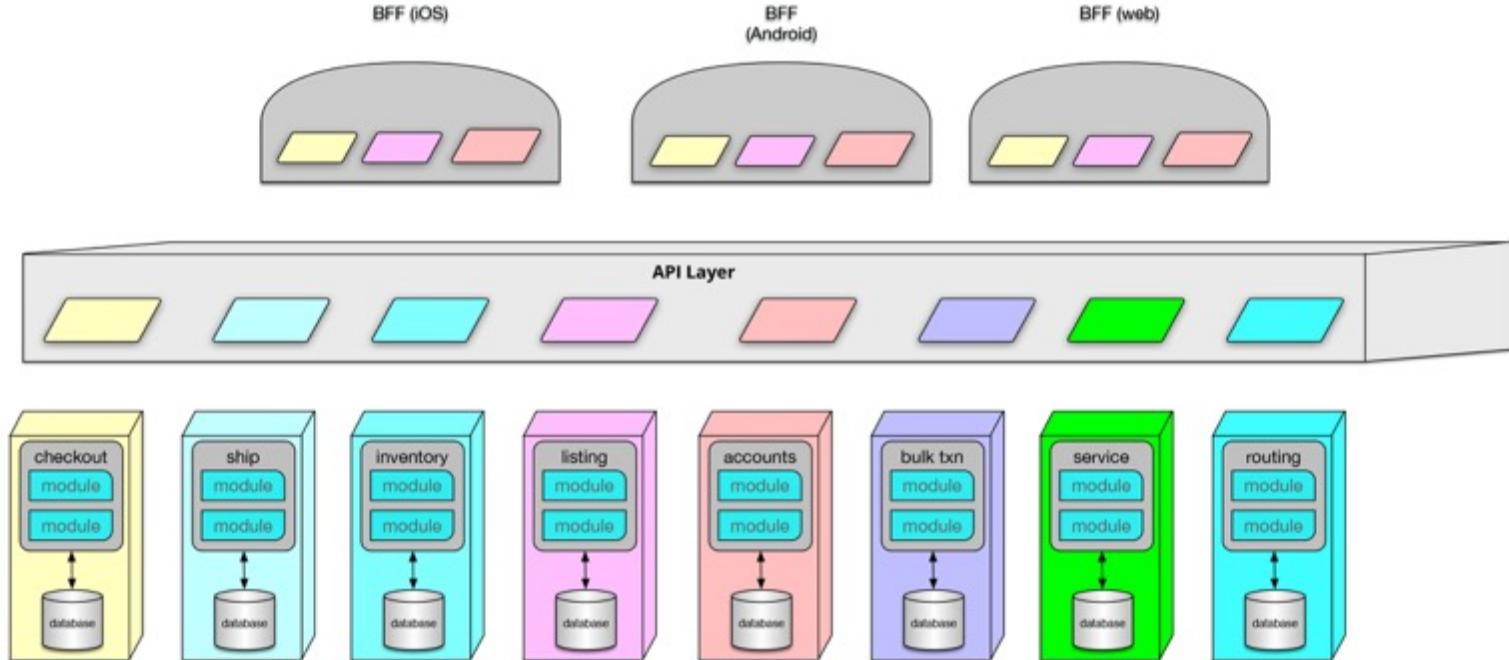
Microservices



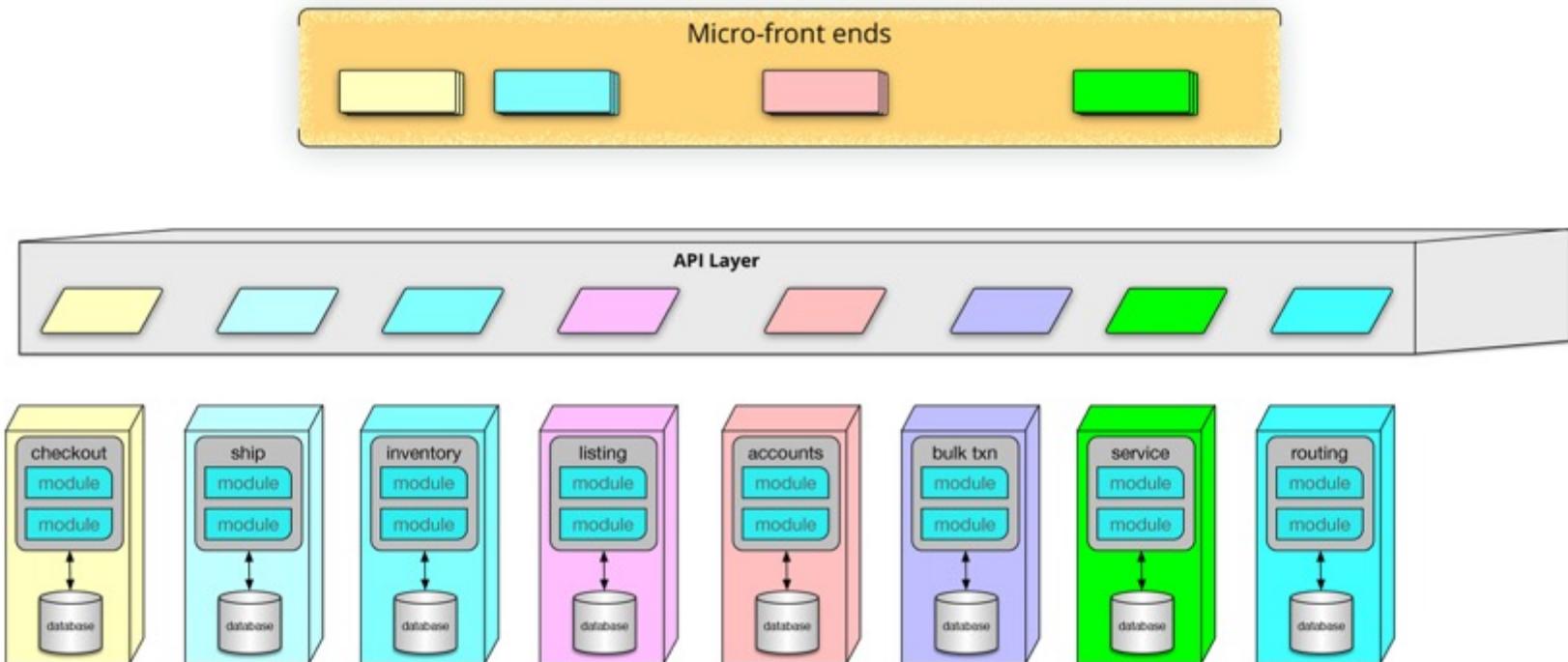
Microservices: Monolithic UI



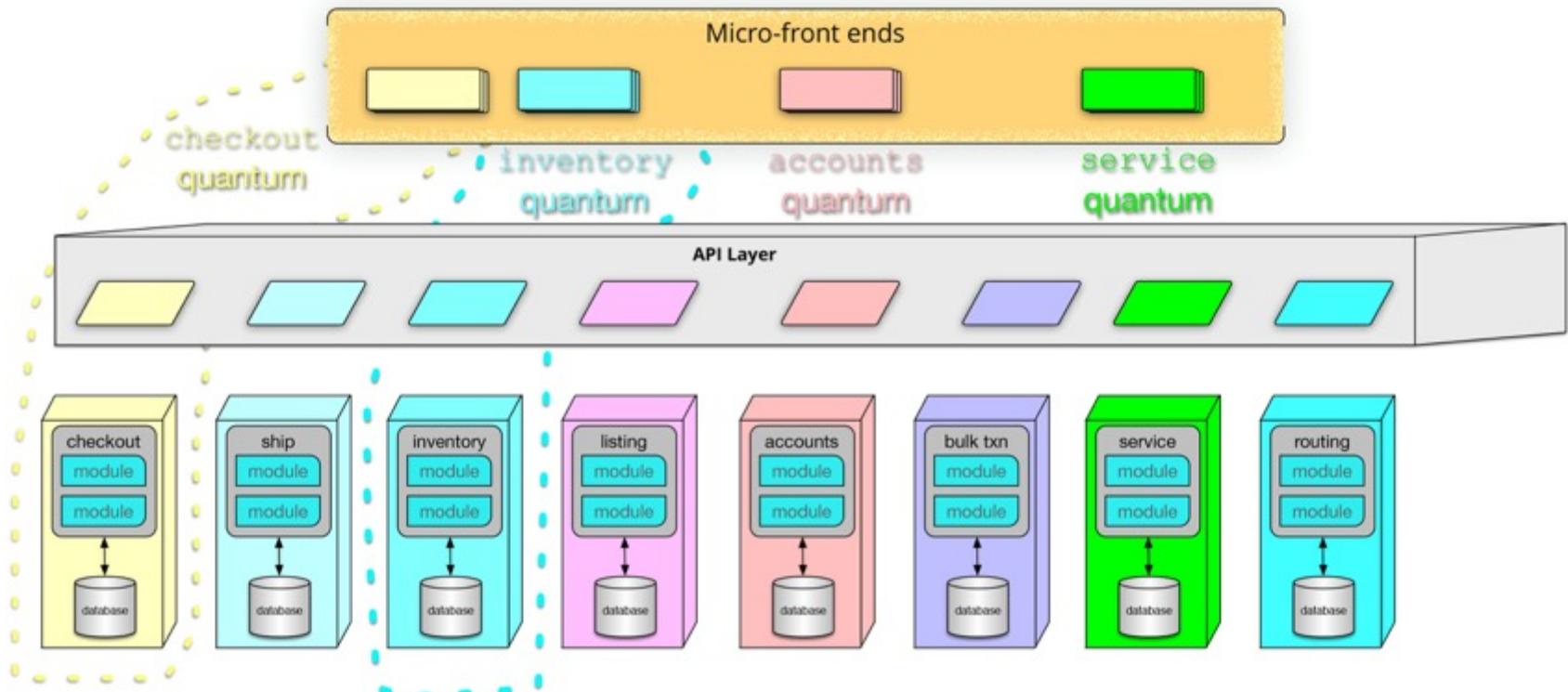
Microservices :BFF



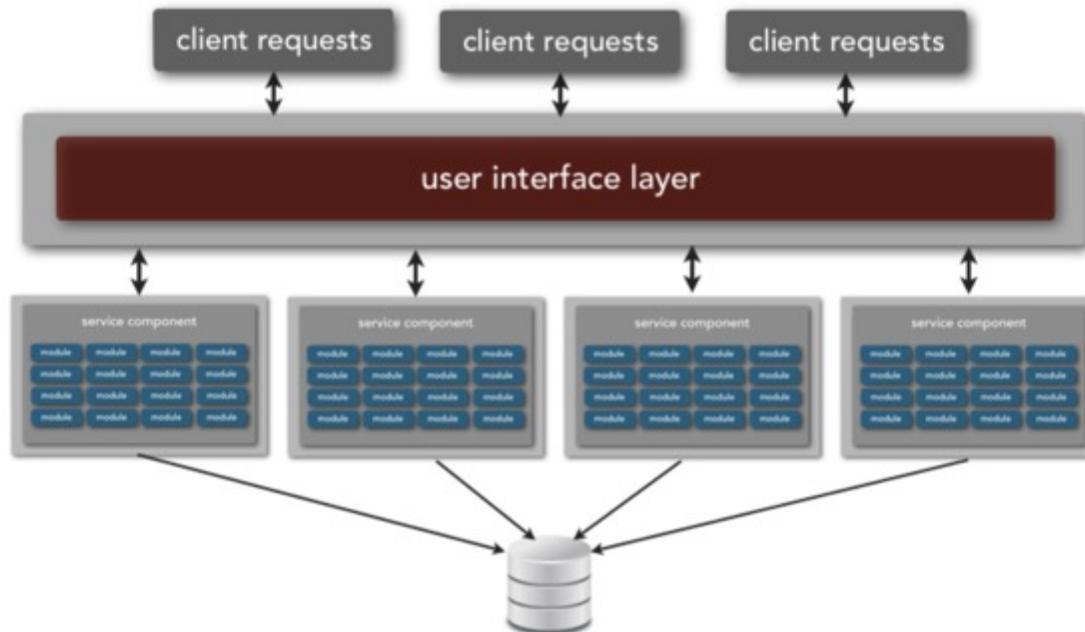
Microservices



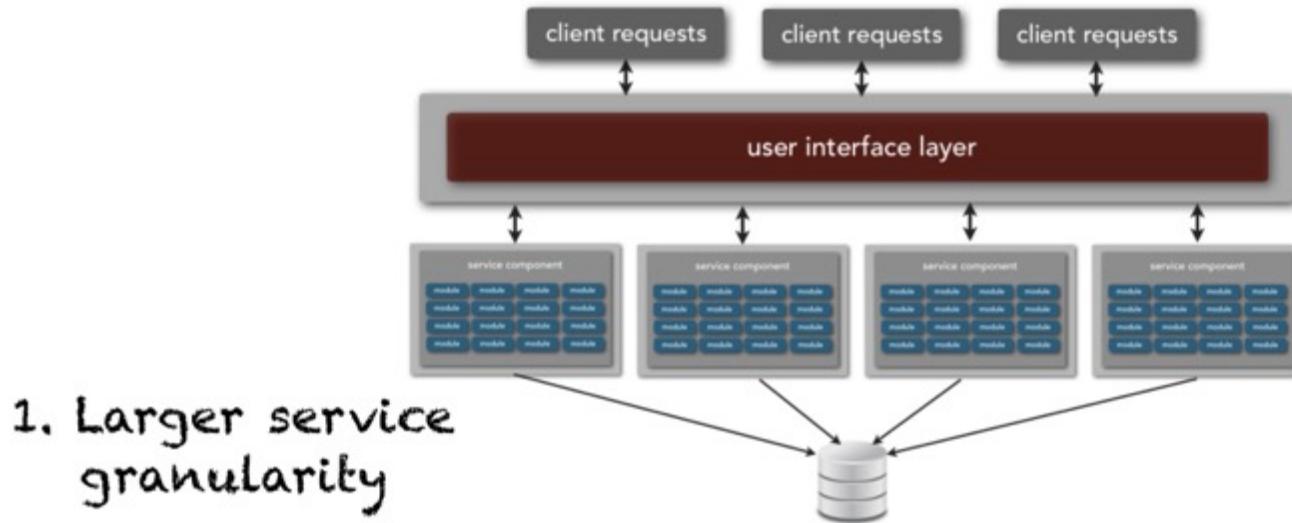
Microservices



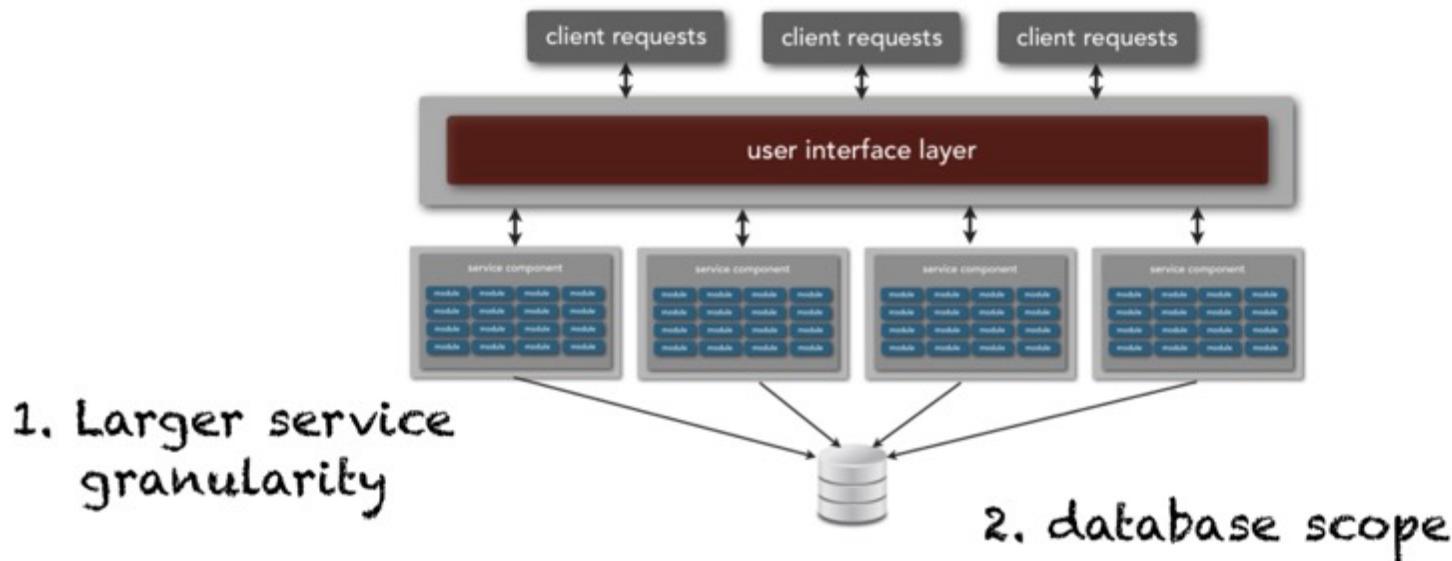
Service-based Architectures



Service-based Architectures

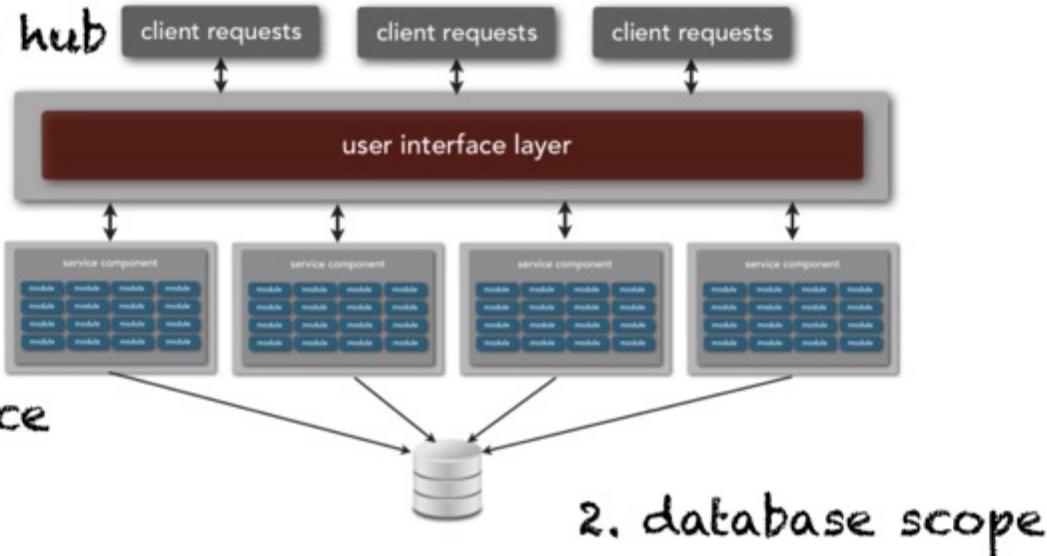


Service-based Architectures



Service-based Architectures

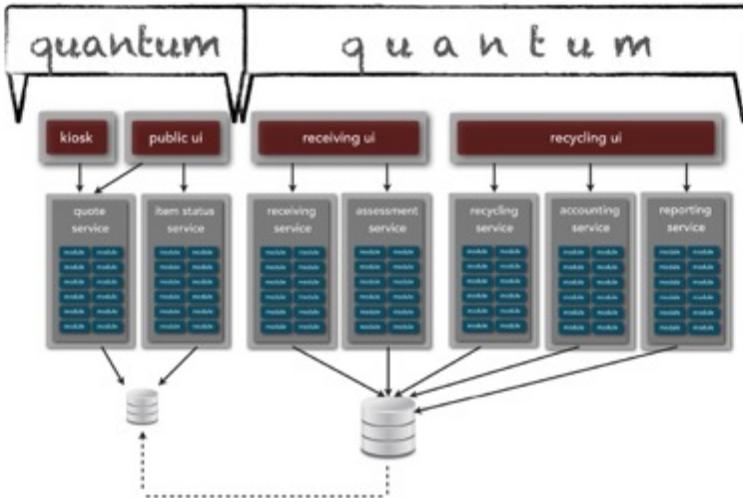
3. use of service bus
as integration hub



1. Larger service
granularity

2. database scope

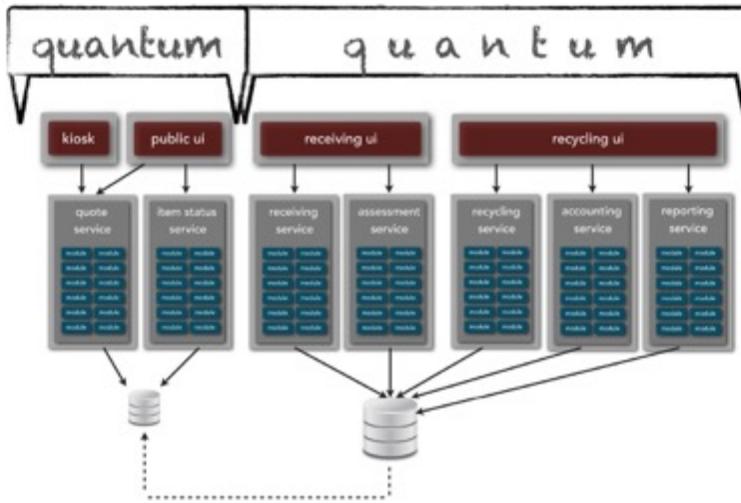
Reducing Quanta Size



Reducing Quanta Size



useful for
architectural analysis

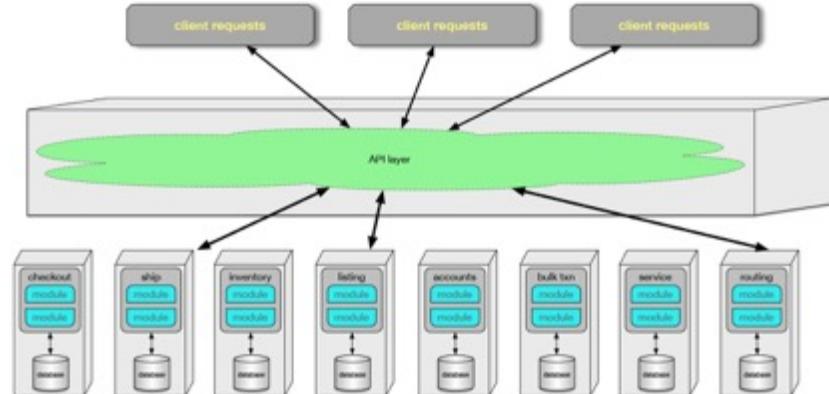
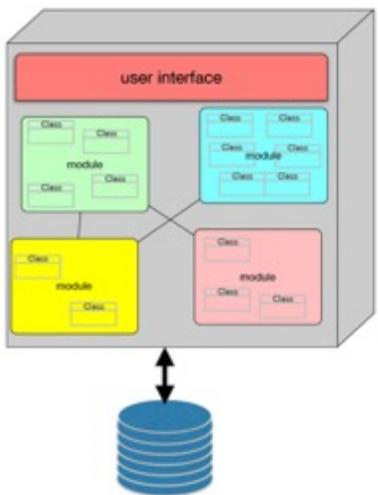


helps analyze
coupling

Smaller Quanta Size

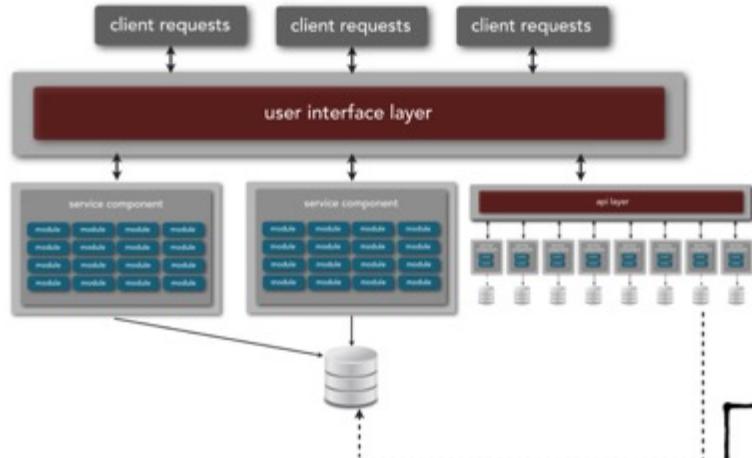
△

Evolutionary





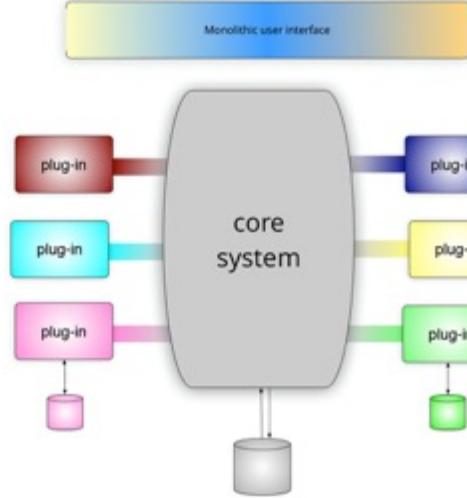
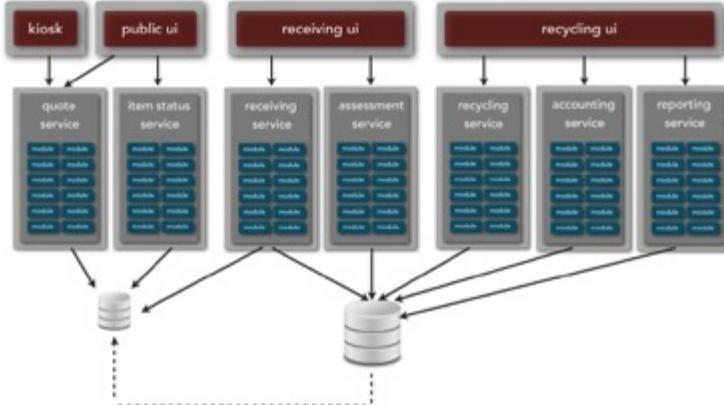
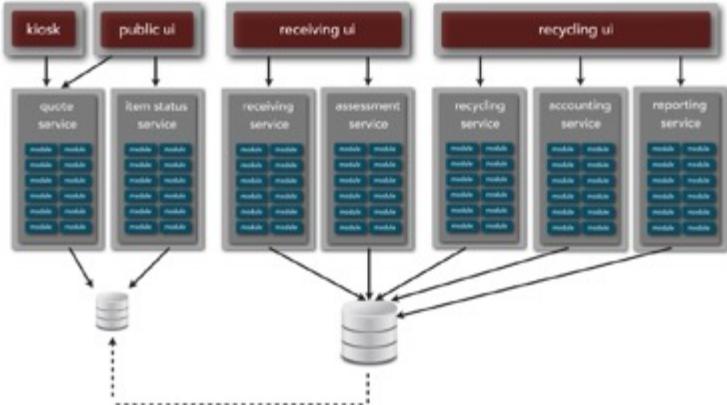
identify the quanta



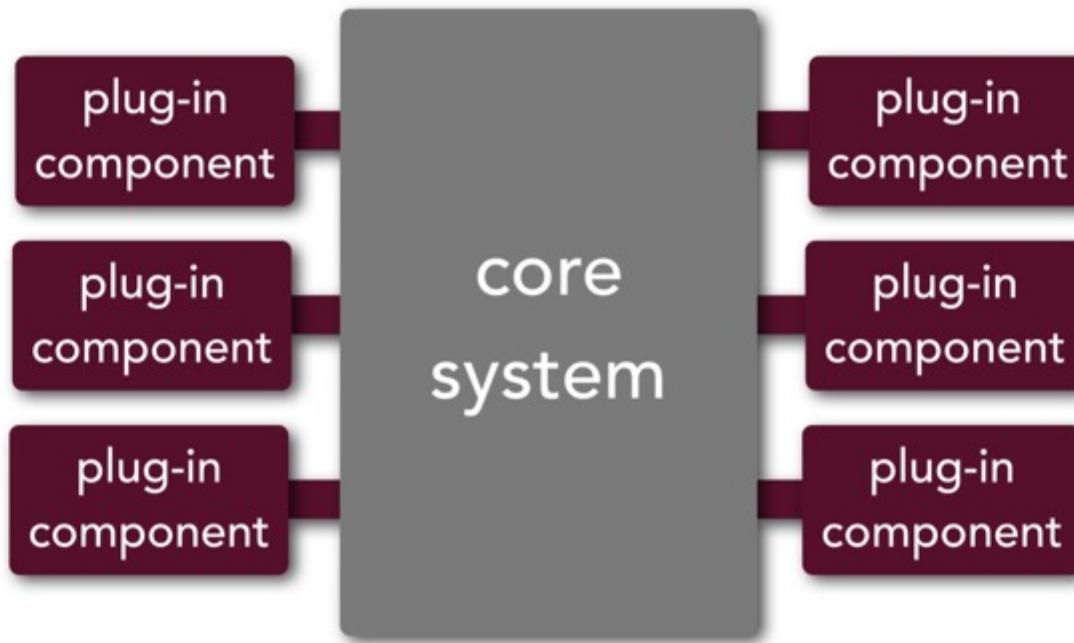
A B

identify
the quanta

C D



Microkernel Redux



Last 10% Trap



Last 10% Trap

80%



what the user wants

Last 10% Trap



what the user wants

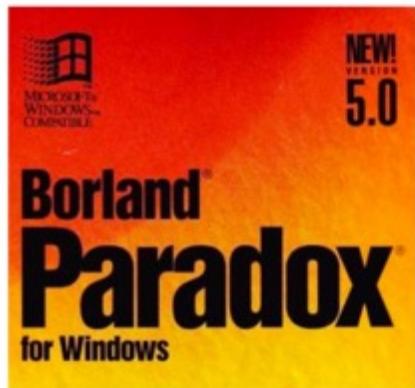
Last 10% Trap

“Users always want 100% of what they want (& are never satisfied with less).”



What Happened to the 4GLs?

What Happened to the 4GLs?



What Happened to the 4GLs?



DSL

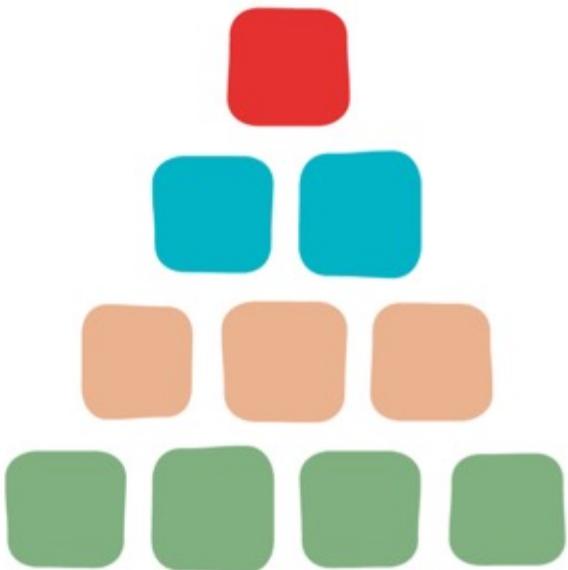
see also: Vendor King



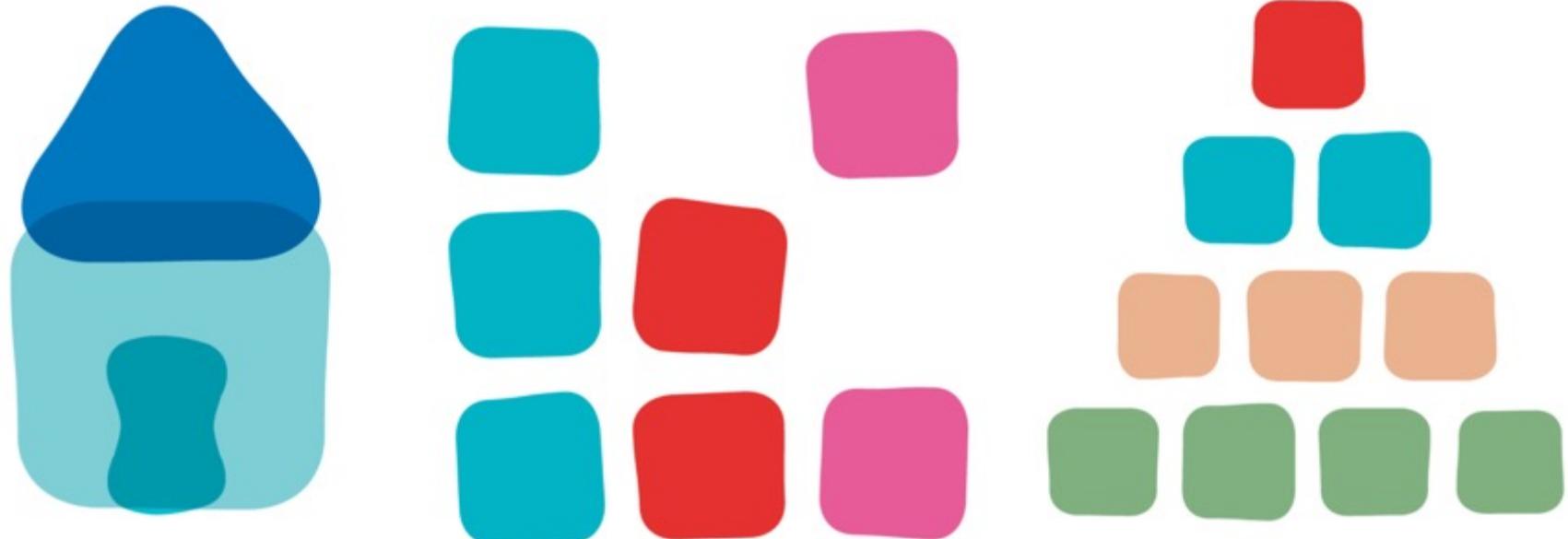


**Be careful of the *Last
10% Trap* when choosing
tools & frameworks.**

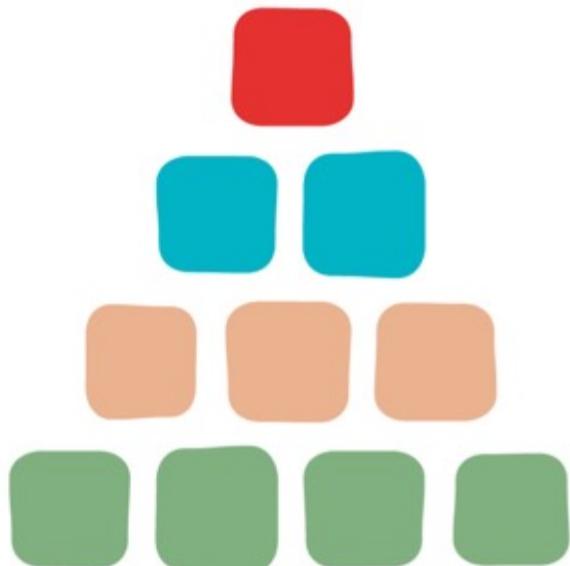
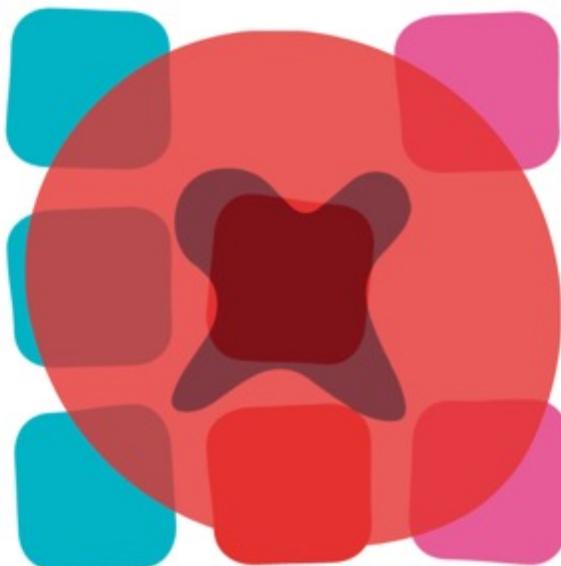
Domain/Architecture Isomorphism



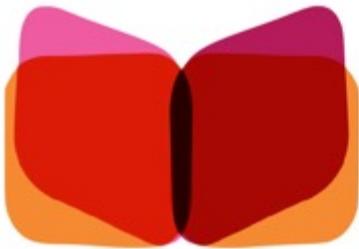
Domain/Architecture Isomorphism



Domain/Architecture Isomorphism



Agenda Part 1



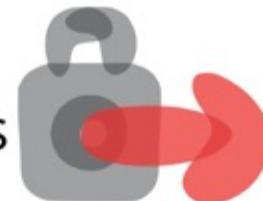
Definition



Guided Change via Fitness Functions



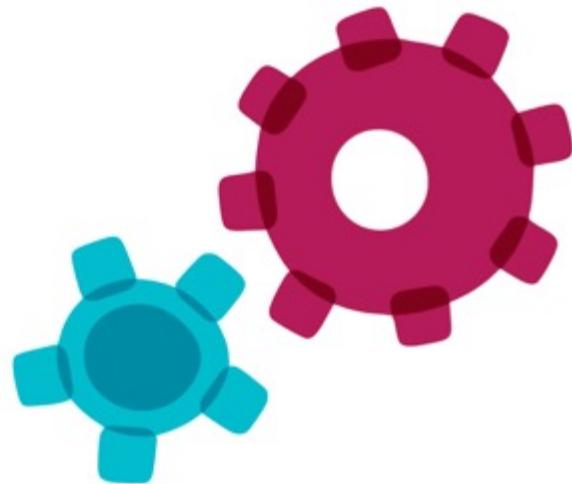
Incremental Change

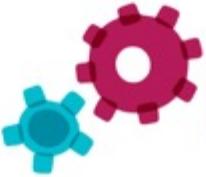


Structuring architecture for evolution



Putting evolutionary architecture into Practice





Mechanics

1. Identify dimensions



1. Identify dimensions

auditability 

performance 

security 



data 

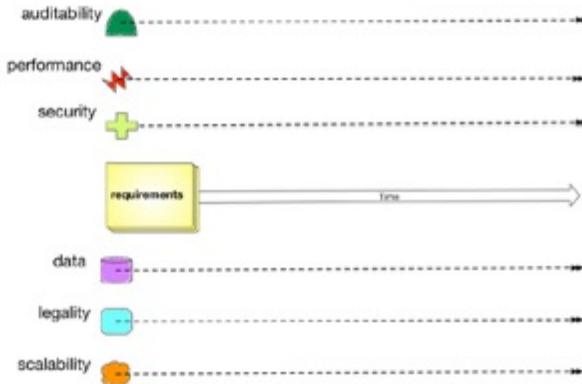
legality 

scalability 



1. Identify dimensions

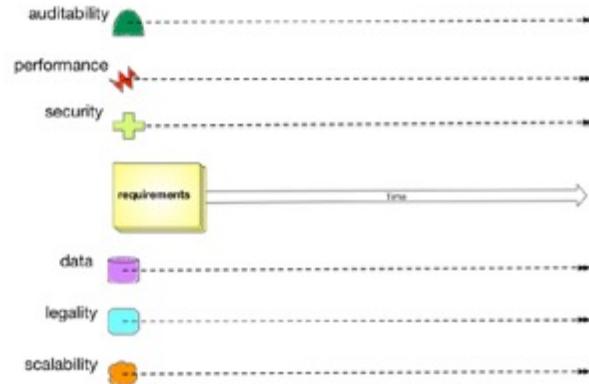
prioritization





1. Identify dimensions

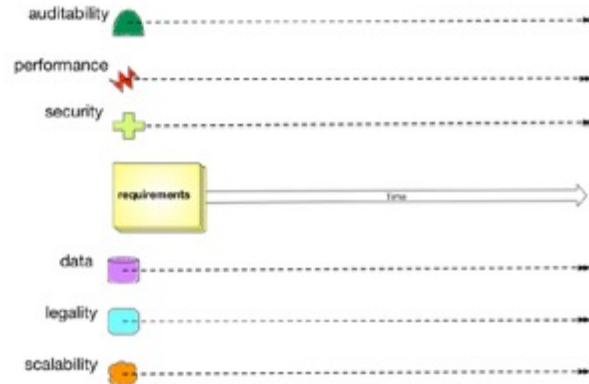
prioritization



cost to implement & maintain



1. Identify dimensions



prioritization

cost to implement & maintain

change variable “fix it” cost to constant
maintenance cost

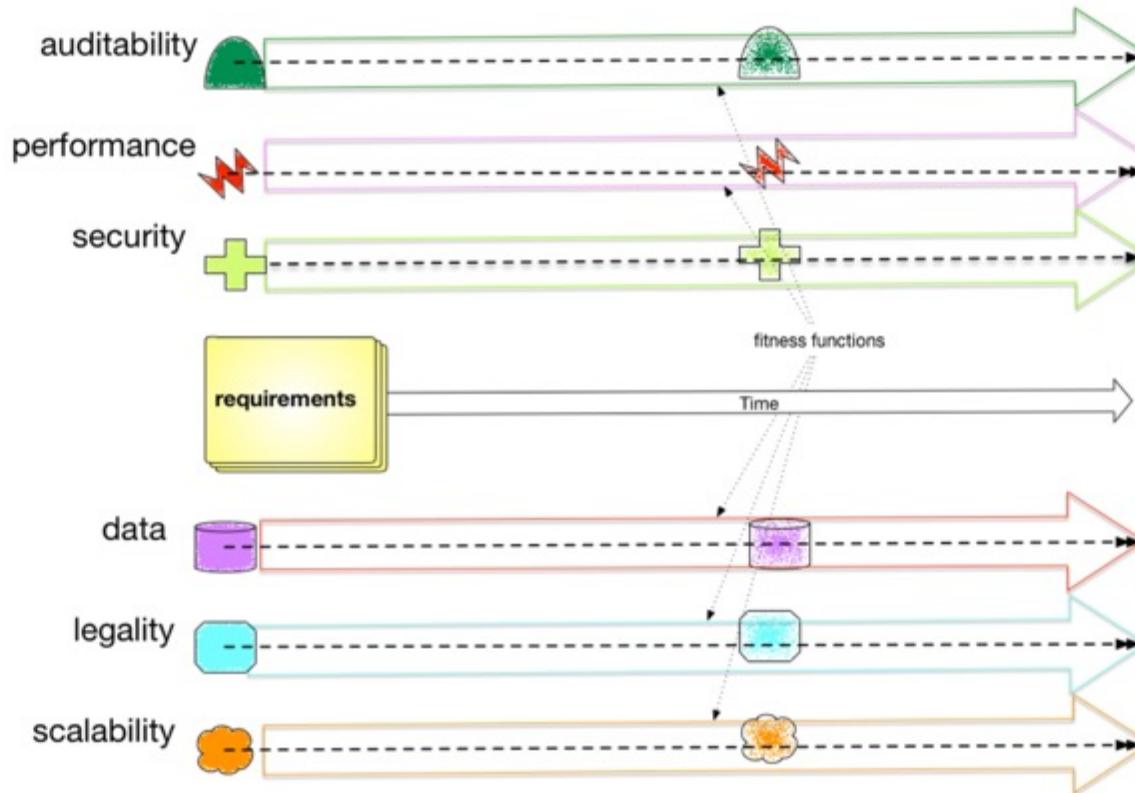


Mechanics

1. Identify dimensions
2. Define fitness function(s)



2. Define Fitness Function(s)



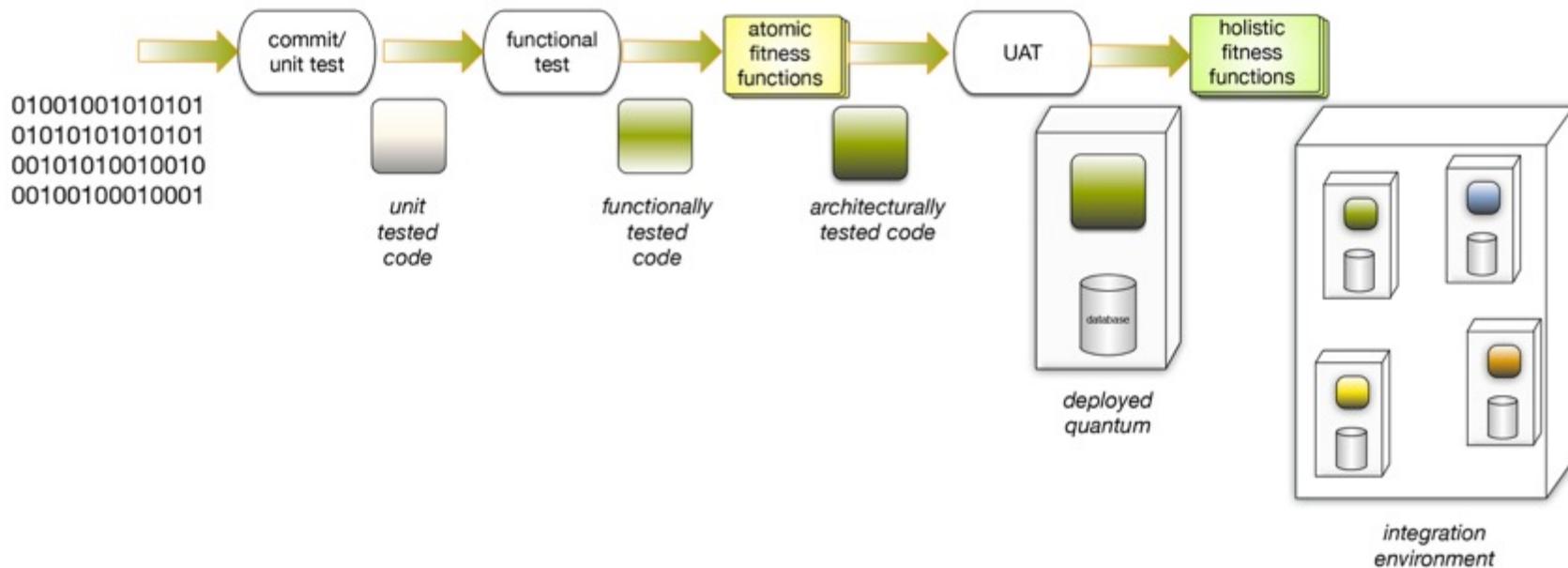


Mechanics

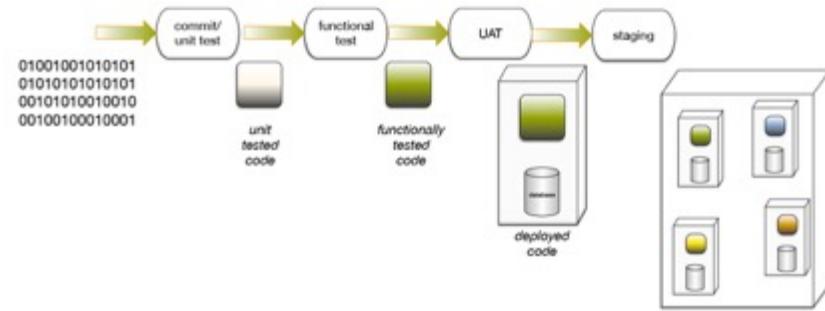
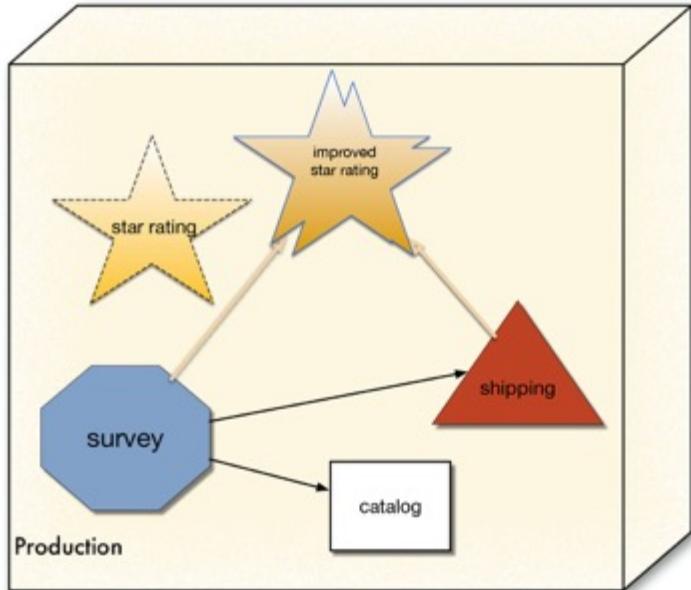
1. Identify dimensions
2. Define fitness function(s)
3. Use deployment pipelines and/or continuous fitness functions



3. Use deployment pipelines and/or continuous fitness functions



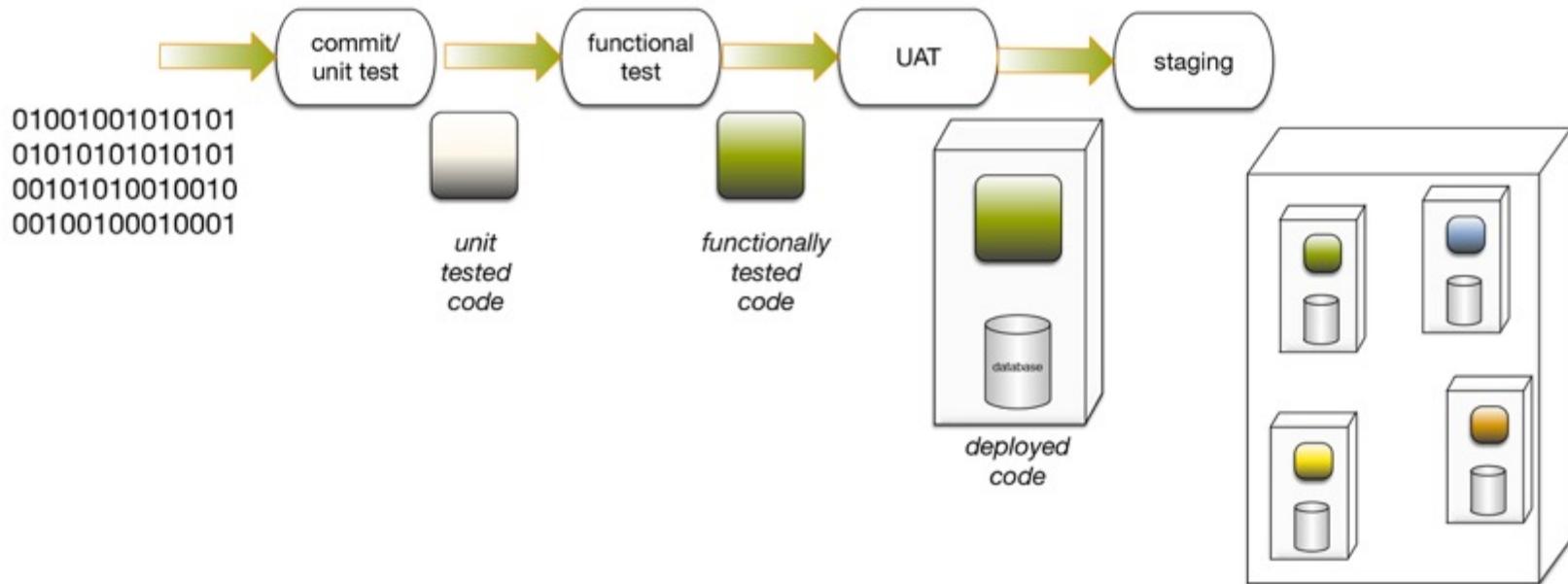
+++++ incremental



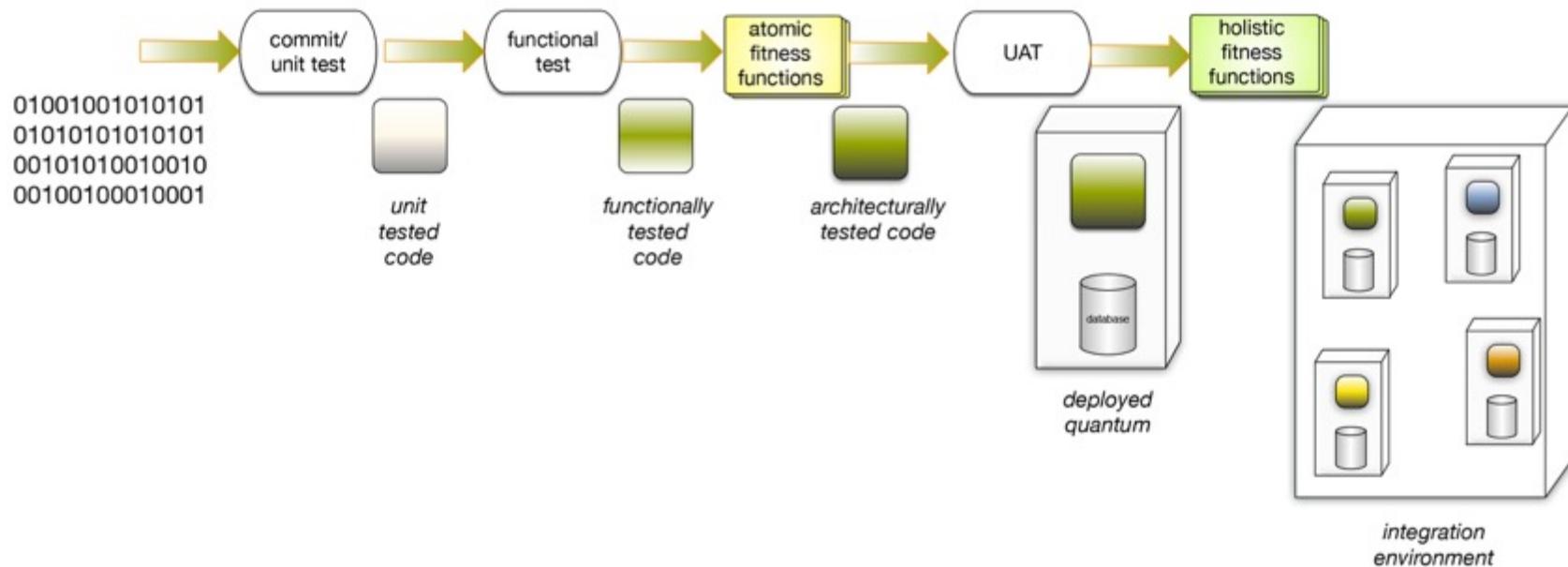
Deployment Pipelines



Deployment Pipeline

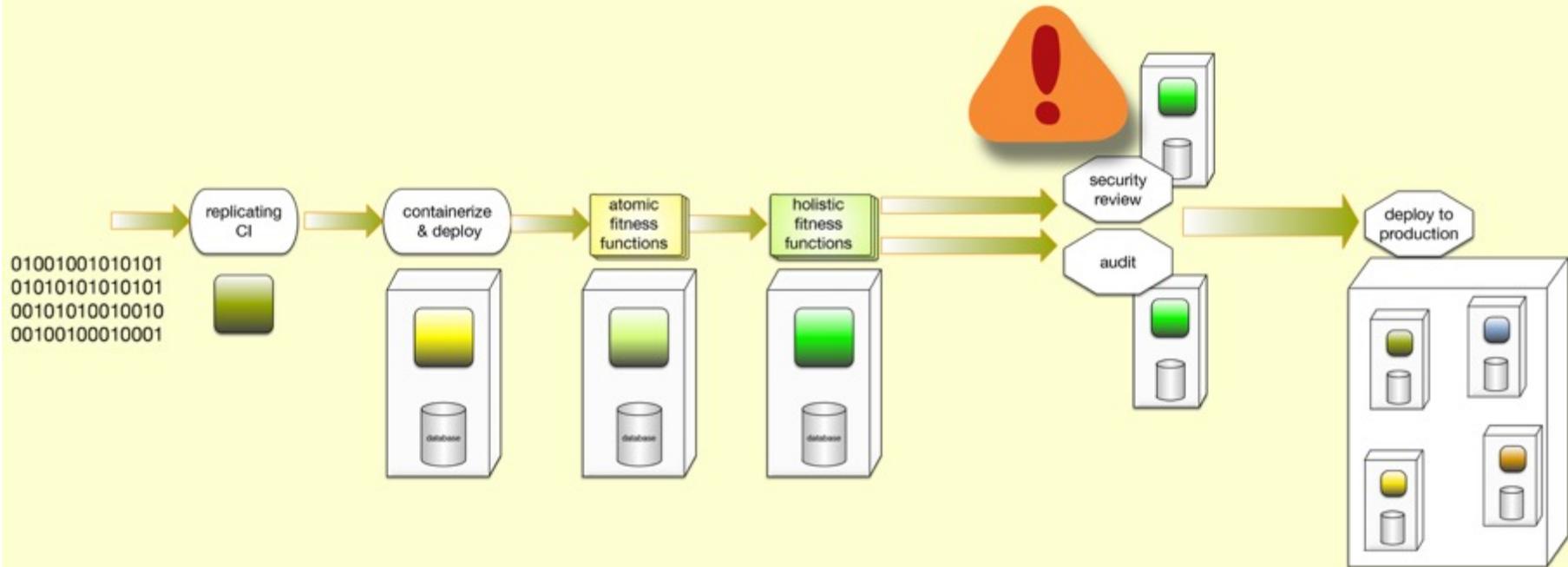


Deployment Pipeline + Fitness Functions





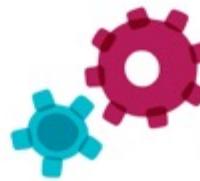
Penultimate Deployment Pipeline





Mechanics

1. Identify dimensions
2. Define fitness function(s)
3. Use deployment pipelines and/or continuous fitness functions

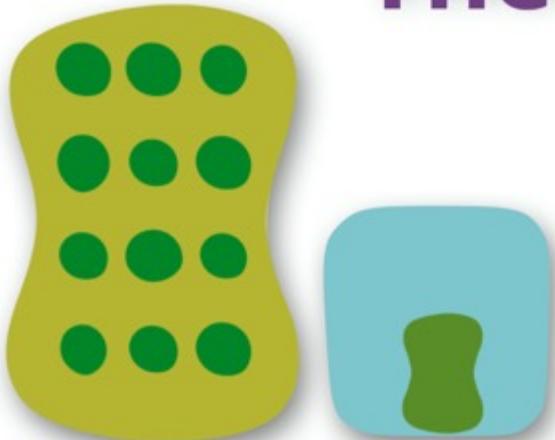


Mechanics

1. Identify dimensions
2. Define fitness function(s)
3. Use deployment pipelines and/or continuous fitness functions



The Business Case





BARRY O'REILLY

experimental by design, free thinking, curious, I use a lot of pen & paper

PRESENTATIONS LEAN ENTERPRISE CONSULTING ABOUT HOME

OCT 21 2013 10 COMMENTS

ADULT CONTINUOUS LEARN LEARN INNOVATION LEAN LEAN ENTERPRISE

HOW TO IMPLEMENT HYPOTHESIS-DRIVEN DEVELOPMENT →

Remember back to the time when we were in high school science class. Our teachers had a framework for helping us learn – an experimental approach based on the best available evidence at hand. We were asked to make observations about the world around us, then attempt to form an explanation or hypothesis as to what would happen if we had certain ideas. We then tested this hypothesis by predicting an outcome based on our theory that would be achieved in a controlled experiment. If the outcome was achieved, we had proven our theory to be correct.

We could then apply this learning to inform and test other hypotheses by conducting more sophisticated experiments, and tuning, refining or abandoning any hypothesis as we made further observations from the results we achieved.

Experimentation is the foundation of the scientific method, which is a systematic means of exploring the world around us. Although some experiments take place in laboratories, it is possible to perform an experiment anywhere, at any time, even in software development.

Practicing (Hypothesis-Driven Development)^[1] is thinking about the development of new ideas, products and services – like organizational change – as a series of experiments to determine whether an expected outcome will be achieved. The process is iterative until a desirable outcome is created or the idea is determined to be non-viable.

The need to change our mindset to view our proposed solution to a problem statement as a hypothesis, especially in new product or service development – the market we are targeting, how a business model will work, how costs will measure and even how the customer will react.

We do not do projects anymore, only experiments. Customer discovery and Lean startup strategies are designed to test assumptions about customers. Quality Assurance is testing system behavior against defined specifications. The experimental principle also applies in Test-Driven Development – we write the test first, then use the test to validate that our code is correct, and success if the code passes the test. Ultimately, product or service development is a process to have a hypothesis about system behavior in the environment or market it is developed for.

The key outcome of an experimental approach is measurable evidence and learning. Learning is the information we have gained from conducting the experiment. Did what we expected to occur actually happen? If not, what did and how does that inform what we should do next?

In order to learn we need use the scientific method for investigating phenomena, acquiring new knowledge, and correcting and integrating previous knowledge back into our thinking.

As the software development industry continues to mature, we now have an opportunity to leverage increased capabilities such as Continuous Design and Delivery to maximize our potential to learn quickly what works and what does not. By taking an experimental approach to information discovery, we can more rapidly test our solutions against the problems we have identified in the products or services we are attempting to build. With the goal to optimize our effectiveness of solving the right problems, ever simply becoming a feature factory to continually building solutions.

The steps of the scientific method are:

- Make observations
- Formulate a hypothesis
- Design an experiment to test the hypothesis

Follow

barryoreilly.com/2013/10/21/how-to-implement-hypothesis-driven-development/

Requirements

As A.... <role>
I Want... <goal/desire>
So That... <receive benefit>

Behavior-Driven Development

In Order To... <receive benefit>
As A... <role>
I Want... <goal/desire>

Hypothesis-driven Development

Hypothesis-driven development

We believe <this capability>

Will result in <this outcome>

We will have confidence to
proceed when

<we see a measurable signal>

@barryoreilly, <http://barryoreilly.com/2013/10/21/how-to-implement-hypothesis-driven-development/>

Hypothesis-driven Development

Hypothesis-driven development

We believe *<this capability>*

Will result in *<this outcome>*

We will have confidence to proceed when

<we see a measurable signal>

@barryoreilly, <http://barryoreilly.com/2013/10/21/how-to-implement-hypothesis-driven-development/>

Hypothesis-driven Development

Hypothesis-driven development

We believe *<this capability>*

Will result in *<this outcome>*

We will have confidence to proceed when

<we see a measurable signal>

@barryoreilly, <http://barryoreilly.com/2013/10/21/how-to-implement-hypothesis-driven-development/>

Business Story



We Believe That increasing the size of hotel images on the booking page

Will Result In improved customer engagement and conversion

We Will Have Confidence To Proceed When we see a 5% increase in customers who review hotel images who then proceed to book in 48 hours.

Hypothesis Driven UX

Hypothesis driven UX design

The value of design changes when you enable whole teams to learn instead of just looking at pretty mockups.

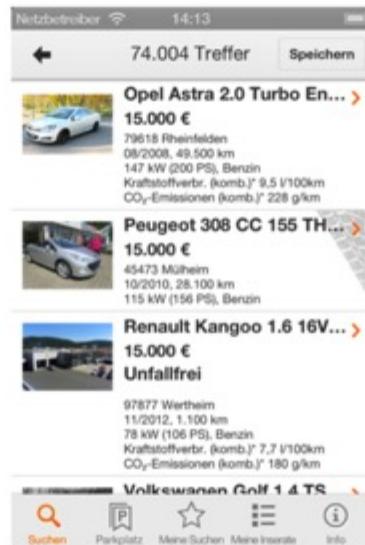
Shortly before I joined [mobile.de](#) (an ebay owned automotive platform in Germany with millions of visits per month) a smart product manager had an idea: The iPhone app should be redesigned not only to match current visual design guidelines and trends but also to get rid of usability issues that were caused by the current design approach.

Shortly after my start at the company I was handed the project. We tried to tackle the project with an iterative approach. Knowing that people are mostly reluctant to change we changed only parts of the app, released and when everything went well we took on the next elements of the app. Everything went pretty well: numbers stayed stable, no "redesign drop" and feedback from users was overall positive.

Curious case of a result view

<https://medium.com/@mwambach1/hypotheses-driven-ux-design-c75fbf3ce7cc#.gk3dpip81>

Hypothesis Driven UX



Three Hypotheses

More Listings

If we provide more listings on the screen then we can provide better comparability and offer more diversity on our platform because users like to compare a lot of listings on the result page

Better Structure

If we provide more structure to our listings then we achieve a better scanability because the user is able to scan the relevant information quicker

Better Prioritization

If we prioritize information according to user needs then we achieve better guidance because the user can see all relevant information at a glance

Experiments to Perform

11:11
6.703 Treffer

Audi A6 Avant 2.0 TDIe 6-Gang
17.200 € 100 kW (136 PS)
EZ 10/2010 126.107 km
Diesel 5,3 l/100 km (komb.)
139 g CO₂/km (komb.)

Audi A6 Avant 2.7 TDI, schöne Le..
17.500 € Neu 132 kW (176 PS)
EZ 09/2006 120 km
Hybrid 7,2 l/100 km (komb.)
(Benzin/Elektro) 192 g CO₂/km (komb.)

Audi A6 Avant 2.6/SV/ZV/SHZ/.. P
599 € 258.027 km
EZ 02/1996 100 kW (136 PS)

Audi A6 Avant 2.0 TDIe 6-Gang
17.200 € 100 kW (136 PS)
EZ 10/2010 126.107 km
Diesel 5,3 l/100 km (komb.)
139 g CO₂/km (komb.)

Audi A6 Avant 2.7 TDI, schöne Le..
17.500 € EZ 09/2006 120 km
Hybrid 7,2 l/100 km (komb.)
(Benzin/Elektro) 192 g CO₂/km (komb.)

Suchen **Parkplatz** **Meine Suchen** **Meine Inserat** **Info**

More Listings

11:11
6.703 Treffer

Audi A6 Avant 2.0 TDIe 6-Gang
17.200 € (inkl. 19% MwSt.)

Unfallfrei, Qualitätssiegel
30179 Hohen Neuendorf, OT Bergfelde
EZ 10/2010 126.107 km
100 kW (136 PS) Diesel
5,3 l/100 km^{*} 139 g/km^{*}

Audi A6 Avant 2.7 TDI, schöne Lederausstattung..
17.500 € Neu

Unfallfrei, Qualitätssiegel
65193 Wiesbaden
EZ 06/2006 120 km
132 kW (179 PS) Hybrid (Benzin/Elektro)
7,2 l/100 km^{*} 192 g/km^{*}

Audi A6 Avant 2.6/SV/ZV/SHZ/Klimatron.. P
599 €

66679 Losheim am See
02/1996, 258.027 km, 100 kW (136 PS), Benzin

Suchen **Parkplatz** **Meine Suchen** **Meine Inserat** **Info**

Better Structure

11:11
6.703 Treffer

Audi A6 Avant 2.0 TDIe 6-Gang
17.200 €

Unfallfrei, Qualitätssiegel
30179 Hohen Neuendorf, OT Bergfelde
10/2010, 126.107 km, 100 kW (136 PS), Diesel
-5,3 l/100 km (komb.), -139 g CO₂/km (komb.)

Audi A6 Avant 2.7 TDI, ..
17.500 € Neu

Unfallfrei, Qualitätssiegel
65193 Wiesbaden
06/2006, 120 km, 132 kW (179 PS),
Hybrid (Benzin/Elektro)
-7,2 l/100 km (komb.), -192 g CO₂/km (komb.)

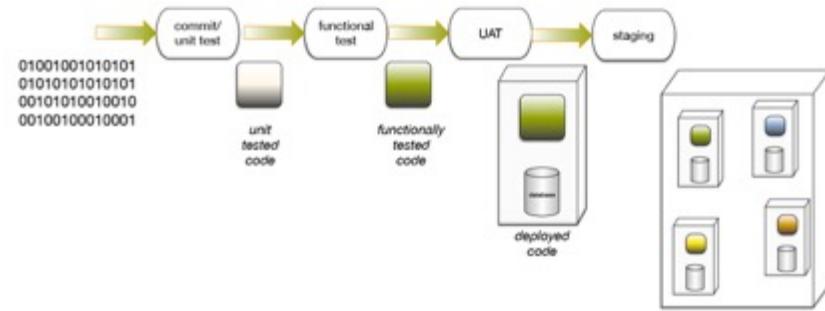
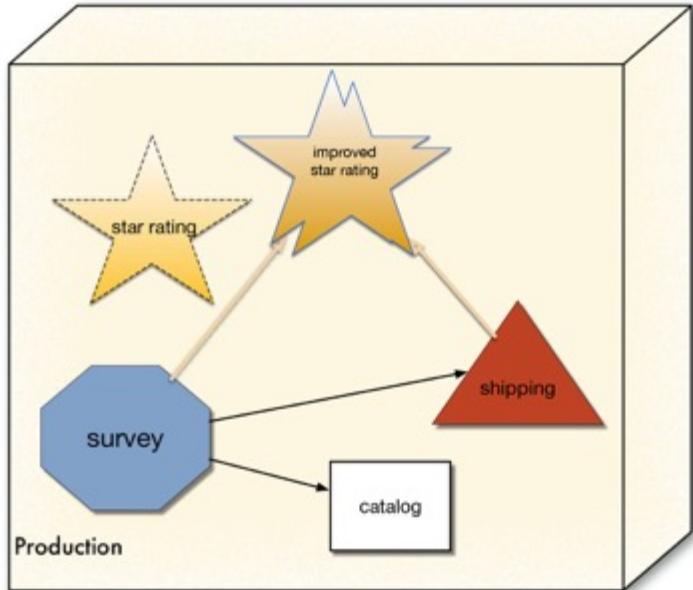
Audi A6 Avant 2.6/SV/ZV/SHZ/Klimatron.. P
599 €

66679 Losheim am See
02/1996, 258.027 km, 100 kW (136 PS), Benzin

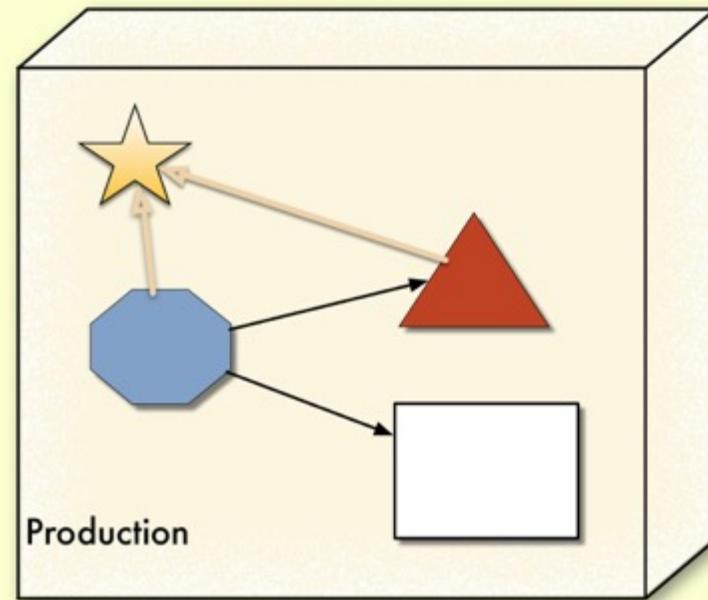
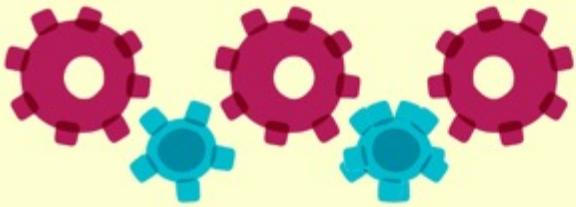
Suchen **Parkplatz** **Meine Suchen** **Meine Inserat** **Info**

Better Prioritization

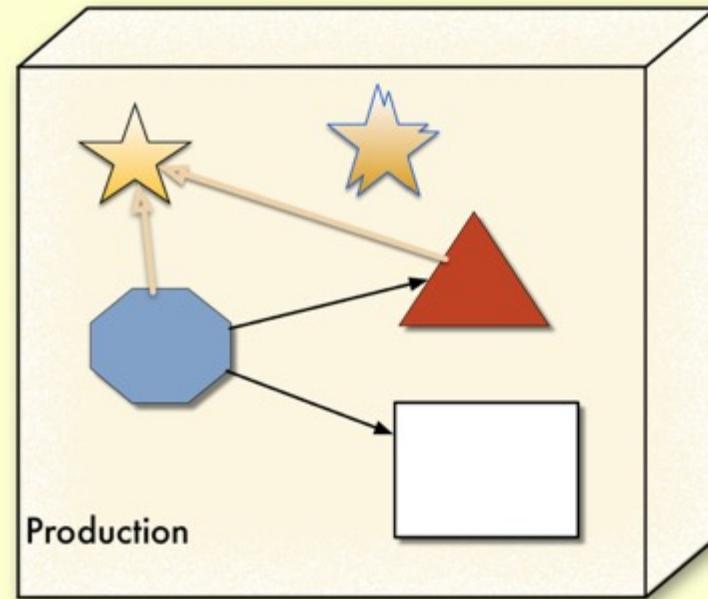
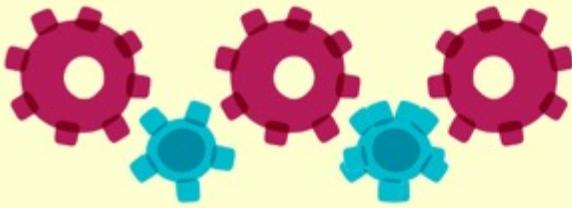
+++++ incremental



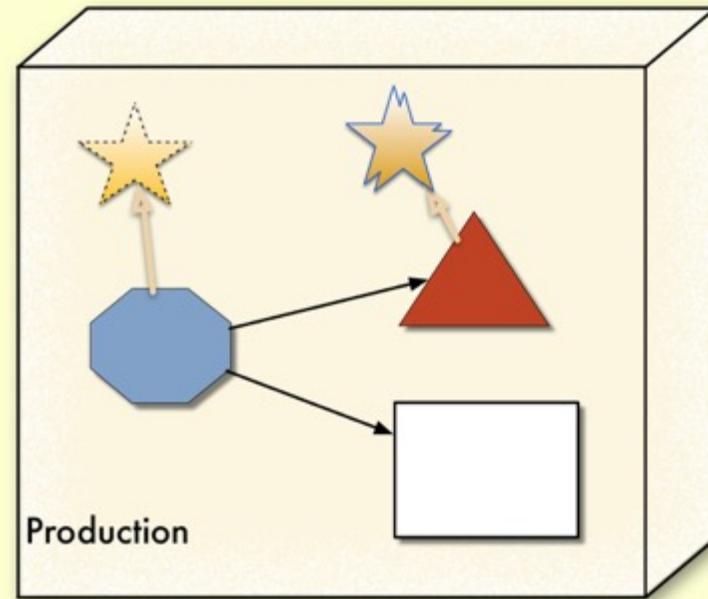
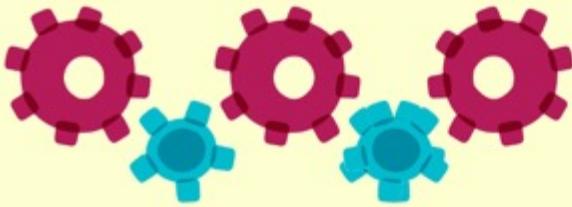
Penultima↑e



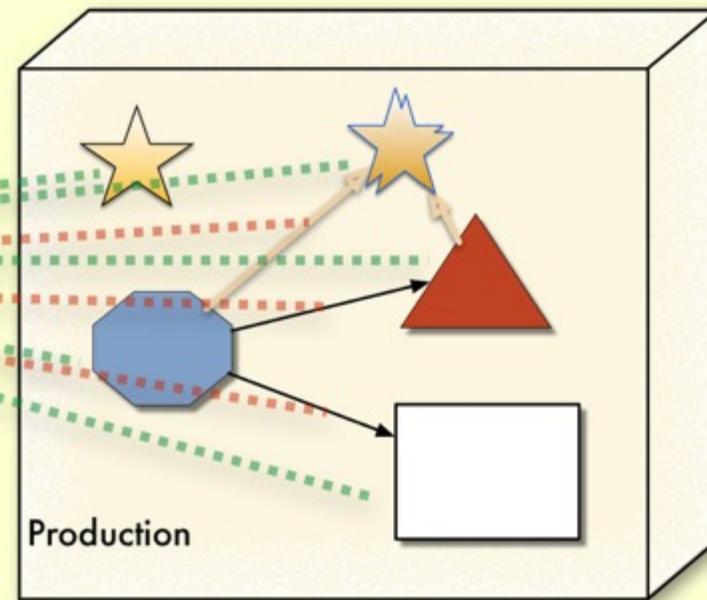
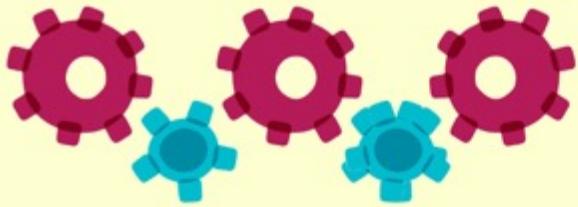
Penultima↑e



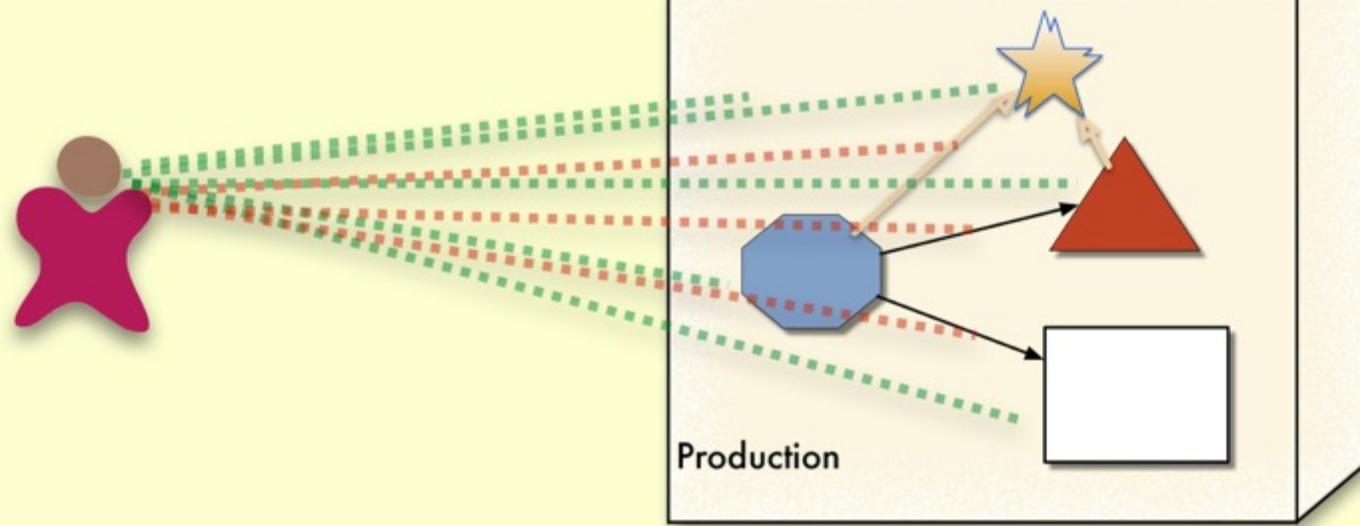
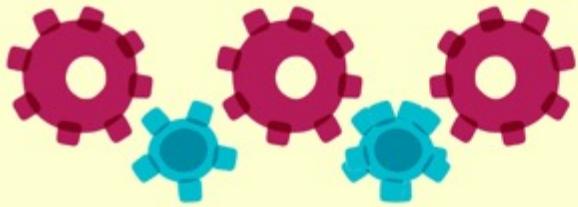
Penultima↑e



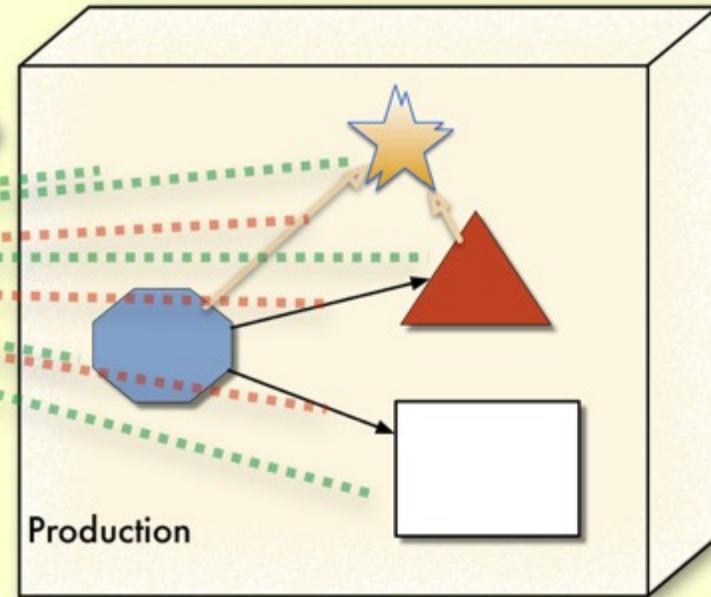
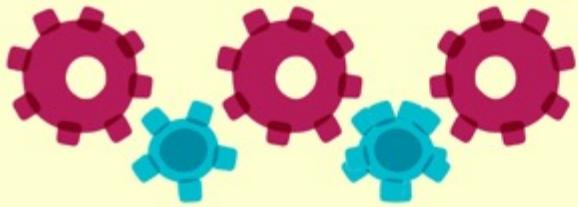
Penultima ↑ e



Penultima ↑ e



Penultima[↑]e



The screenshot shows a web browser displaying a blog post from GitHub Engineering. The title of the post is "Move Fast and Fix Things". Below the title, it says "December 15, 2015". The main content discusses technical debt and GitHub's efforts to manage it. It includes a section on "Merges in Git" and a note about file storage. At the bottom, there is a footer note about a shell script.

Move Fast and Fix Things

vng December 15, 2015

Anyone who has worked on a large enough codebase knows that technical debt is an inescapable reality: The more rapidly an application grows in size and complexity, the more technical debt is accrued. With GitHub's growth over the last 7 years, we have found plenty of nooks and crannies in our codebase that are inevitably below our very best engineering standards. But we've also found effective and efficient ways of paying down that technical debt, even in the most active parts of our systems.

At GitHub we try not to brag about the "shortcuts" we've taken over the years to scale our web application to more than 12 million users. In fact, we do quite the opposite: we make a conscious effort to study our codebase looking for systems that can be rewritten to be cleaner, simpler and more efficient, and we develop tools and workflows that allow us to perform these rewrites efficiently and reliably.

As an example, two weeks ago we replaced one of the most critical code paths in our infrastructure: the code that performs merges when you press the Merge Button in a Pull Request. Although we routinely perform these kind of refactorings throughout our web app, the importance of the merge code makes it an interesting story to demonstrate our workflow.

Merges in Git

We've talked at length in the past about the storage model that GitHub uses for repositories in our platform and our Enterprise offerings. There are many implementation details that make this model efficient in both performance and disk usage, but the most relevant one here is the fact that repositories are always stored "bare".

This means that the actual files in the repository (the ones that you would see on your working directory when you clone the repository) are not actually available on disk in our infrastructure: they are compressed and delta'ed inside packfiles.

Because of this, performing a merge in a production environment is a nontrivial endeavour. Git knows several merge strategies, but the recursive merge strategy that you'd get by default when using `git merge` to merge two branches in a local repository assumes the existence of a working tree for the repository, with all the files checked out on it.

The workaround we developed in the early days of GitHub for this limitation is effective, but not particularly elegant: instead of using the default `git-merge-recursive` strategy, we wrote our own merge strategy based on the original one that Git used back in the day: `git-merge-resolve`. With some tweaking, the old strategy can be adapted to not require an actual checkout of the files on disk.

To accomplish this, we wrote a shell script that sets up a temporary working directory, in

Move
Fast
&
Fix
Things



```

def create_merge_commit(base, head, author, commit_message)
  base = resolve_commit(base)
  head = resolve_commit(head)
  commit_message = Rugged::PrettyPrinter.pretty_print(commit_message)

  merge_base = rugged.merge_base(base, head)
  return [nil, "already_merged"] if merge_base == head.old

  ancestor_tree = merge_base && Rugged::Commit.lookup(rugged, merge_base).tree
  merge_options = {
    :fail_on_conflict => true,
    :rakka_recur => true,
    :no_recursive => true,
  }
  index = base.tree.merge(head.tree, ancestor_tree, merge_options)
  return [nil, "merge_conflict"] if (index.mint? || index.conflicts?)

  options = {
    :message => commit_message,
    :committer => author,
    :author => author,
    :parents => [base, head],
    :tree => index.write_tree(rugged)
  }

  [Rugged::Commit.create(rugged, options), nil]
end

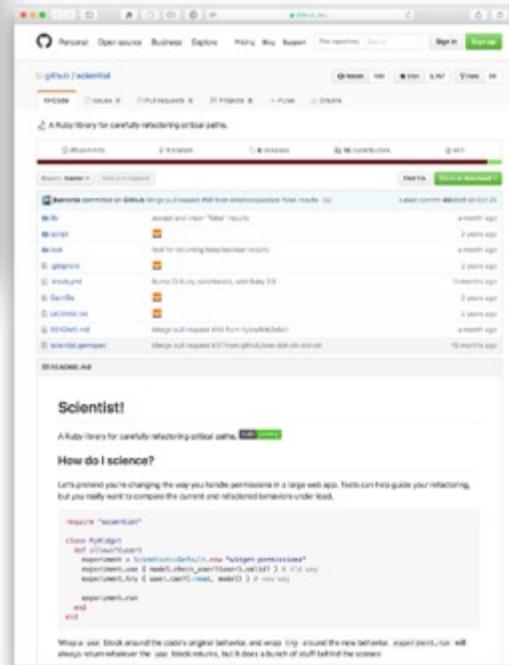
```

```

def create_merge_commit(author, base, head, options = {})
  commit_message = options[:commit_message] || "Merge #{head} into #{base}"
  now = Time.current

  science "create_merge_commit" do |e|
    e.context :base => base.to_s, :head => head.to_s, :repo => repository.repo
    e.use { create_merge_commit_git(author, now, base, head, commit_message) }
    e.try { create_merge_commit_rugged(author, now, base, head, commit_message) }
  end
end

```



<https://github.com/github/scientist>



```
require "scientist"

class MyWidget
  def allows?(user)
    experiment = Scientist::Default.new "widget-permissions"
    experiment.use { model.check_user?(user).valid? } # old way
    experiment.try { user.can?(:read, model) } # new way

    experiment.run
  end
end
```

- It decides whether or not to run the try block,
- Randomizes the order in which use and try blocks are run,
- Measures the durations of all behaviors,
- Compares the result of try to the result of use,
- Swallows (but records) any exceptions raised in the try block
- Publishes all this information.



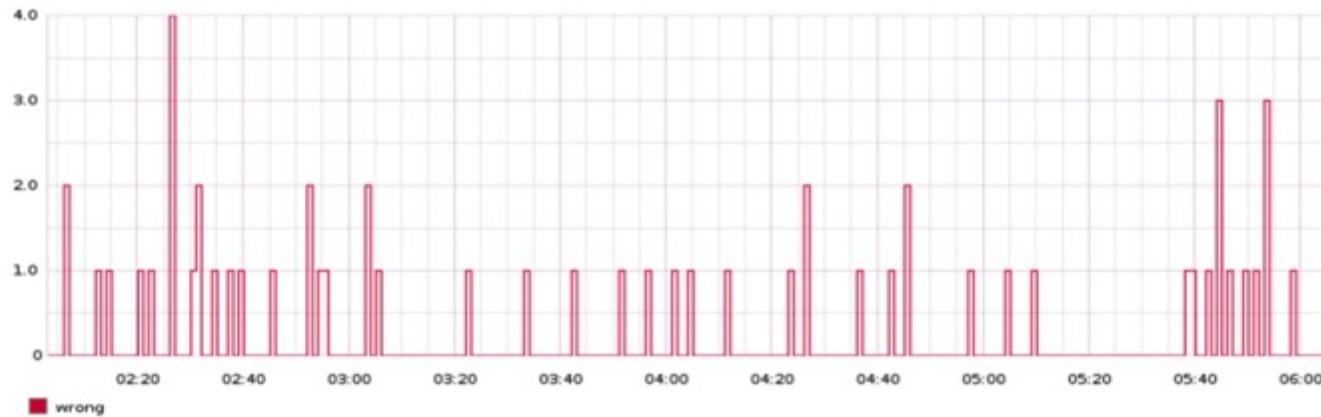
Accuracy

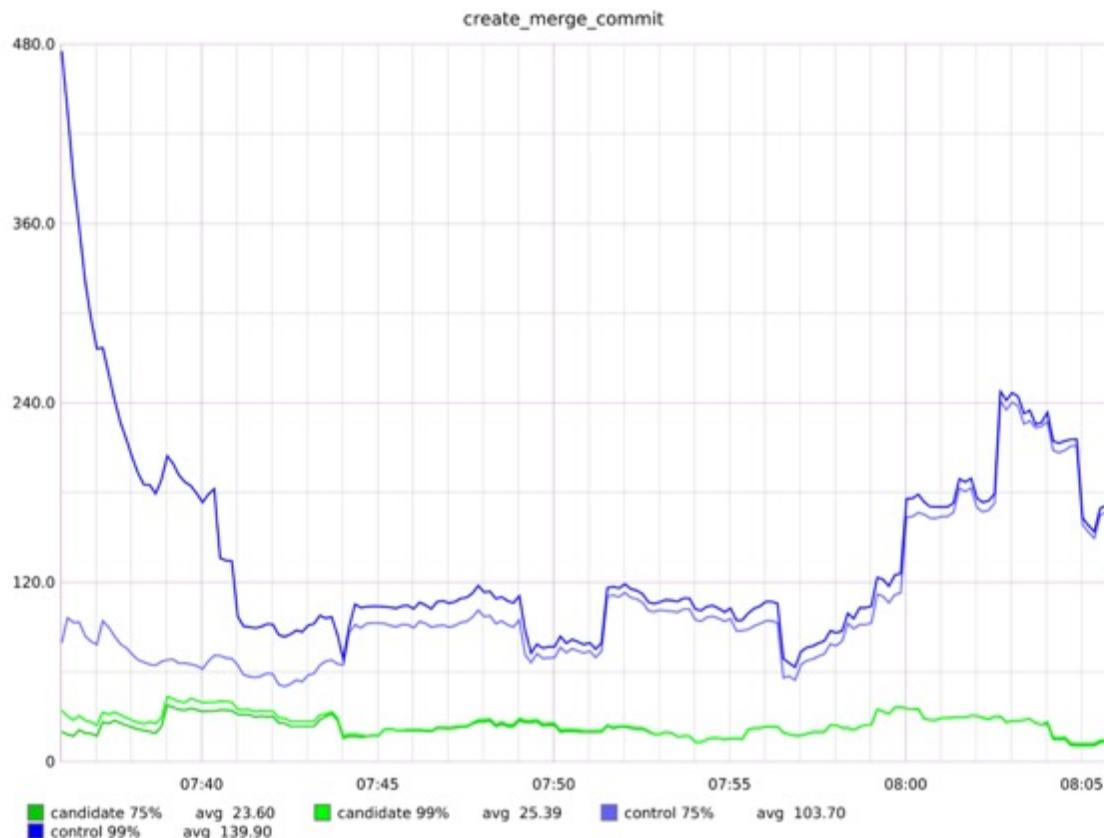
The number of times that the candidate and the control agree or disagree. [View mismatches](#)





The number of incorrect/ignored only.





4 days

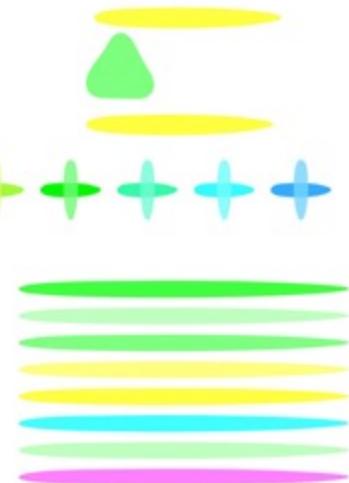
24 hours/
no mismatches or slow cases

> 10,000,000
comparisons



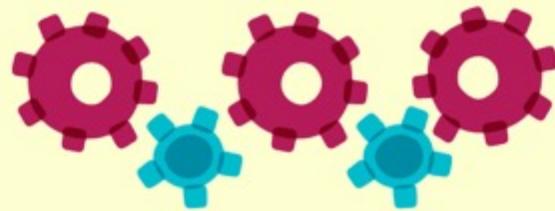
Evolutionary Architecture

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



Agenda Part 2

The Evolution of
Penultima↑e

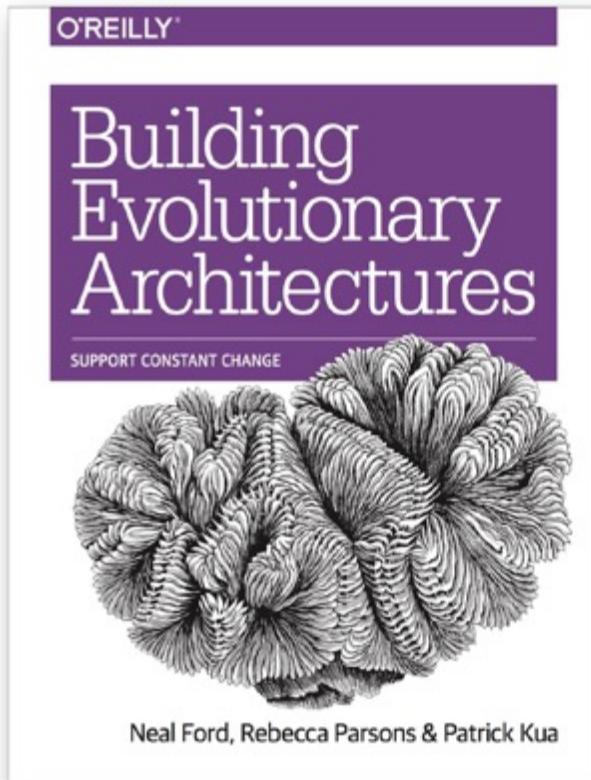


Agenda Part 2



- A11yAllTheThings
- AuditTheAccounting
- AvoidComplexCode
- BreakOnUpgrade
- ConfigureSomeOfTheThingsAllOfTheTime
 - CycleTimeGuard
 - DebugAllTheThings
 - DegradeGracefully
 - DependOnDependencies
 - DeployAllTheThings
- DeterministicDistributability
 - DireDeltas
 - DiscoverAllTheThings
 - DocSyncWithApi
- ElasticityOfMicroservices
- InstallAllTheThings
- LegalityOfOpenSourceLibraries
- MaintainTheMock
- MaintainingGoodThroughput
- MakeSureNewReallyReplacesOld
 - MonitorAllTheThings
 - NoMoreViewModels
 - ReplaceTheCruftyCore
 - RespondResponsibly
 - SellThePlatform
 - UpgradeAllTheThings
 - ZeroDaySecurity

iMpLemENtNg eVoLuTiONaRy ARcHiEcTuREs



 @neal4d
nealford.com



 @rebeccaparsons



 @patkua

