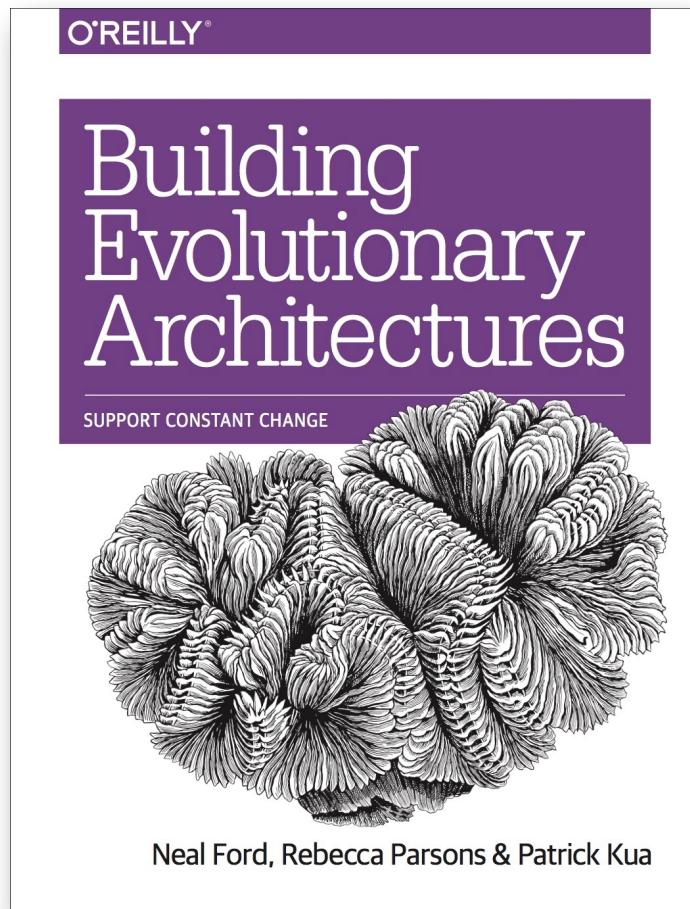


iMpLemENtiNg eVoLuTiONaRy ARcHiEcTuReS



SUPPORT CONSTANT CHANGE

 @neal4d
nealford.com

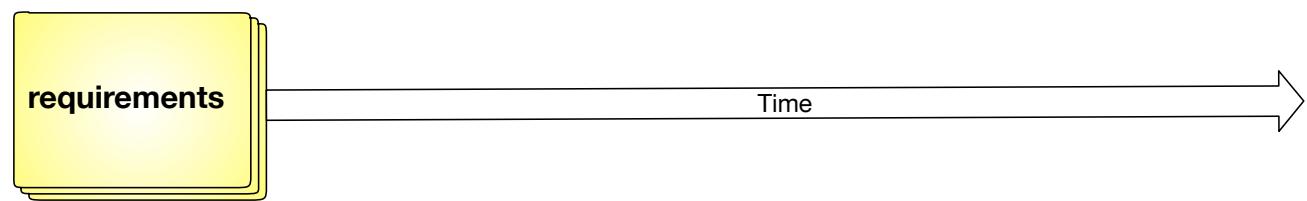


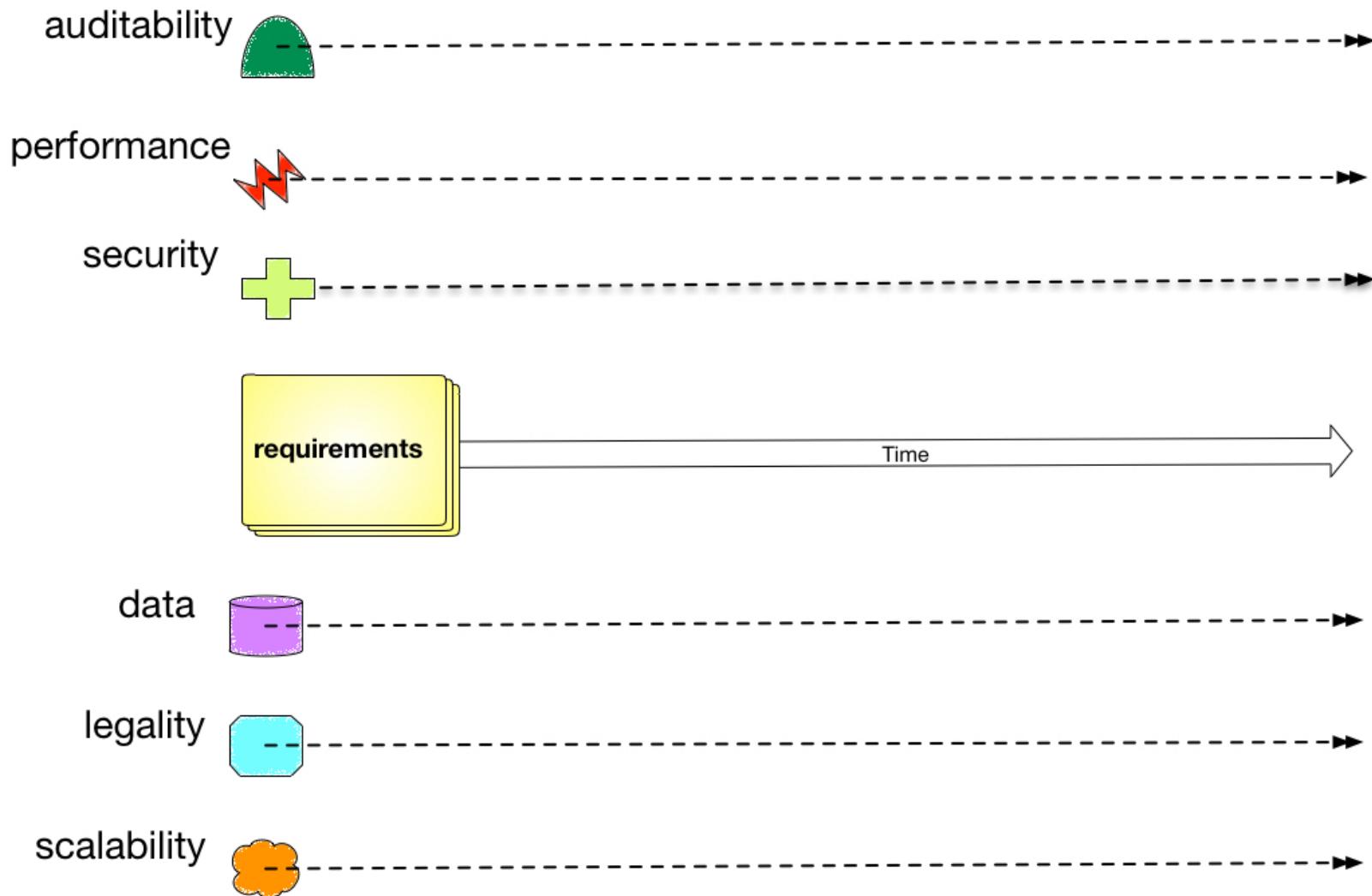
 @rebeccaparsons



 @patkua









evolvability

degradability
determinability
demonstrability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency

operability
orthogonality
portability
precision
predictability
process capabilities
productivity
provability
recoverability
relevance

sustainability
tailorability
testability
timeliness
traceability
transparency
ubiquity
understandability
upgradability
usability

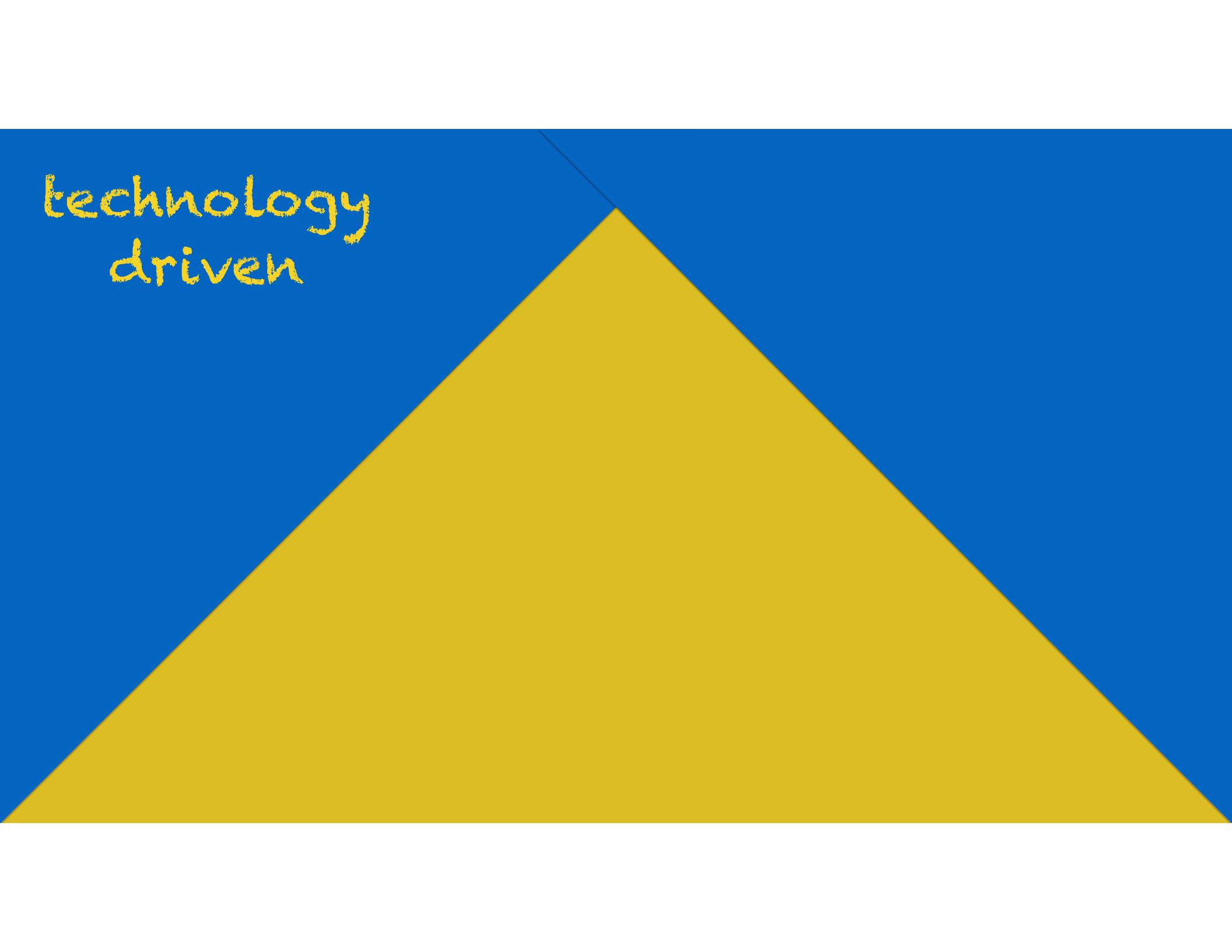
accessibility
accountability
accuracy
adaptability
administrability
affordability
agility
auditability
autonomy

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

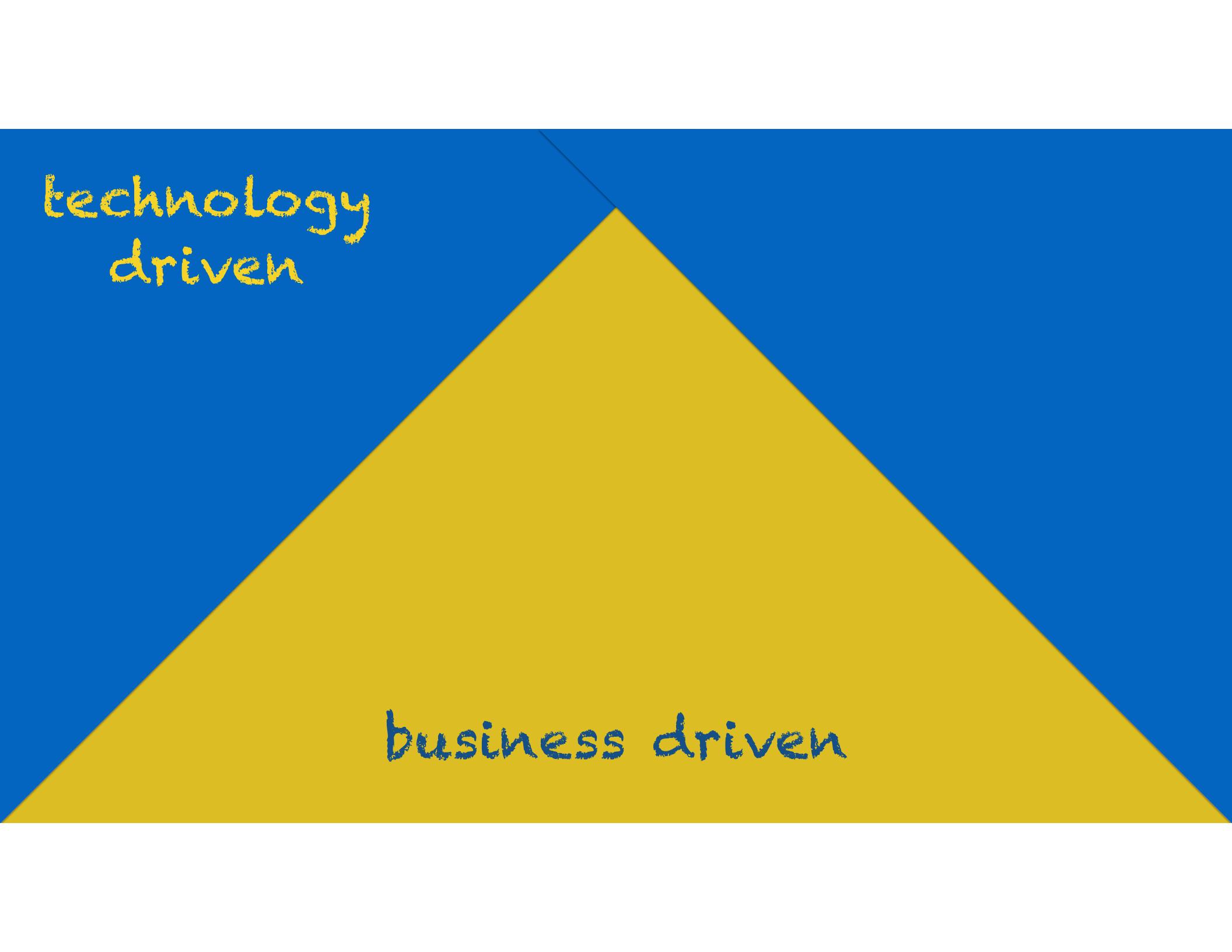
repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness



CHaNgE



technology
driven



technology
driven

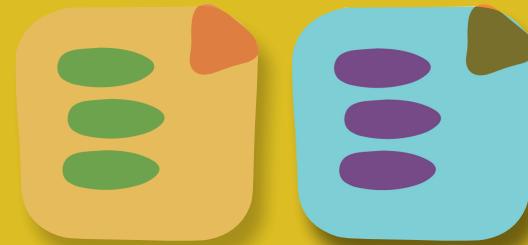
business driven

technology
driven



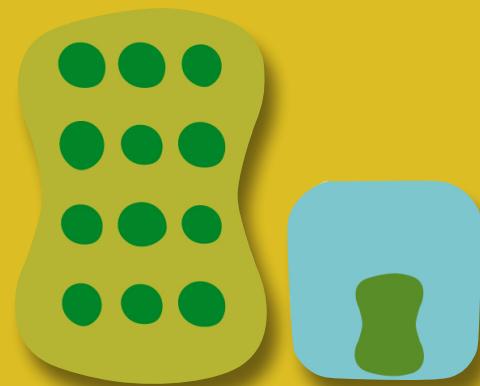
business driven

technology
driven

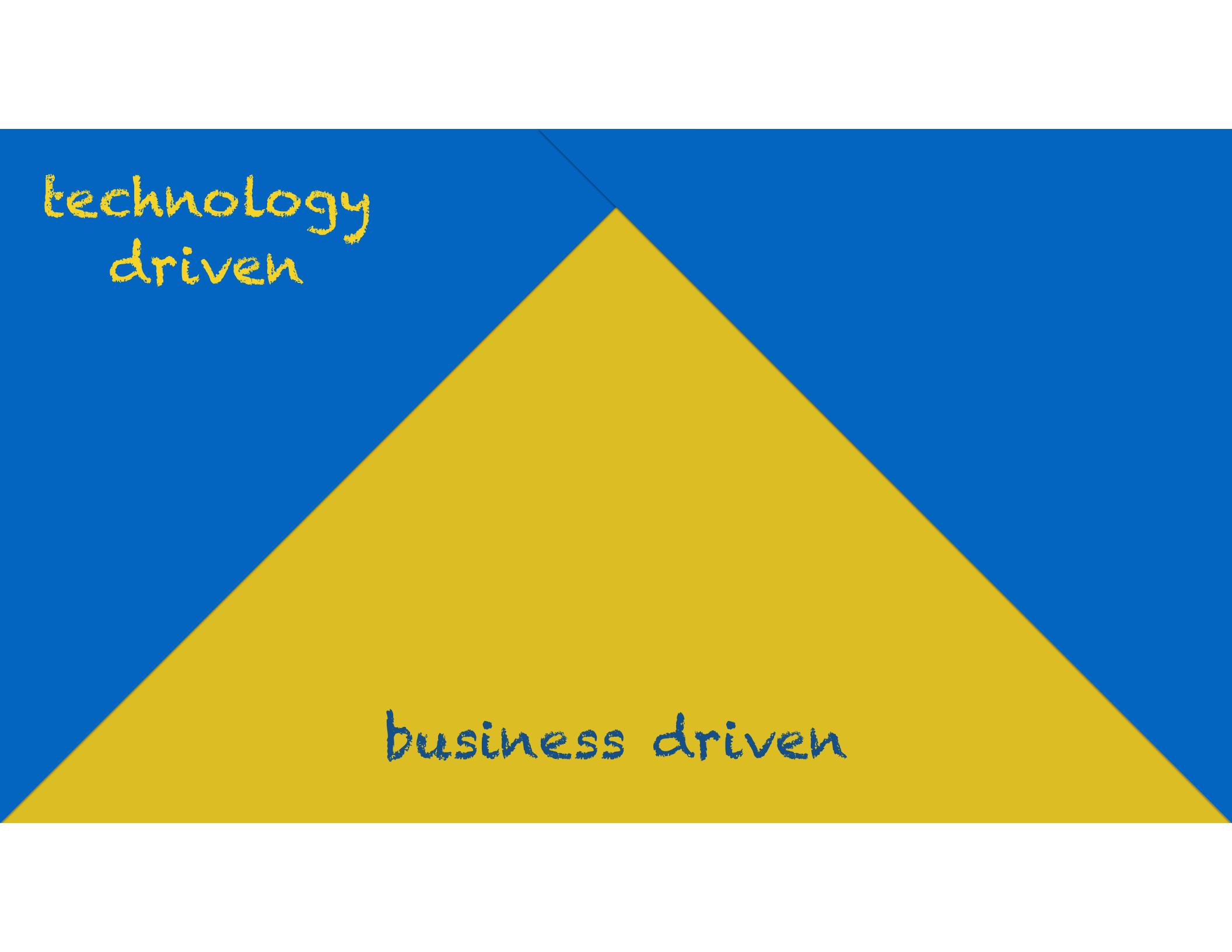


business driven

technology
driven

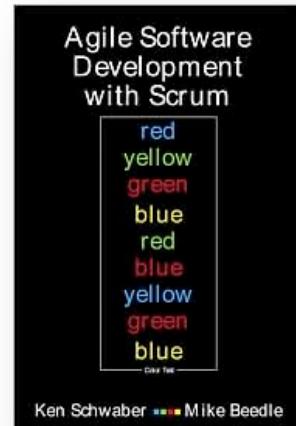
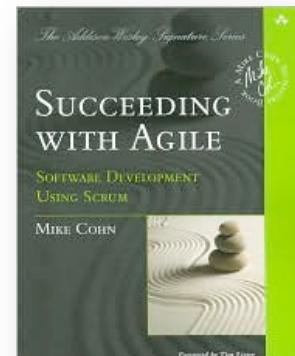
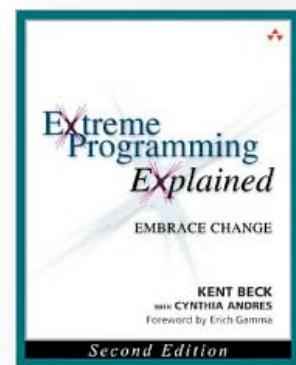
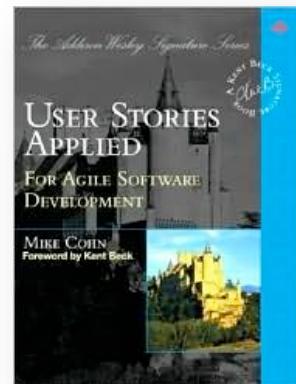
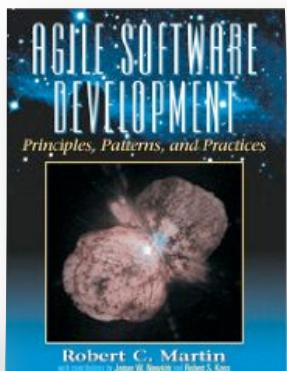
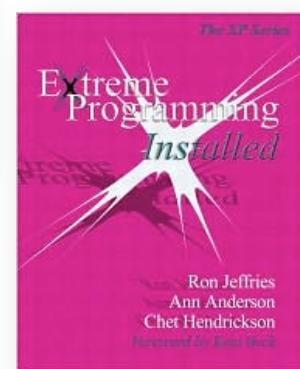


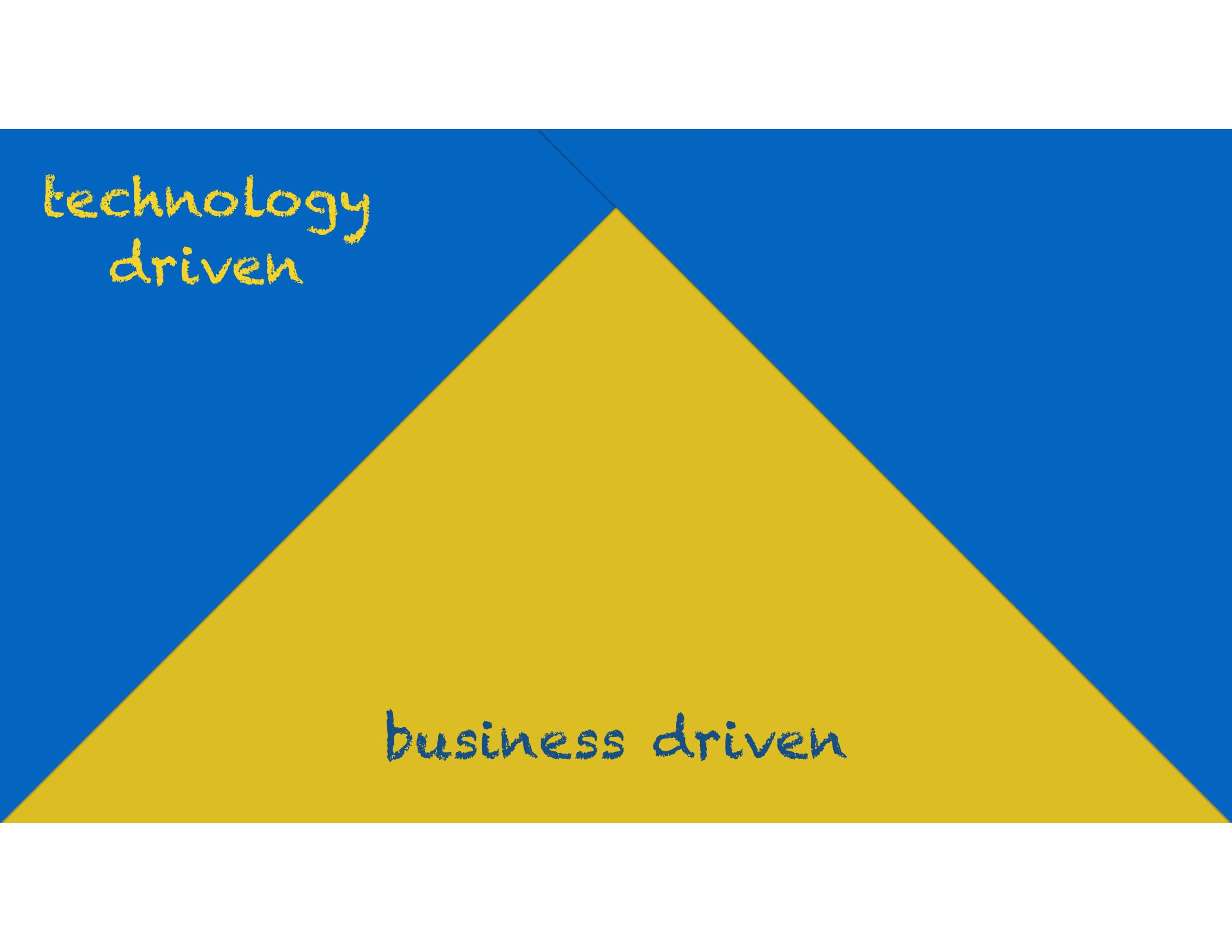
business driven



technology
driven

business driven



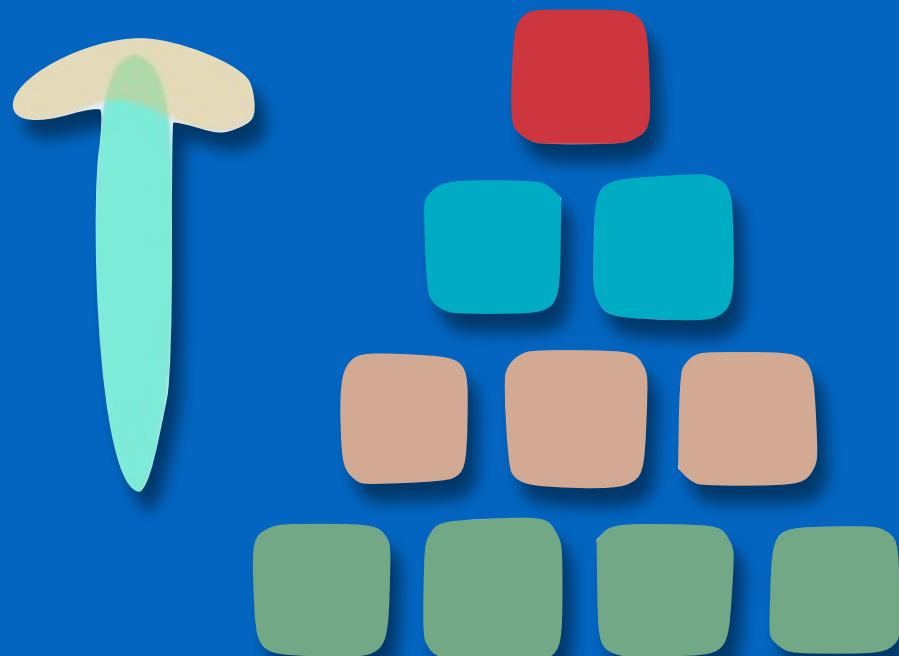


technology
driven

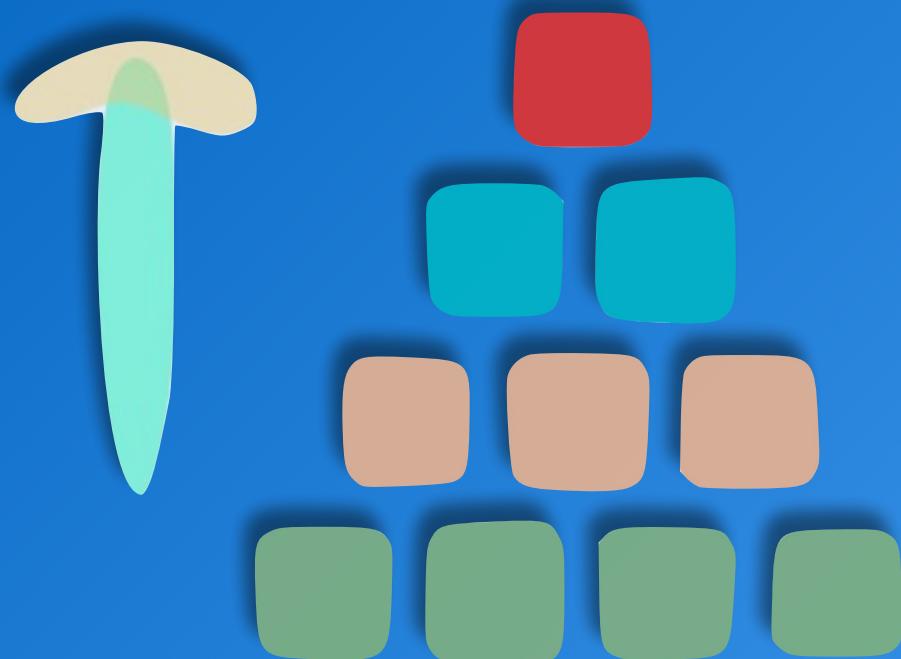
business driven

everything chAnGEs
all the time!

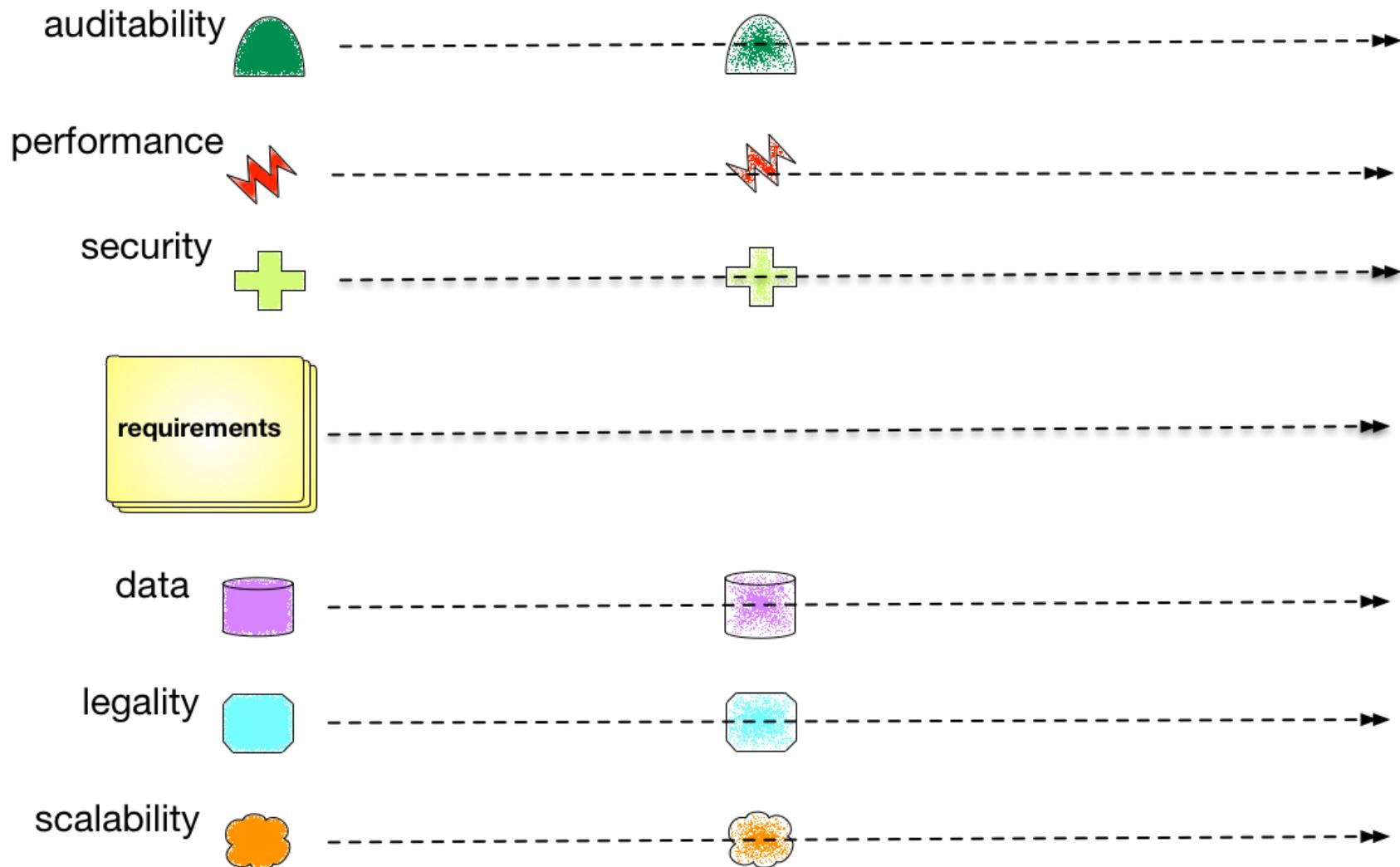
dYNaMic equUiLiBRiuM



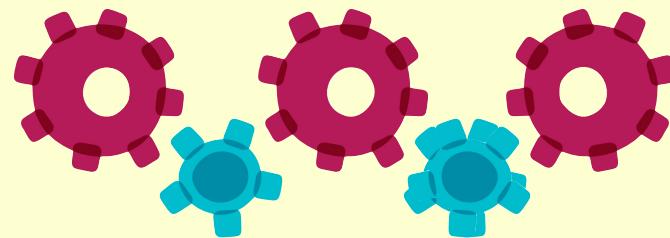
dYNaMic equUiLiBRiuM



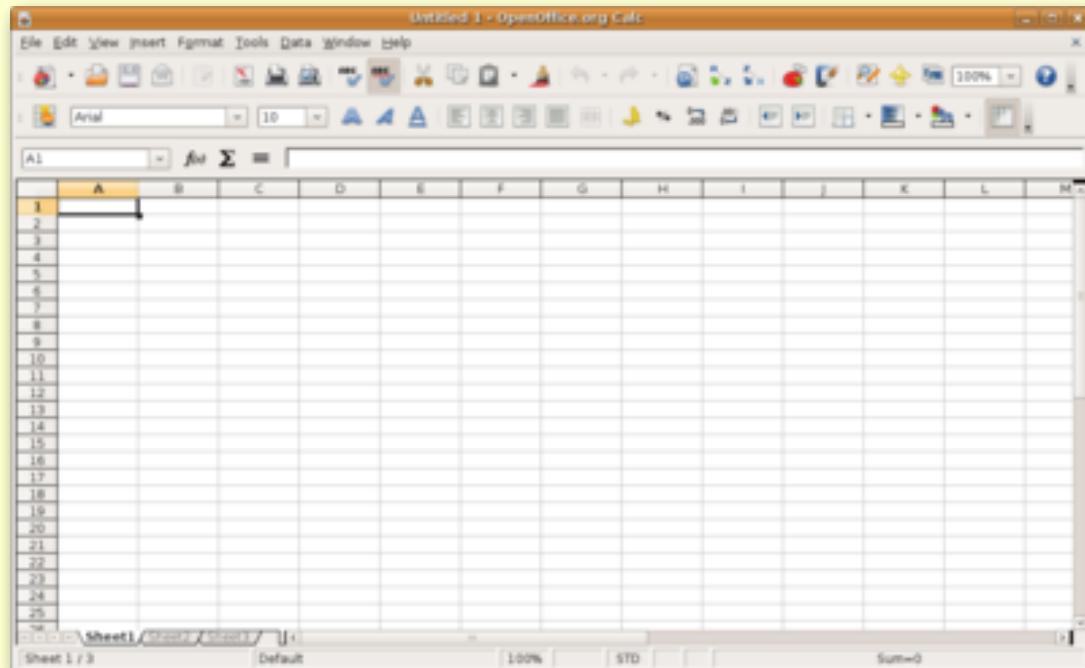
**How is long term planning
possible when things constantly
change in unEXpECtEd ways?**



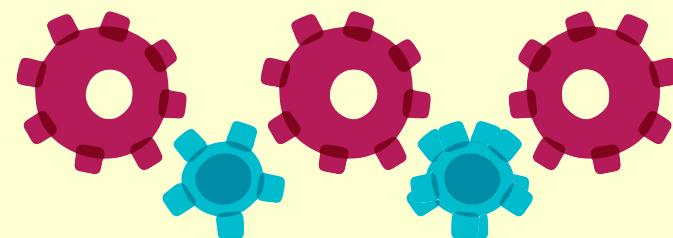
Penultima ↑ e



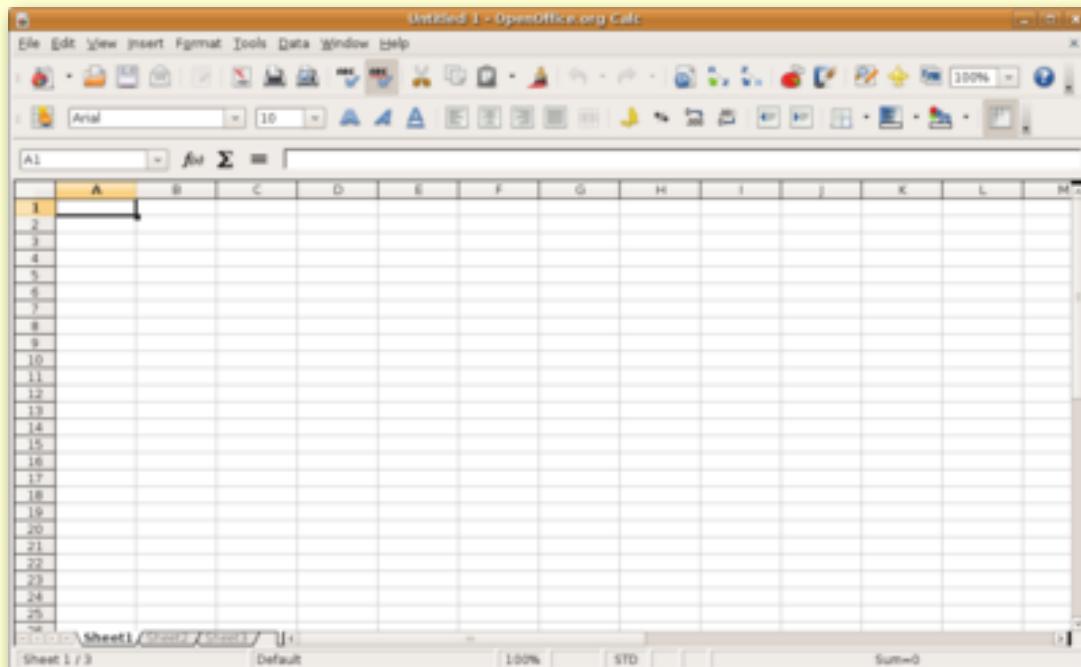
EA Spreadsheet



Penultima ↑ e

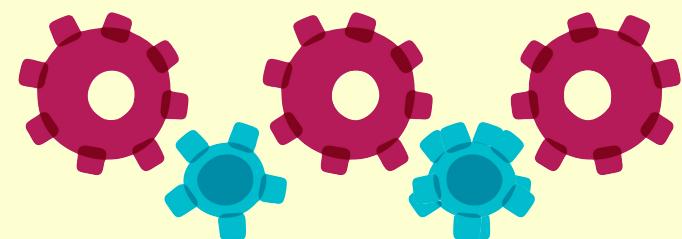


EA Spreadsheet

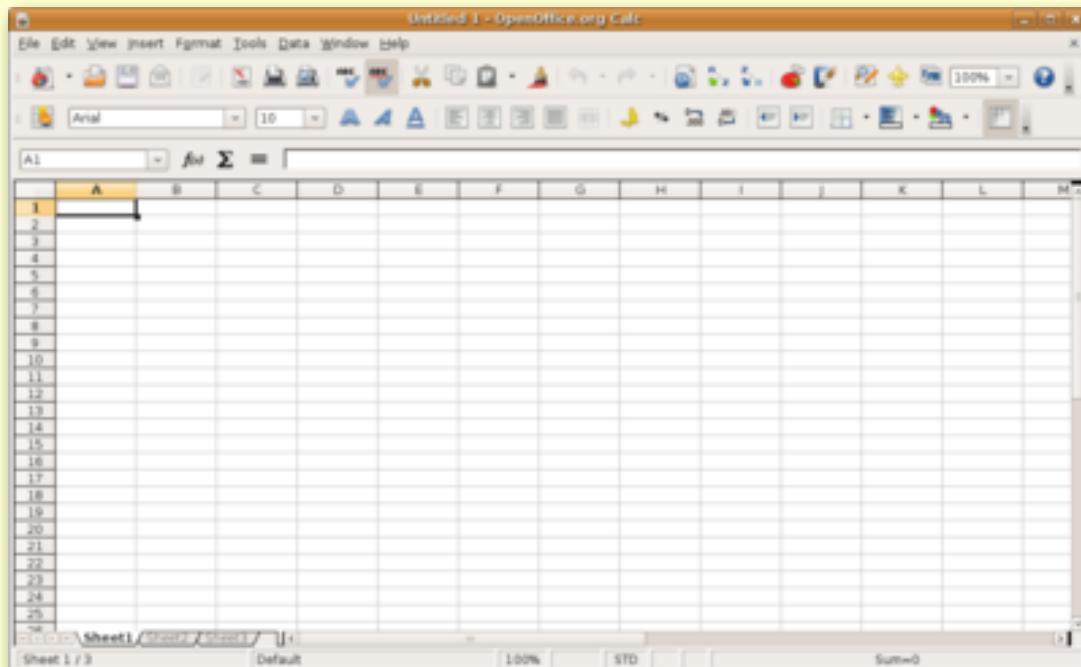


✓ definition

Penultima ↑ e



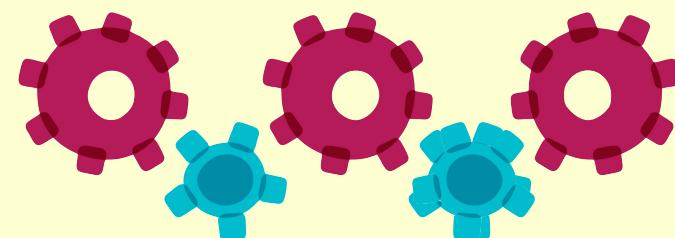
EA Spreadsheet



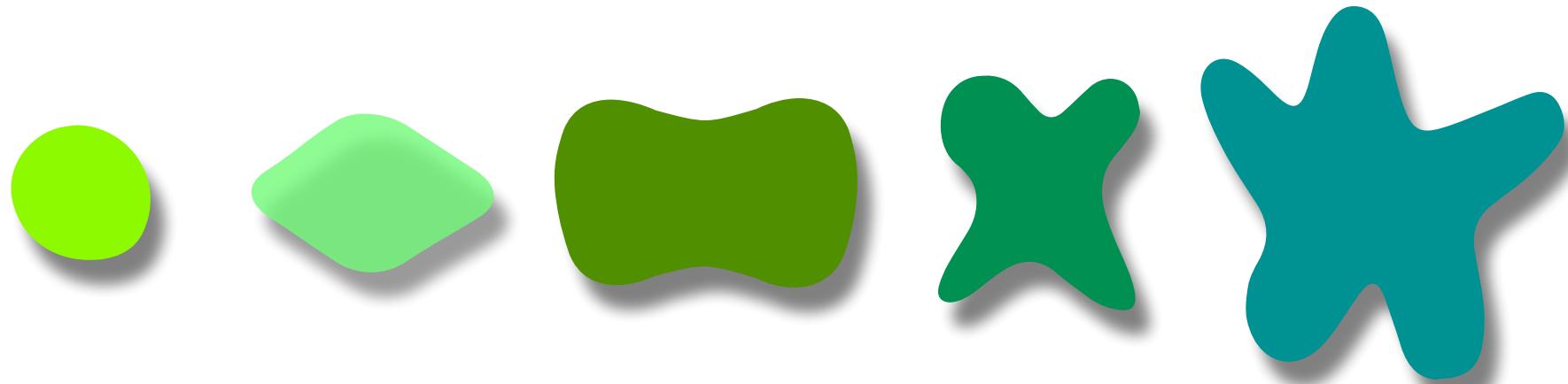
✓ definition

! verification

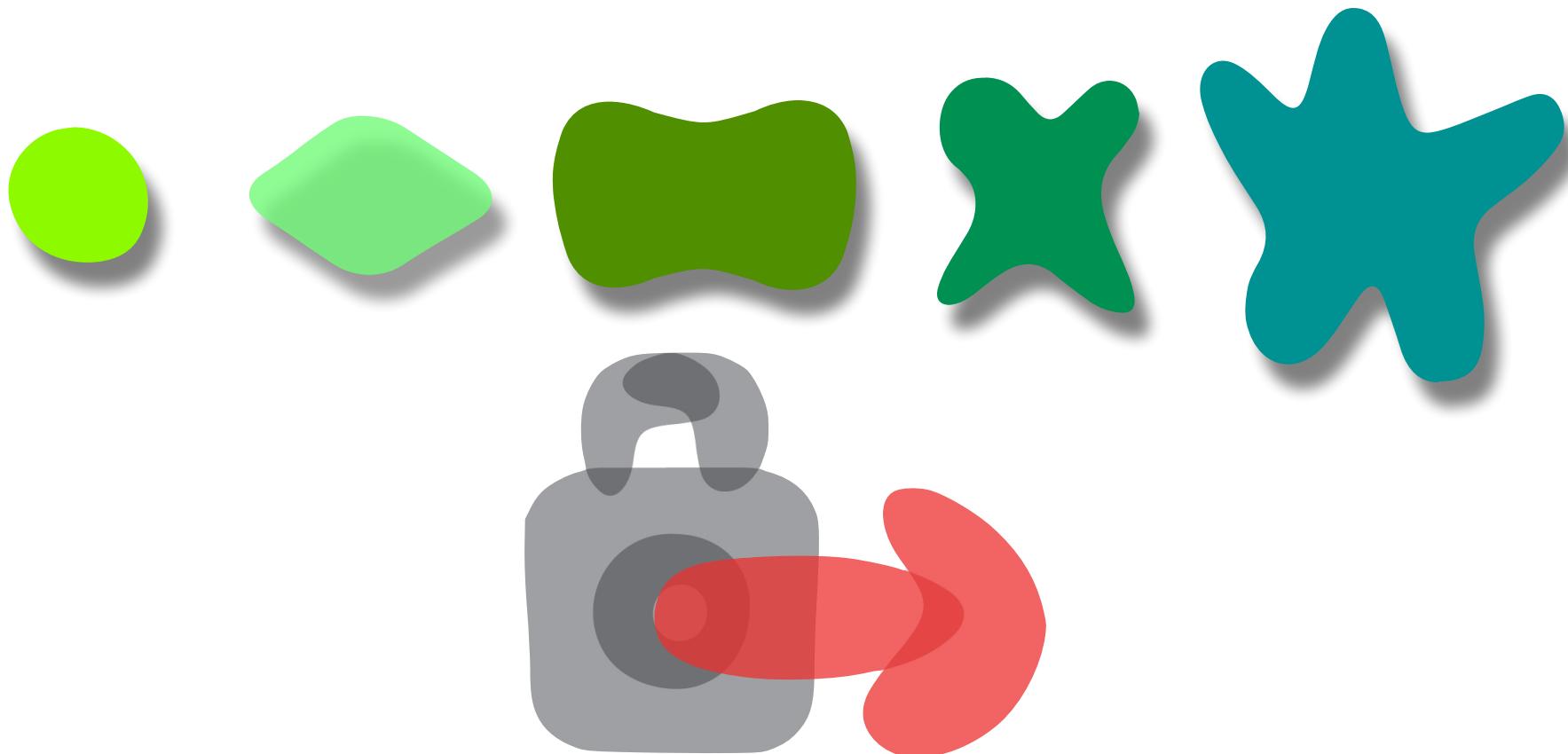
Penultima ↑ e



sEcOND-ORdeR eFfEcT

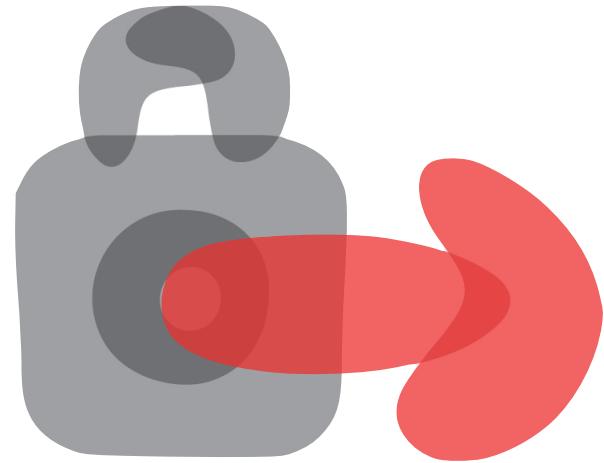


sEcOND-ORdeR eFfEcT



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

governance



Evolutionary Architecture

Continuous Architecture?



Continuous Architecture?

Incremental Architecture?



Continuous Architecture?



Incremental Architecture?



Agile Architecture?



Continuous Architecture?



Incremental Architecture?



Agile Architecture?

Adaptable Architecture?

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

Adaptable Architecture?

Evolutionary 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.

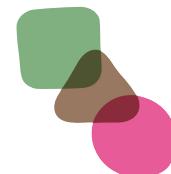




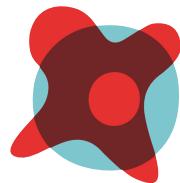
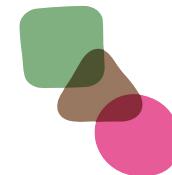
evolutionary computing fitness function:

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

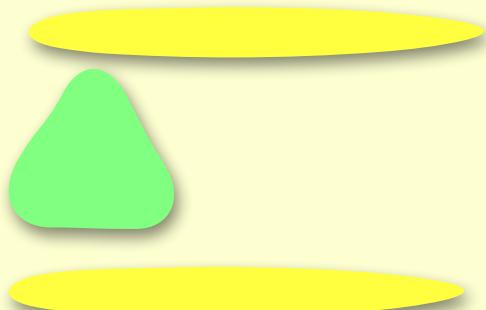




architectural fitness function:

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

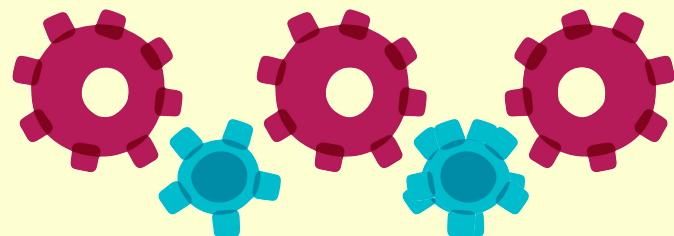
EA Spreadsheet



✓ definition

✓ verification

Penultima ↑ e



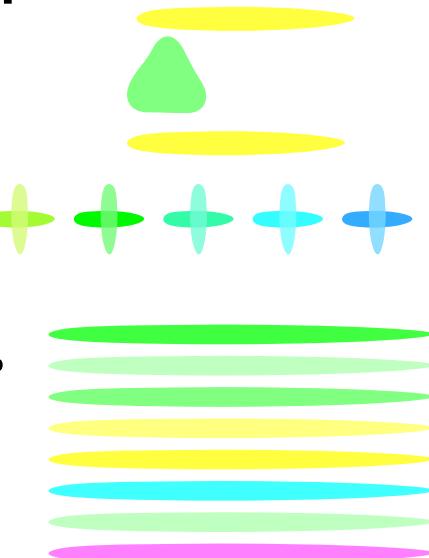
Evolutionary Architecture

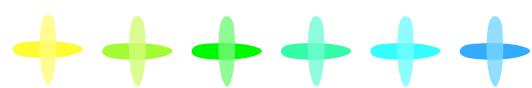
An evolutionary architecture supports
guided
incremental change
across multiple dimensions.



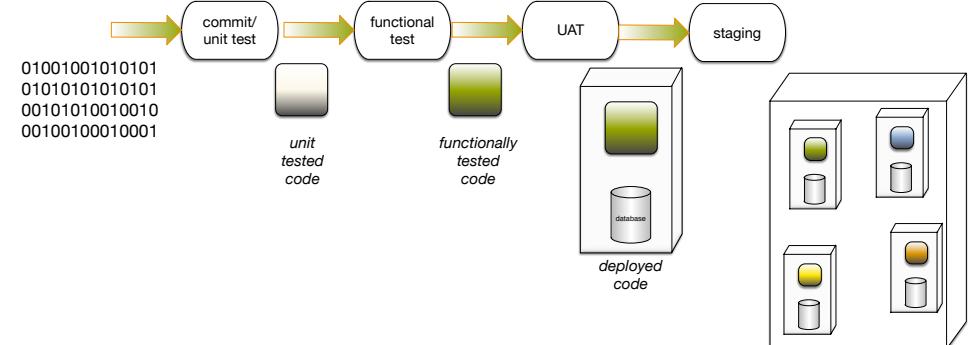
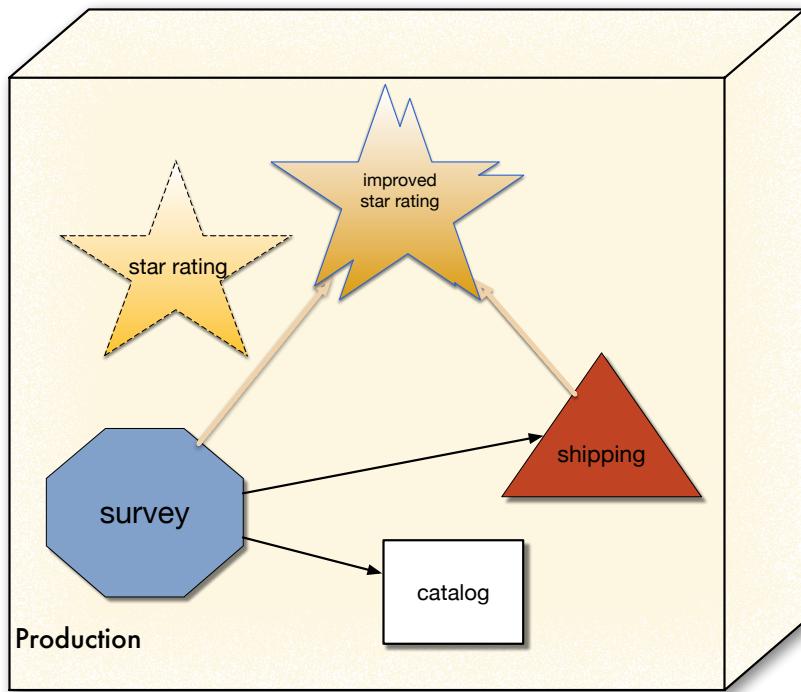
Evolutionary Architecture

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



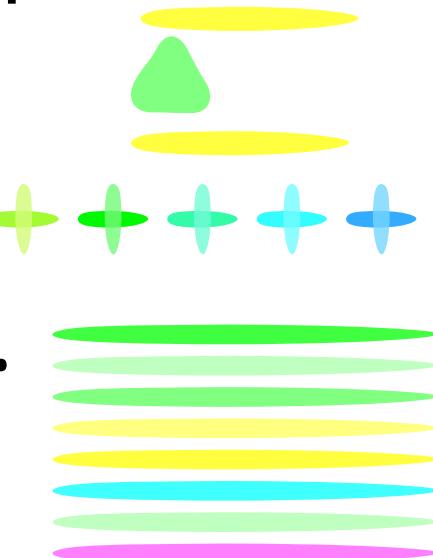


incremental



Evolutionary Architecture

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

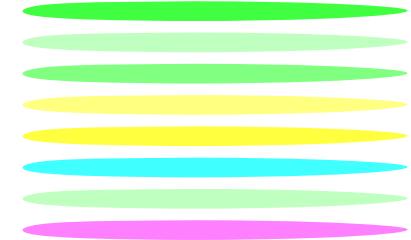


Evolutionary Architecture

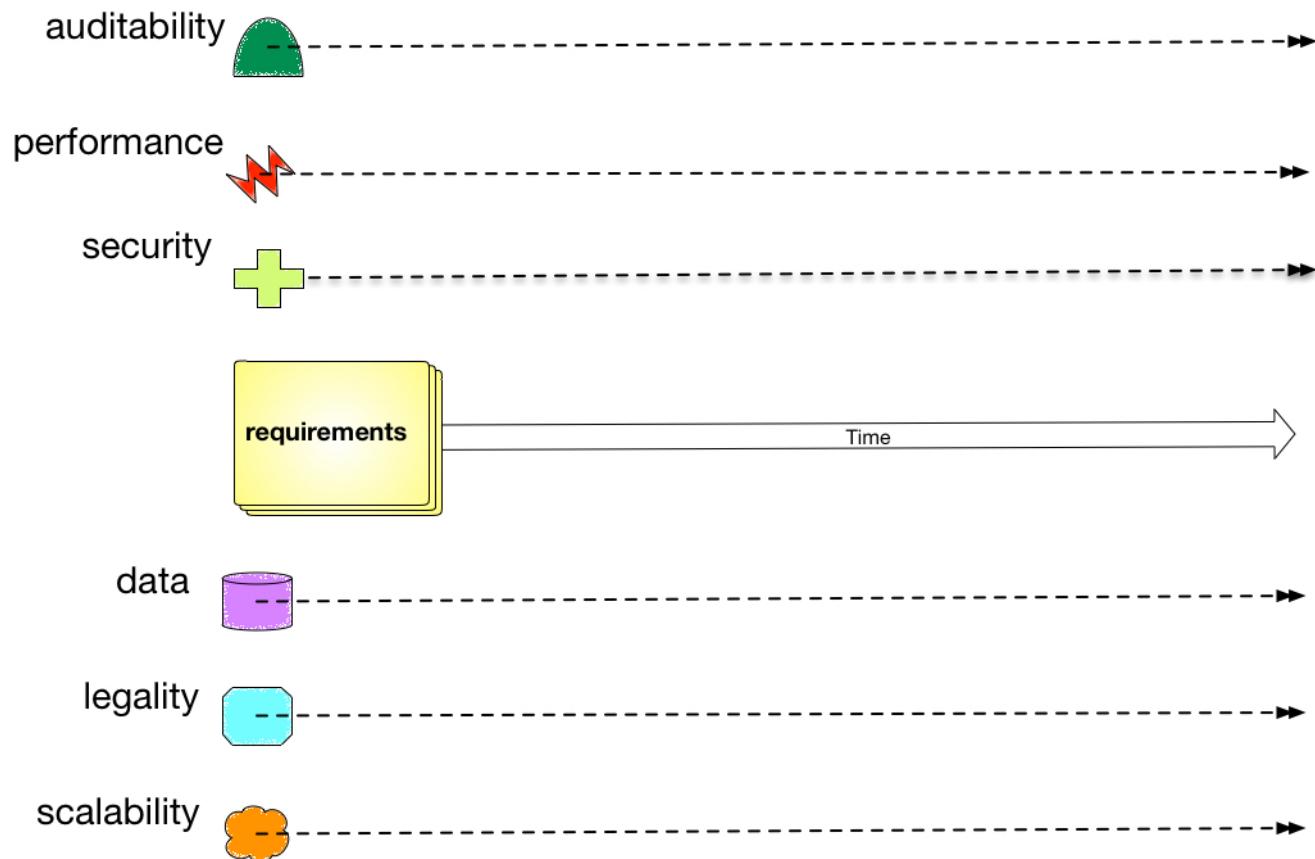
An evolutionary architecture supports
guided,
incremental change

across **multiple dimensions**





multiple dimensions

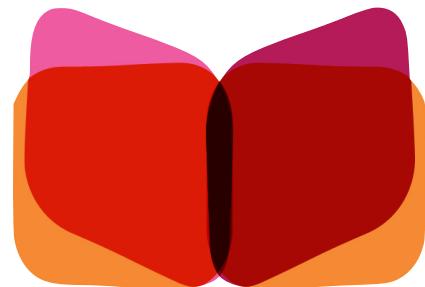


Evolutionary Architecture

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



Agenda Part 1



Definition



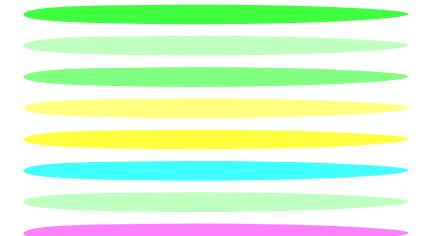
Guided Change via Fitness Functions



Structuring architecture for evolution

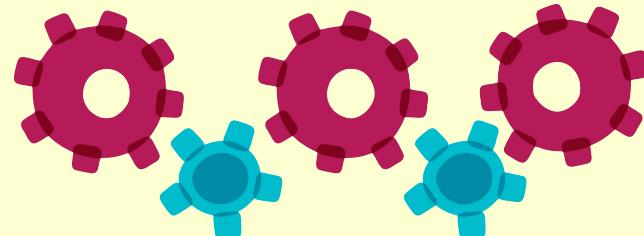


Incremental Change



Agenda Part 2

The Evolution of
Penultima↑e



Agenda Part 2

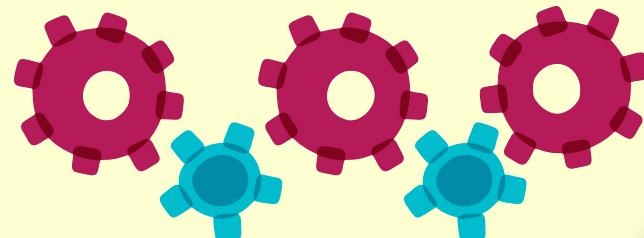
fitness functions

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.

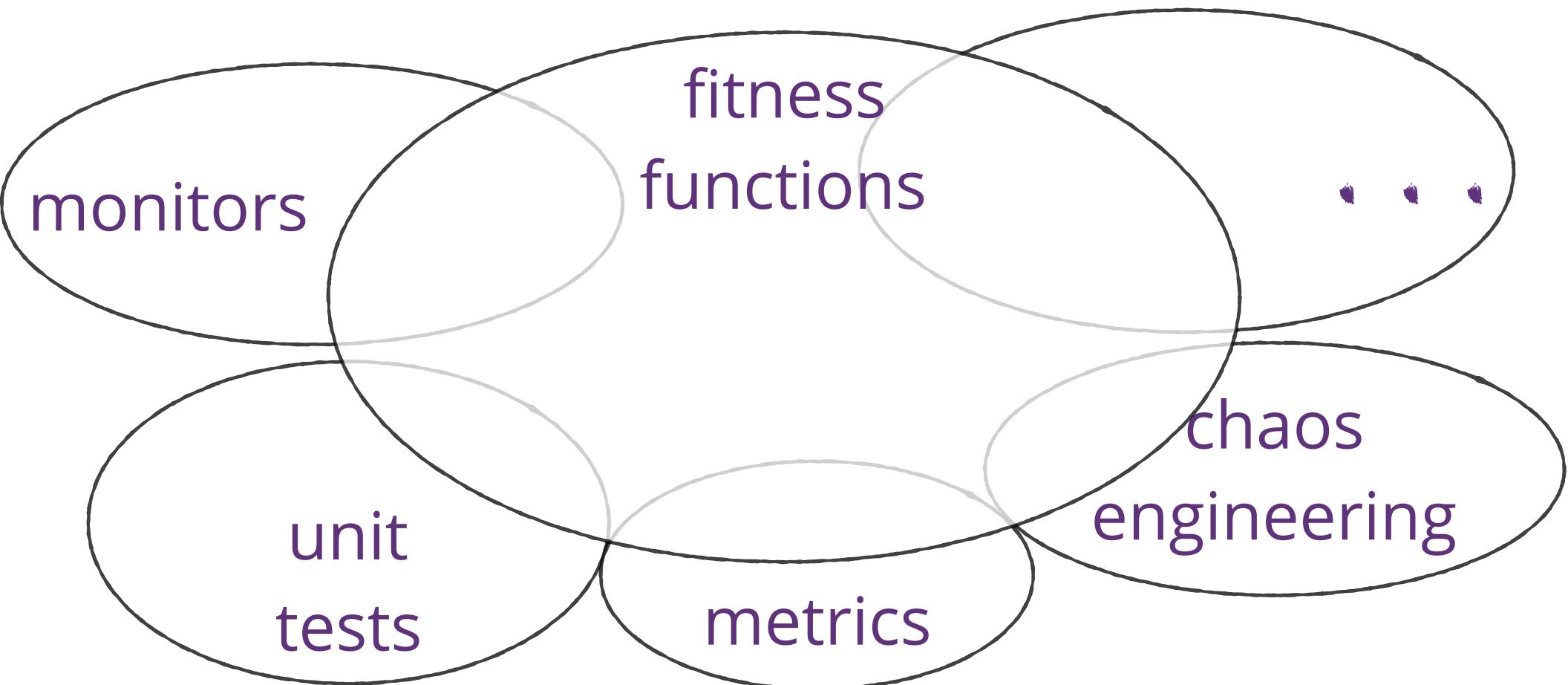




architectural fitness function:

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

Fitness Functions



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
 - deployment pipeline running tests
 - timed task

Categories of Fitness Functions

triggered



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

continuous



executes constant verification of architectural aspect(s)

Categories of Fitness Functions

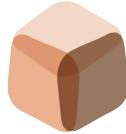
static



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

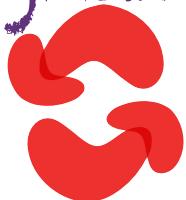
Categories of Fitness Functions

static



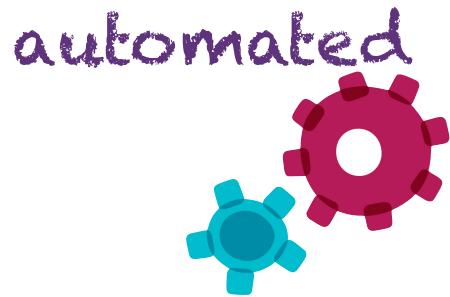
have a fixed result, such as the binary pass/fail of a unit test.

dynamic



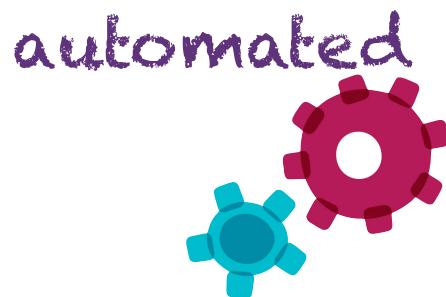
rely on a shifting definition based on extra context.

Categories of Fitness Functions



tests and other verification mechanism that run without human interaction.

Categories of Fitness Functions



tests and other verification mechanism that run without human interaction.



must involve at least one human.

Categories of Fitness Functions

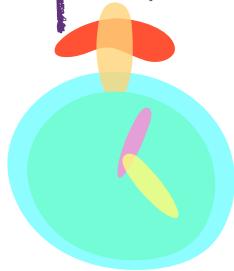
temporal



architects may want to build a time component into assessing fitness

Categories of Fitness Functions

temporal

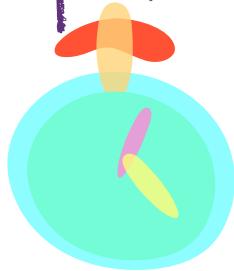


architects may want to build a time component into assessing fitness

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

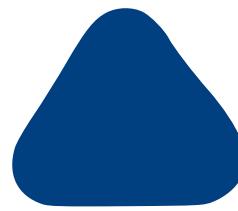
architectural
characteristic

domain-specific



problem domain

unit



functional



UAT

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



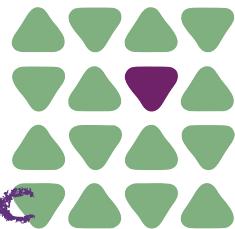
architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...



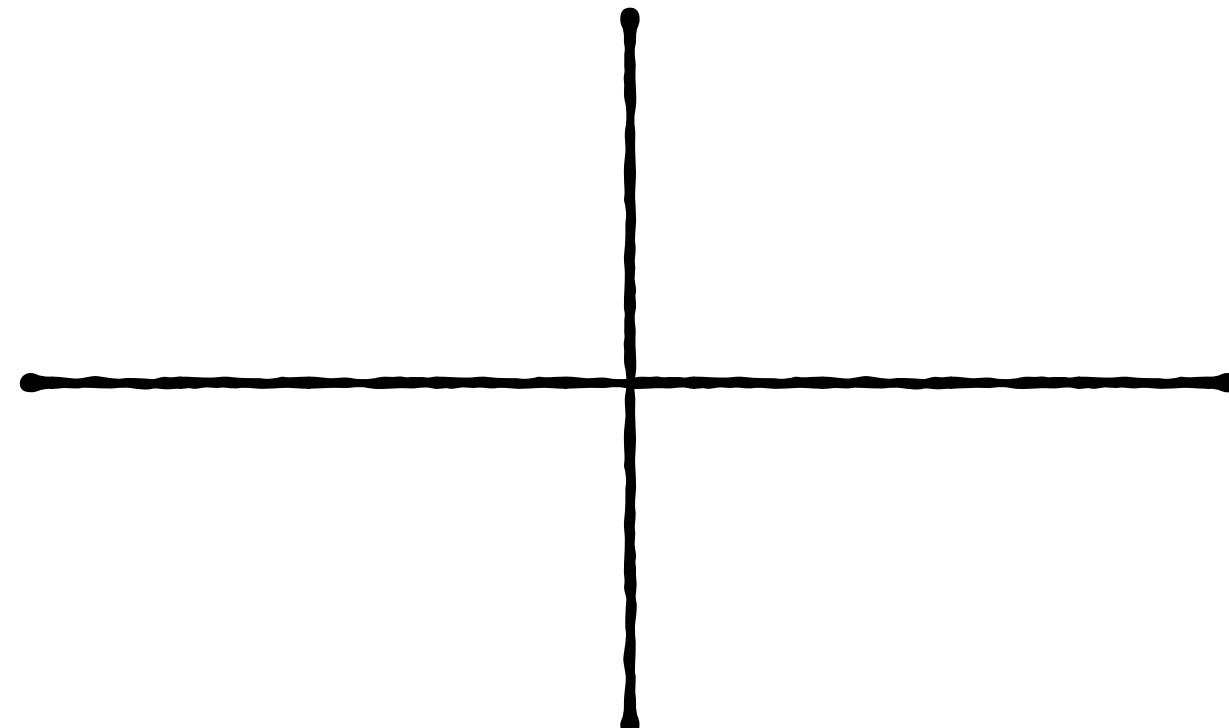
...some fitness functions will emerge during development of the system

Fitness Function

atomic



holistic



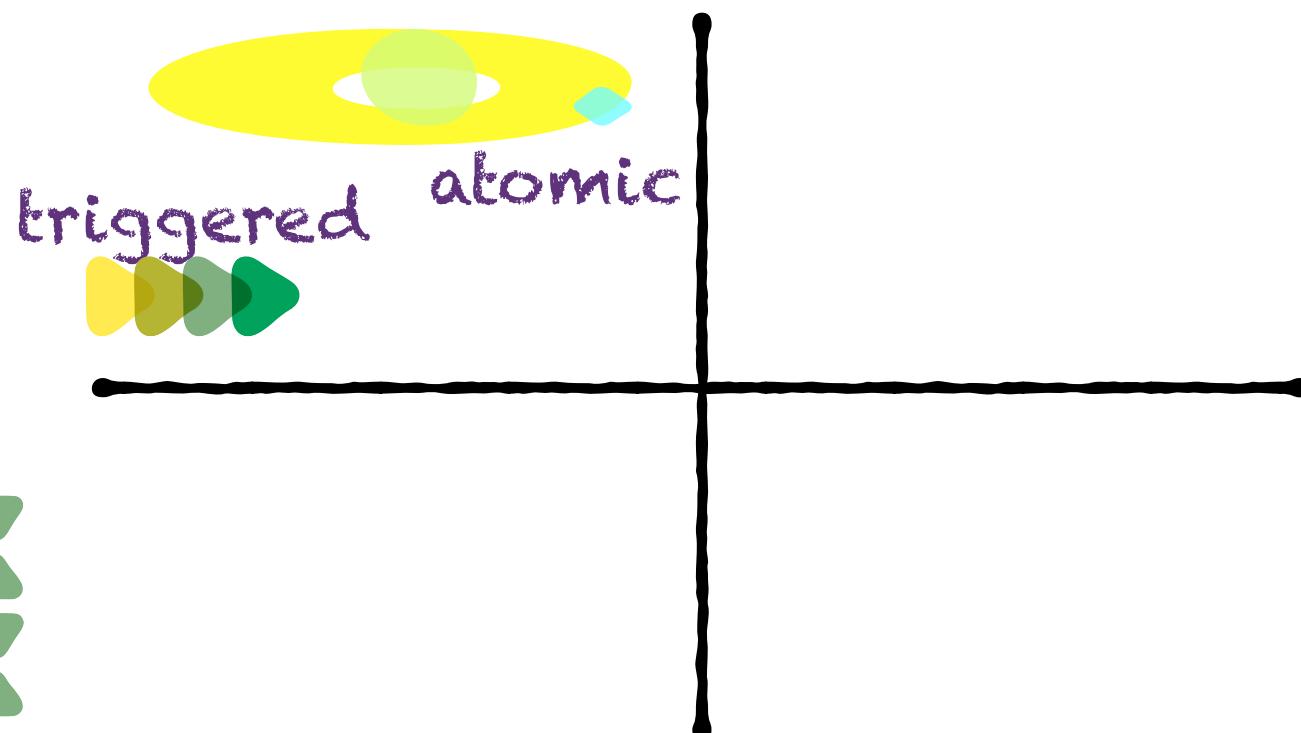
triggered



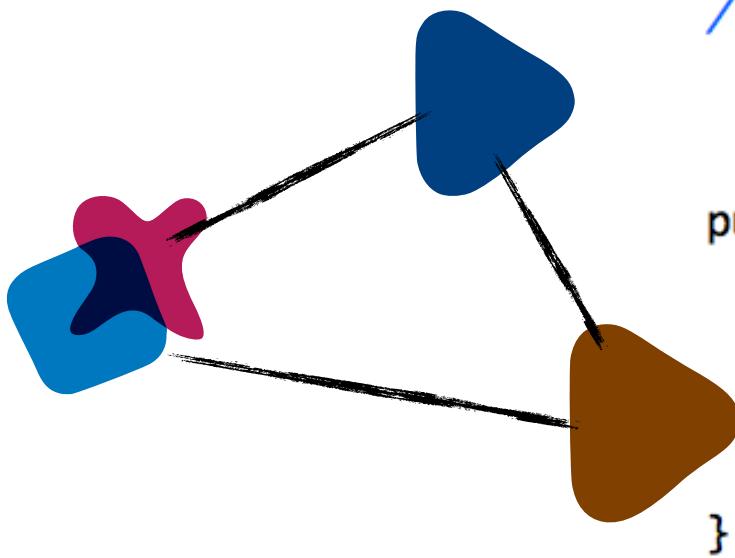
continuous



Fitness Function



Cyclic Dependency Function

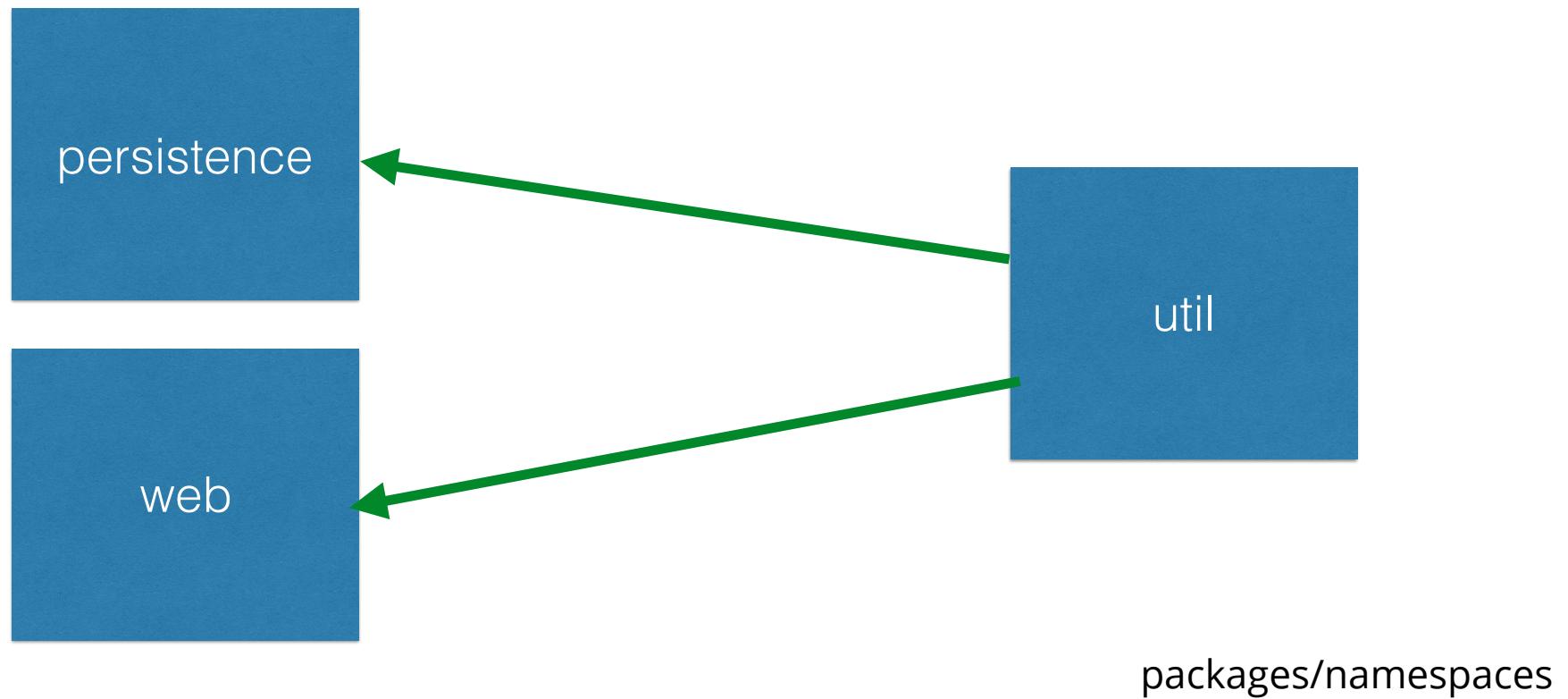


```
/**  
 * Tests that a package dependency cycle does not  
 * exist for any of the analyzed packages.  
 */  
public void testAllPackages() {  
  
    Collection packages = jdepend.analyze();  
  
    assertEquals("Cycles exist",  
                false, jdepend.containsCycles());  
}
```

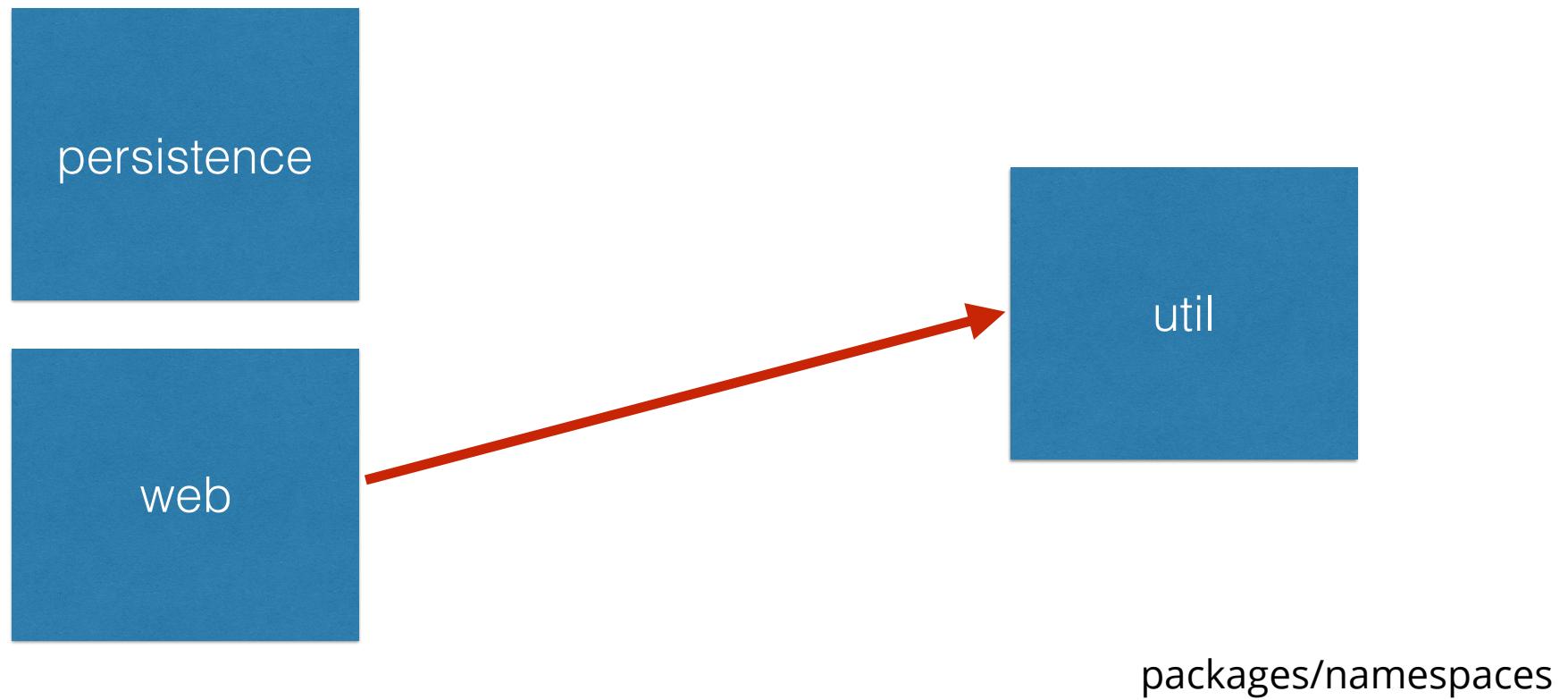
clarkware.com/software/JDepend.html



Directionality of Imports



Directionality of Imports



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");

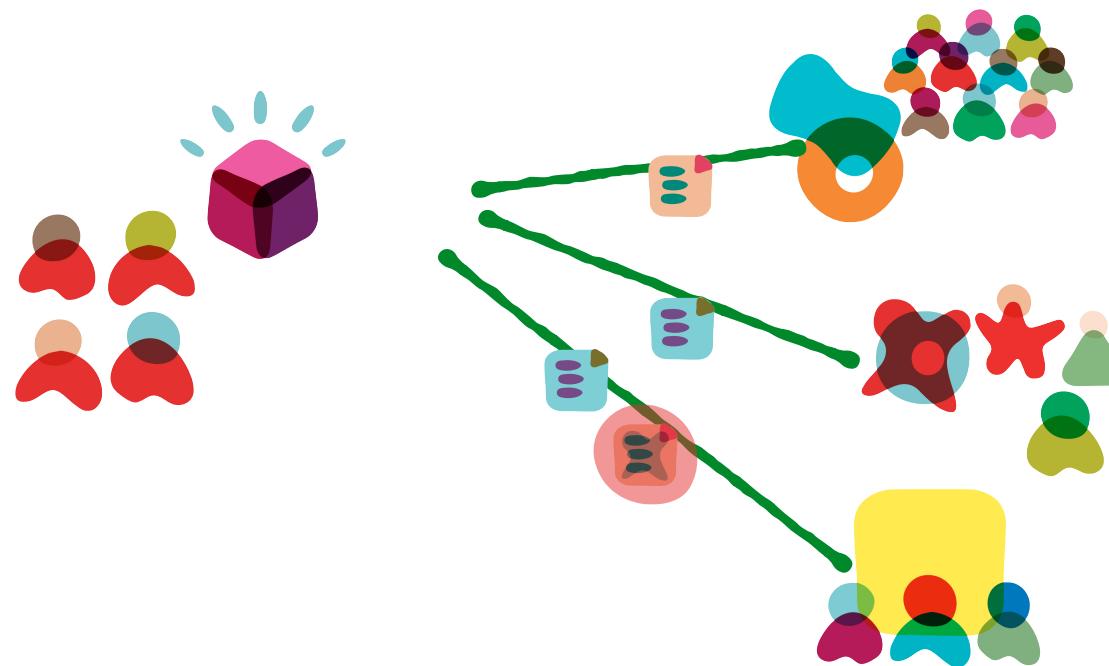
    persistence.dependsUpon(util);
    web.dependsUpon(util);

    jdepend.analyze();

    assertEquals("Dependency mismatch",
                true, jdepend.dependencyMatch(constraint));
}
```

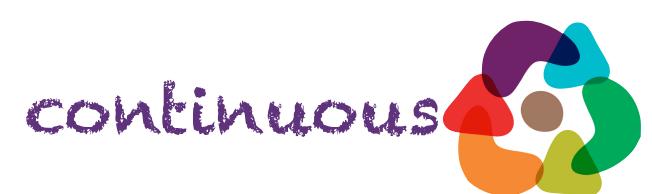
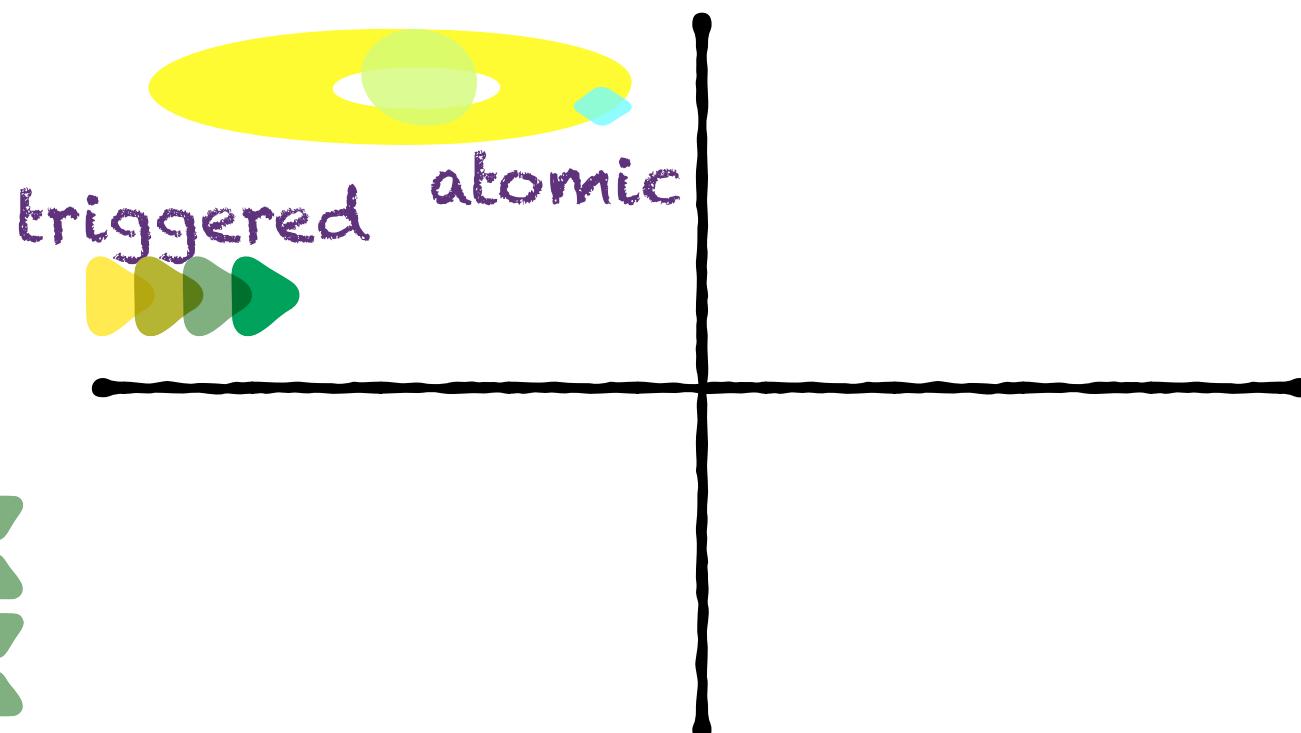


Consumer Driven Contracts



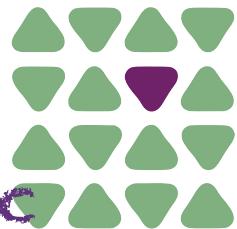
martinfowler.com/articles/consumerDrivenContracts.html

Fitness Function

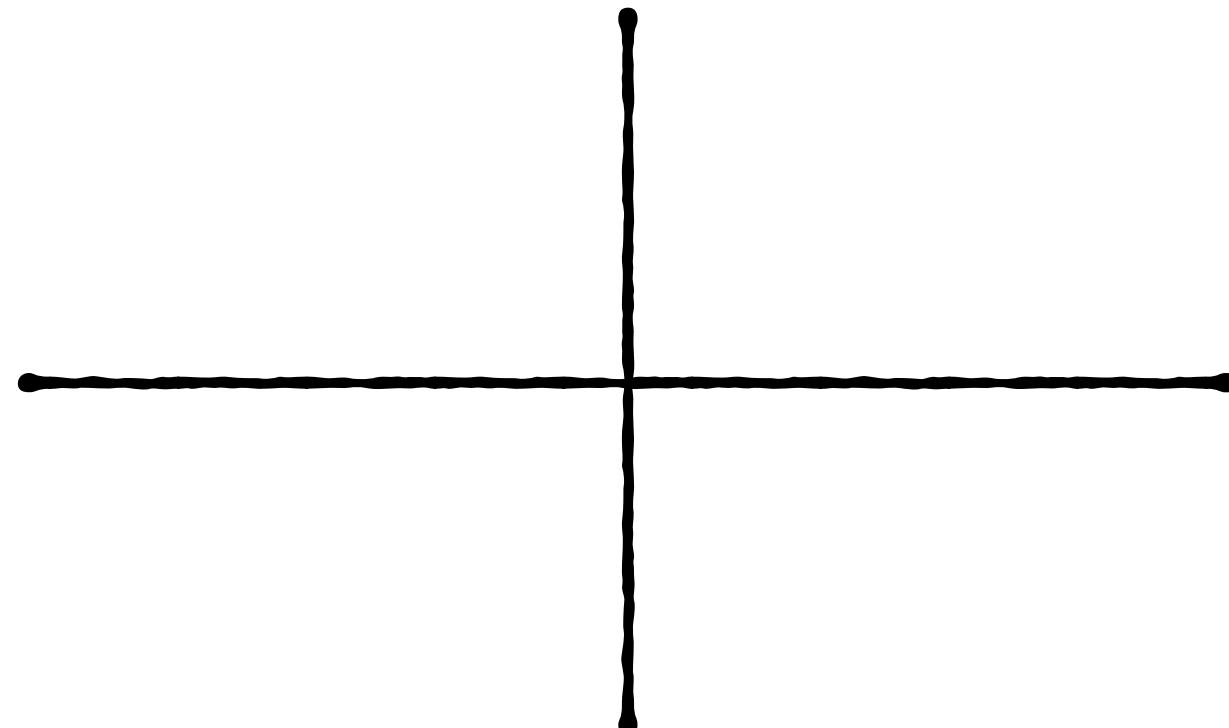


Fitness Function

atomic



holistic



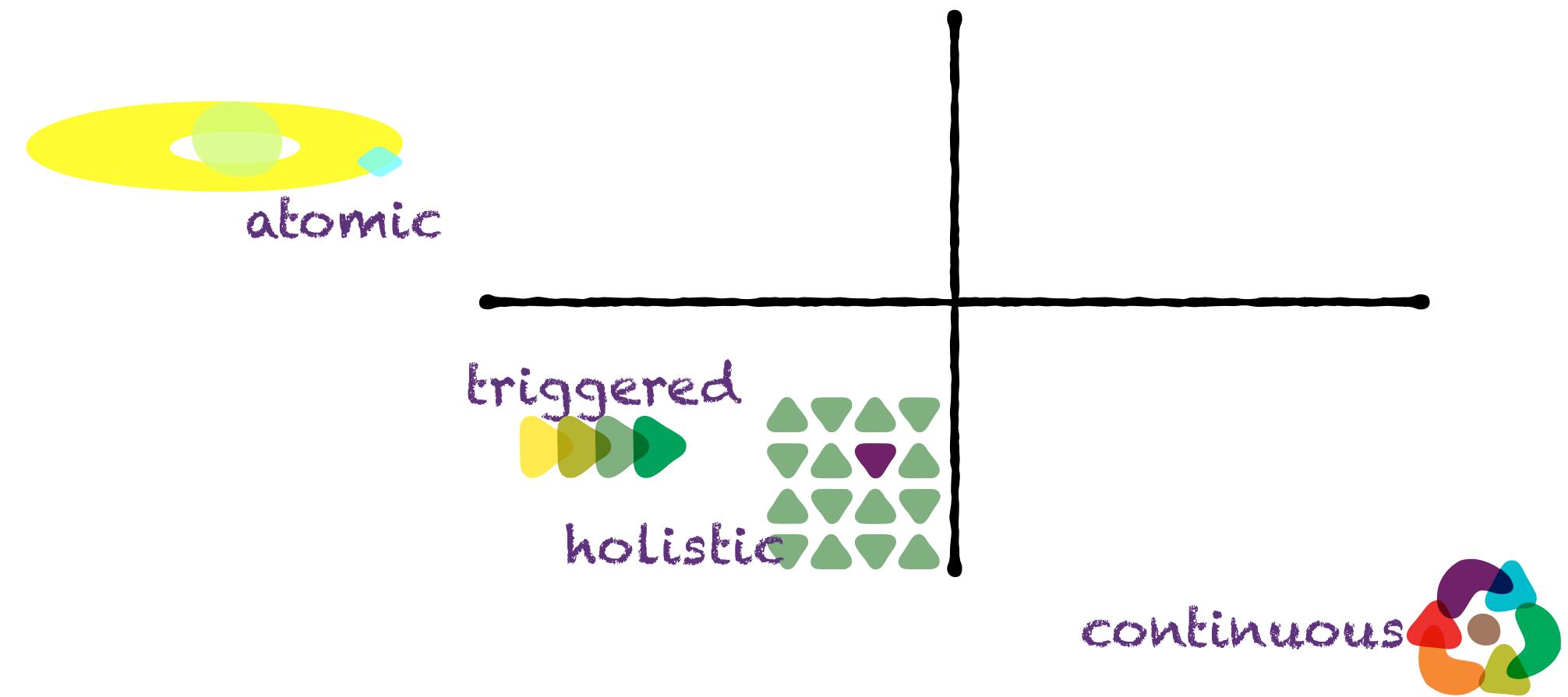
triggered



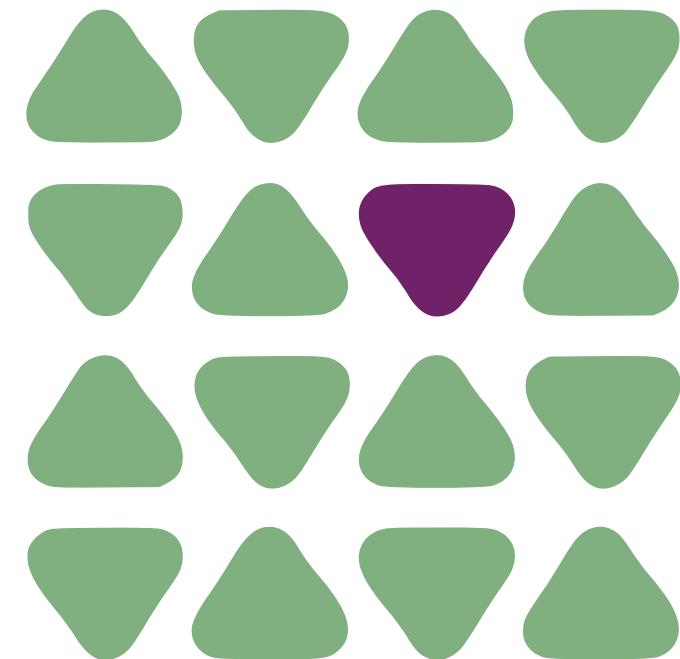
continuous



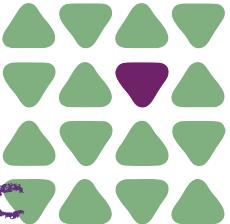
Fitness Function

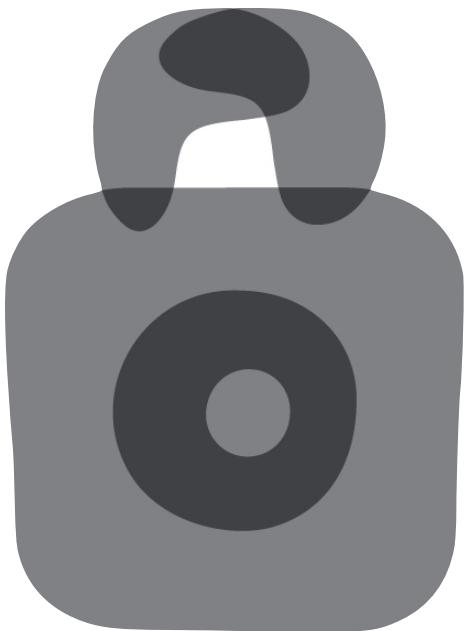


triggered

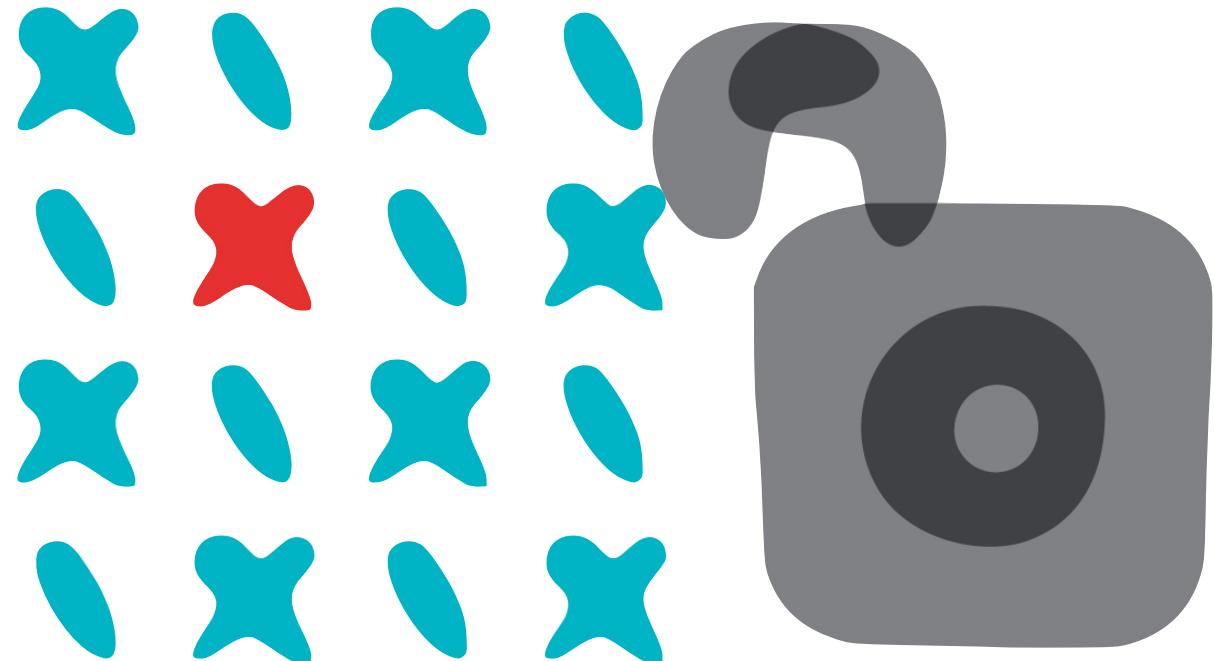


holistic

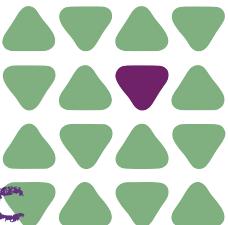




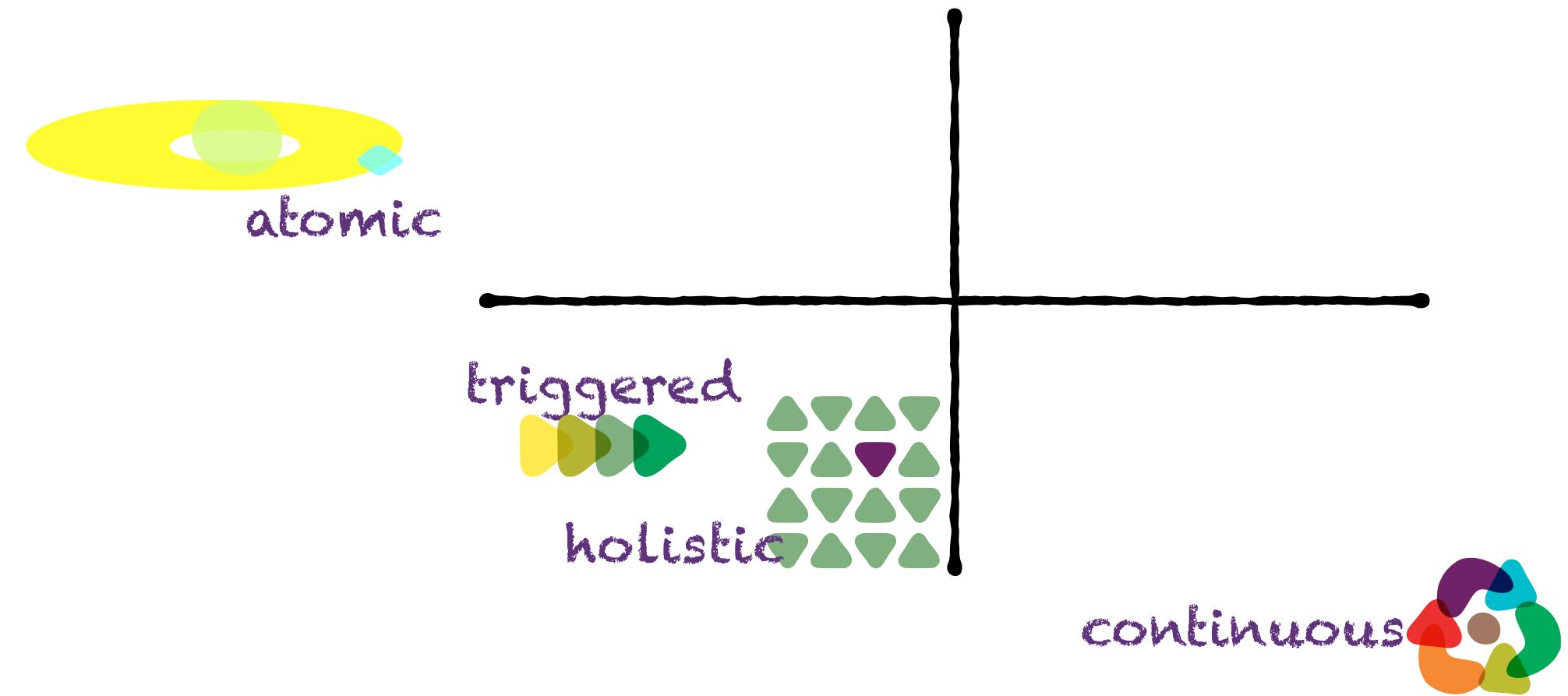
triggered



holistic

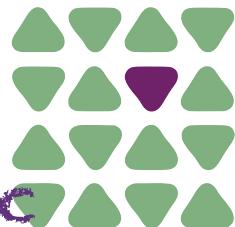


Fitness Function

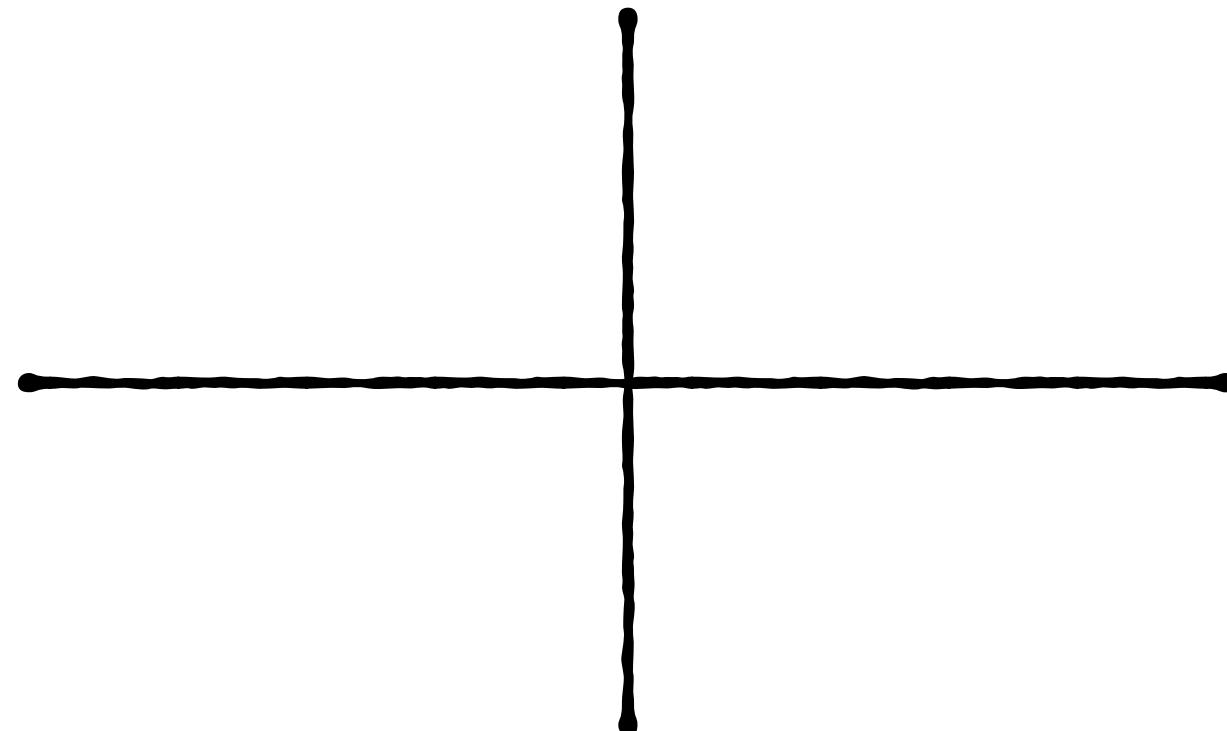


Fitness Function

atomic



holistic



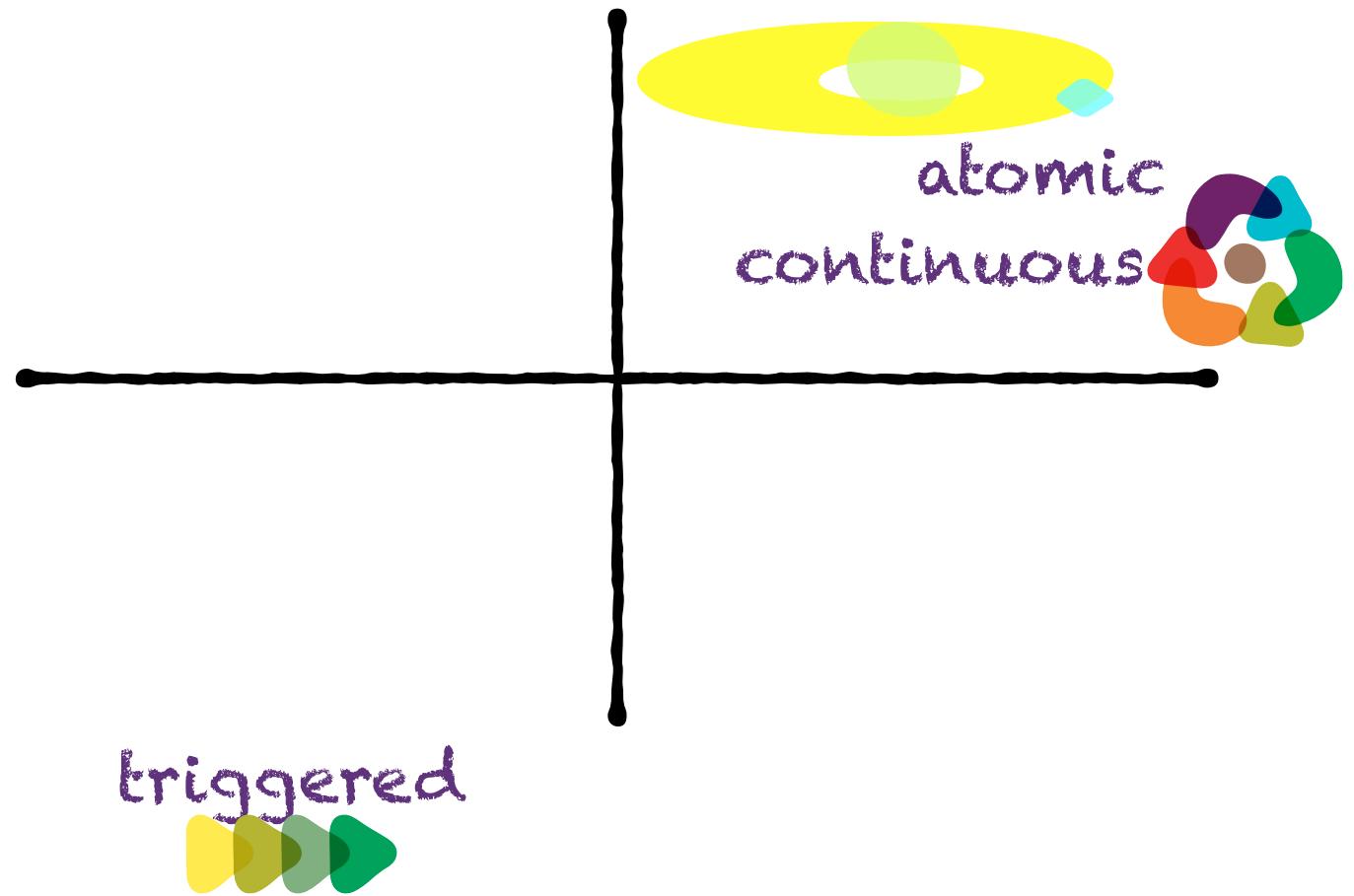
triggered



continuous



Fitness Function

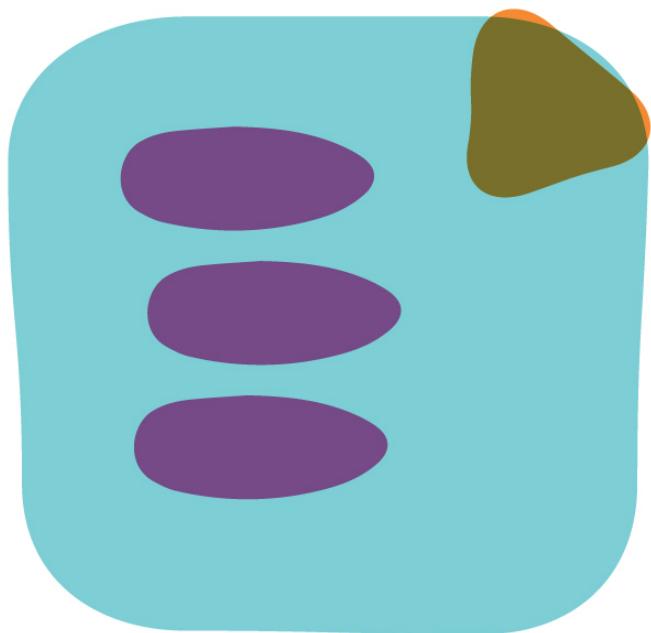
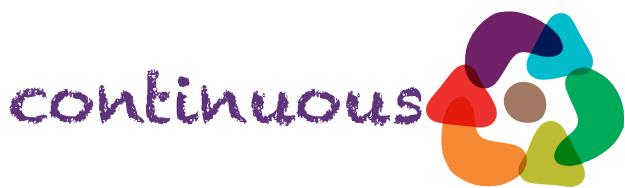




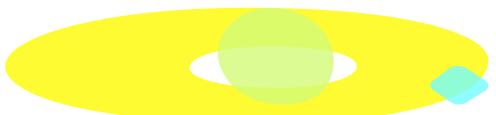
atomic



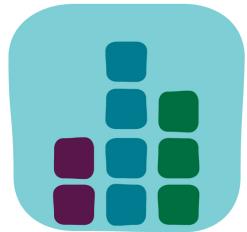
monitoring



logging



atomic



monitoring

Nagios XI

Home Views Dashboards Reports Configure Tools Help Admin

Quick View

- Host Status Summary
- Service Status Summary
- Hostgroup Status Summary
- Status Summary For All Host Groups

Details

- Service Detail
- Host Detail
- Hostgroup Summary
- Hostgroup Overview
- Hostgroup Grid
- Servicegroup Summary
- Servicegroup Overview
- Servicegroup Grid
- BPI
- Metrics

Graphs

- Performance Graphs
- Graph Explorer

Maps

- BMap
- Google Map
- HyperMap
- Minimap
- Nagvis
- Network Status Map
- Legacy Network Status Map

Incident Management

- Latest Alerts
- Acknowledgements
- Scheduled Downtime
- Mass Acknowledge
- Recurring Downtime
- Notifications

Monitoring Process

- Process Info
- Performance
- Event Log

Host Status Summary

Up	Down	Unreachable	Pending
53	61	3	0
Unhandled	Problems	All	
64	64	117	

Last Updated: 2017-10-05 16:06:57

Service Status Summary

Ok	Warning	Unknown	Critical	Pending
226	12	64	371	2
Unhandled	Problems	All		
366	367	595		

Last Updated: 2017-10-05 16:06:57

Top Alert Producers Last 24 Hours

Metrics Overview

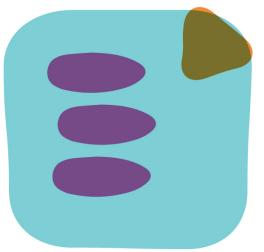
Disk Usage

Host	Service	% Utilization	Details
localhost	Root Partition	78.67%	DISK WARNING - free space: / 1207 MB (17% inode=68%);
vs1.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%);

Last Updated: 2017-10-05 16:06:58

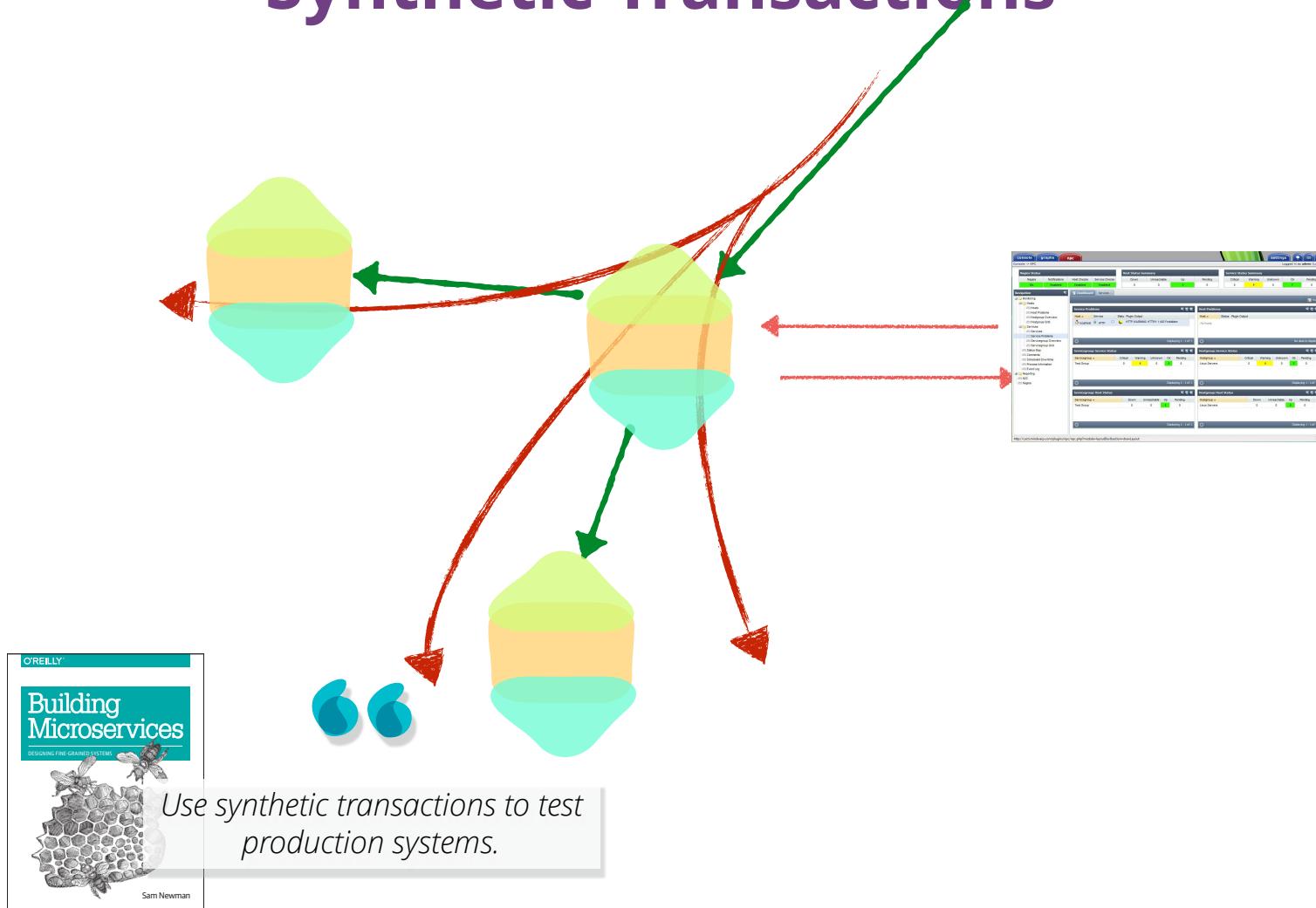
Nagios XI 5.4.10 • Check for Updates

About | Legal | Copyright © 2008-2017 Nagios Enterprises, LLC

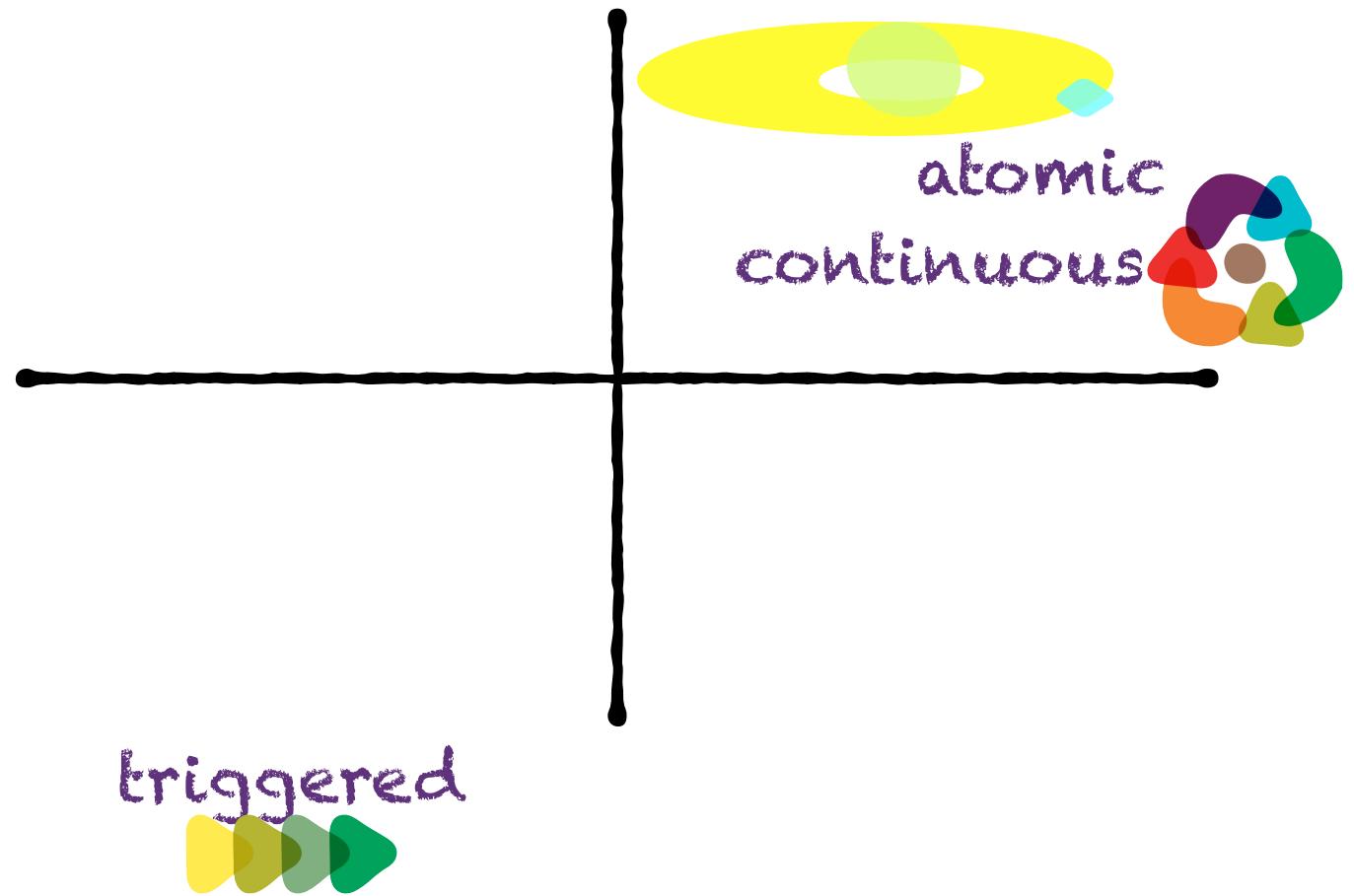


logging

Synthetic Transactions

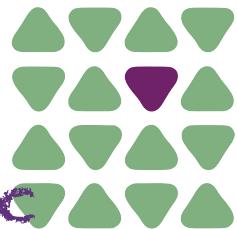


Fitness Function

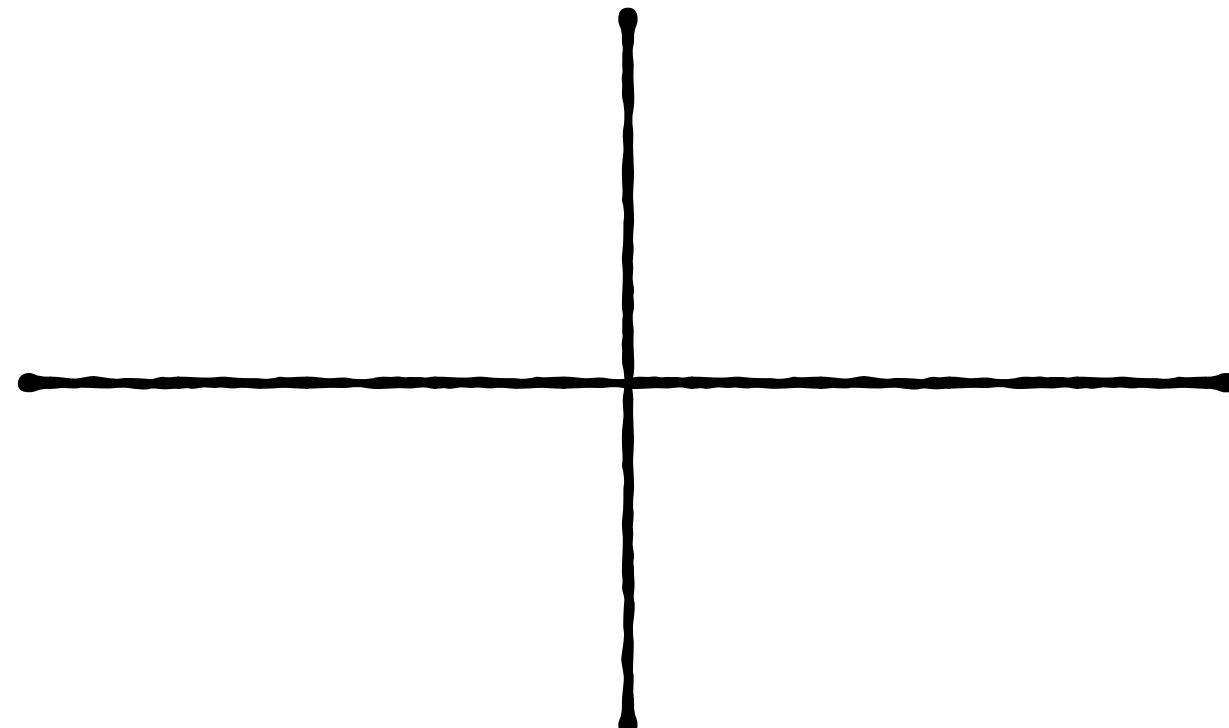


Fitness Function

atomic



holistic



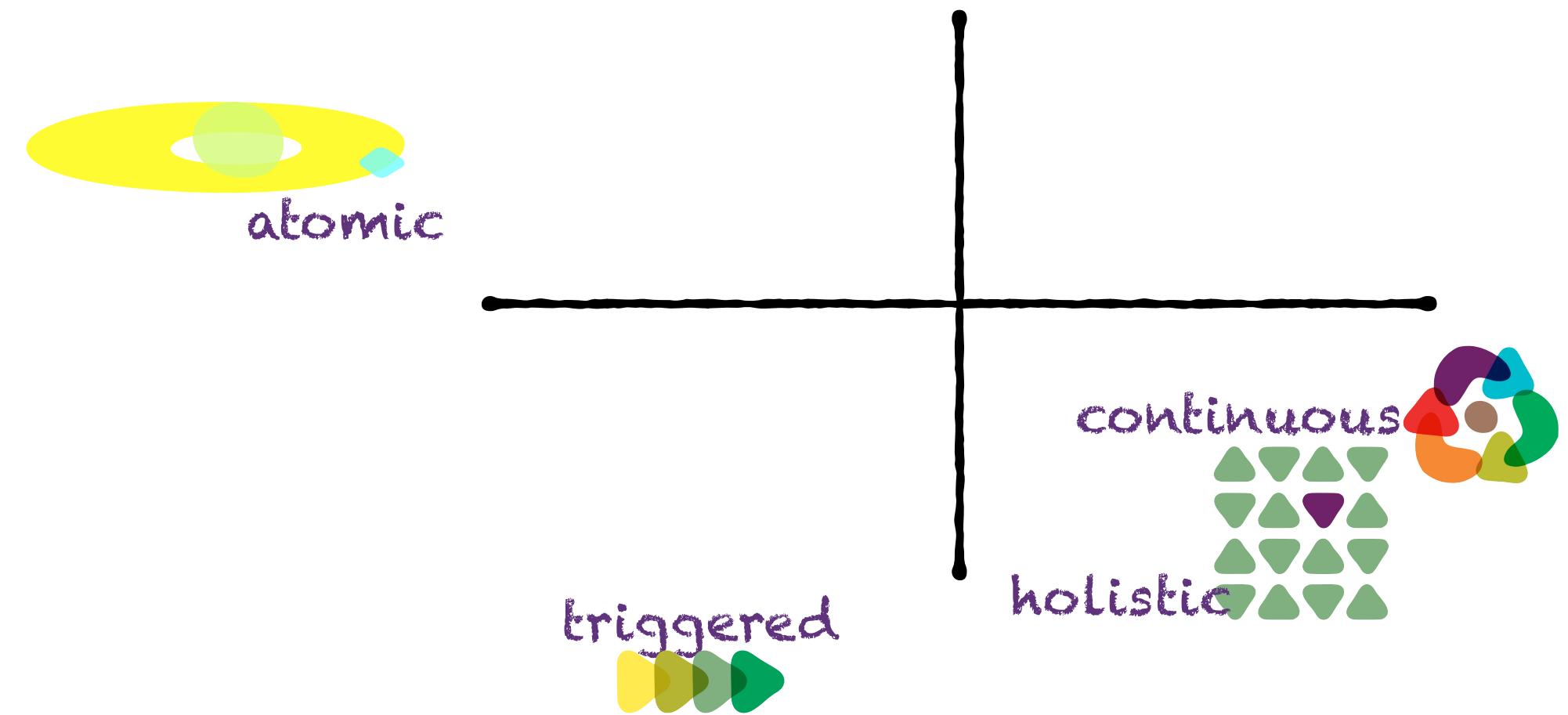
triggered



continuous



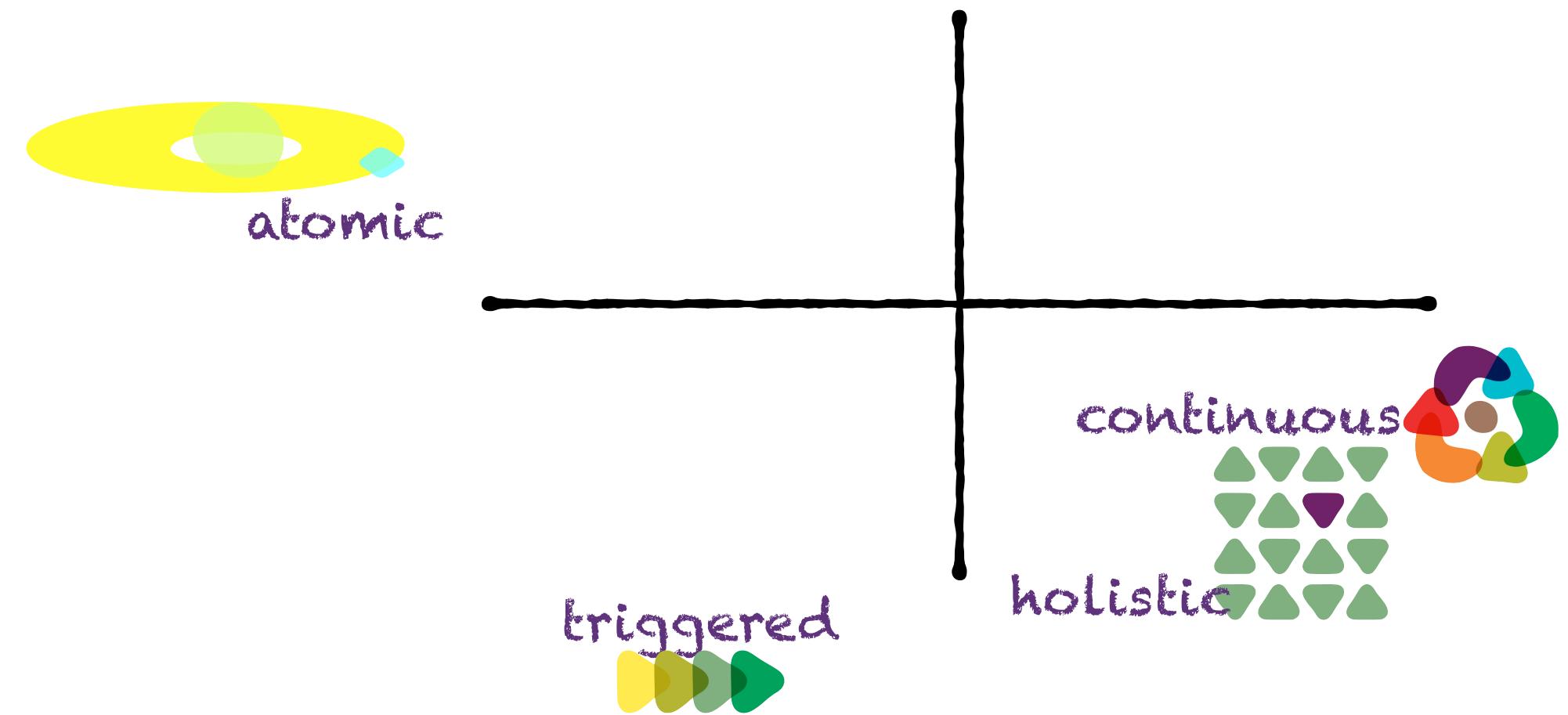
Fitness Function





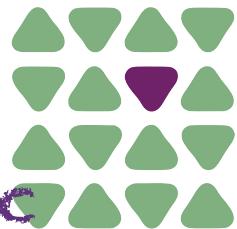


Fitness Function

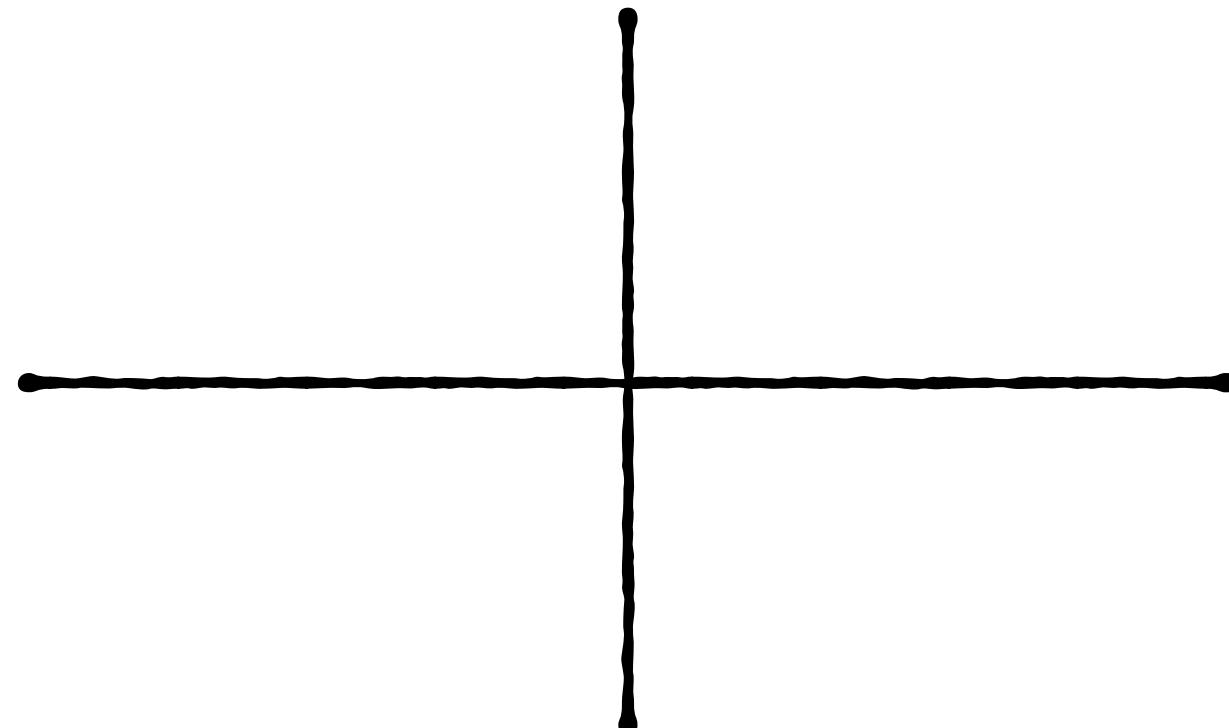


Fitness Function

atomic



holistic



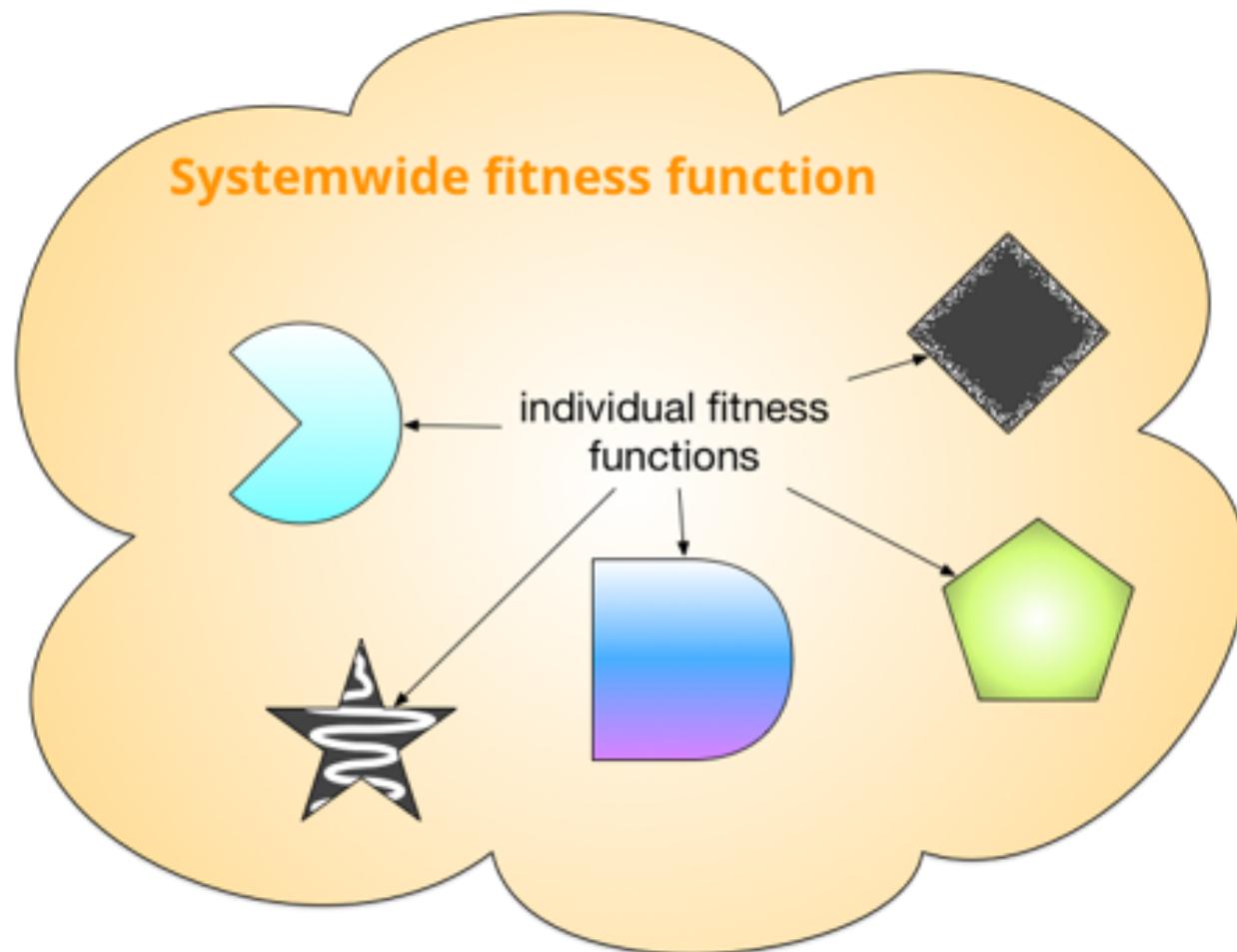
triggered



continuous

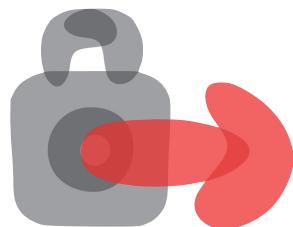
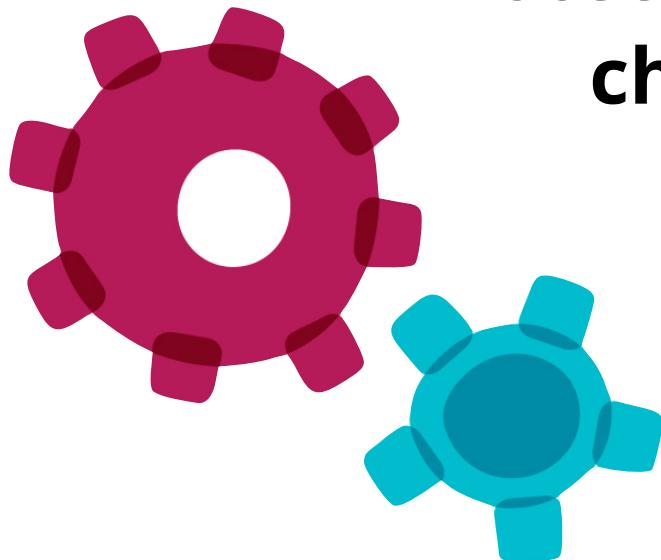


System-wide Fitness Function



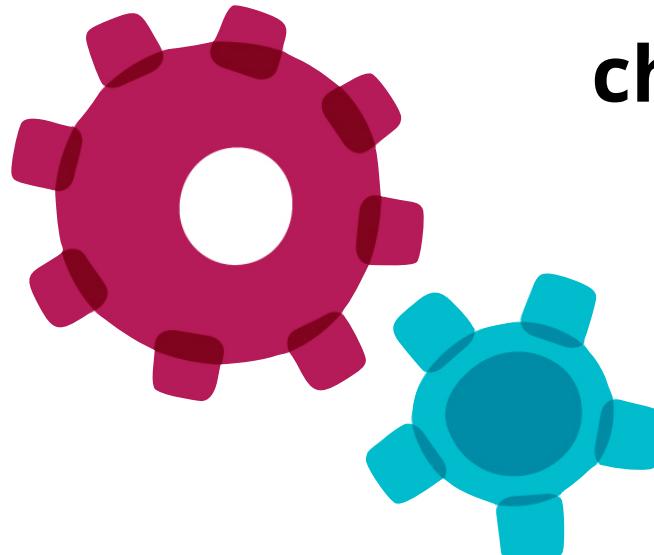
Implementing Fitness Functions

Protecting architectural
characteristics

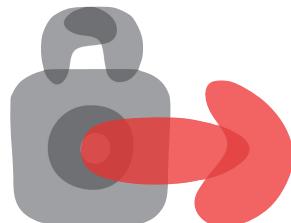


Implementing Fitness Functions

Protecting architectural
characteristics



Automating governance



maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

Postel's Law

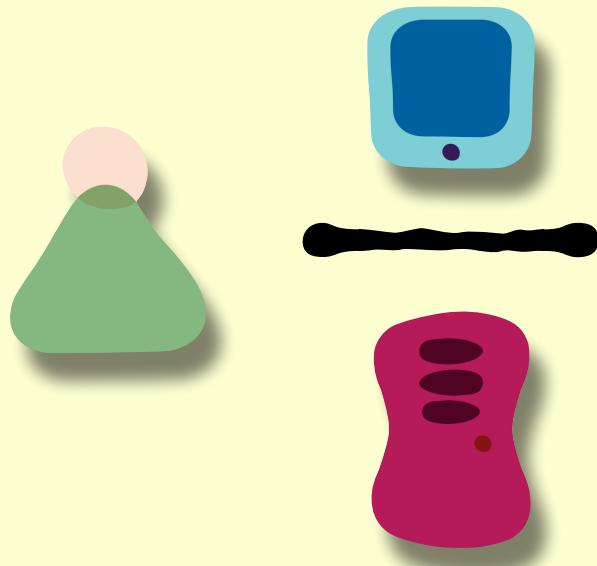
maintainable?

(incoming/outgoing)

Controlled afferent/efferent coupling



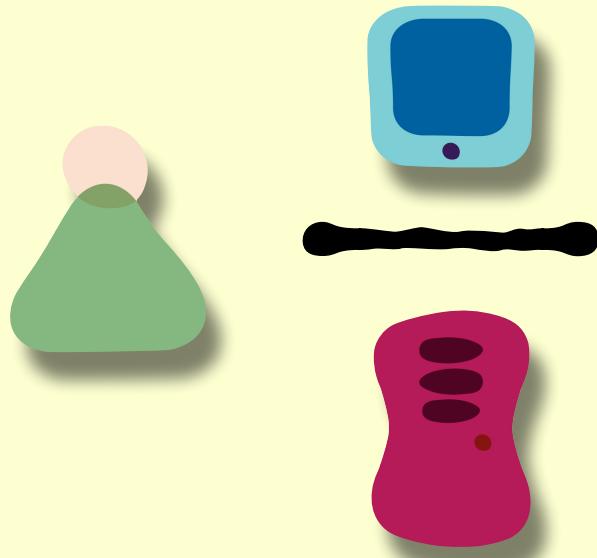
Governing Code Quality



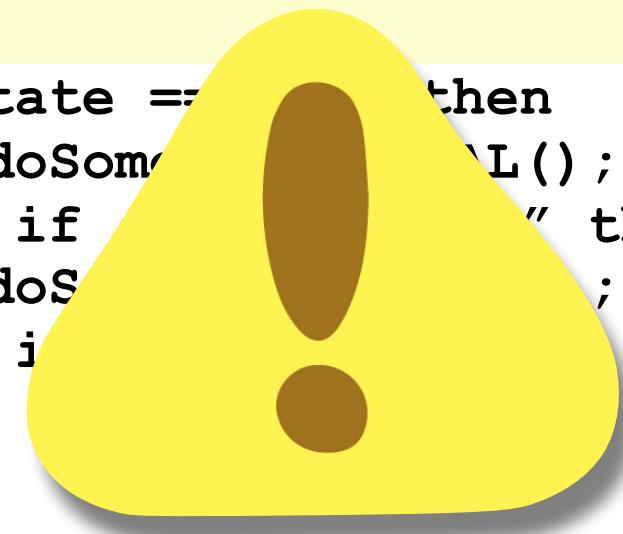
```
if state == "AL" then  
    doSomethingForAL();  
else if state == "GA" then  
    doSomethingForGA();  
else if ...
```



Governing Code Quality



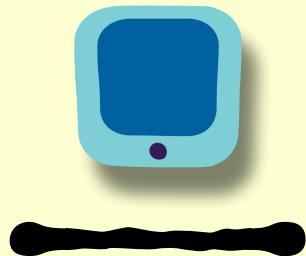
```
if state == "A" then
    doSomethingA();
else if state == "B" then
    doSomethingB();
else if state == "C" then
    doSomethingC();
```





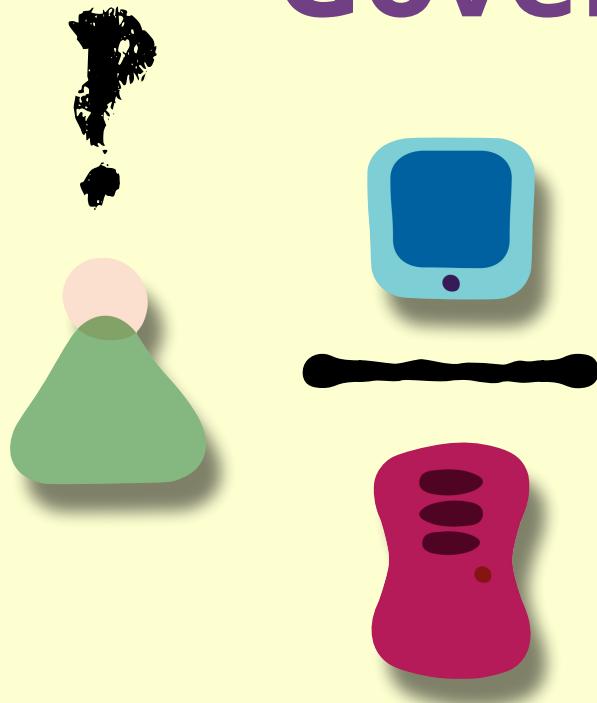
Governing Code Quality

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

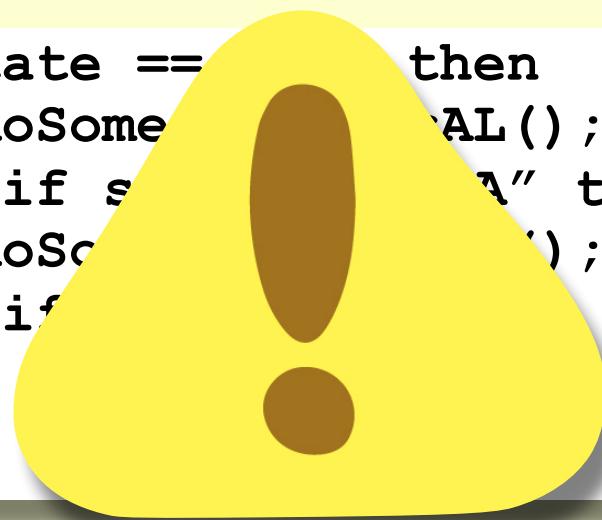




Governing Code Quality

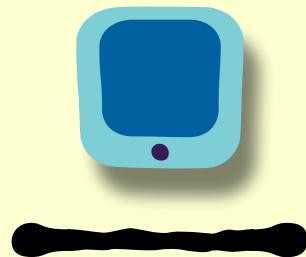
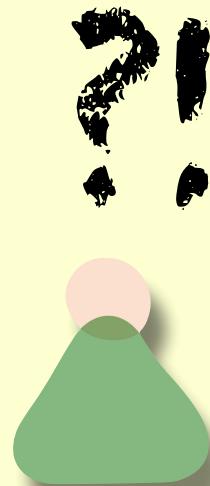


```
if state == "A" then  
    doSomethingA();  
else if state == "B" then  
    doSomethingB();  
else if state == "C" then  
    doSomethingC();
```

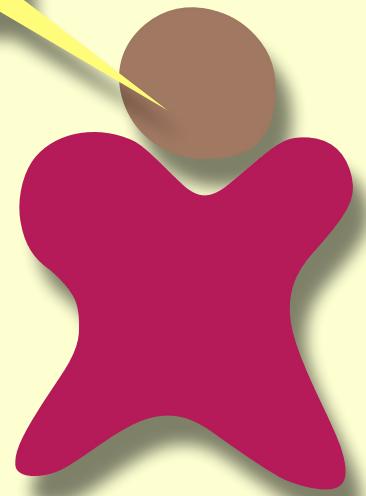


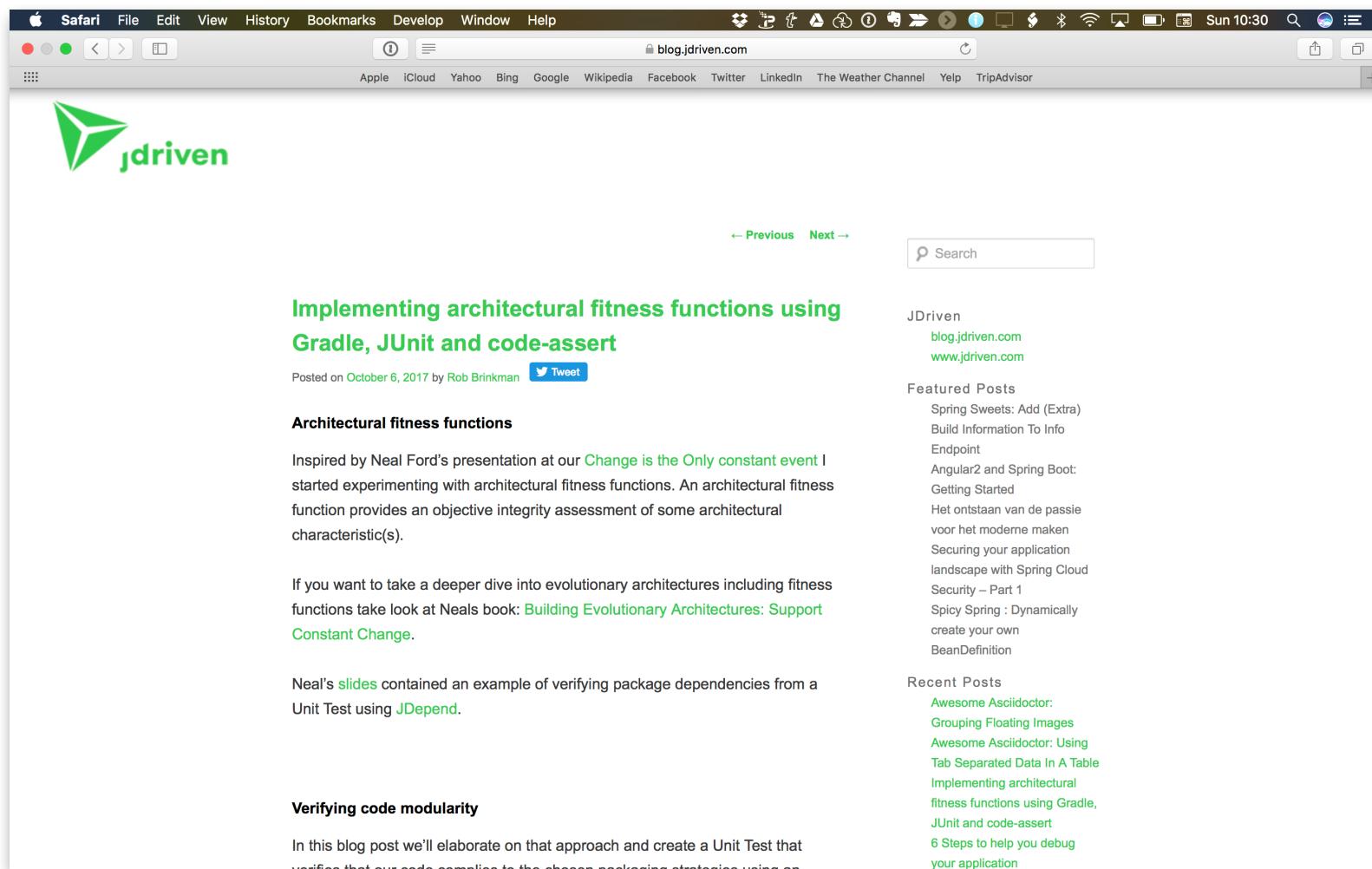


Governing Code Quality



Strategy Design
Pattern





A screenshot of a Safari browser window displaying a blog post from blog.jdriven.com. The page title is "Implementing architectural fitness functions using Gradle, JUnit and code-assert". The post was posted on October 6, 2017, by Rob Brinkman. The content discusses architectural fitness functions, inspired by Neal Ford's presentation at the "Change is the Only constant event". It includes sections on "Architectural fitness functions", "Verifying code modularity", and a quote from Neal's slides. The sidebar features links to JD驱 (JD驱), JD驱 (JD驱), and JD驱 (JD驱). It also lists "Featured Posts" and "Recent Posts".

Safari File Edit View History Bookmarks Develop Window Help

blog.jdriven.com

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



← Previous Next →

Search

JD驱

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 Ascidoctor: Grouping Floating Images
- Awesome Ascidoctor: 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

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

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

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).

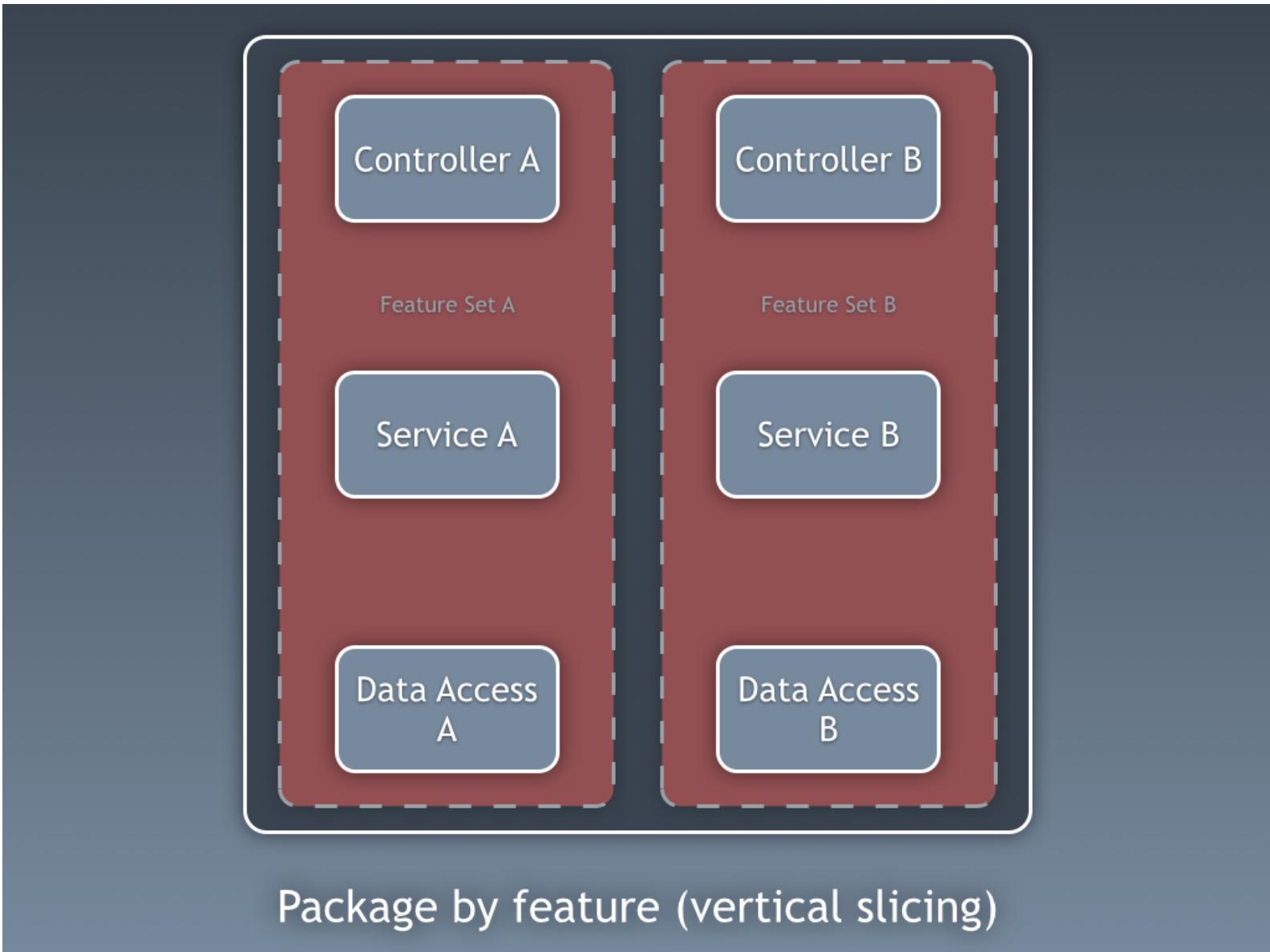
If you want to take a deeper dive into evolutionary architectures including fitness functions take look at Neals book: [Building Evolutionary Architectures: Support Constant Change](#).

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 strategies using an

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



```
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 ->  
com.jdriven.fitness.packaging.by.feature.b (by com.jdriven.fitness.packaging.by.feature.a.ControllerA)
```

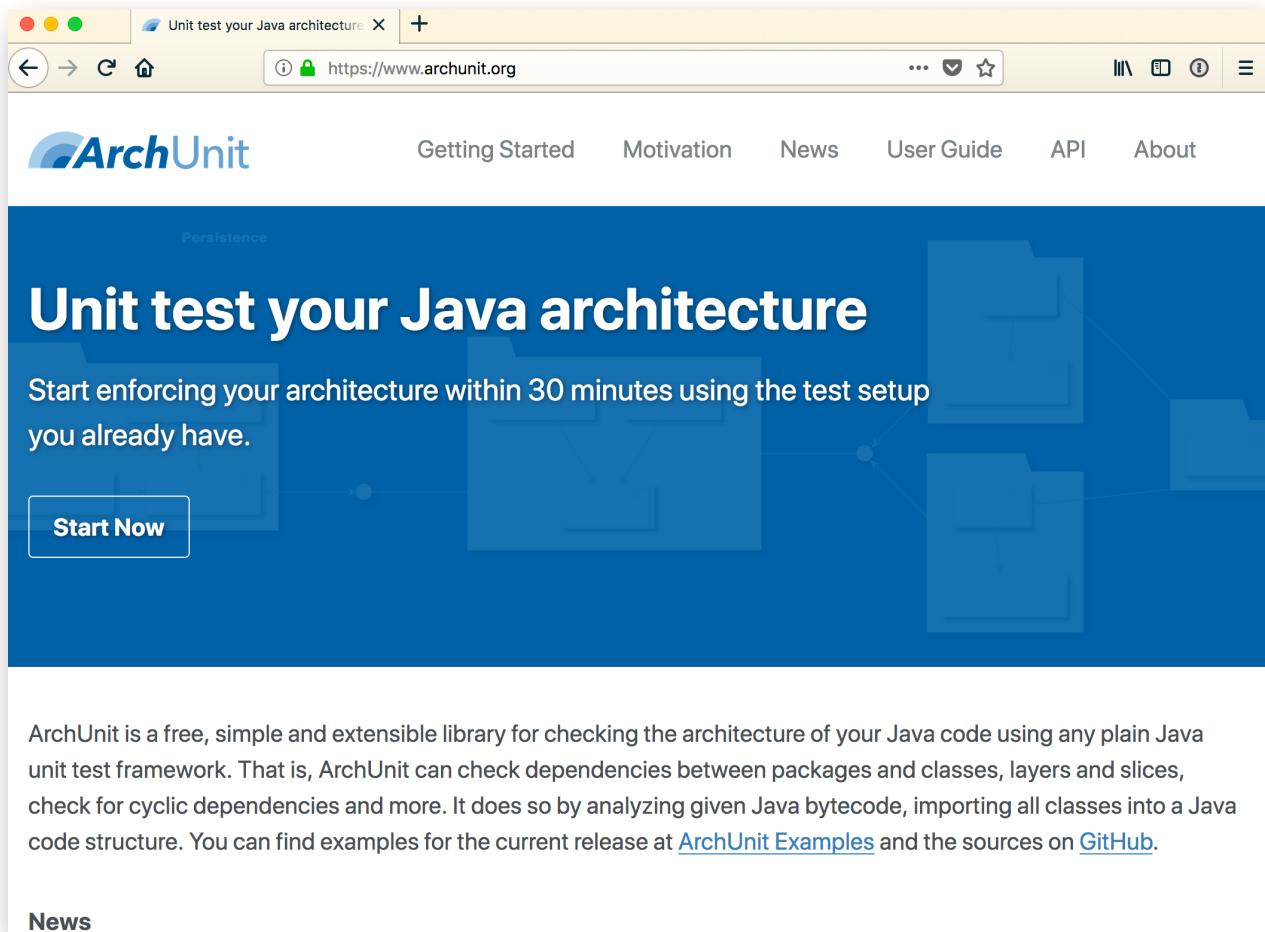
```
public class ControllerA {  
  
    private final ServiceA serviceA;  
    private final ServiceB serviceB;  
  
    public ControllerA(ServiceA serviceA, ServiceB serviceB) {  
        this.serviceA = serviceA;  
        this.serviceB = serviceB;  
    }  
}
```

```
java.lang.AssertionError:  
Expected: Comply with rules  
but: DENIED com.jdriven.fitness.packaging.by.feature.a ->  
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's a navigation bar with links for "Getting Started", "Motivation", "News", "User Guide", "API", and "About". Below the navigation, a large banner features the text "Unit test your Java architecture" and "Start enforcing your architecture within 30 minutes using the test setup you already have." A prominent "Start Now" button is visible. The background of the banner has a blue gradient with abstract white shapes representing code and data flow. At the bottom of the page, there's a "News" section.

Persistence

Unit test your Java architecture

Start enforcing your architecture within 30 minutes using the test setup you already have.

Start Now

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);

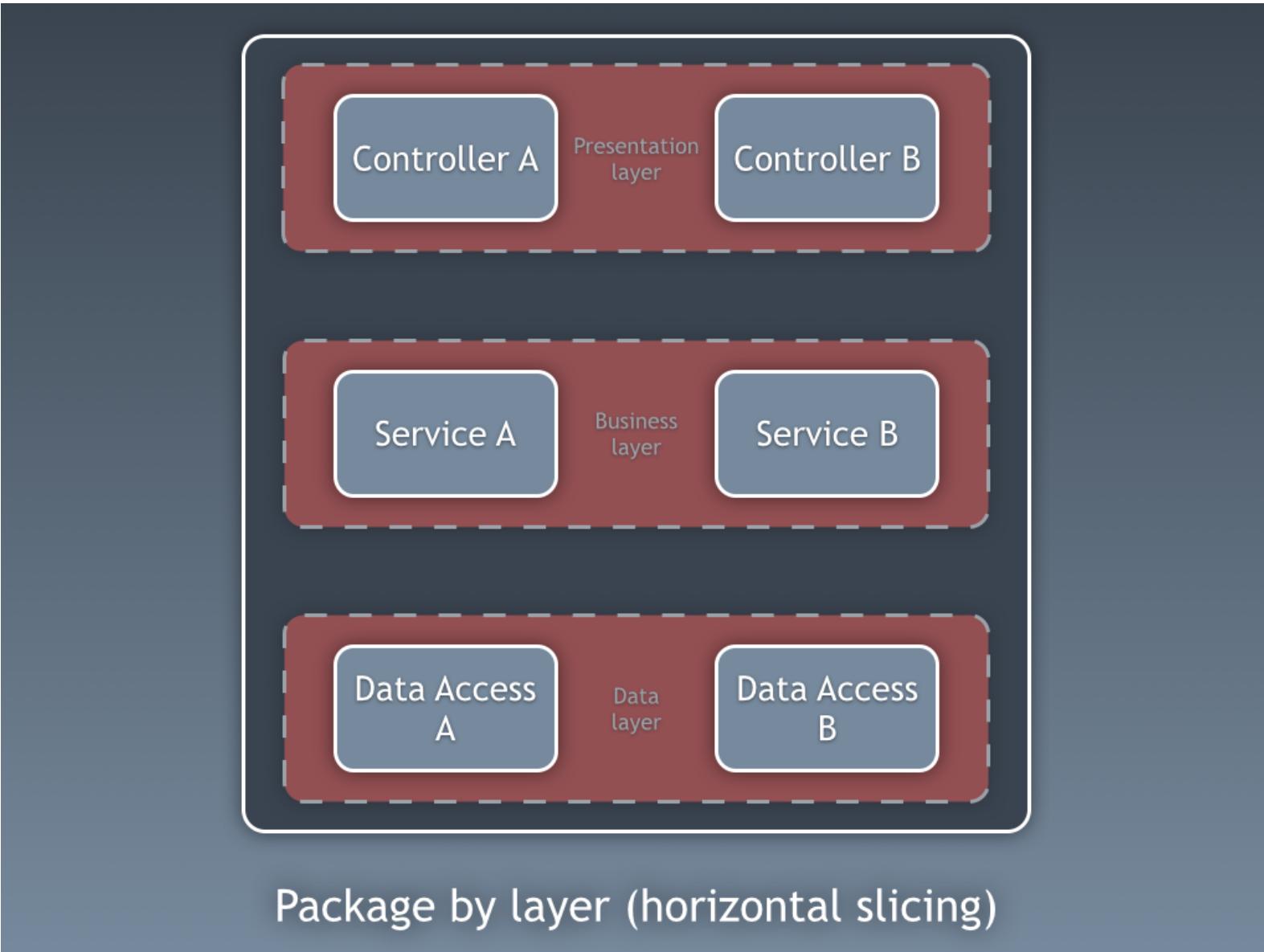
        noClasses().that().resideInAPackage("..impl..").should().beInterfaces().check(classes);
    }
}
```

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
```



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));
}
```

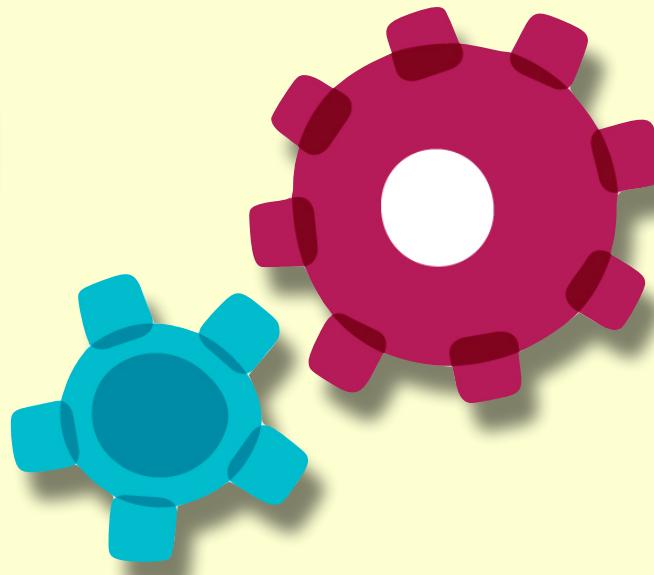
governance

Legality of Open Source Libraries



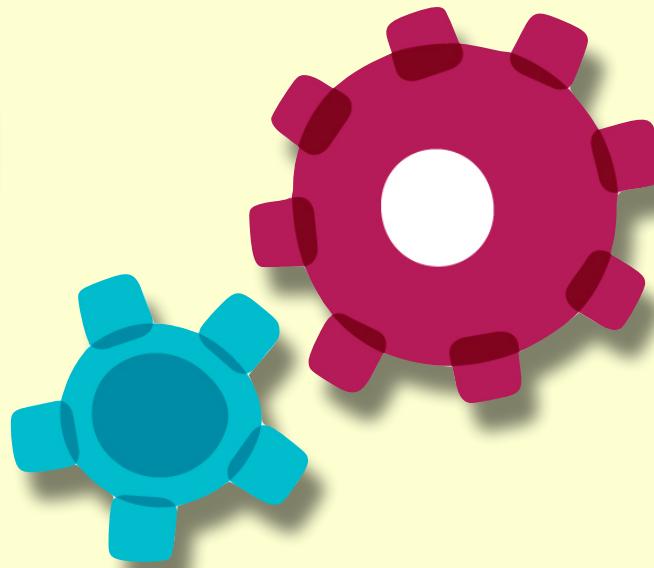
Penultima ↑
A row of three small, colorful gears: one red, one blue, and one green. An upward-pointing arrow is positioned above the text "Penultima".

Legality of Open Source Libraries



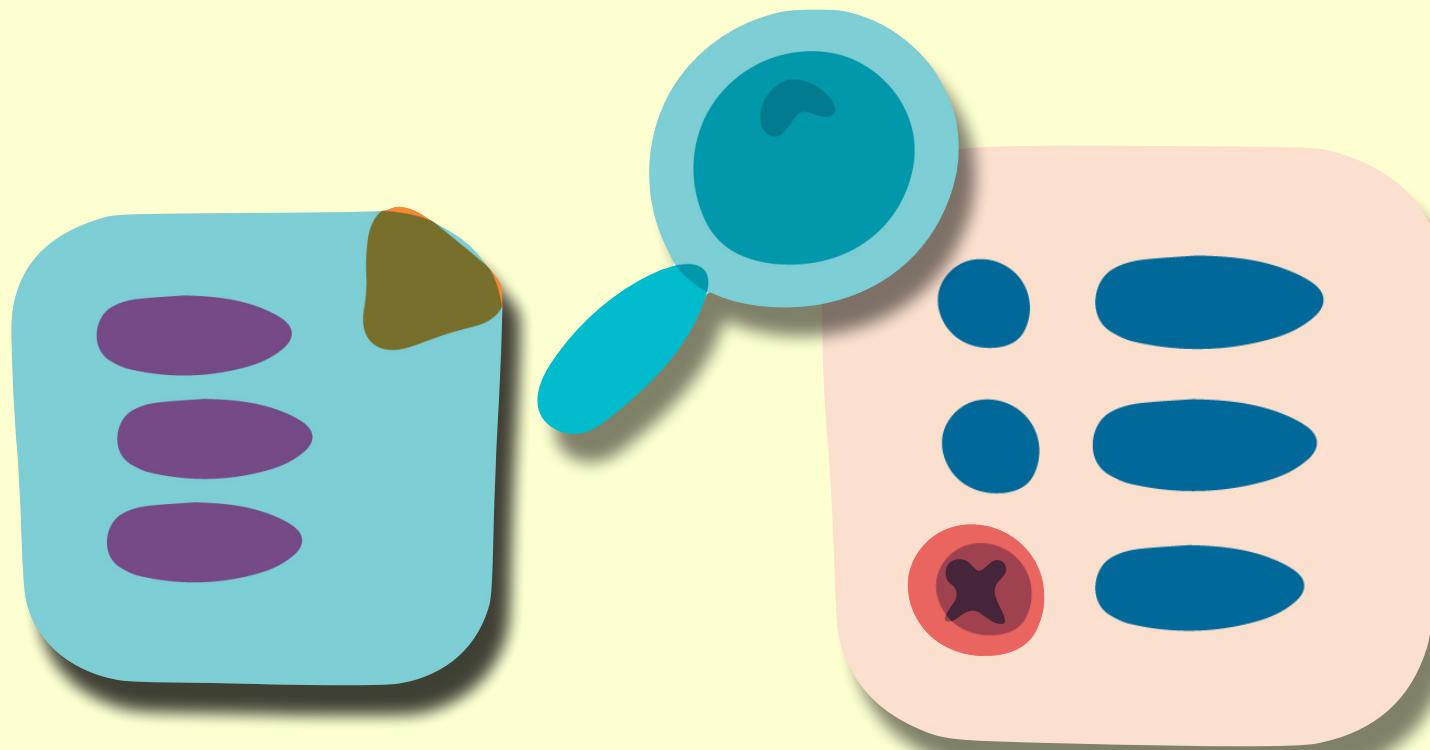
Penultima ↑
Three small gears: two red and one blue, with an arrow pointing to the top red gear.

Legality of Open Source Libraries



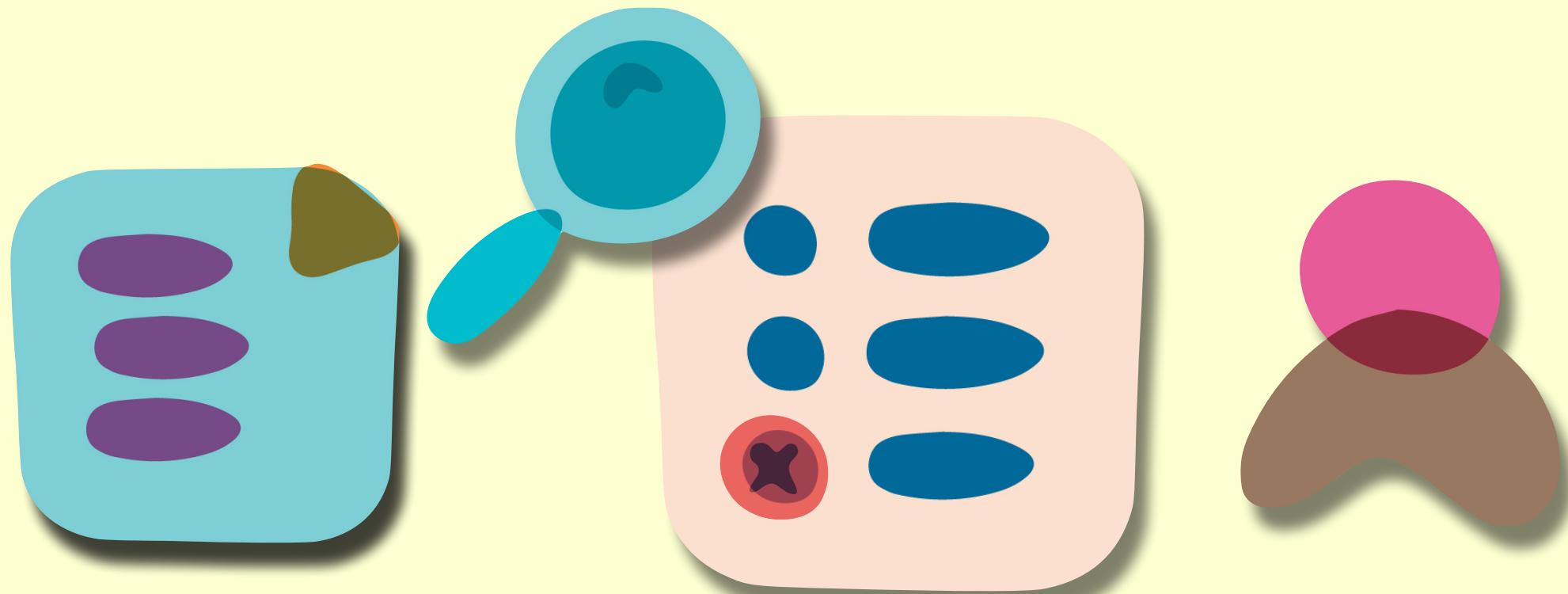
Penultima ↑


Legality of Open Source Libraries



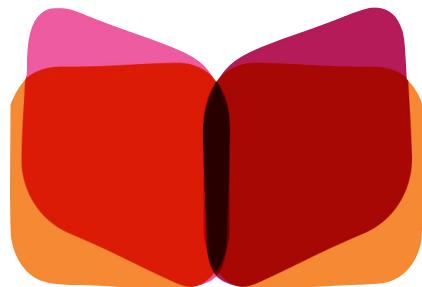
Penultima ↑
A row of four small, colorful gears: one red, one teal, one magenta, and one teal. An upward-pointing arrow is positioned above the magenta gear.

Legality of Open Source Libraries



Penultima ↑
 Cyan Magenta Cyan Magenta

Agenda Part 1



Definition



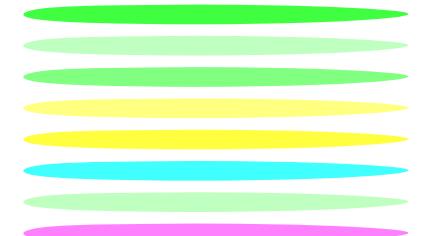
Guided Change via Fitness Functions



Structuring architecture for evolution

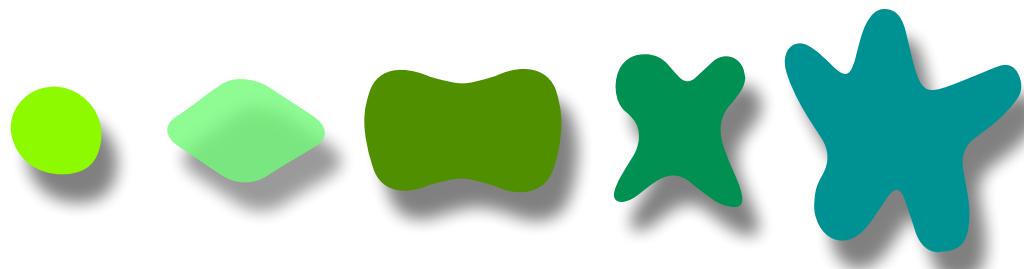


Incremental Change



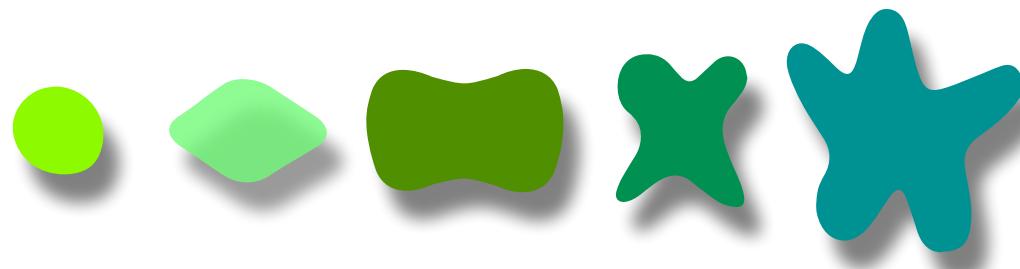
Two Big Ideas

Two Big Ideas



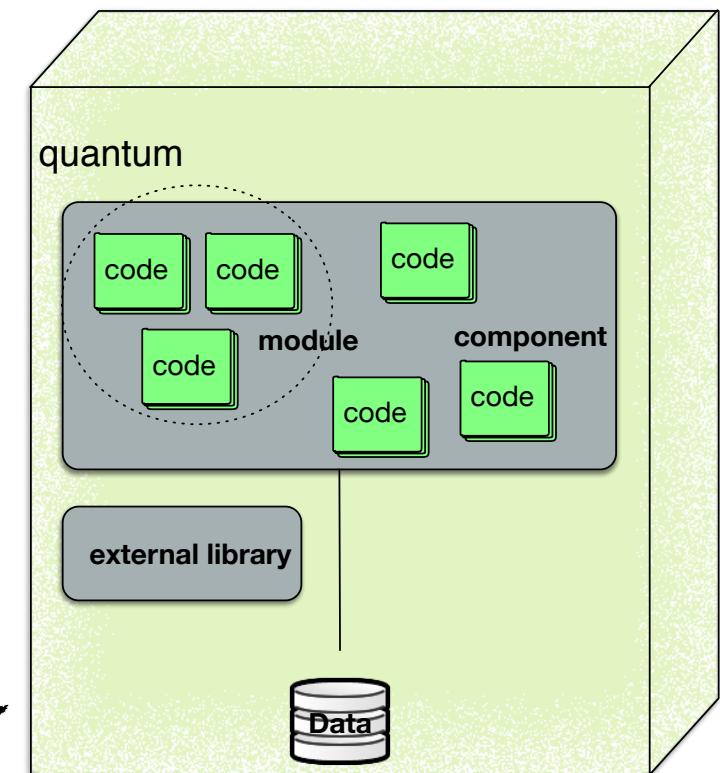
fitness functions for
evolvability

Two Big Ideas

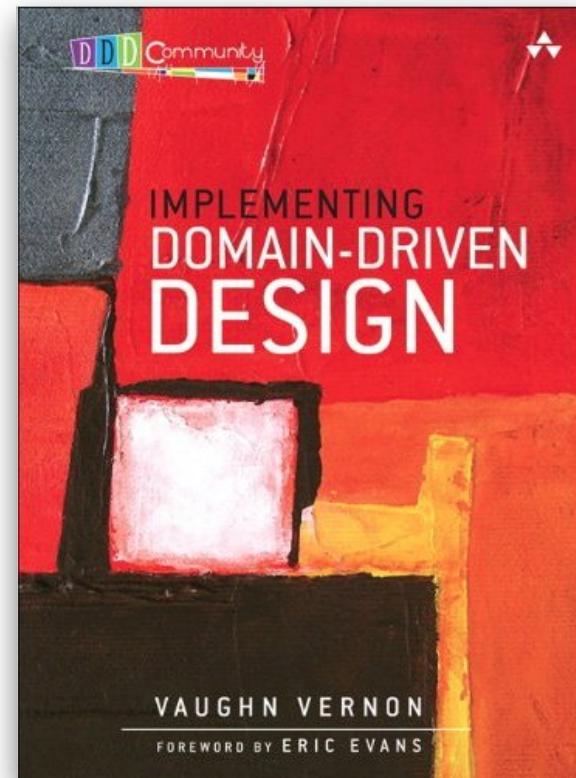
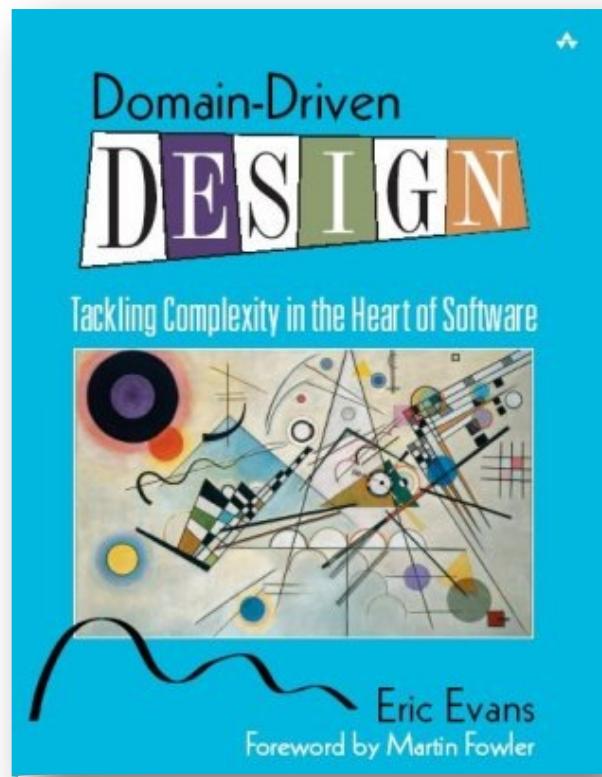


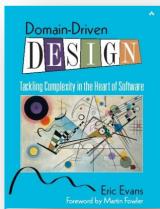
fitness functions for
evolvability

architectural
quantum



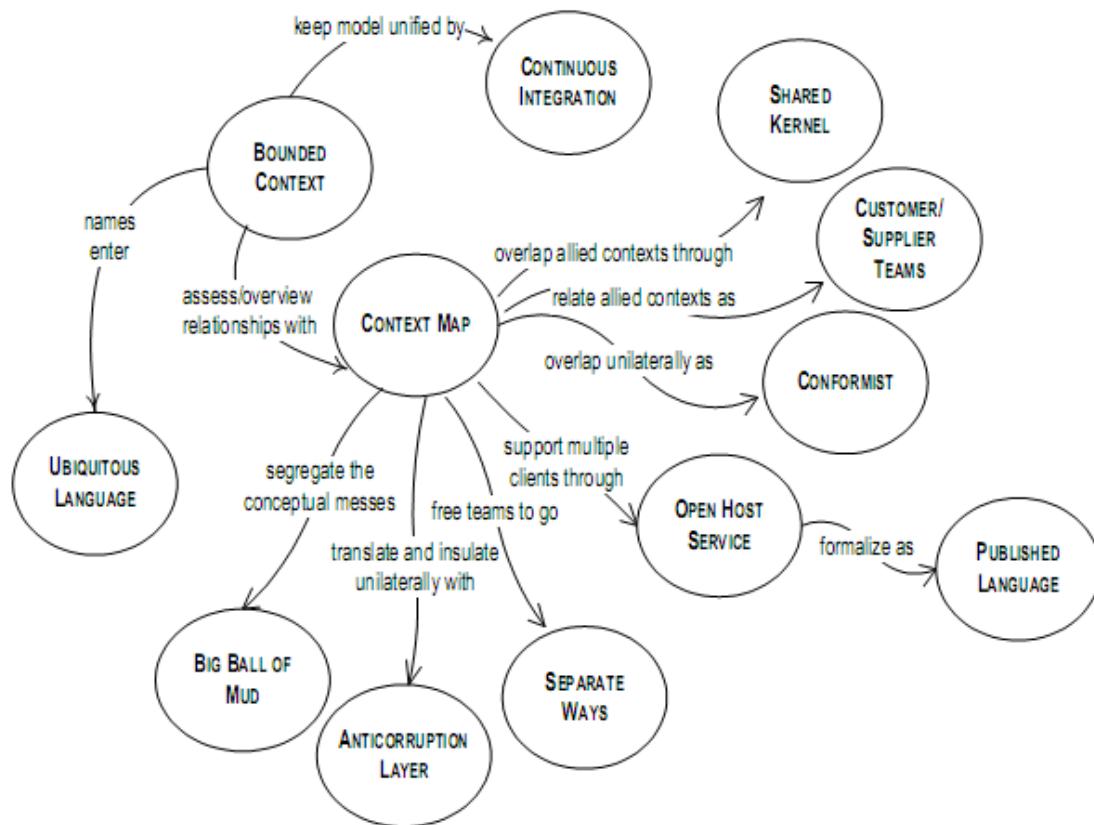
Domain Driven Design





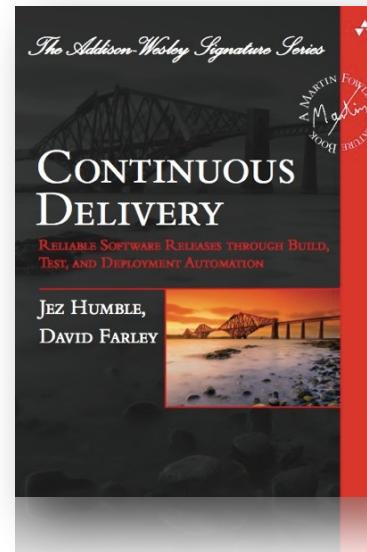
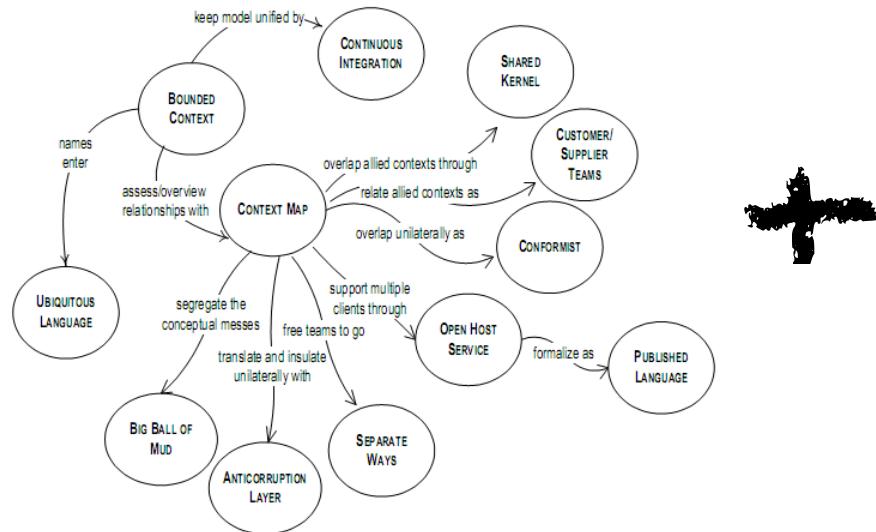
Bounded Context

Maintaining Model Integrity

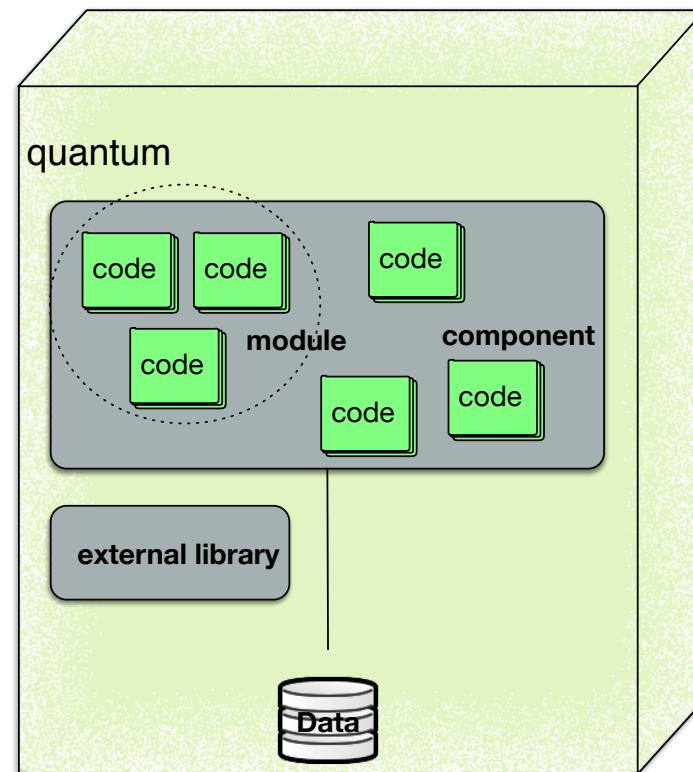


Bounded Context + Continuous Delivery = microservices

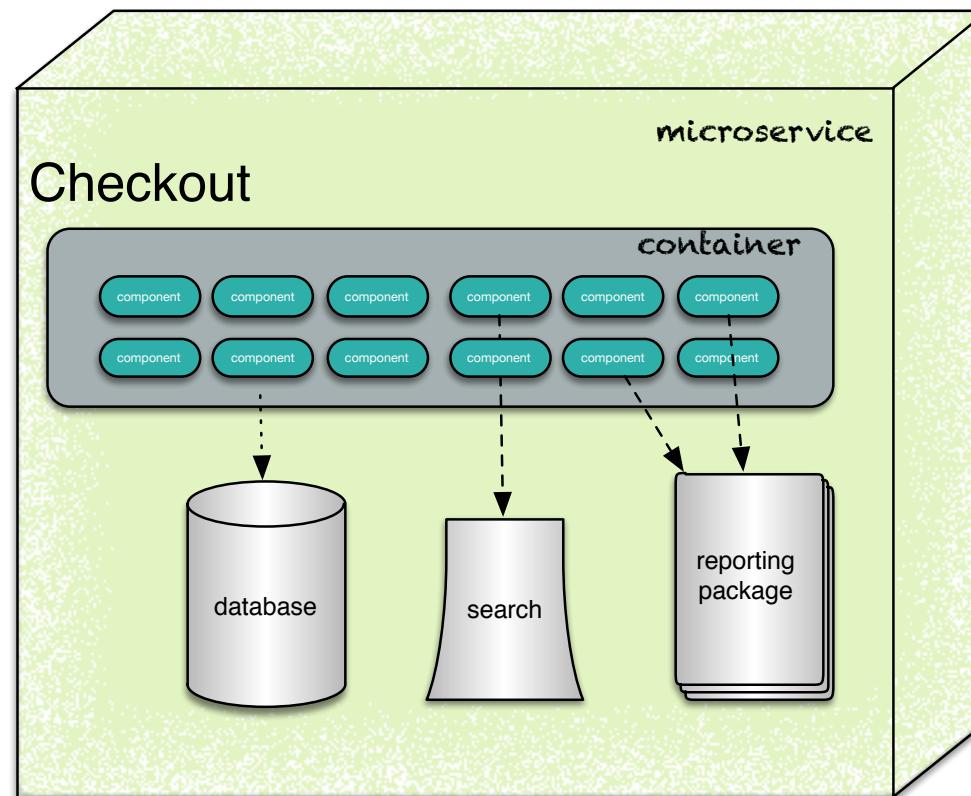
Maintaining Model Integrity



Architectural Quantum



Microservices Quantum

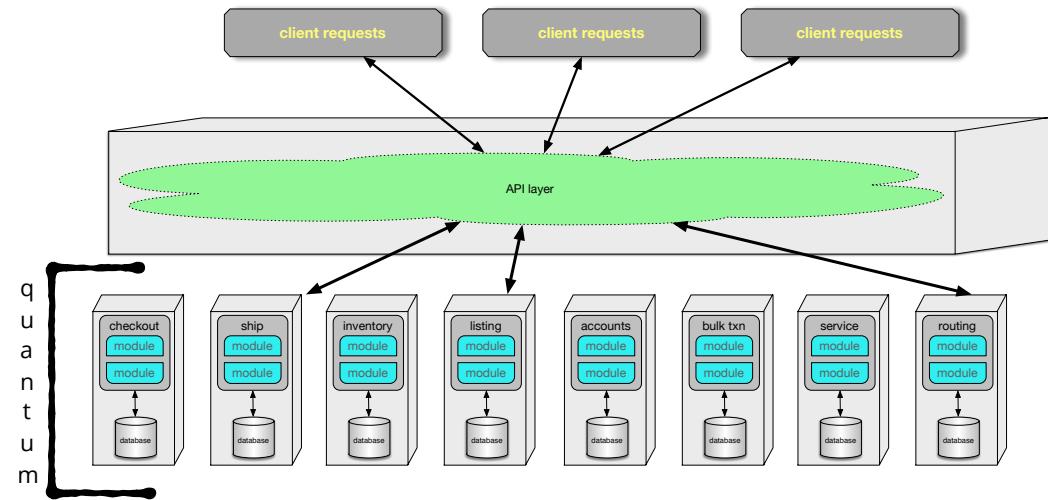
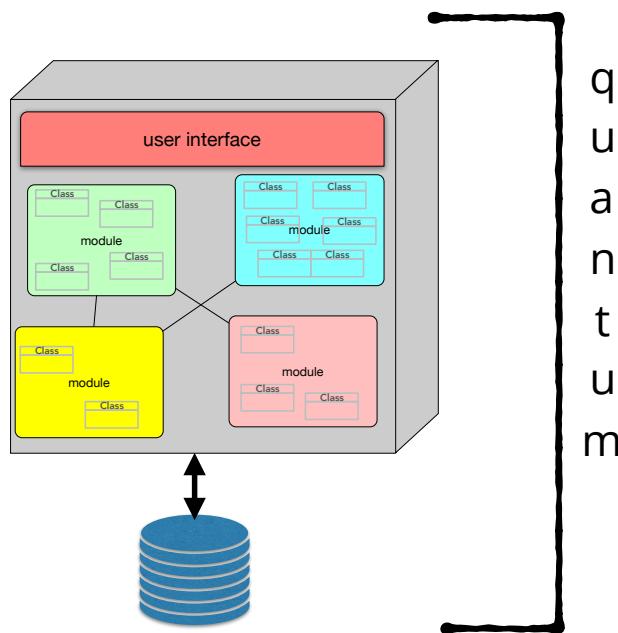


Architectural Quantum

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.

Why Quantum?

The quantum is where architectural characteristics live.



For Each Pattern:

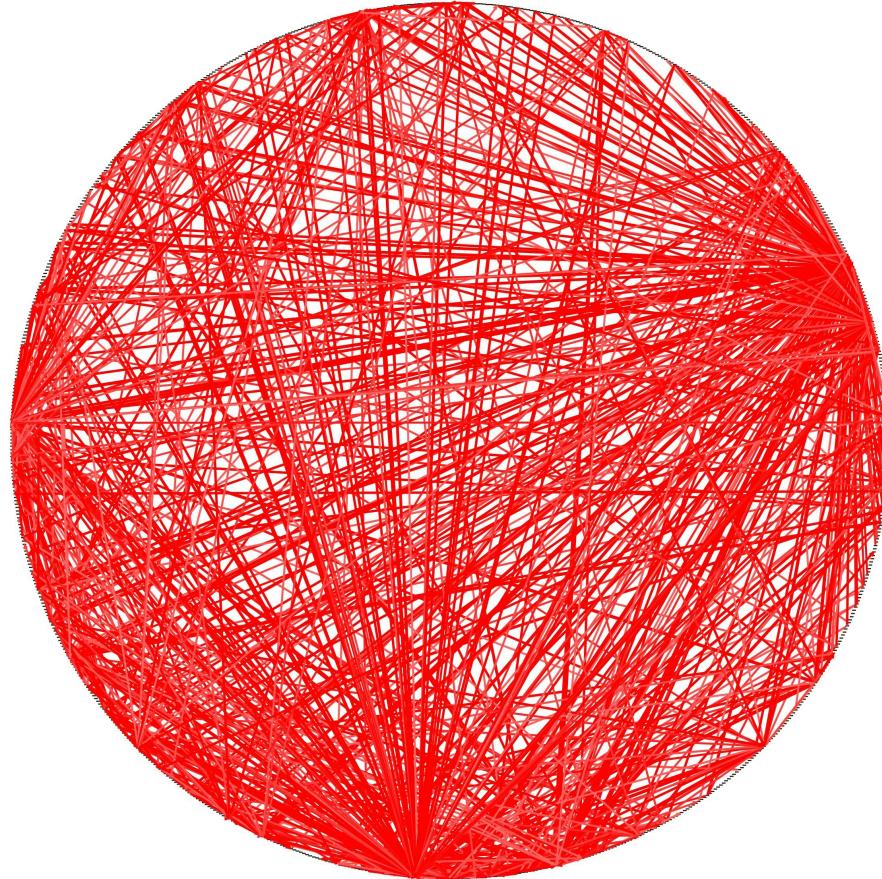


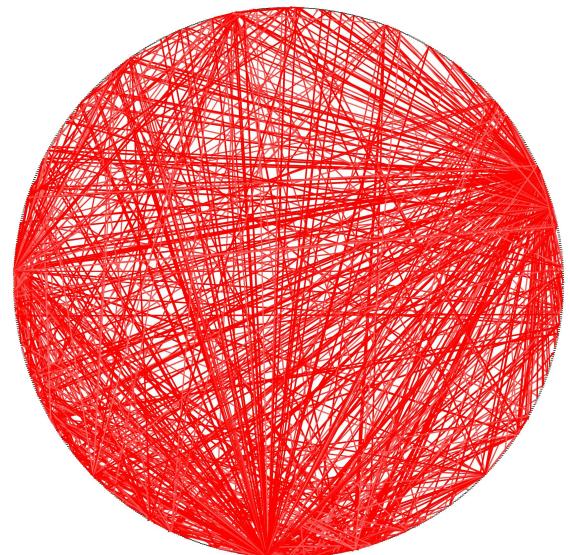
organized by:



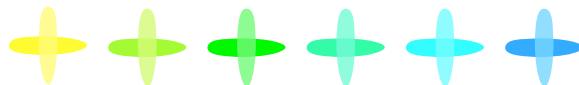
architectural quantum

Big Ball of Mud





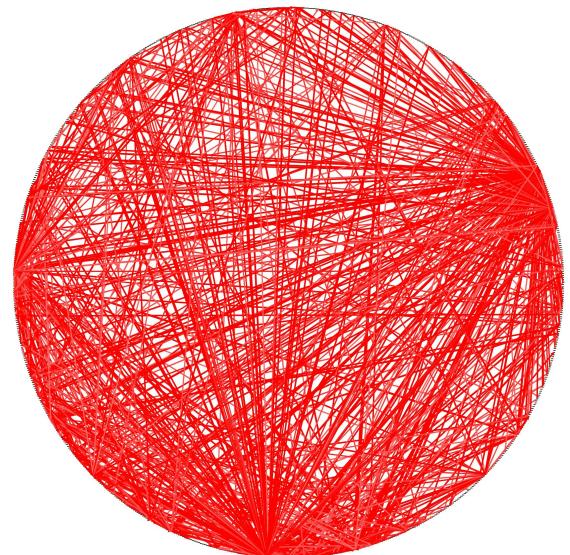
Big Ball of Mud



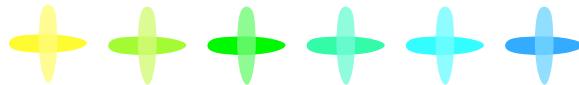
Rippling side effects for any change



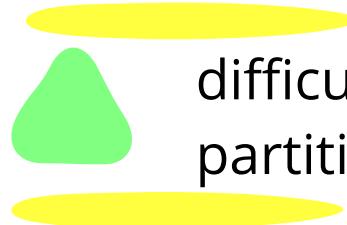
the entire system



Big Ball of Mud



Rippling side effects for any change

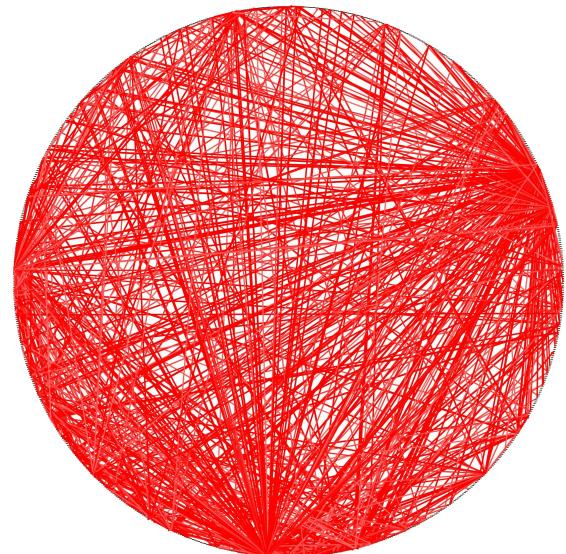


difficult because no clearly defined
partitioning exists

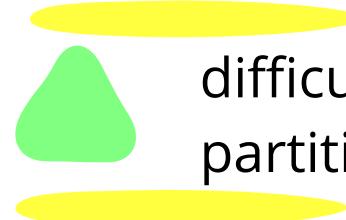


the entire system

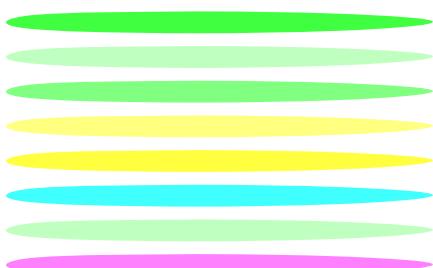
Big Ball of Mud



Rippling side effects for any change



difficult because no clearly defined
partitioning exists

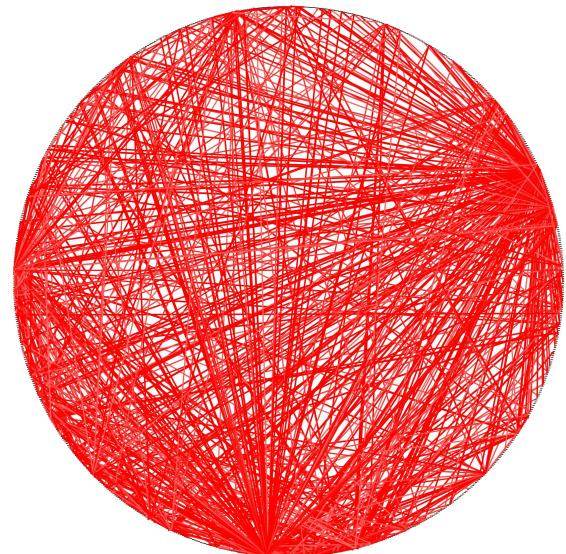


Epitome of **in**appropriate coupling



the entire system

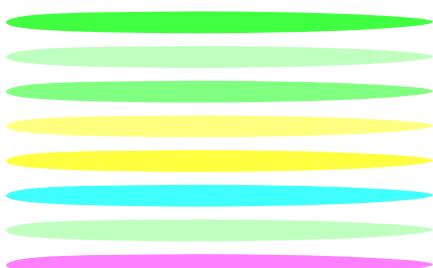
Big Ball of Mud



Rippling side effects for any change



difficult because no clearly defined
partitioning exists

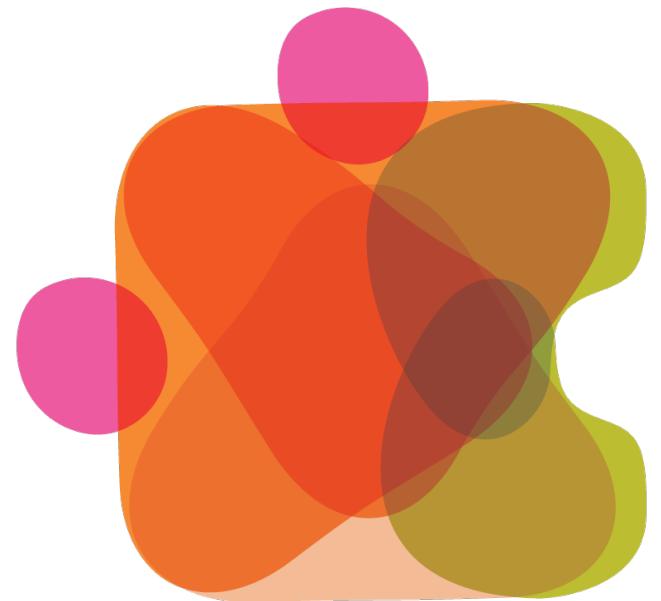


Epitome of **in**appropriate coupling

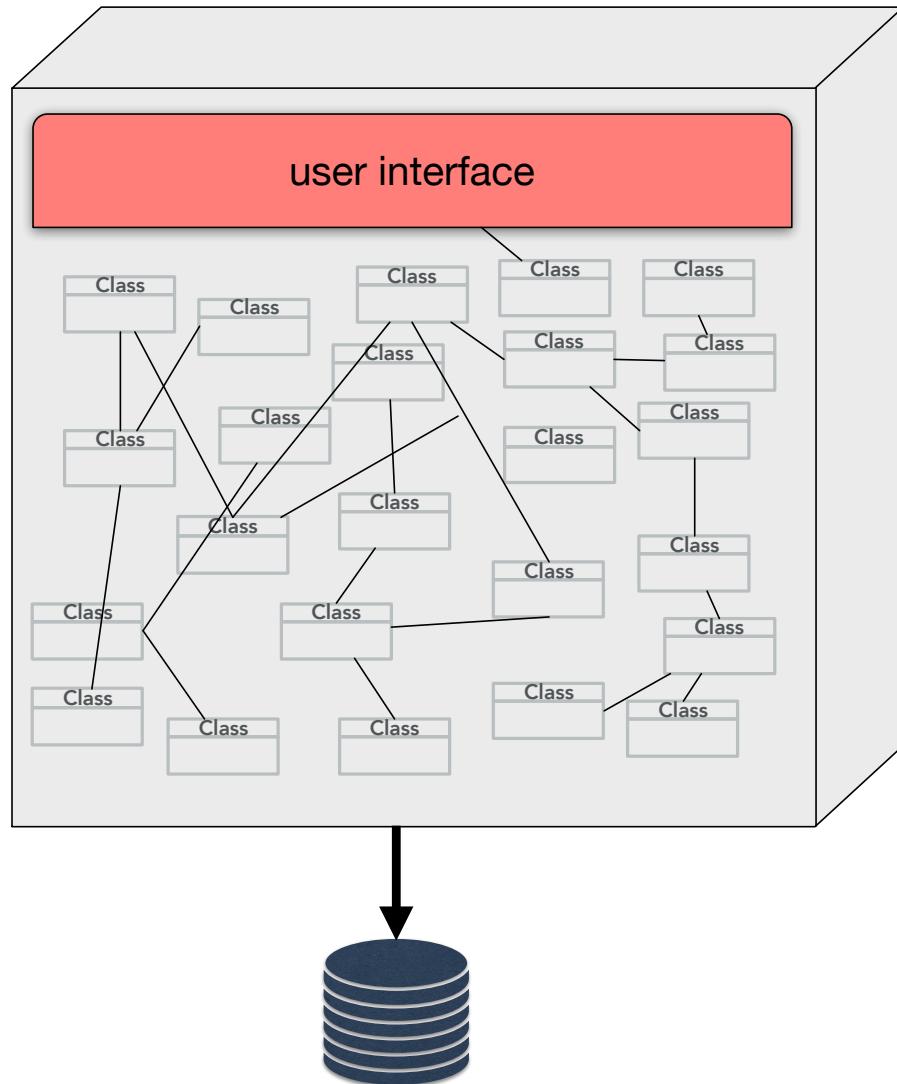


the entire system

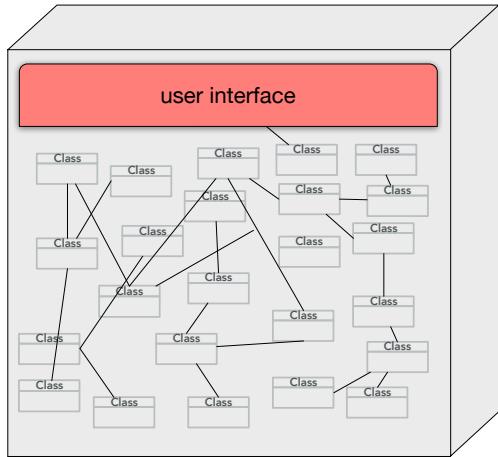
Monoliths



Unstructured Monoliths



Unstructured Monoliths

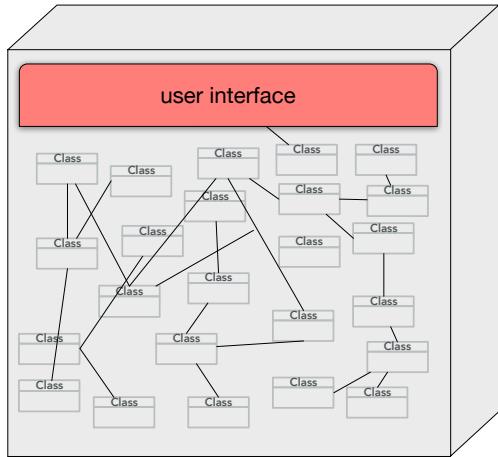


Large quantum size
hinders incremental change



the entire system

Unstructured Monoliths



⊕ ⊕ ⊕ ⊕ ⊕ ⊕ Large quantum size
hinders incremental change

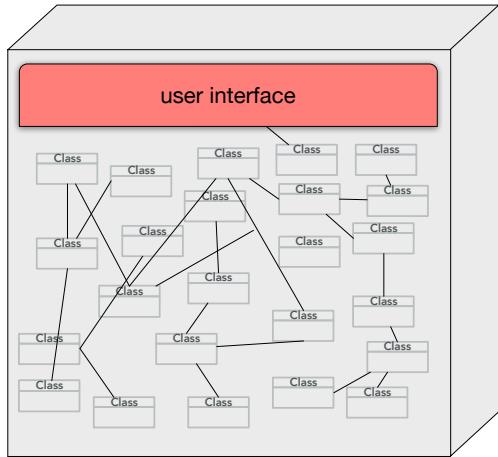


difficult but not impossible

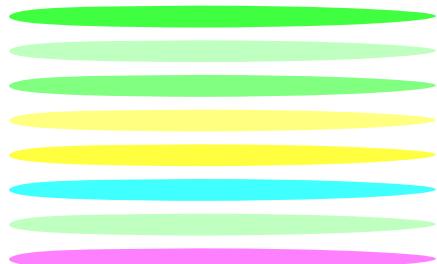


the entire system

Unstructured Monoliths



⊕ ⊕ ⊕ ⊕ ⊕ ⊕ Large quantum size
hinders incremental change



functionally almost as bad as the Big Ball of Mud

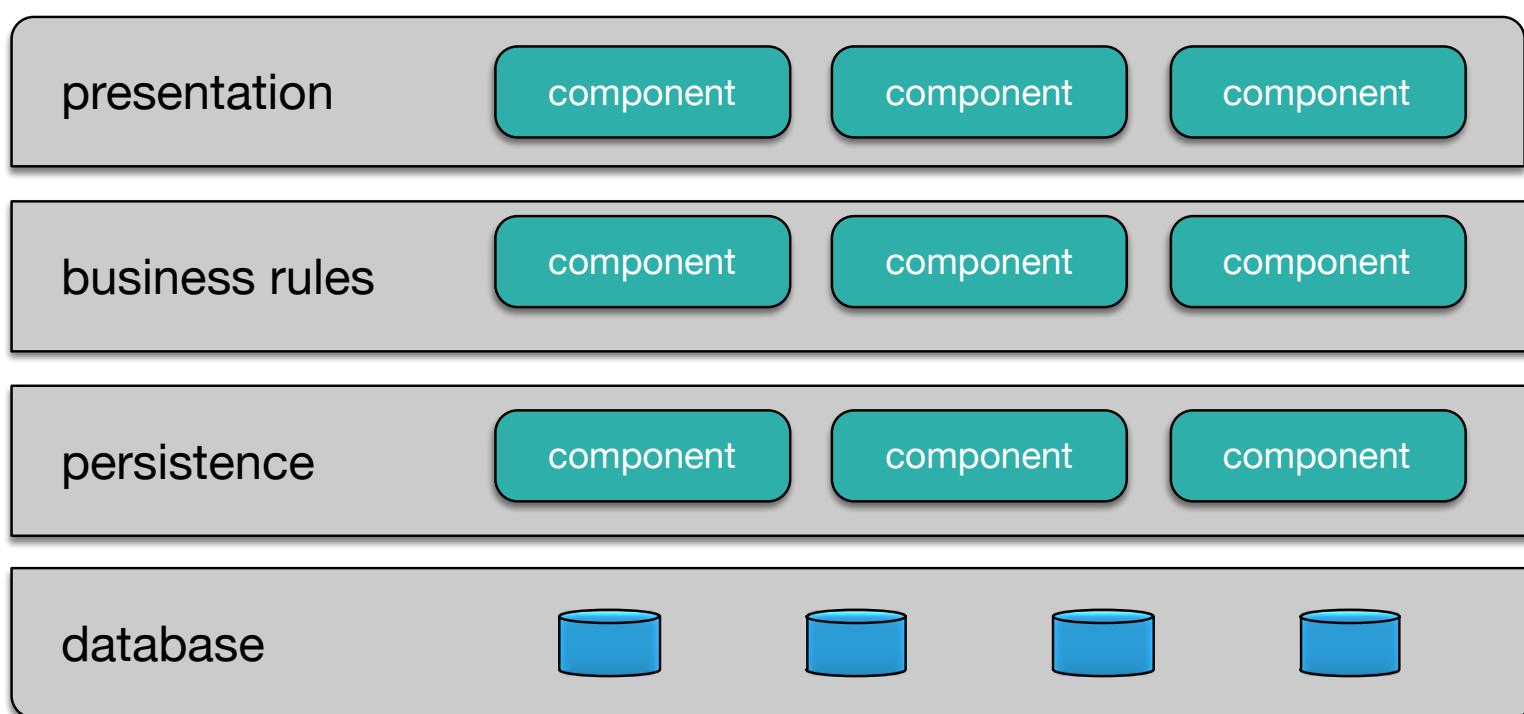


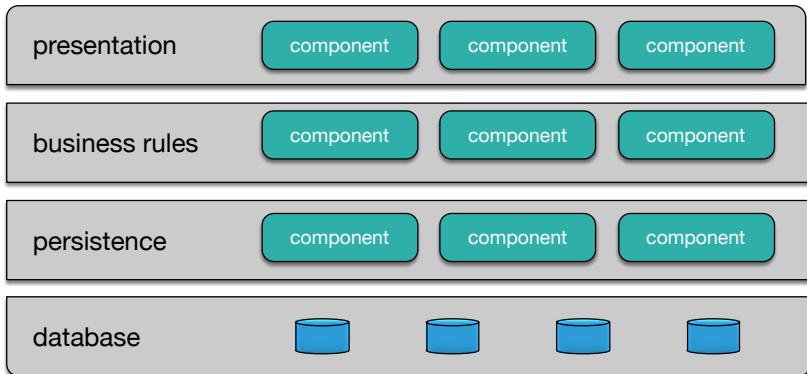
difficult but not impossible



the entire system

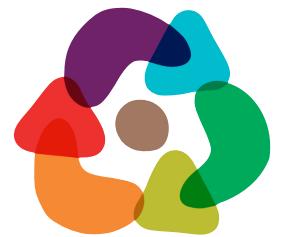
Layered Monolith



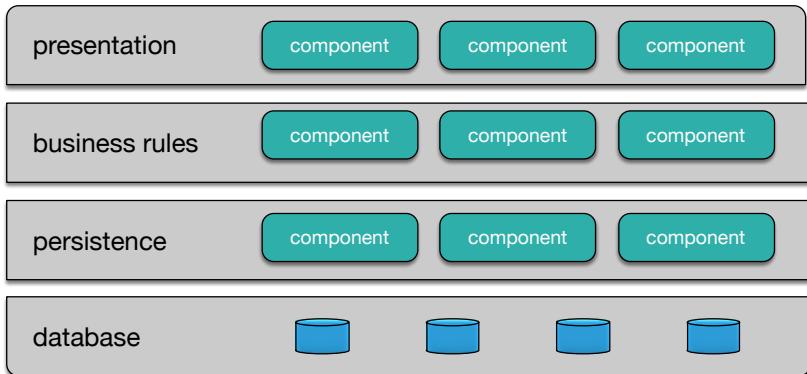


Layered Monolith

⊕ ⊕ ⊕ ⊕ ⊕ ⊕ Selectively easy based on partitioning



the entire system



Layered Monolith

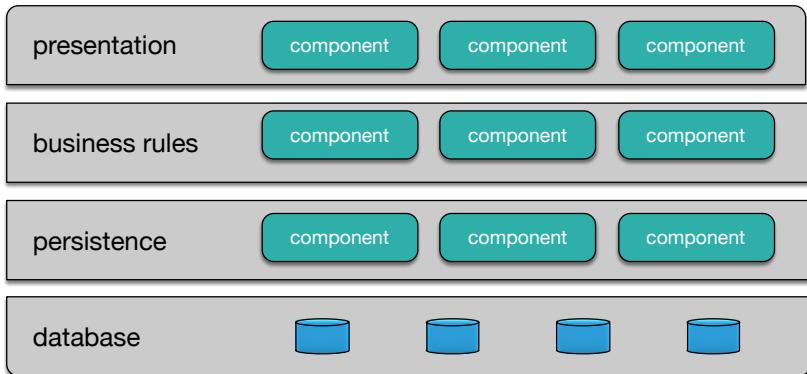
⊕ ⊕ ⊕ ⊕ ⊕ ⊕ Selectively easy based on partitioning



easier because structure is more apparent



the entire system



Layered Monolith

⊕ ⊕ ⊕ ⊕ ⊕ ⊕ Selectively easy based on partitioning

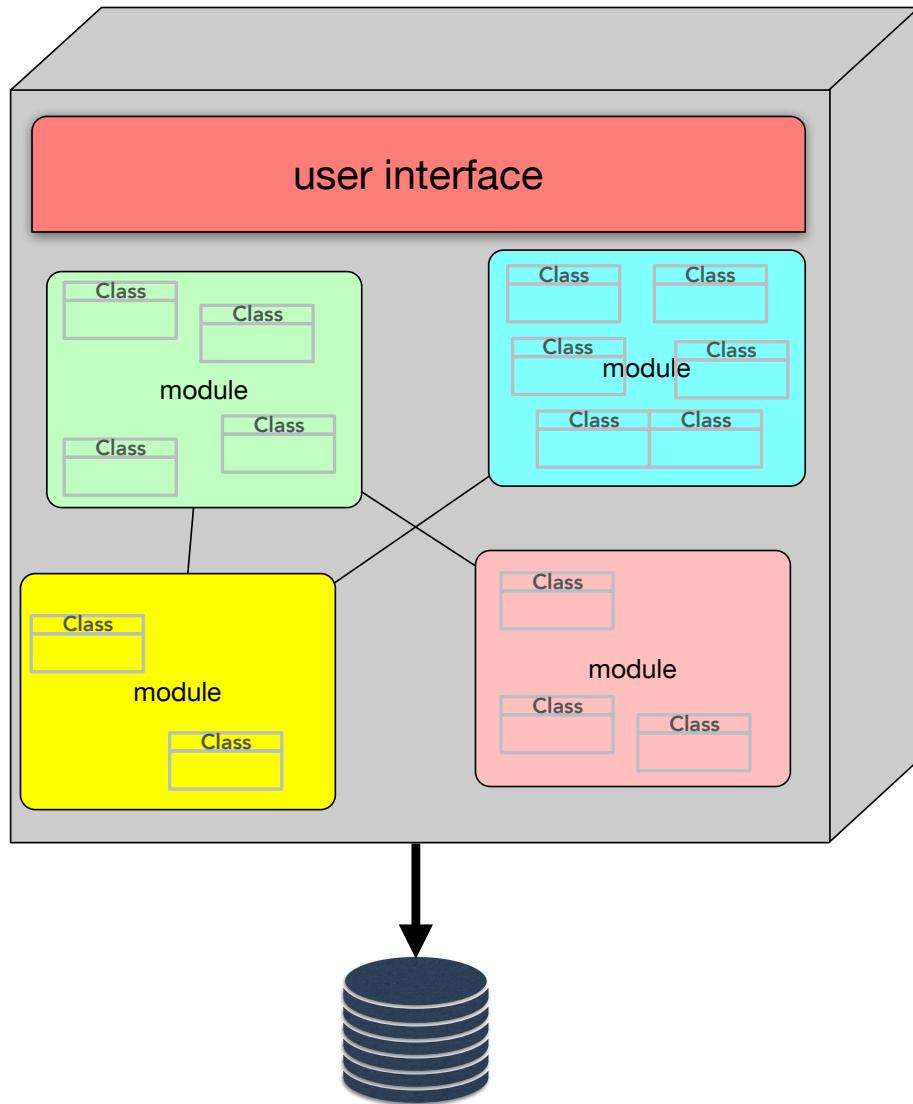
↑ easier because structure is more apparent

- good technical architecture partitioning
- more difficult for orthogonal concerns

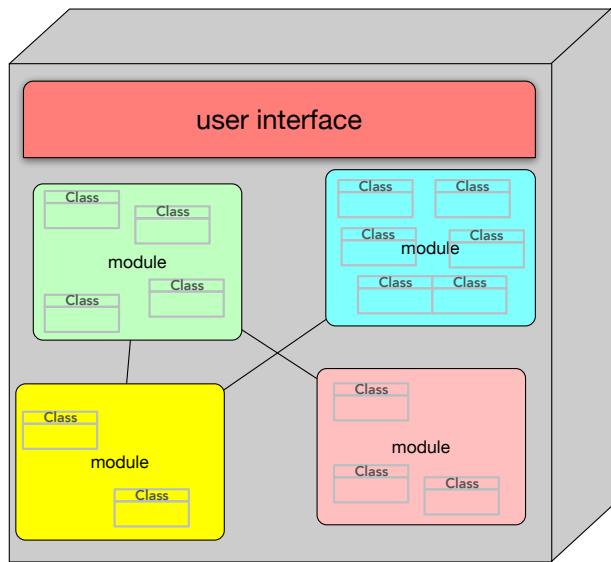


the entire system

Modular Monoliths



Modular Monoliths

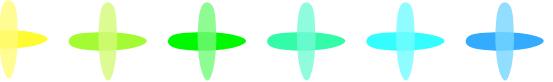


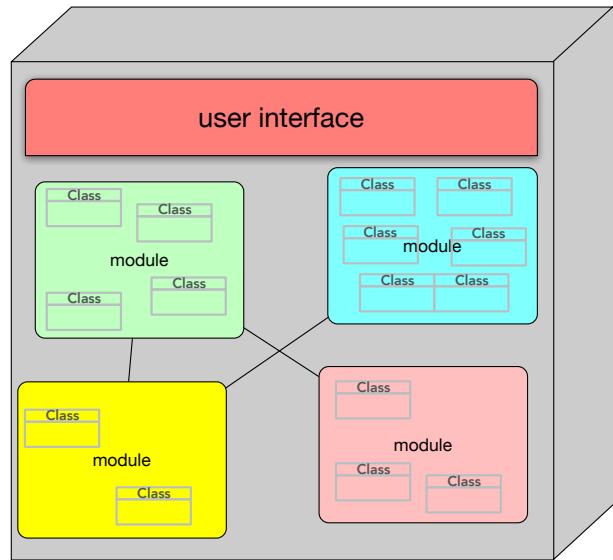
the entire system

Modular Monoliths



the entire system, with selective better granularity

the degree of deployability 
of the components determines the rate of
incremental change.

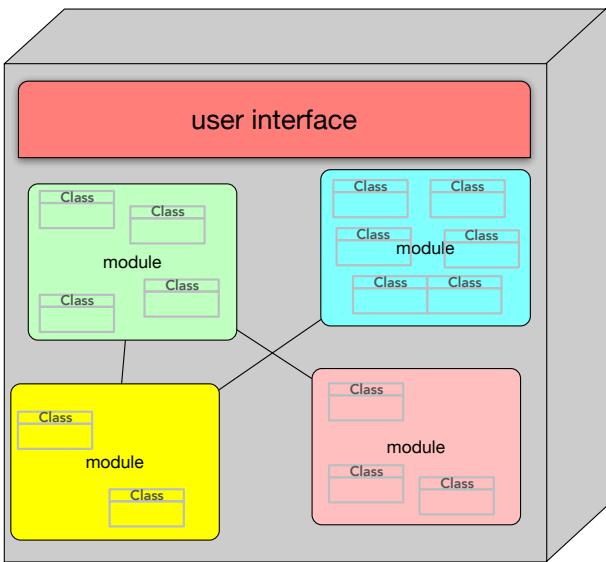


Modular Monoliths



the entire system, with selective better granularity

the degree of deployability of the components determines the rate of incremental change.



easier to design and implement in this architecture because of good separation of components

Modular Monoliths

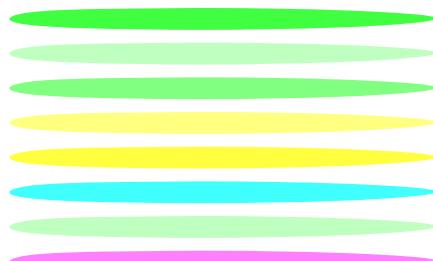
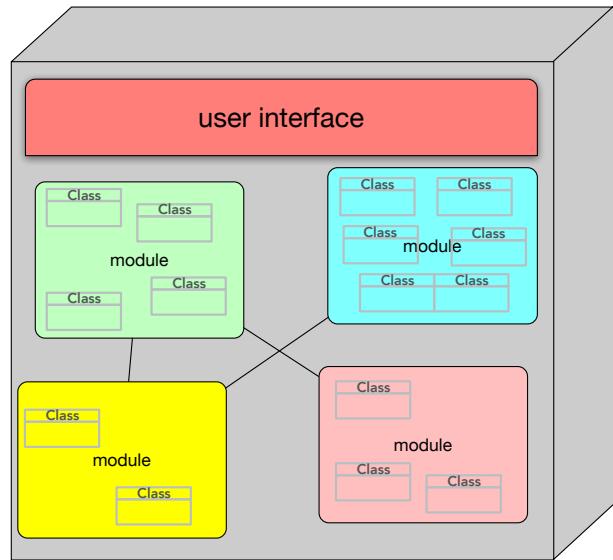
the degree of deployability of the components determines the rate of incremental change.

easier to design and implement in this architecture
(good separation of components)

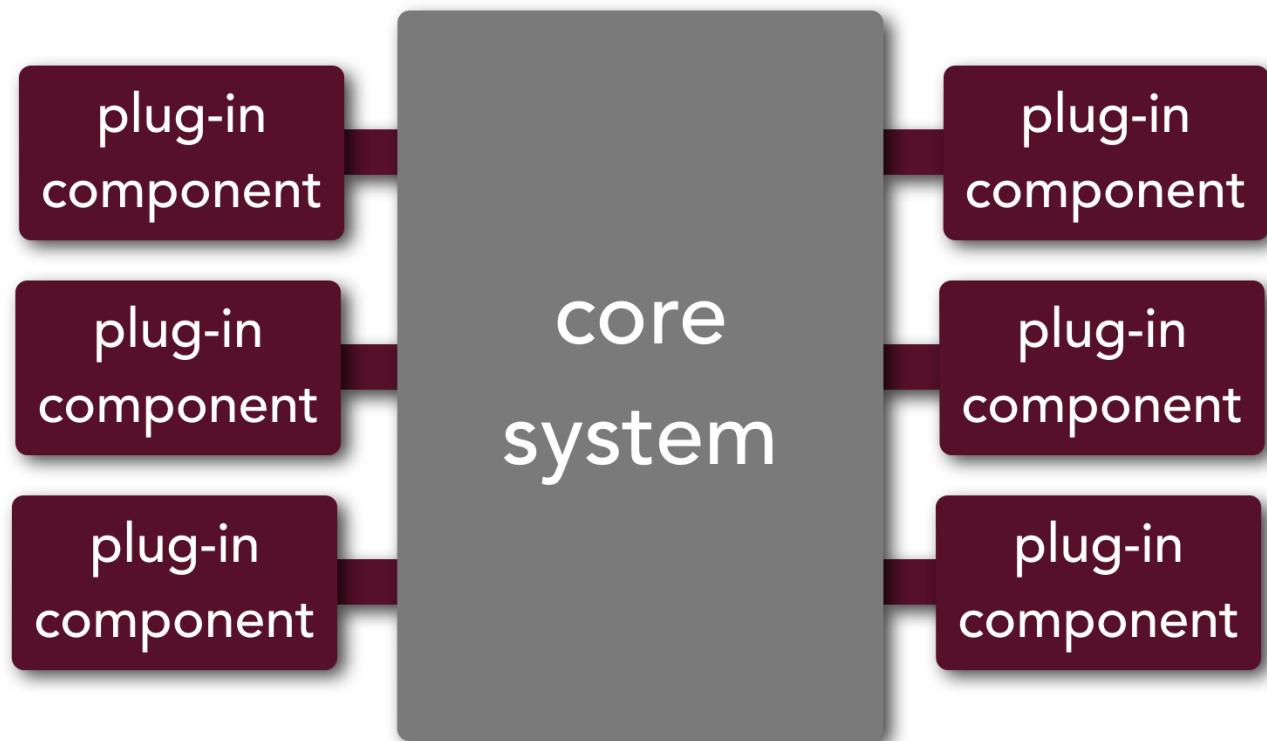
Each component is functionally cohesive, with good interfaces between them and low coupling.



the entire system



Microkernel

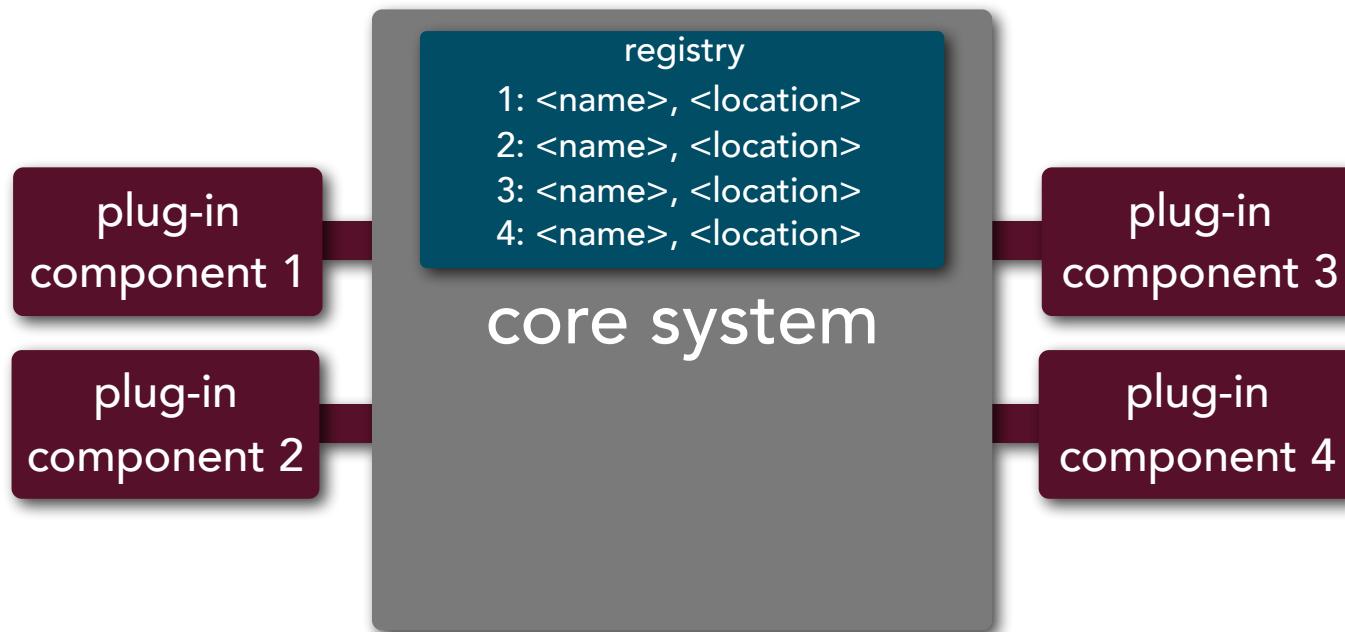


Microkernel

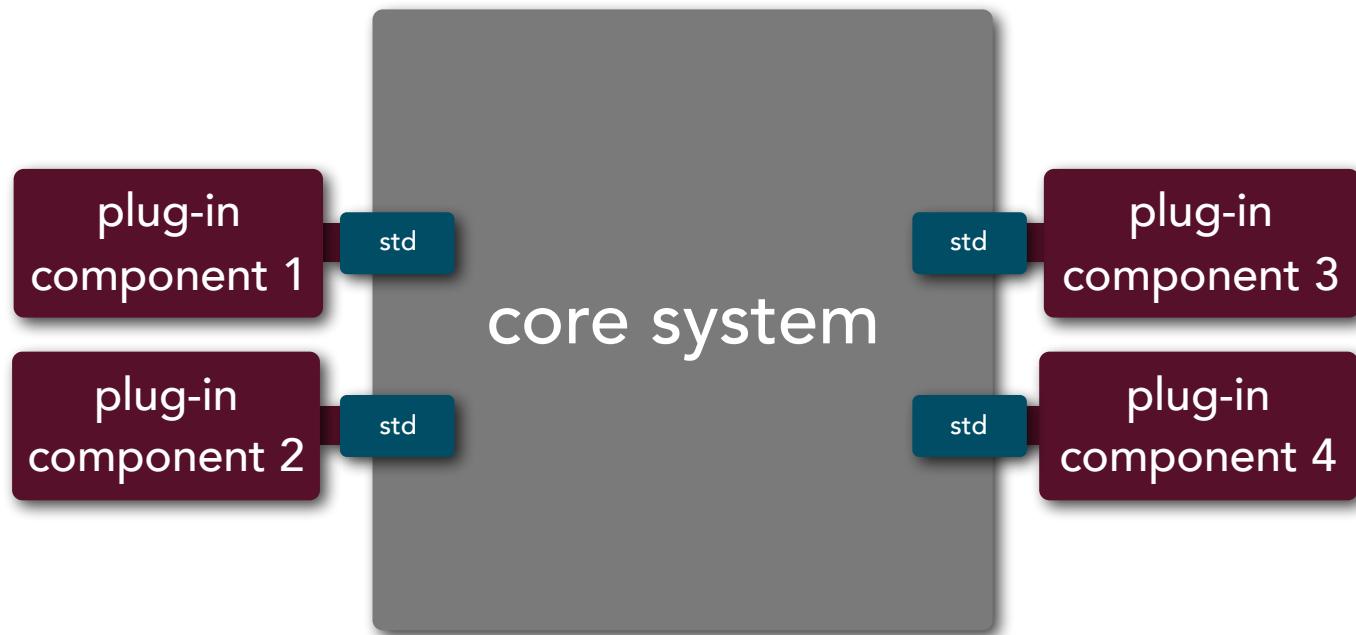
claims processing



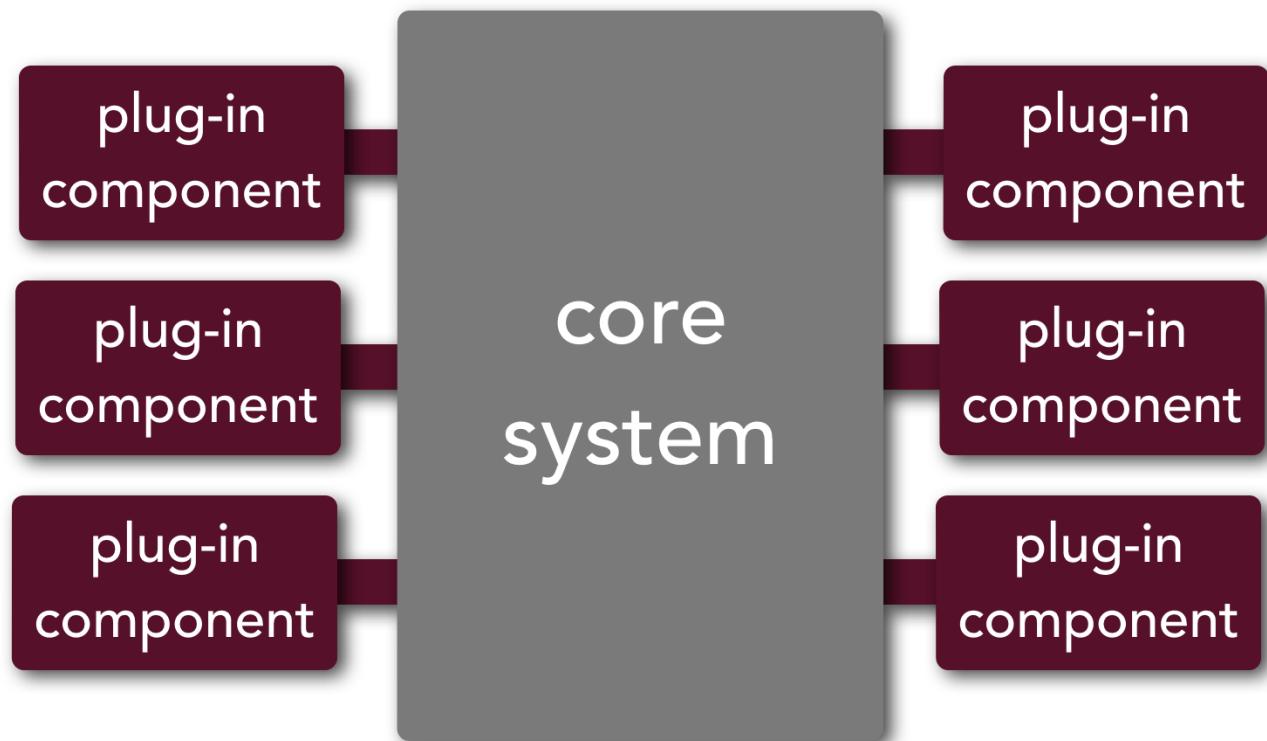
Microkernel Registry



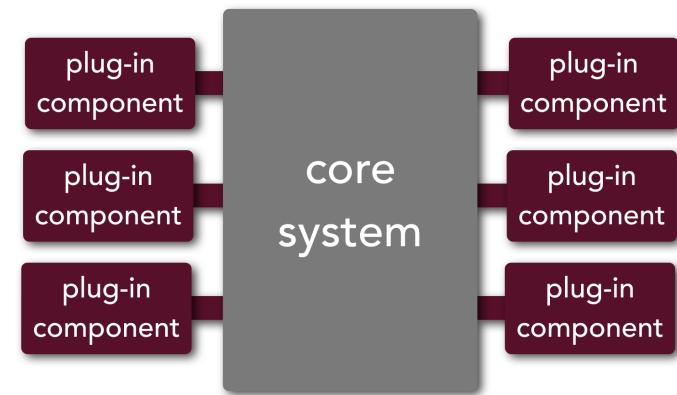
Contracts



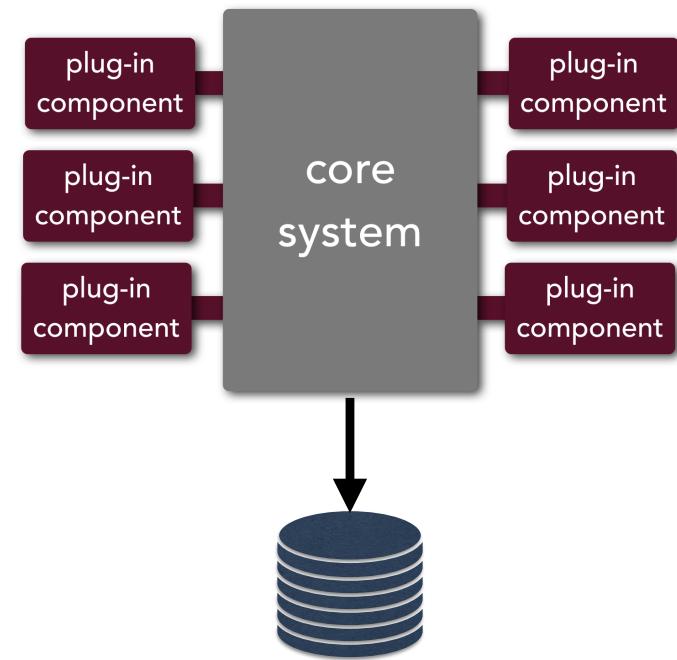
Microkernel



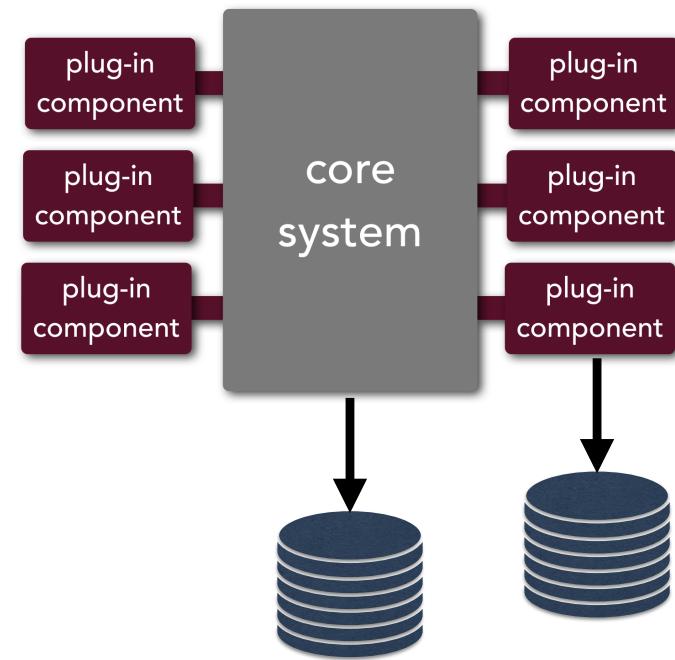
Microkernel



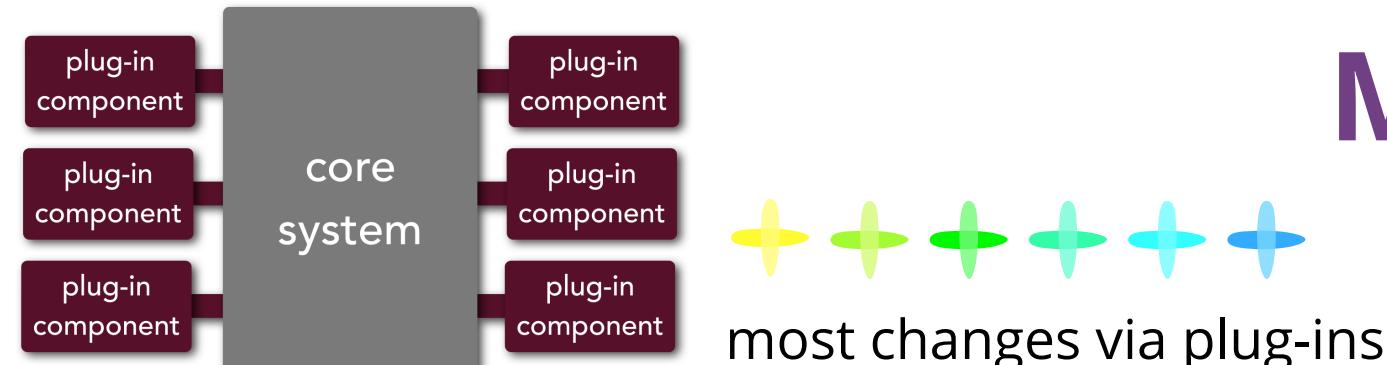
Microkernel



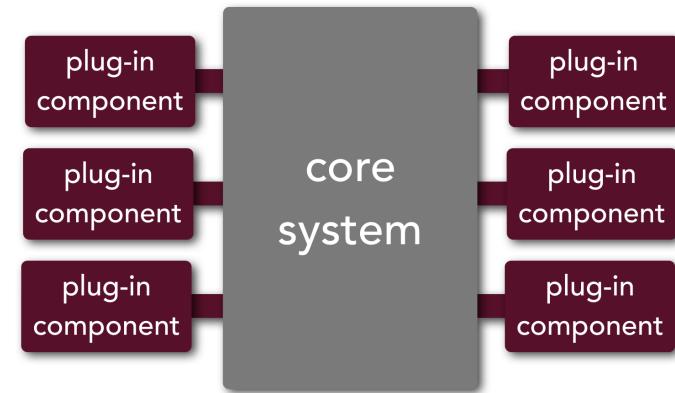
Microkernel



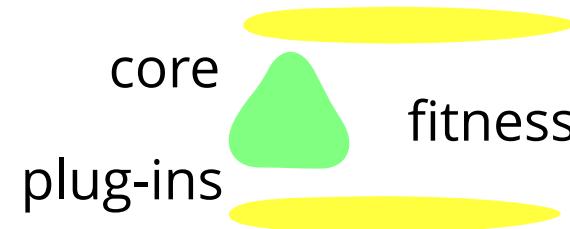
Microkernel



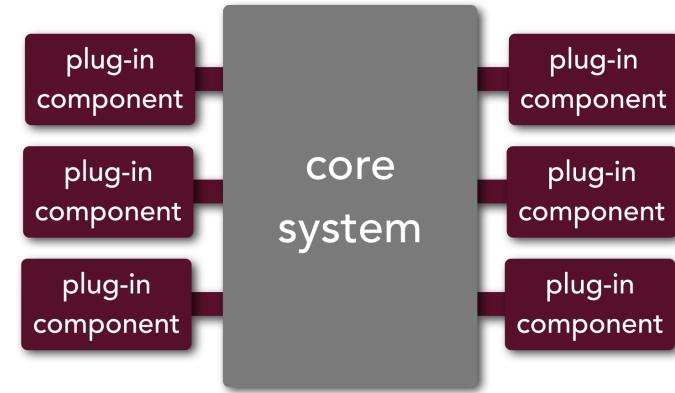
Microkernel



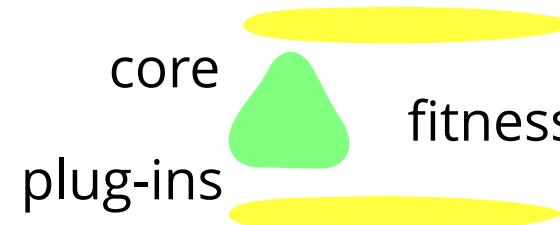
most changes via plug-ins



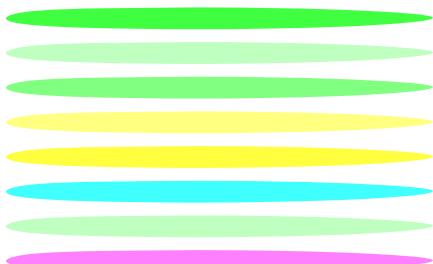
Microkernel



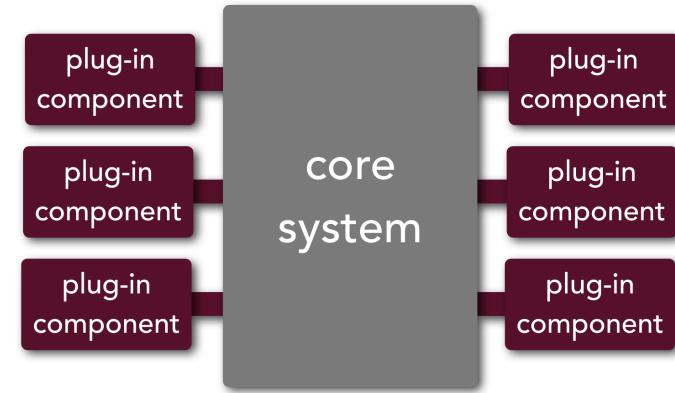
most changes via plug-ins



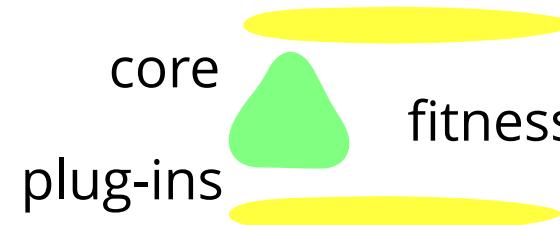
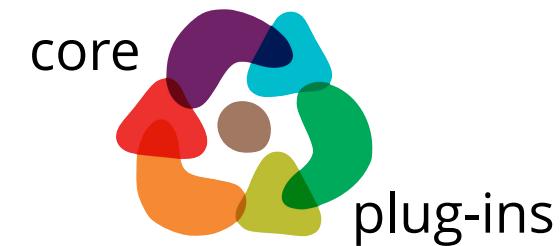
coupling well defined by architectural pattern



Microkernel

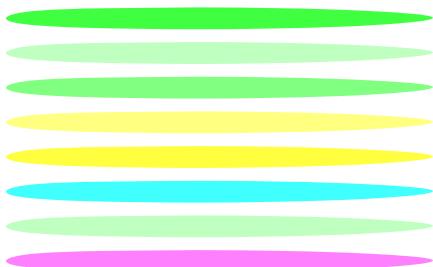


most changes via plug-ins

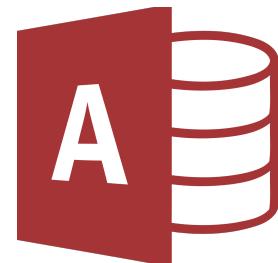


coupling well defined by architectural pattern

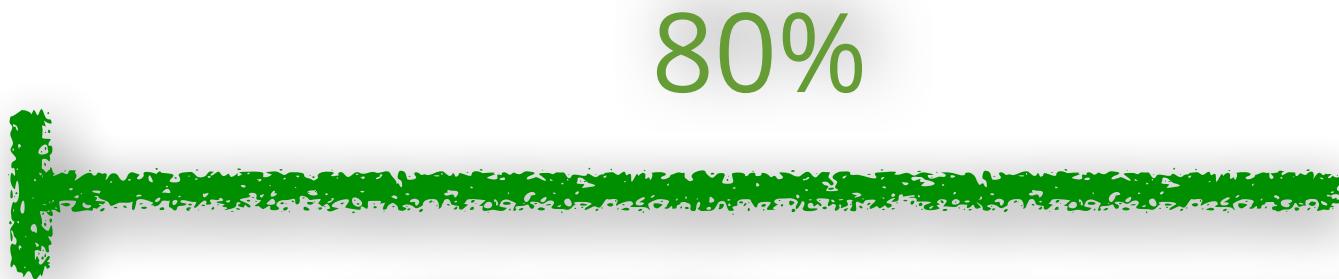
beware the Last 10% Trap



Last 10% Trap

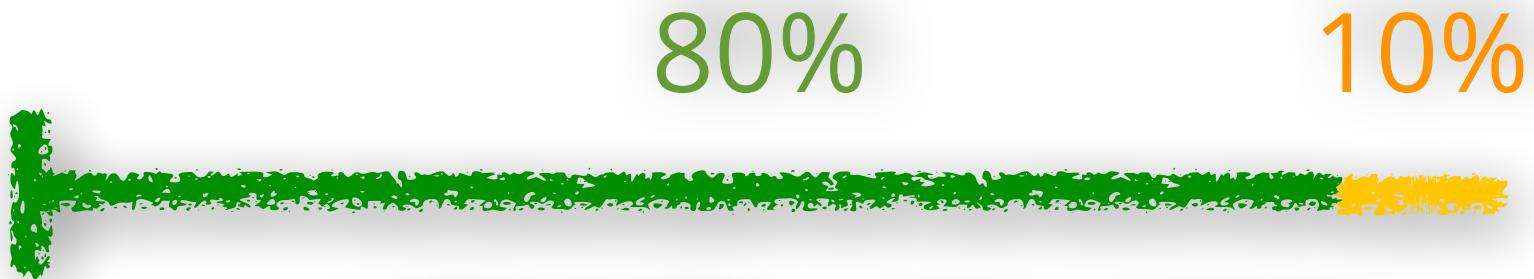


Last 10% Trap



what the user wants

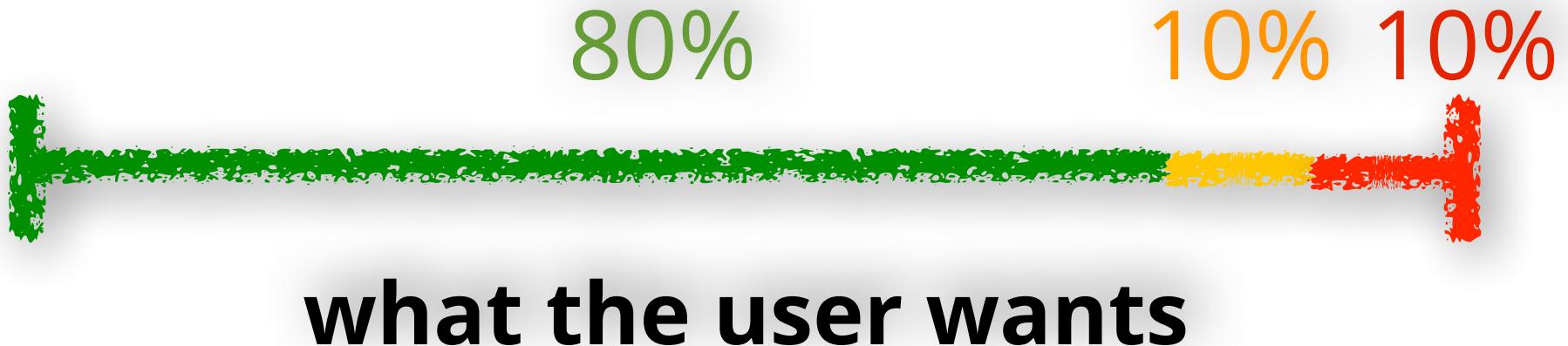
Last 10% Trap



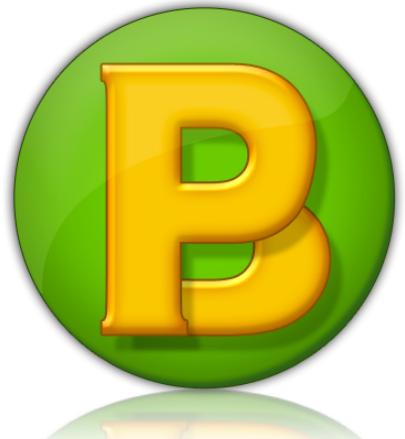
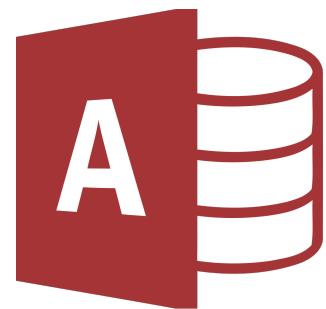
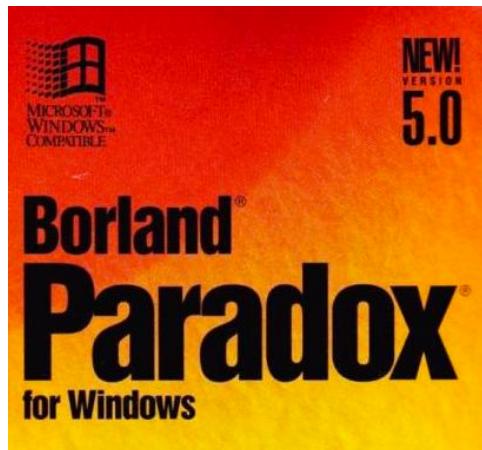
what the user wants

Last 10% Trap

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



What Happened to the 4GLs?

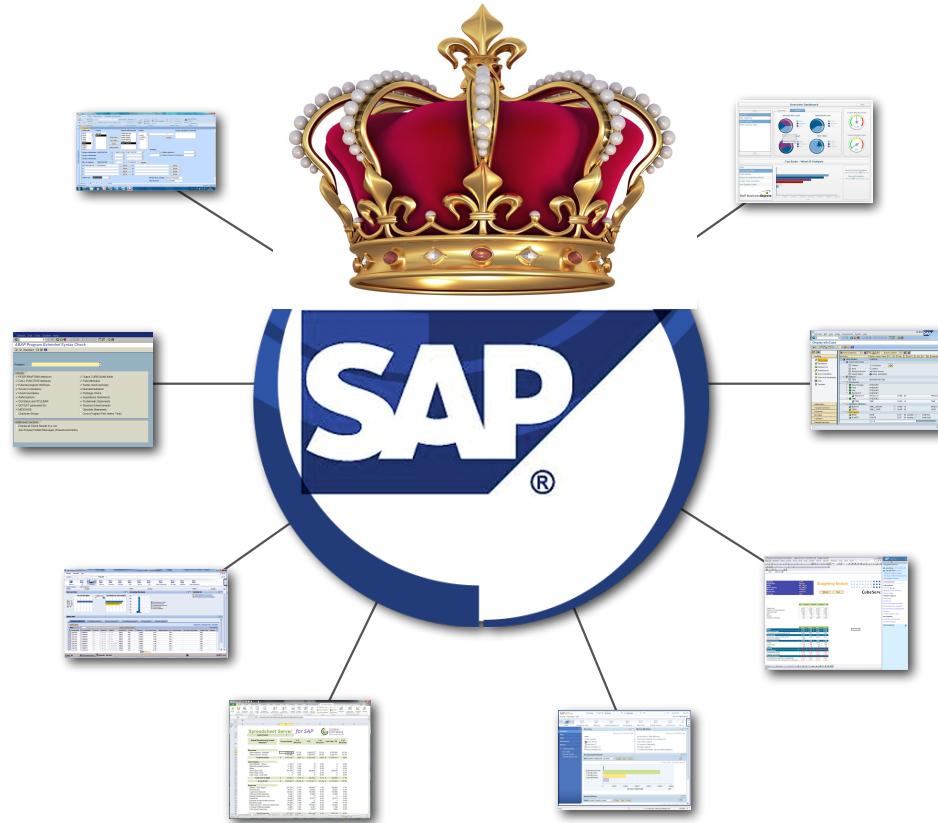


What Happened to the 4GLs?



DSL

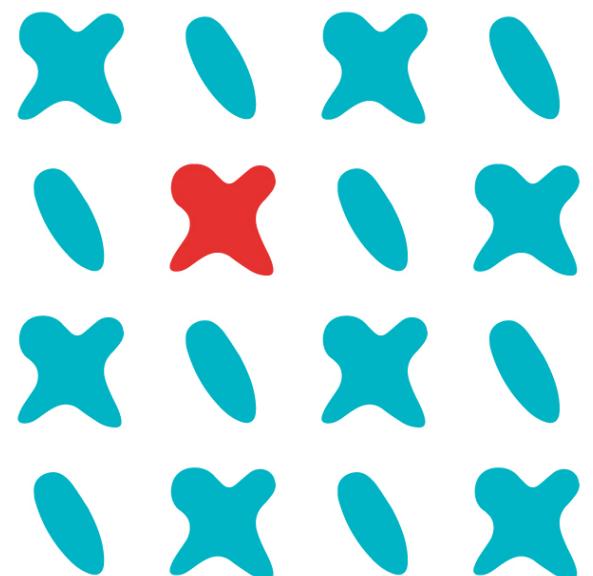
see also: Vendor King



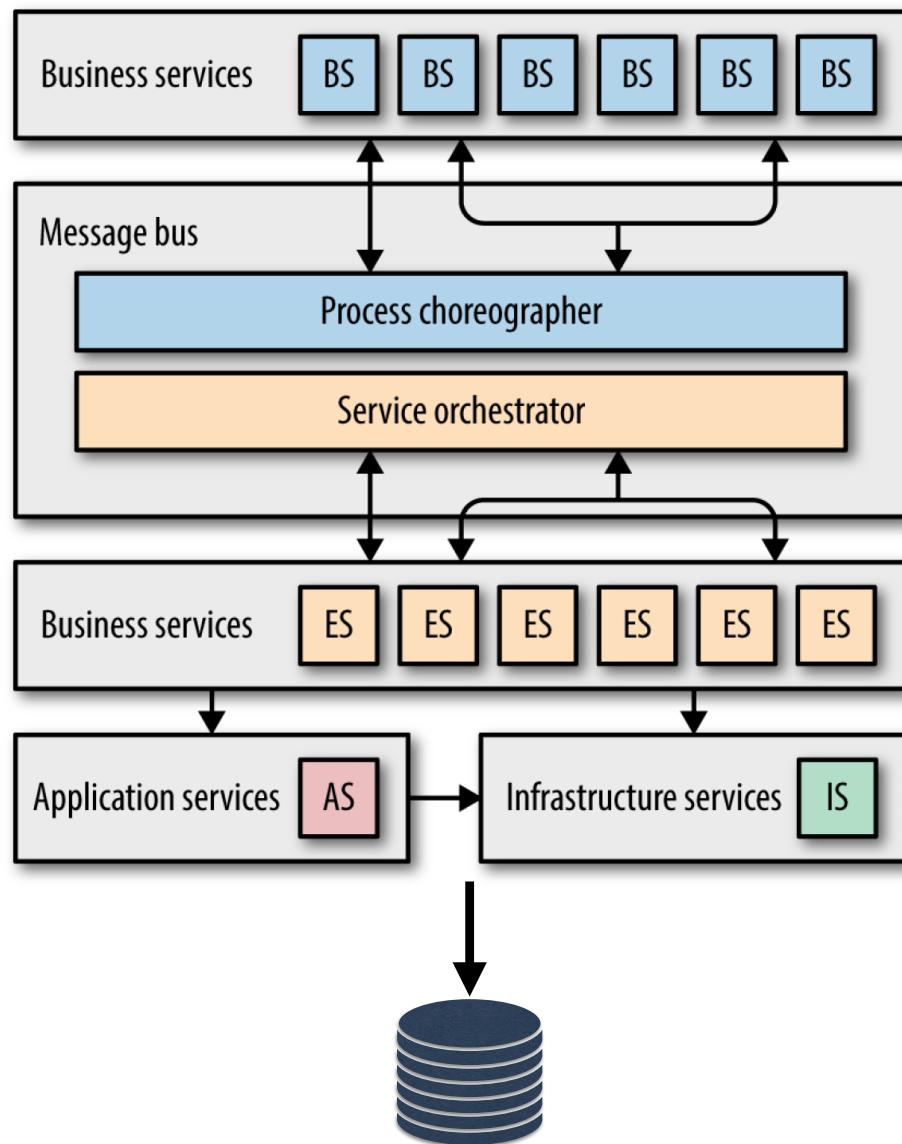


Be careful of the *Last
10% Trap* when choosing
tools & frameworks.

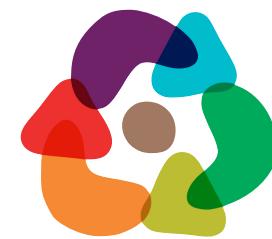
Service-oriented Architectures



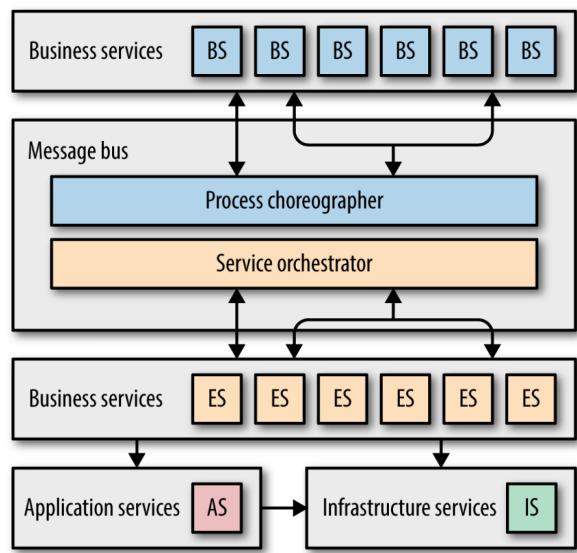
ESB-driven SOA



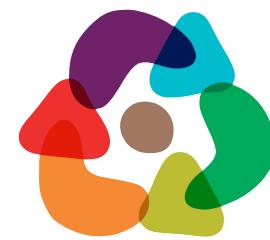
ESB-driven SOA



the entire system

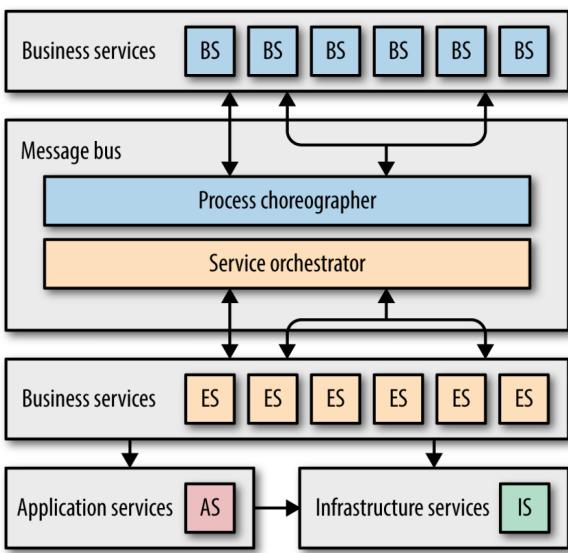


ESB-driven SOA

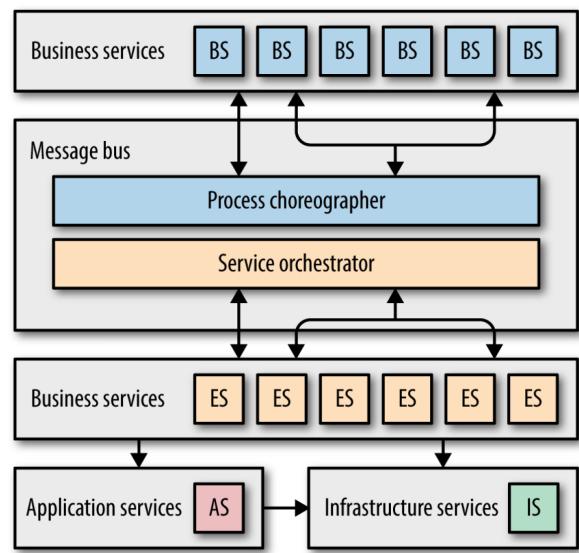


the entire system

virtually impossible



ESB-driven SOA



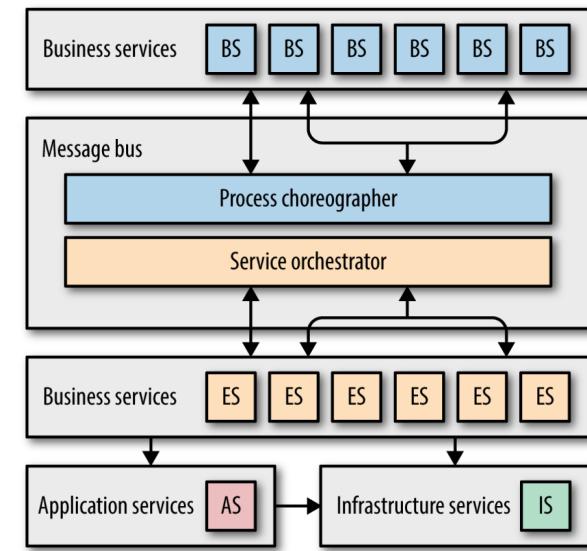
⊕ ⊕ ⊕ ⊕ ⊕ ⊕
virtually impossible



the entire system

extremely difficult

ESB-driven SOA

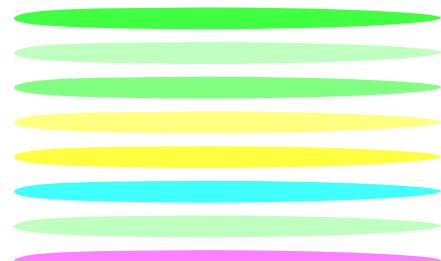


virtually impossible



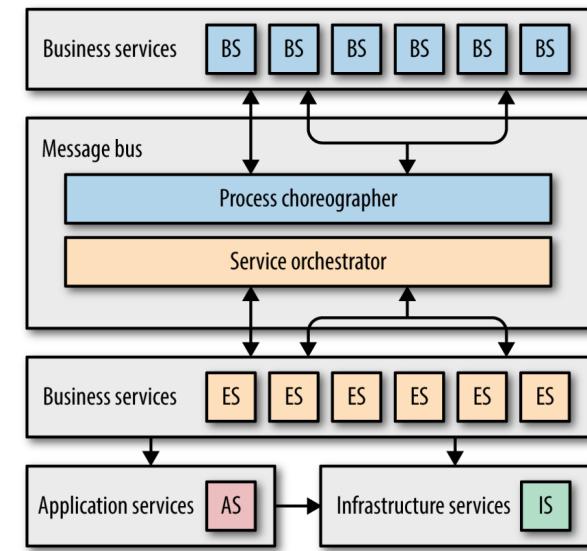
the entire system

extremely difficult



extravagant taxonomy makes sense if reuse is primary goal

ESB-driven SOA

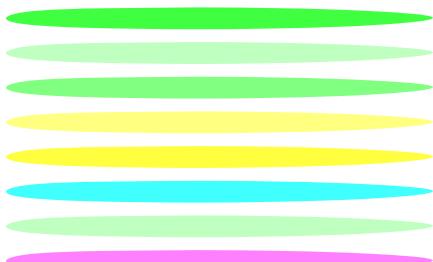


virtually impossible



the entire system

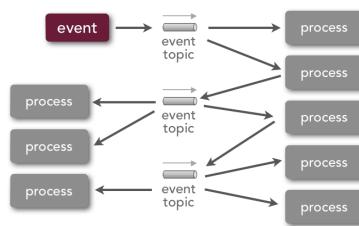
extremely difficult



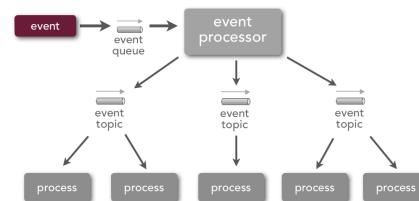
extravagant taxonomy makes sense if reuse is primary goal

extremely coupled in inappropriate ways.

Event-driven Architectures

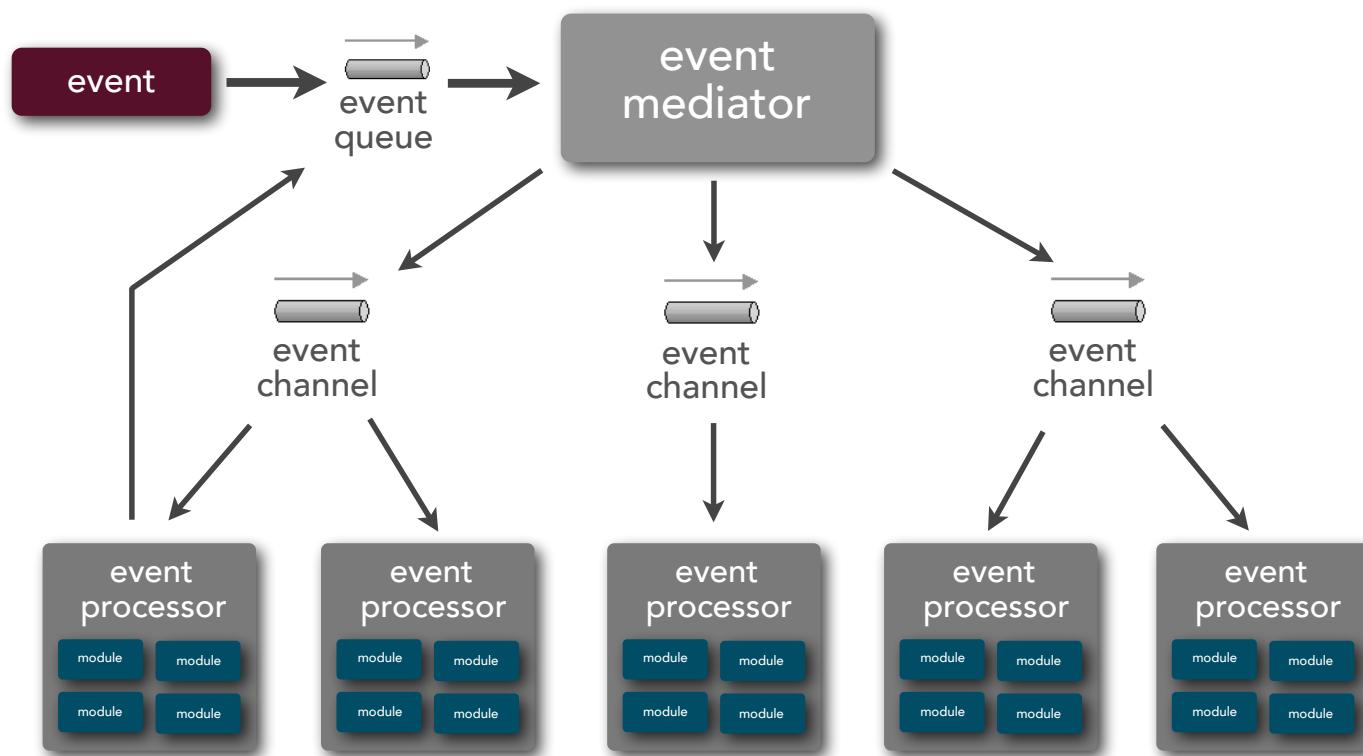


broker topology

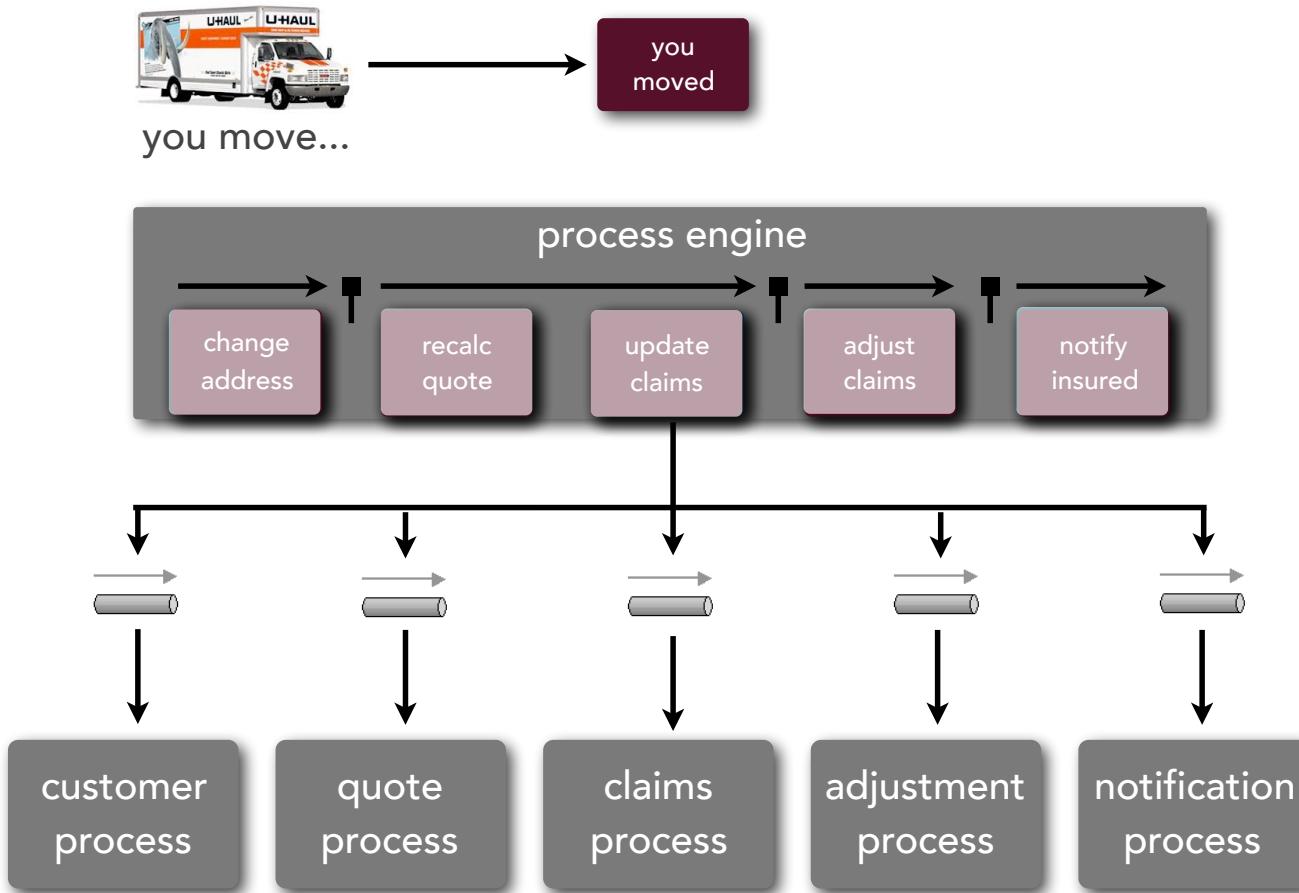


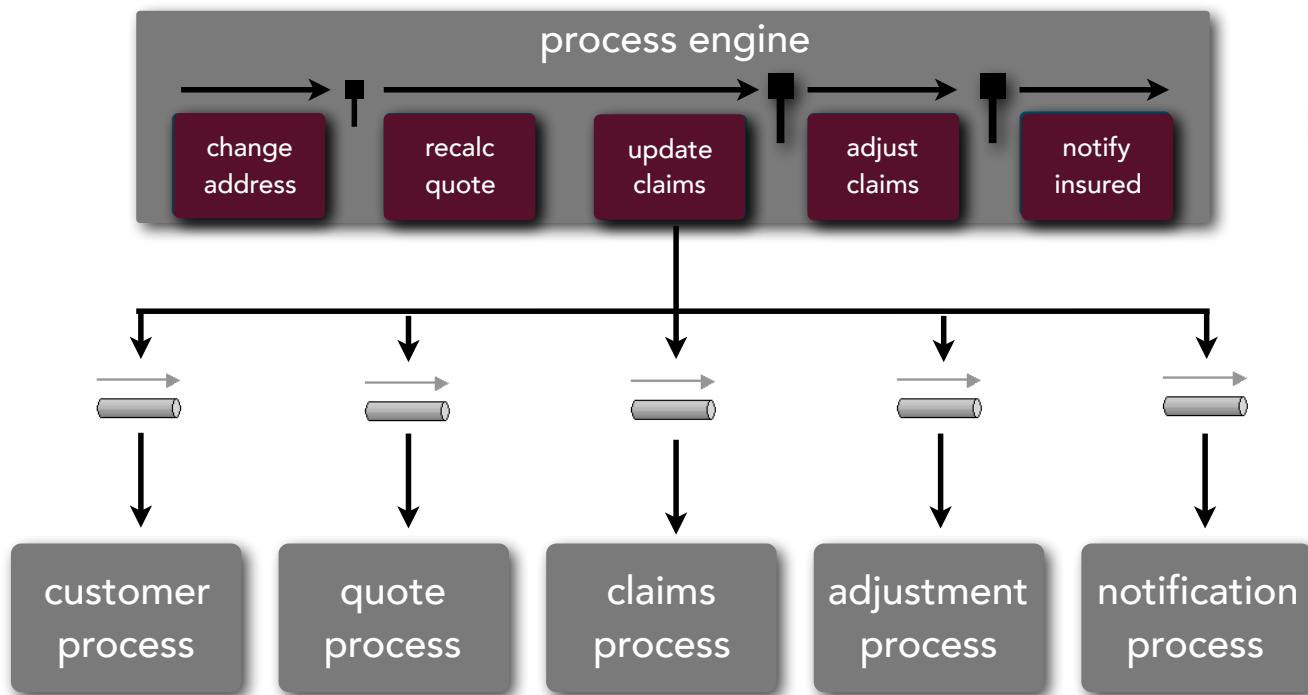
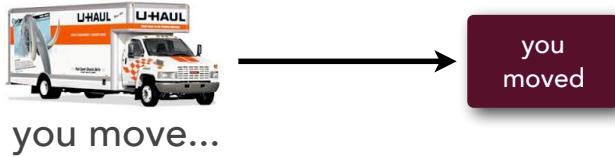
mediator topology

Mediator EDA

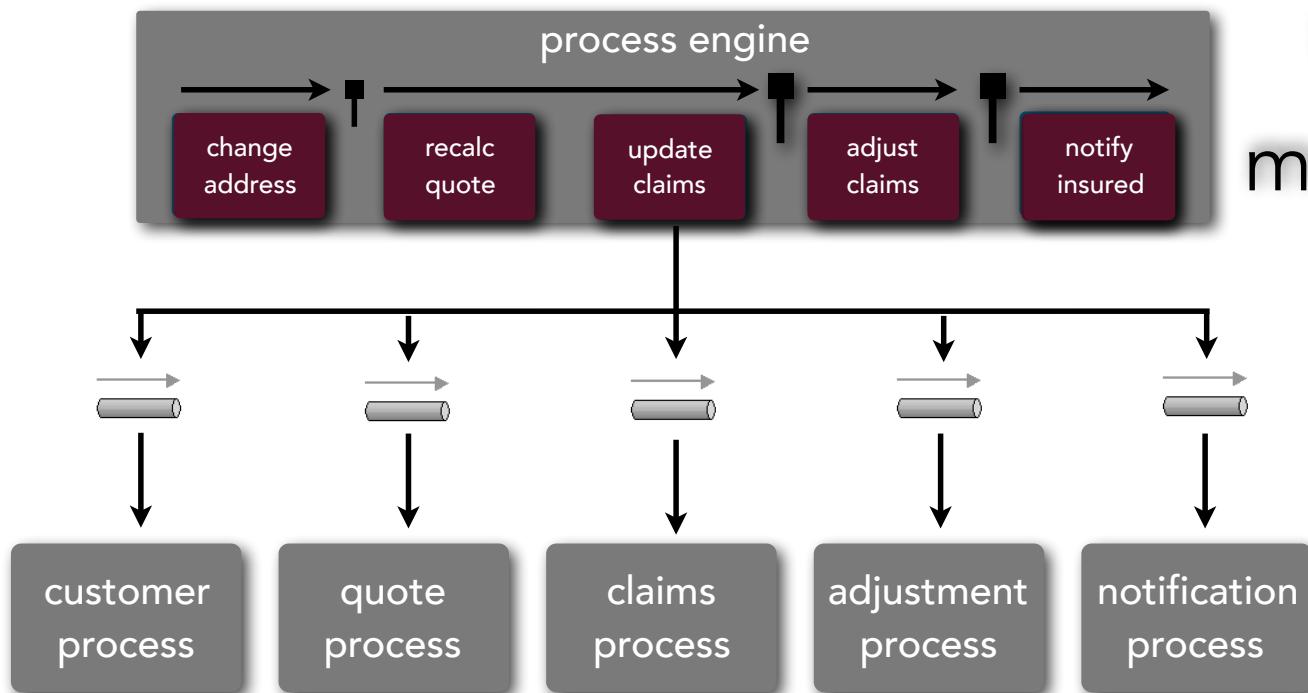
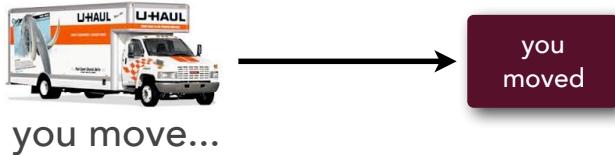


Mediator Message Flow





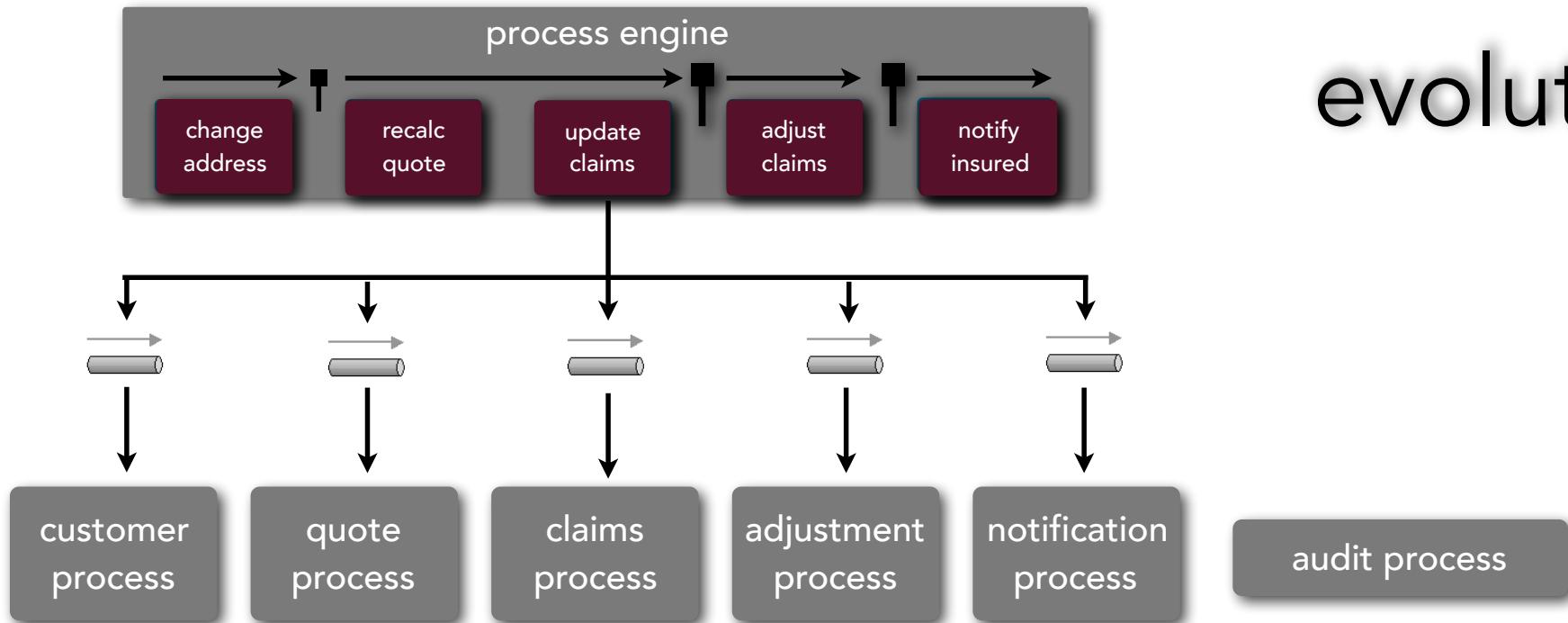
Error handling?



Unified notification message to customer?



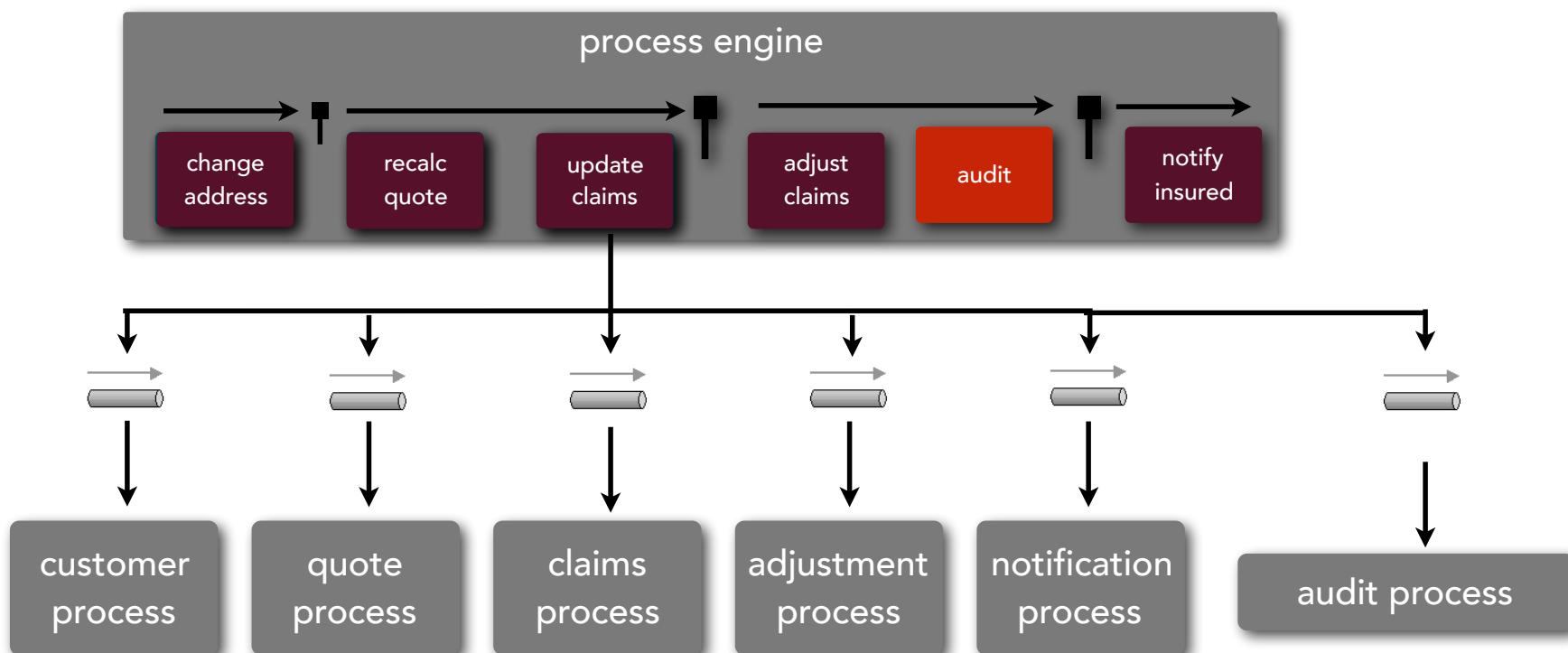
you move...



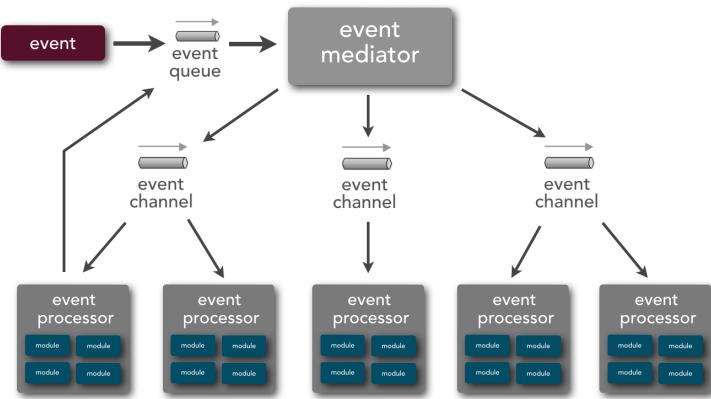


you move... → you moved

evolution

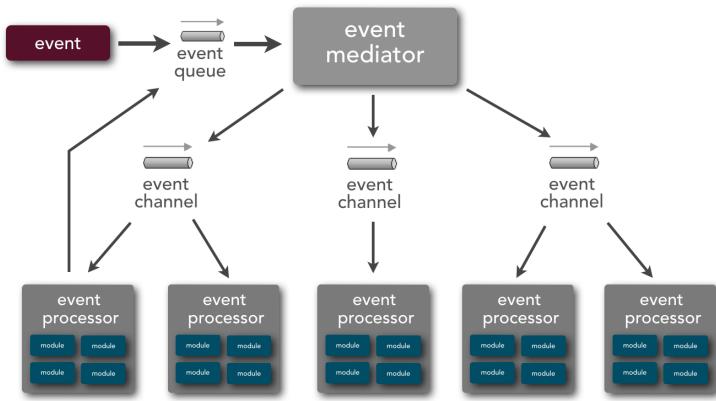


Mediator EDA

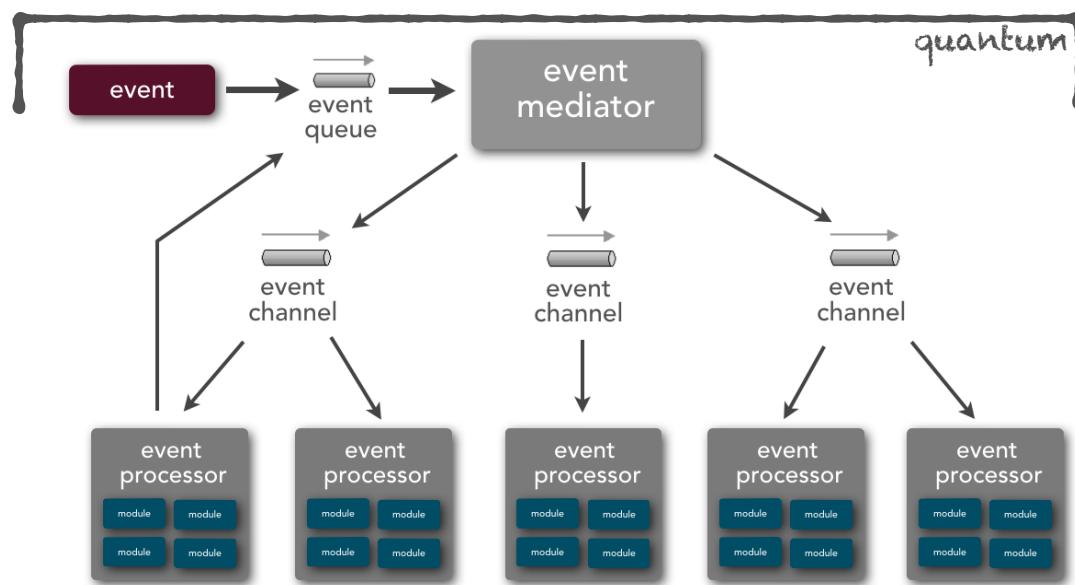


services + meditator

Mediator EDA

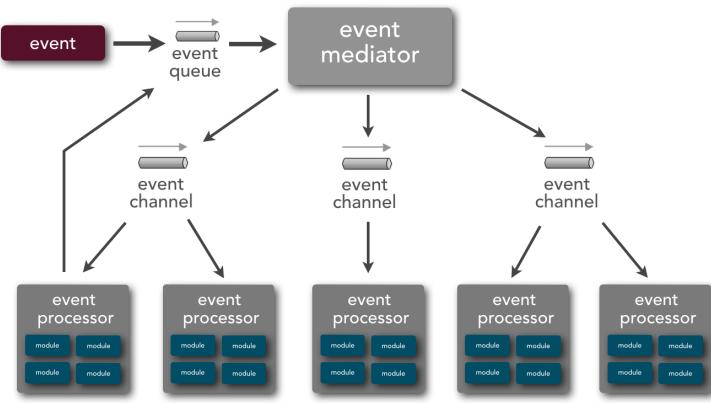


services + mediator



the entire system

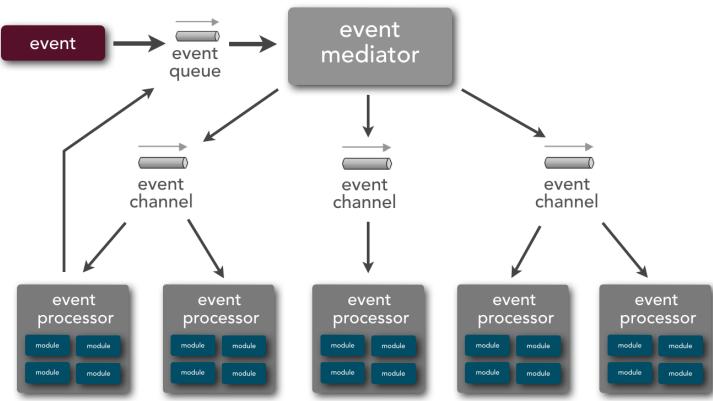
Mediator EDA



services + mediator

similar to broker

Mediator EDA



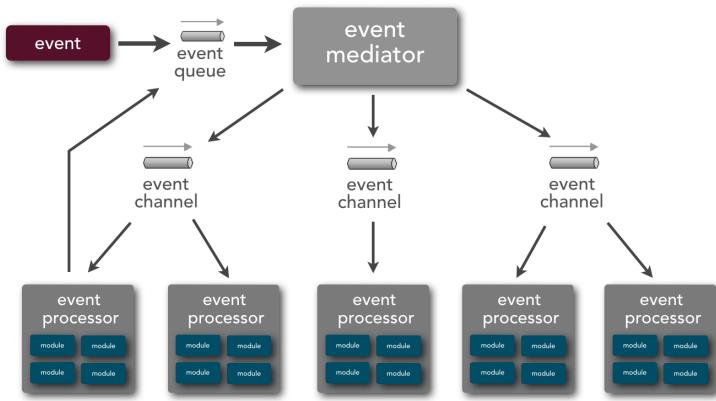
services + mediator

⊕ ⊕ ⊕ ⊕ ⊕ ⊕ similar to broker



easier integration testing via mediator

Mediator EDA

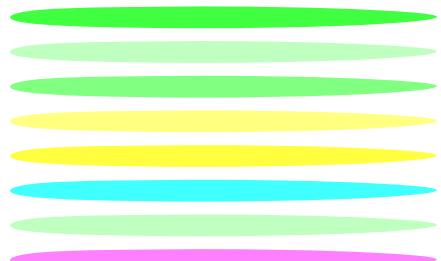


services + mediator

⊕ ⊕ ⊕ ⊕ ⊕ ⊕ similar to broker

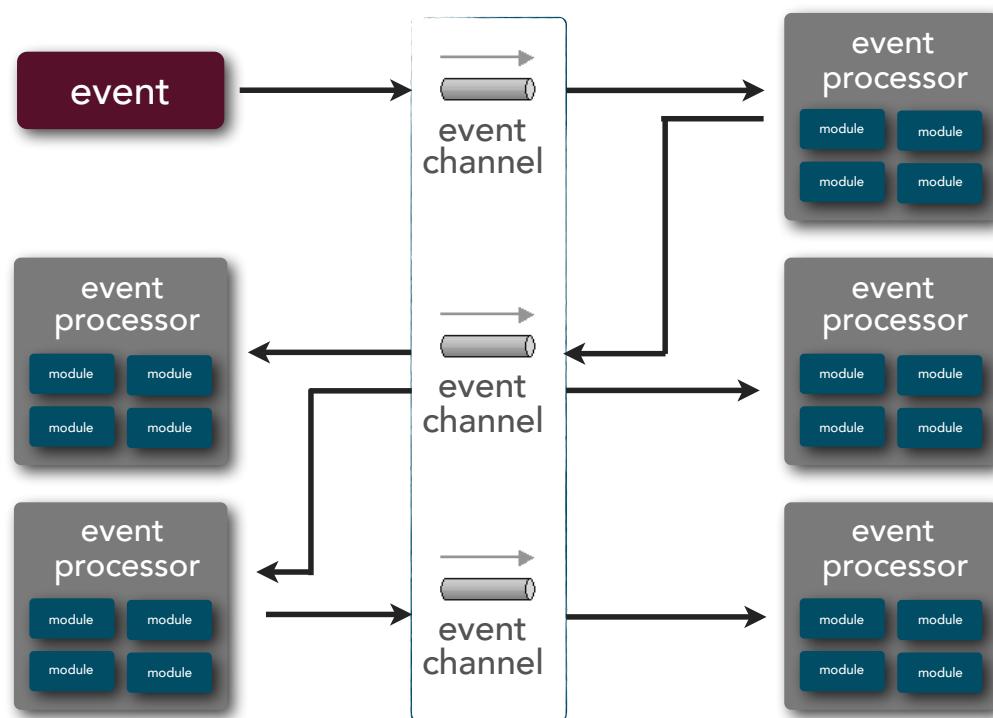


easier integration testing via mediator

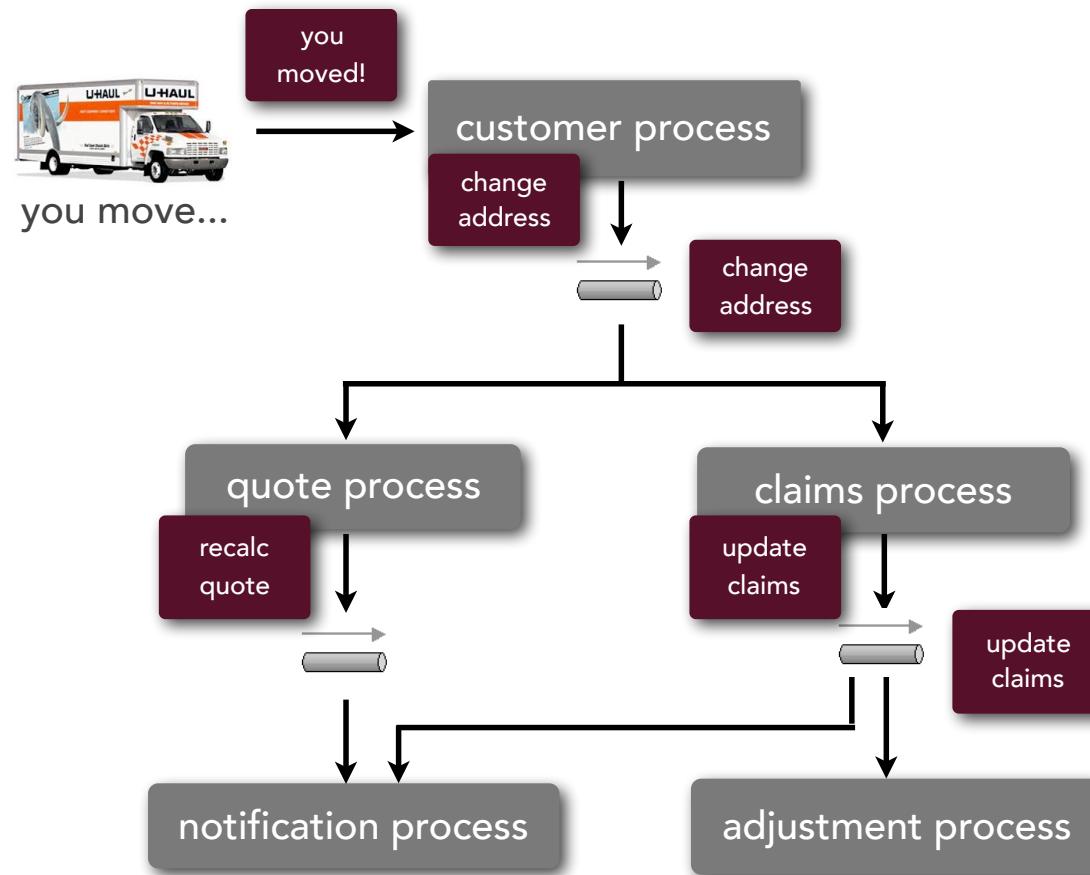


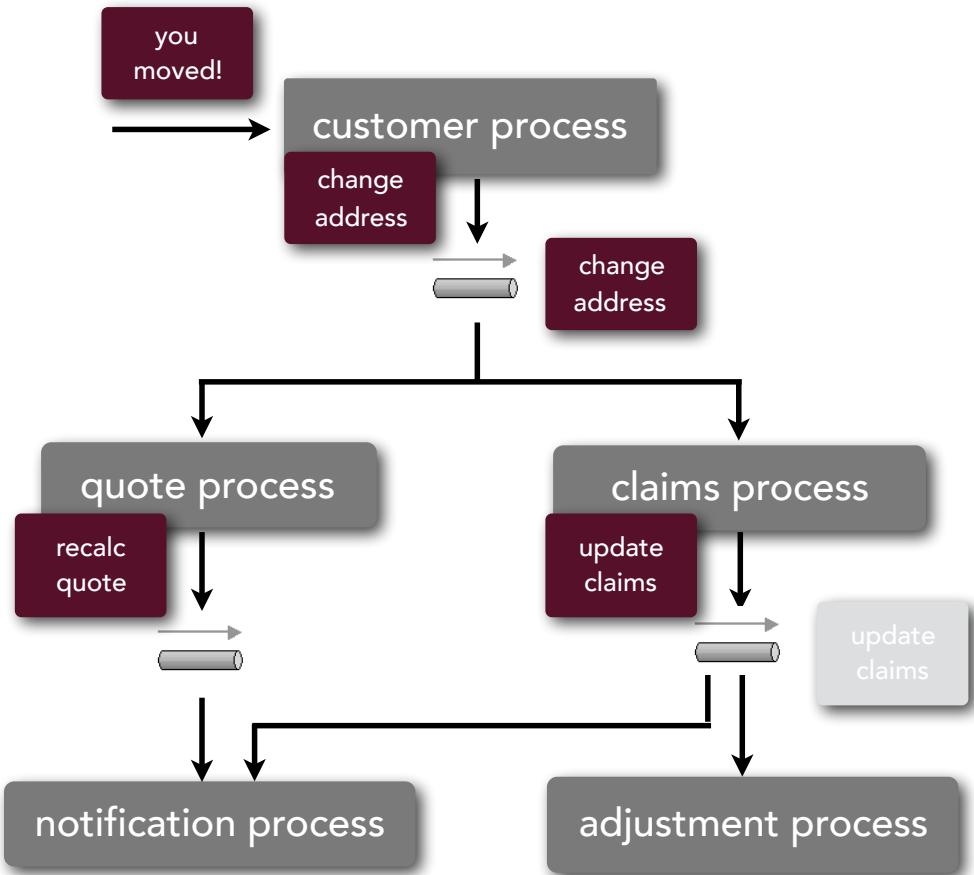
increased semantic coupling

Broker



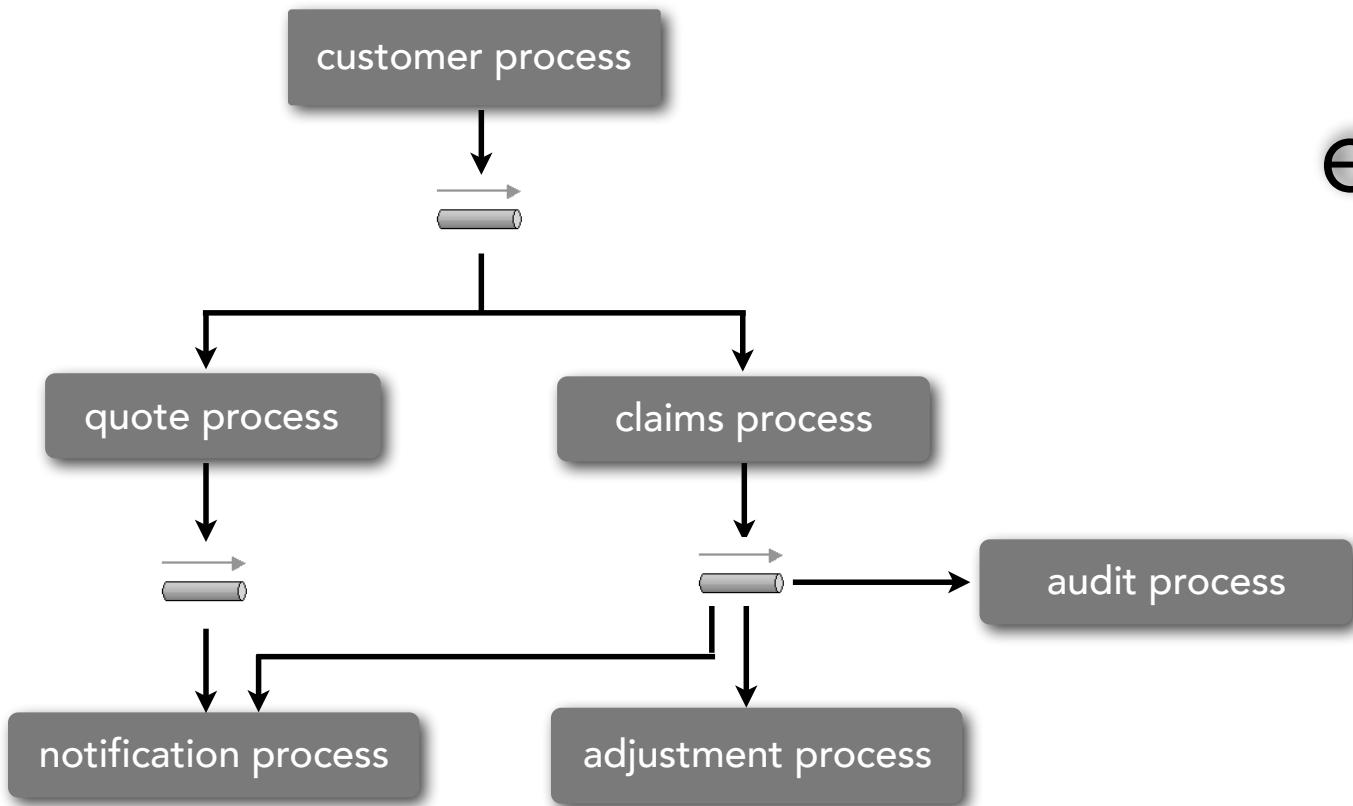
Broker Message Passing

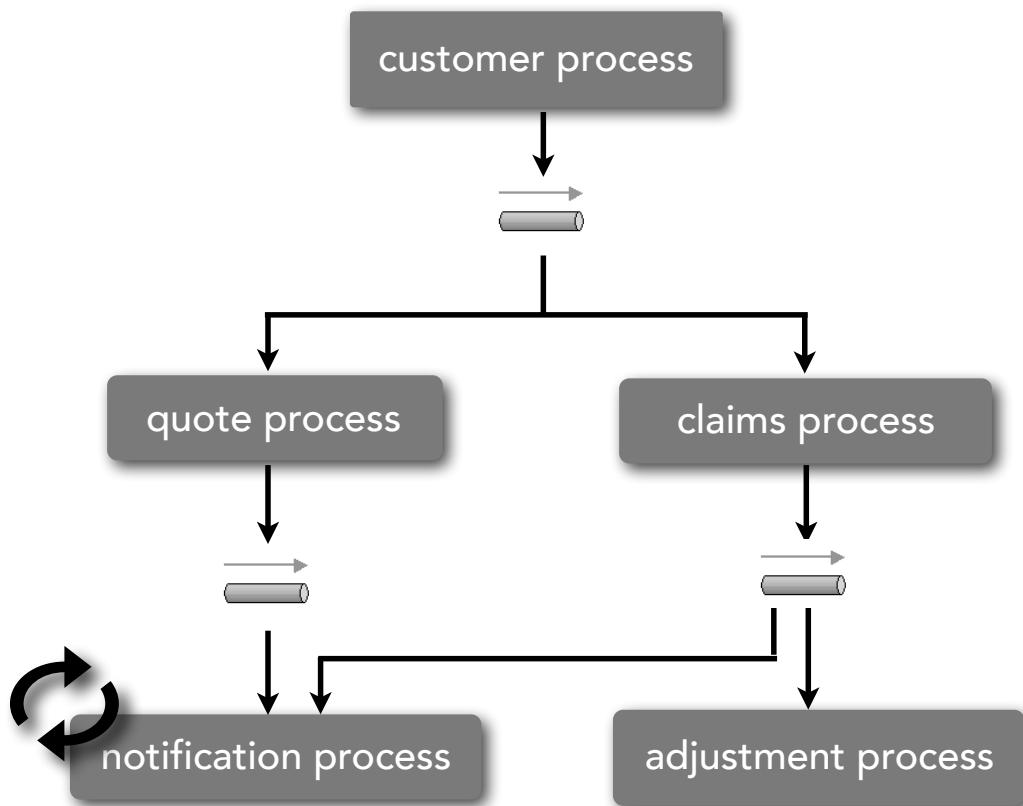




Error handling?

evolution



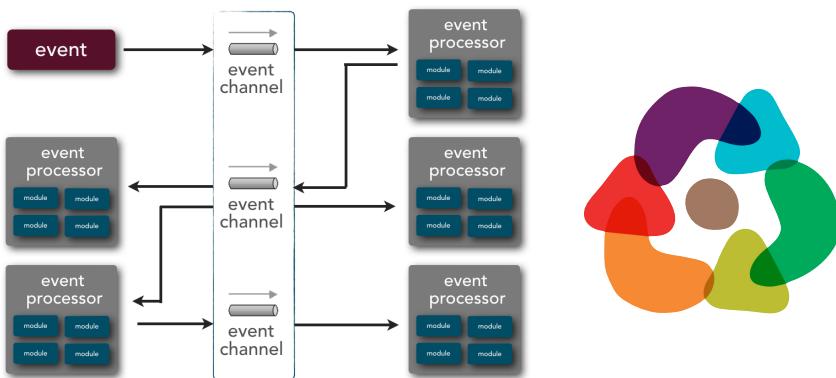


Unified notification
message to customer?

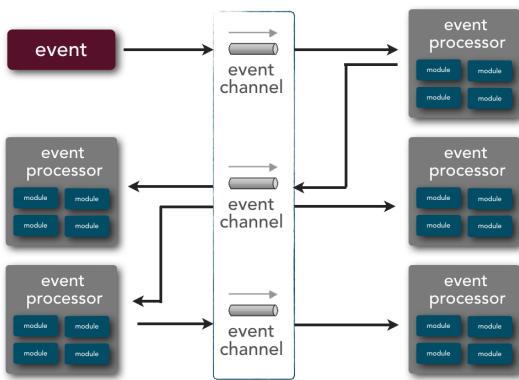
In asynchronous systems,
coordination happens at the end

Broker EDA

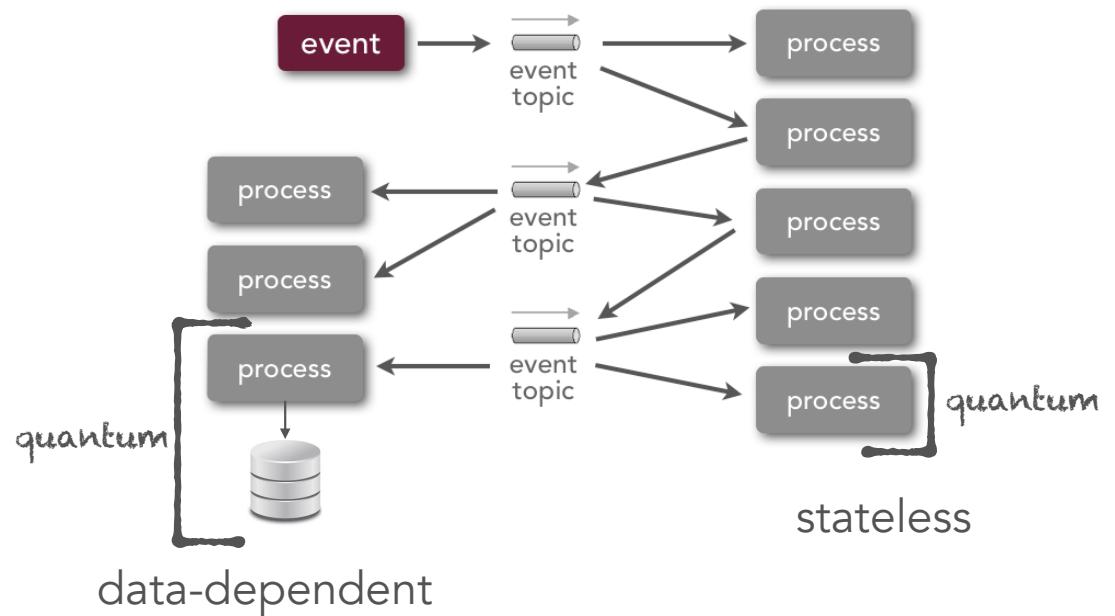
decoupled events

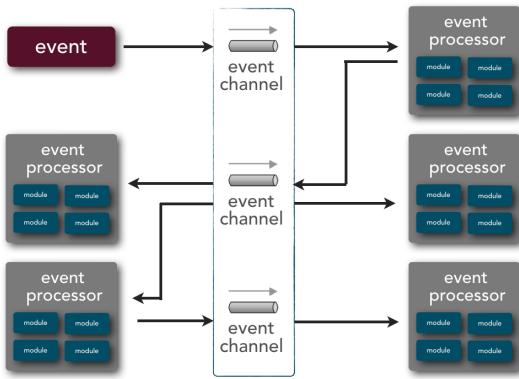


Broker EDA



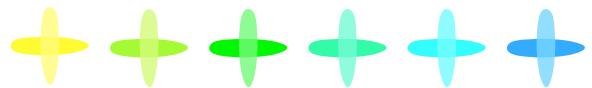
decoupled events





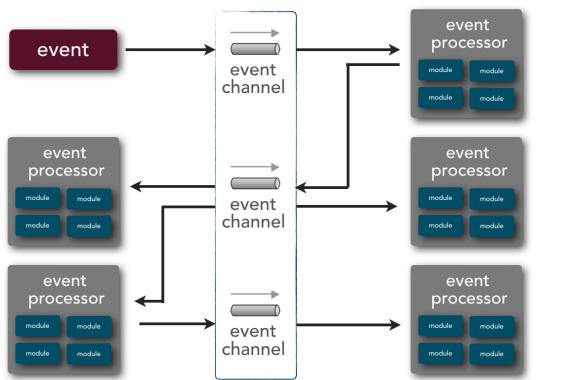
Broker EDA

decoupled events



deployment pipeline integration challenging

Broker EDA



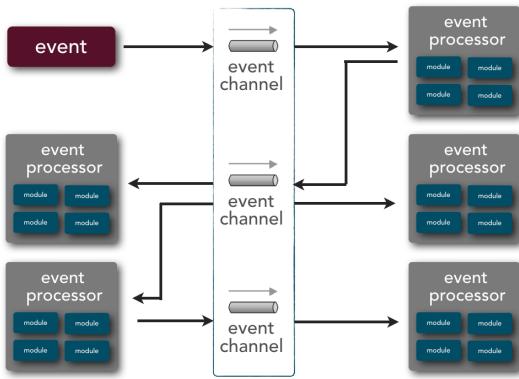
decoupled events



deployment pipeline integration challenging



easy atomic
difficult holistic

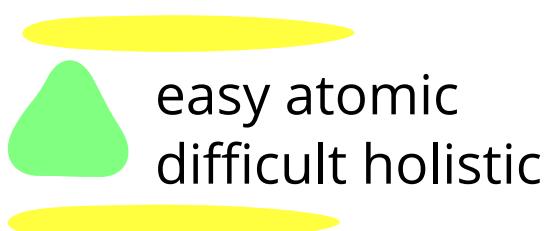


decoupled events

Broker EDA



deployment pipeline integration challenging

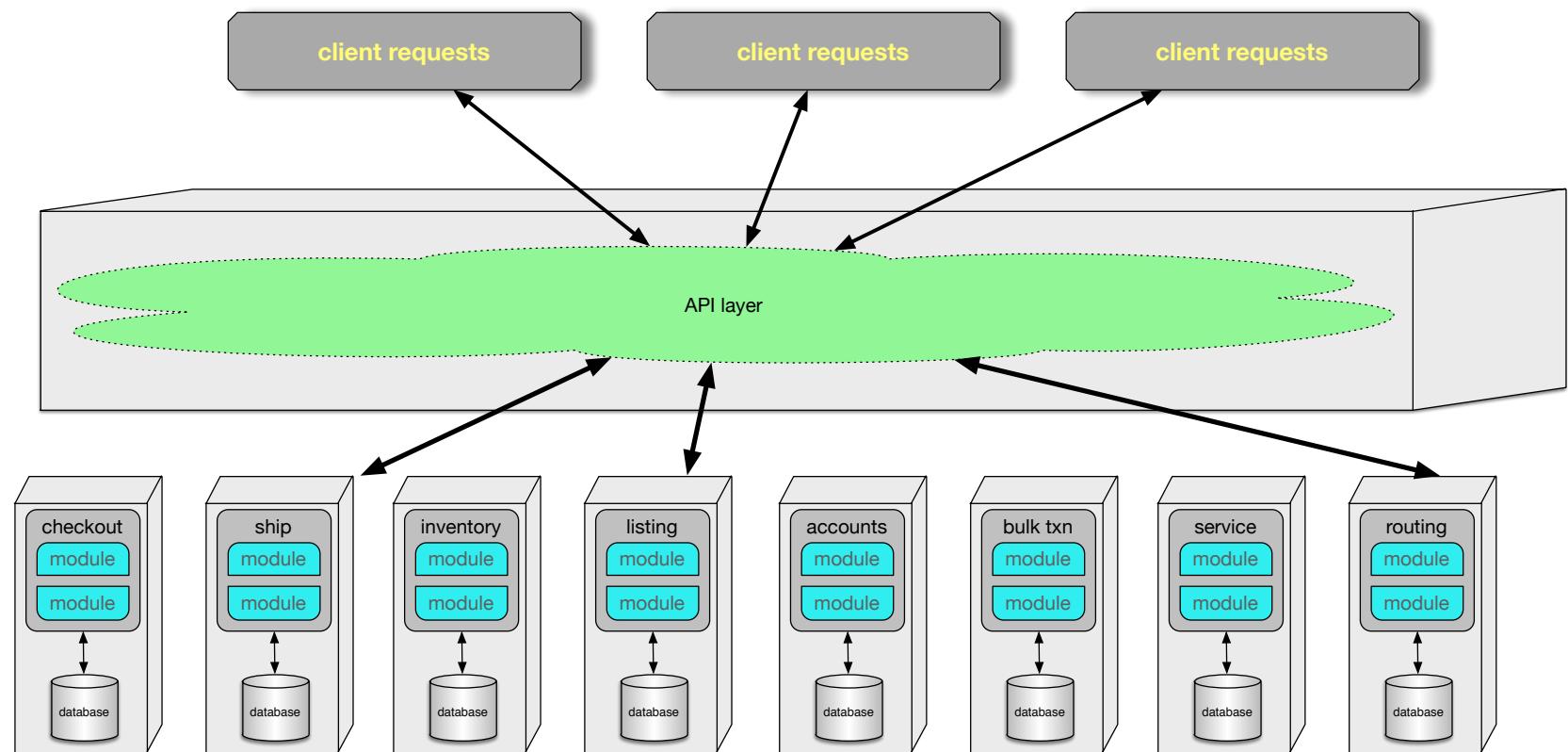


structurally decoupled

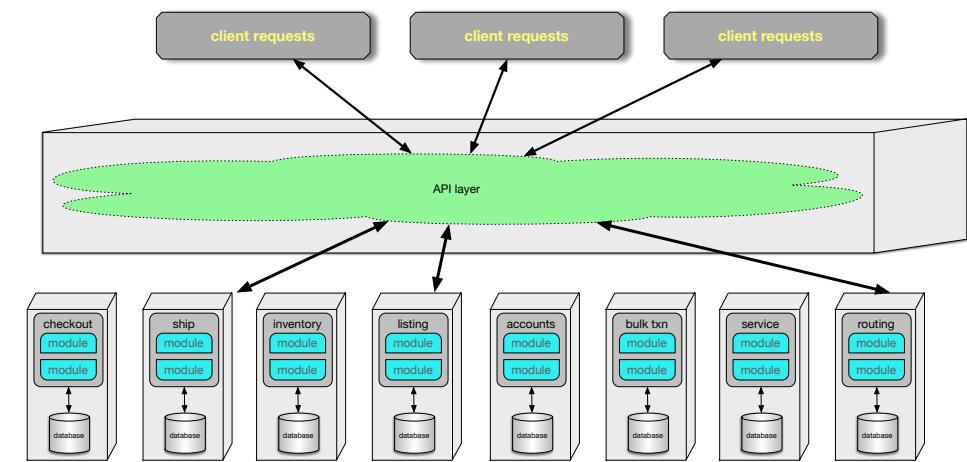


semantically decoupled

Microservices

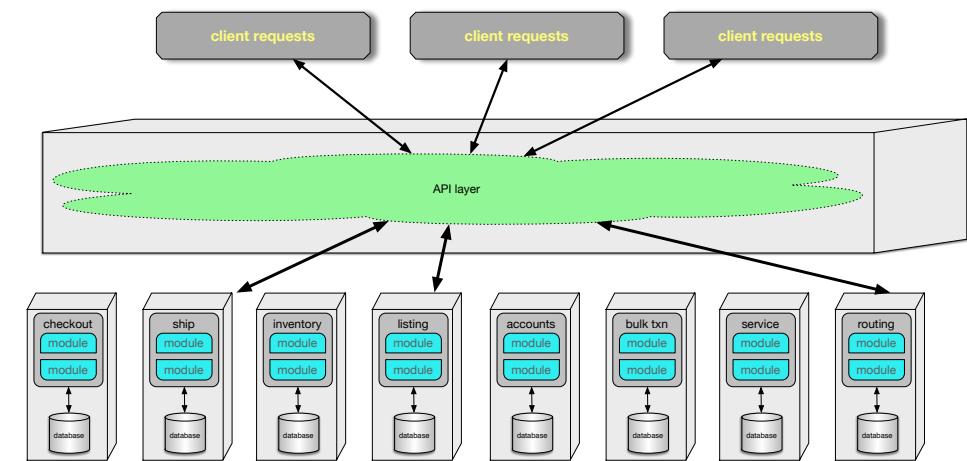


Microservices



extremely fine grained

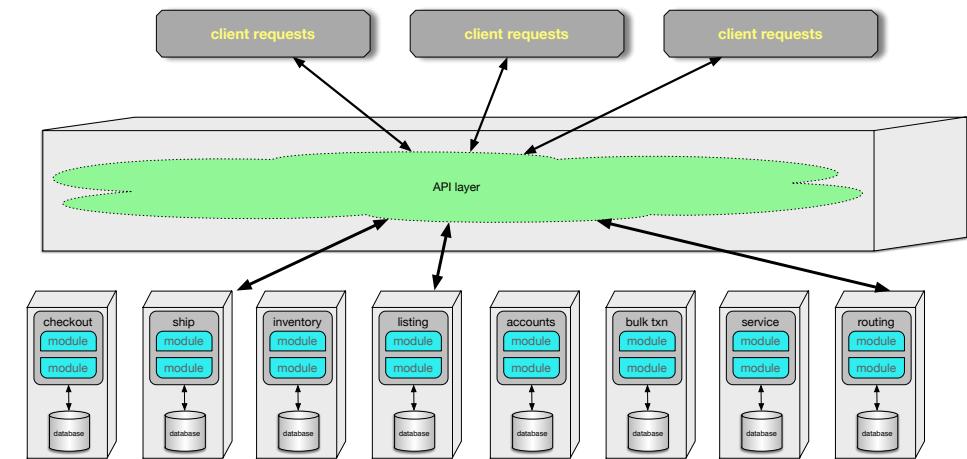
Microservices



extremely fine grained

⊕ ⊕ ⊕ ⊕ ⊕ deployment pipeline considered standard
21st century DevOps practices

Microservices



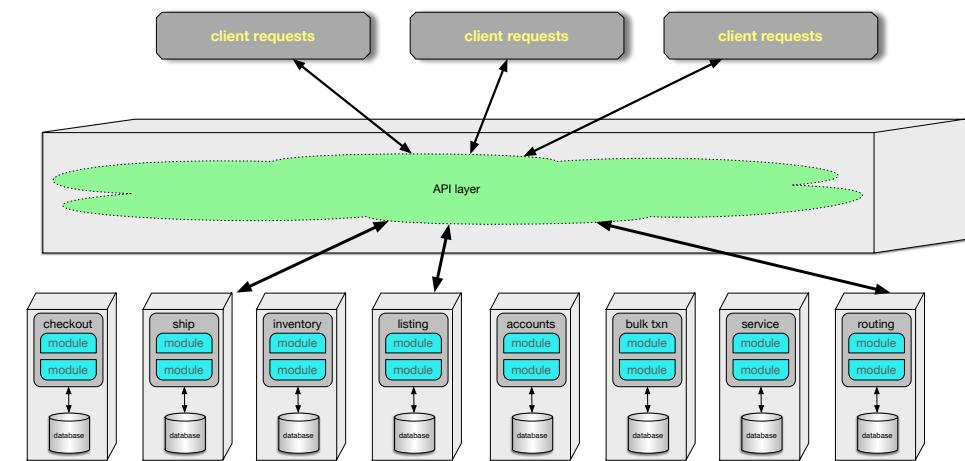
extremely fine grained

⊕ ⊕ ⊕ ⊕ ⊕ deployment pipeline considered standard
21st century DevOps practices



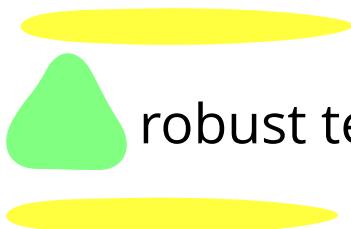
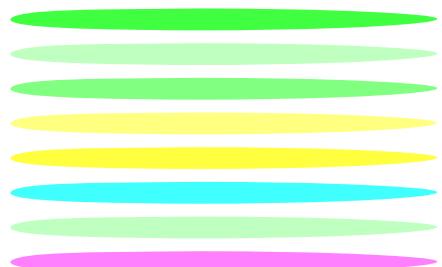
robust testing & fitness function culture

Microservices



extremely fine grained

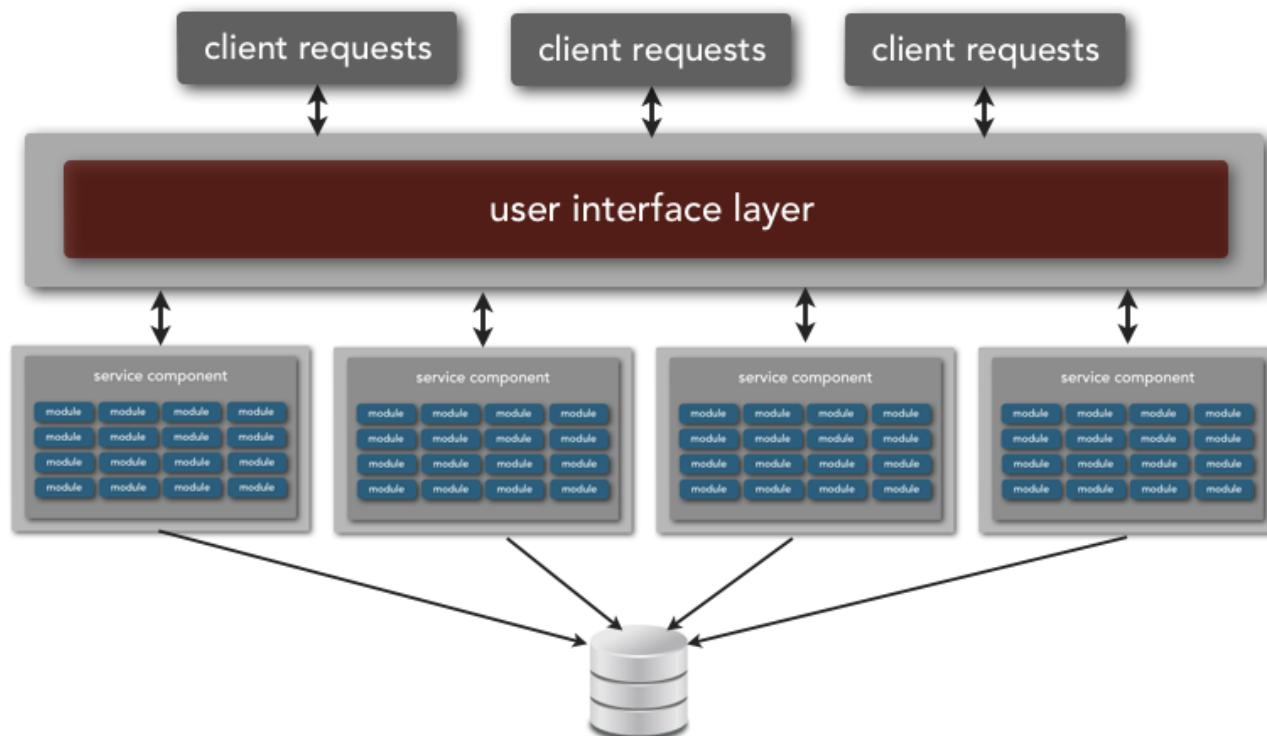
⊕ ⊕ ⊕ ⊕ ⊕ deployment pipeline considered standard
21st century DevOps practices



robust testing & fitness function culture

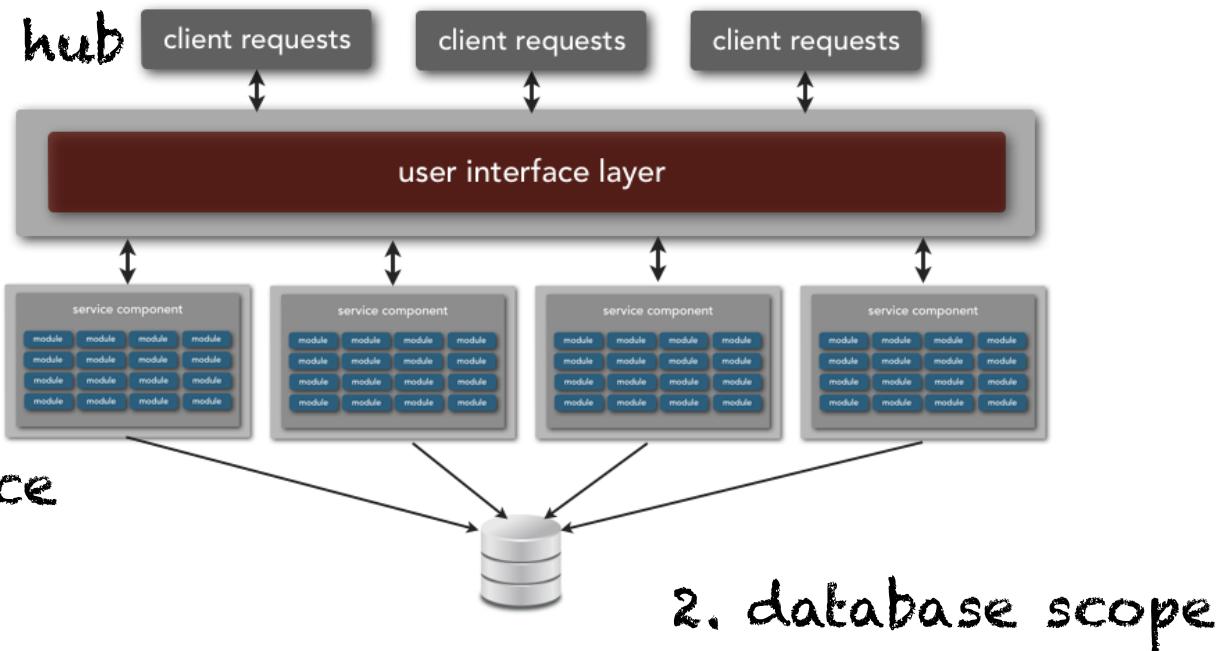
extremely decoupled fine-grained
quanta with well defined boundaries

Service-based Architectures

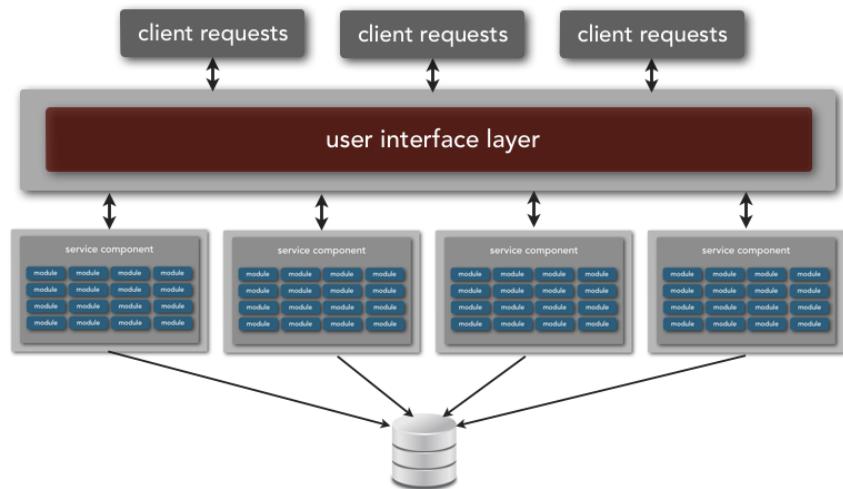


Service-based Architectures

3. use of service bus
as integration hub

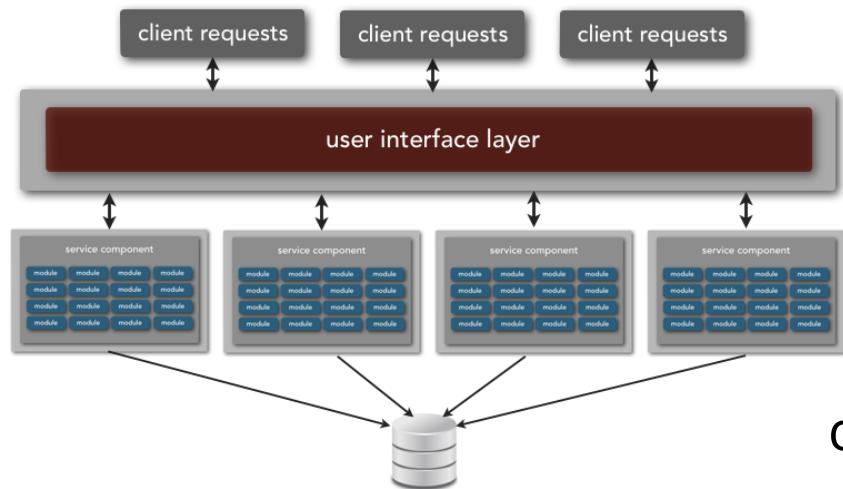


Service-based Architectures



larger than microservices

Service-based Architectures

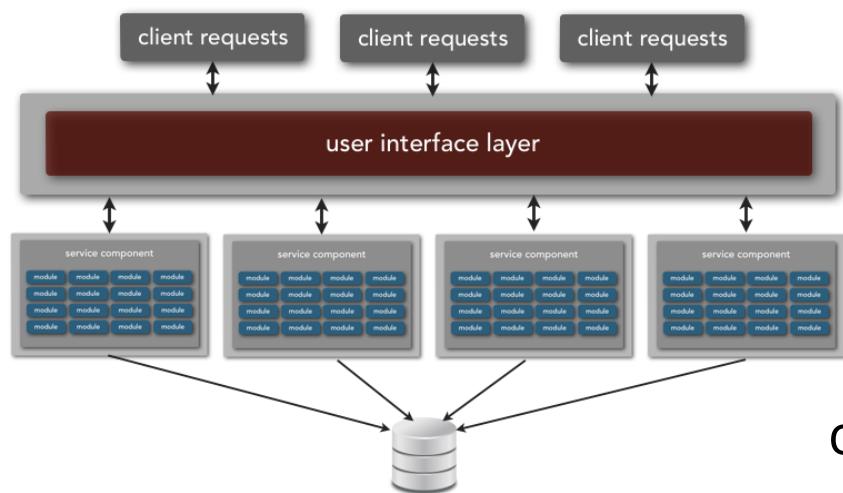


larger than microservices



chunkier granularity harms incremental change

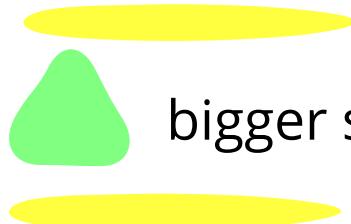
Service-based Architectures



larger than microservices

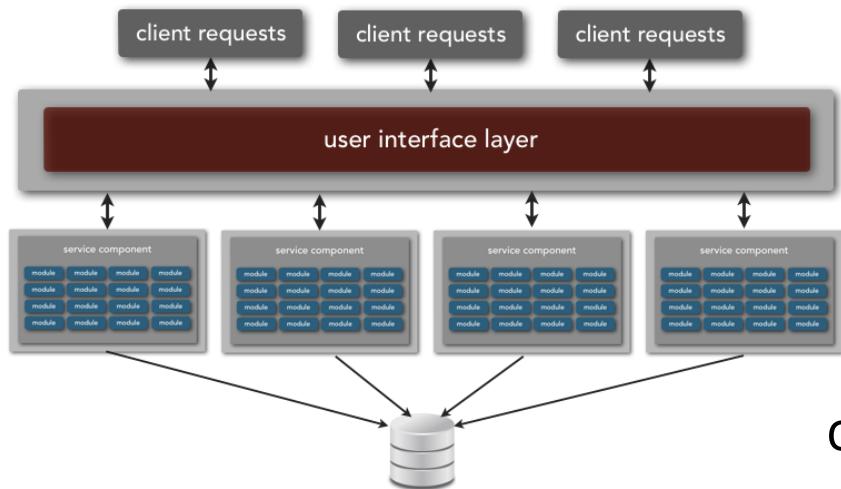


chunkier granularity harms incremental change



bigger scope of tests and other fitness functions

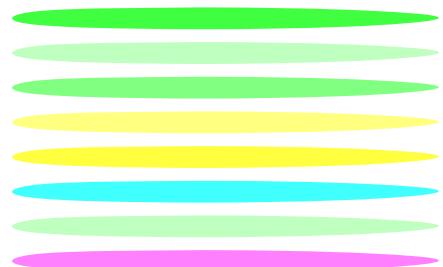
Service-based Architectures



larger than microservices



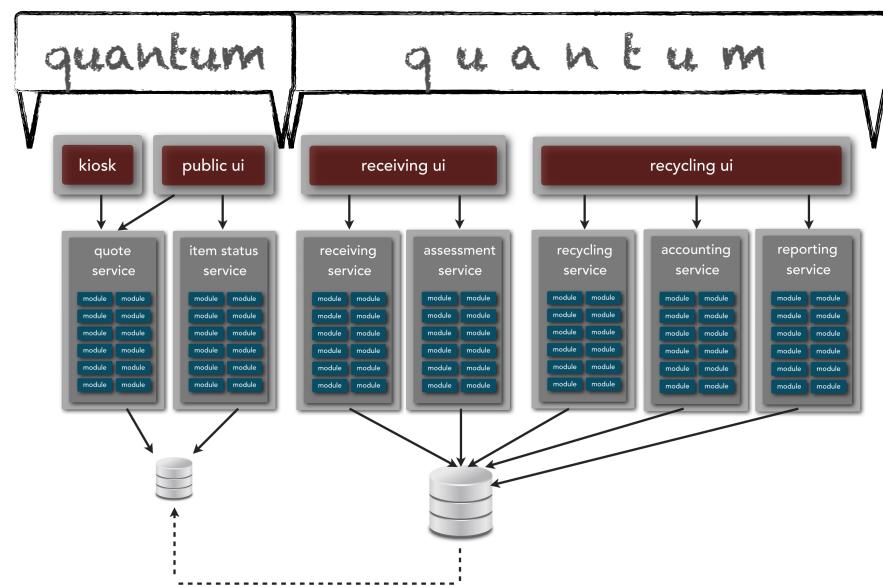
chunkier granularity harms incremental change



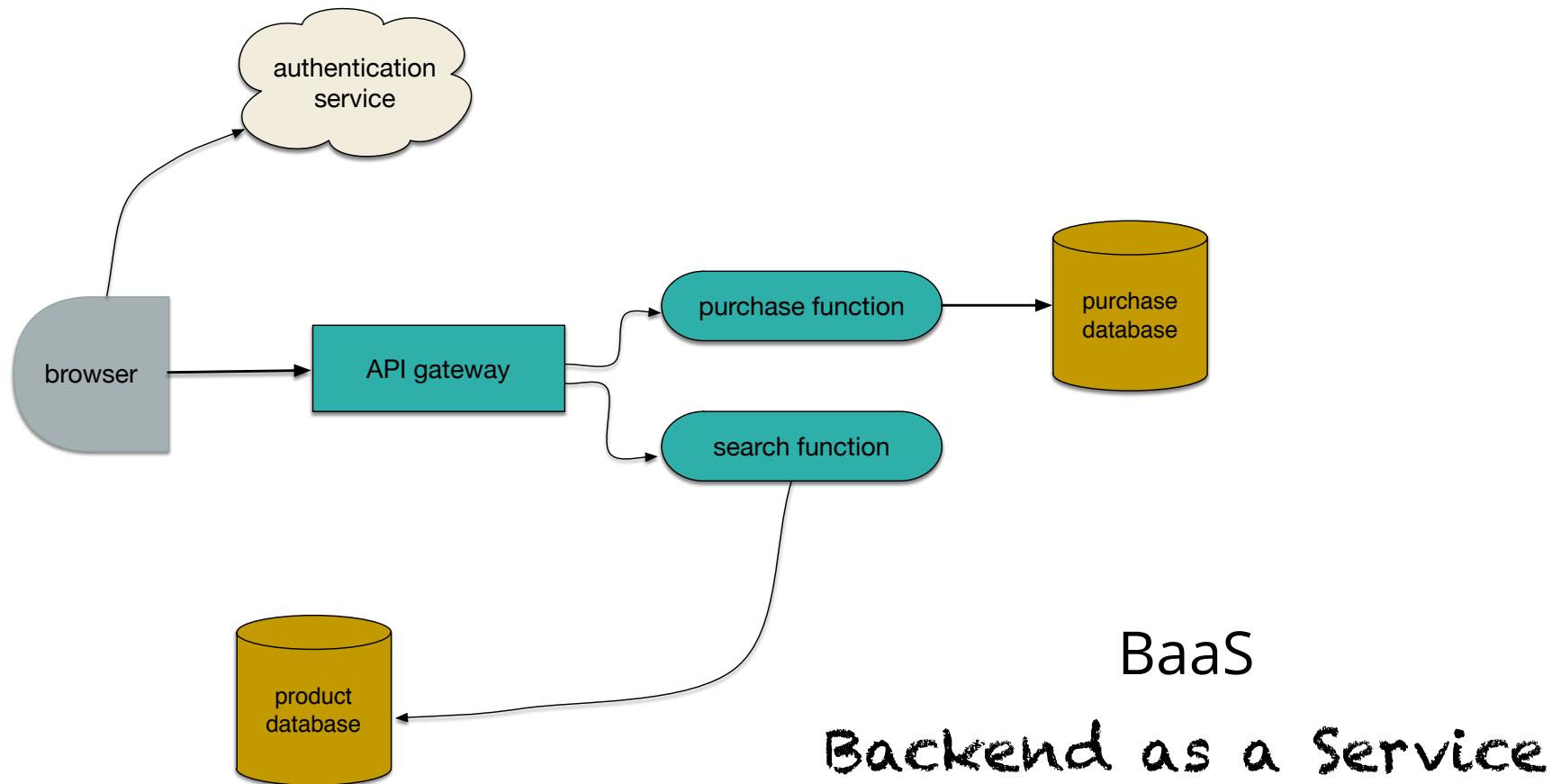
bigger scope of tests and other fitness functions

benefits from domain-centric architecture

Reducing Quanta Size



“Serverless” Architecture



“Serverless” Architecture

FaaS

Function as a Service



API Gateway



Amazon Lambda



DynamoDB

“Serverless” Architecture



Database?

“Serverless” Architecture



Database?



“Serverless” Architecture



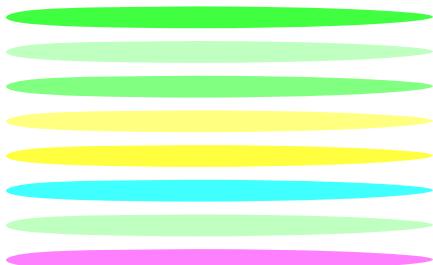
Database?



“Serverless” Architecture



Database?

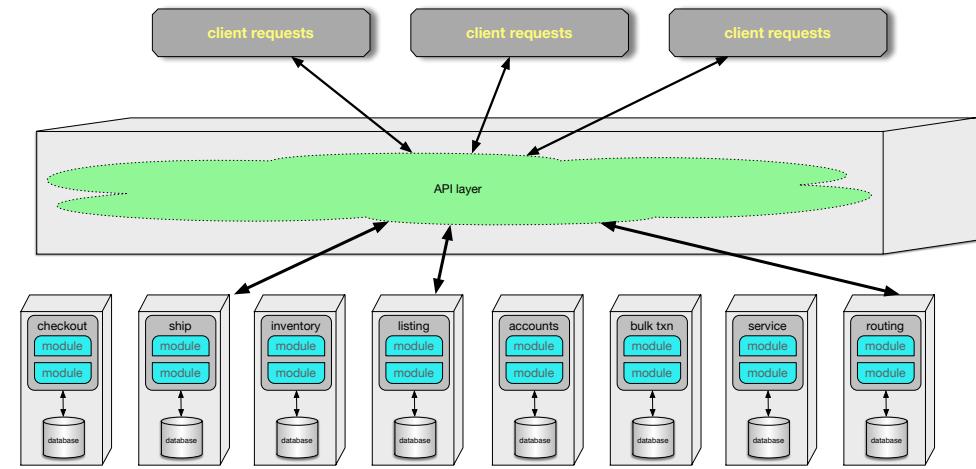
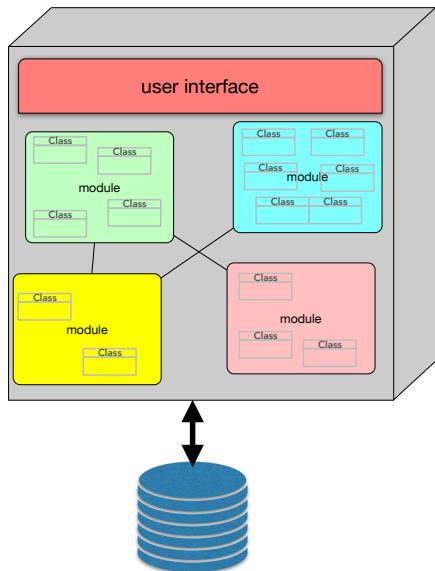


deployment pipeline integration challenging

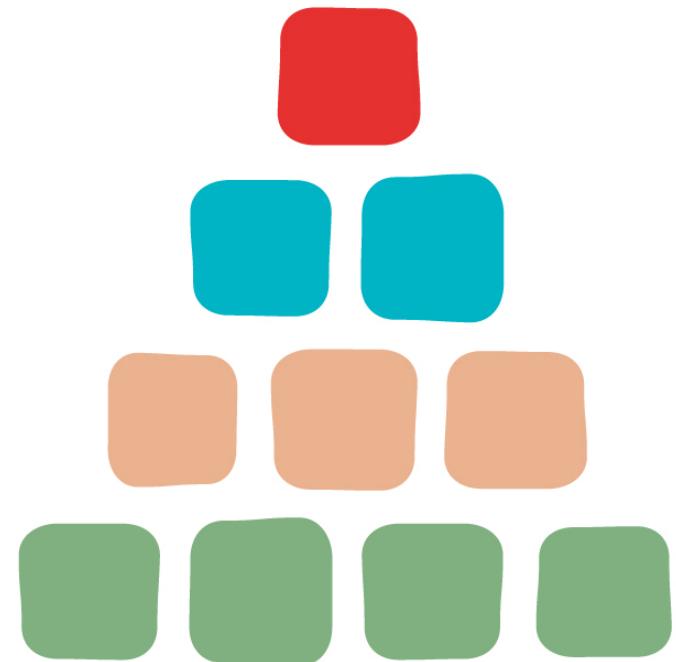
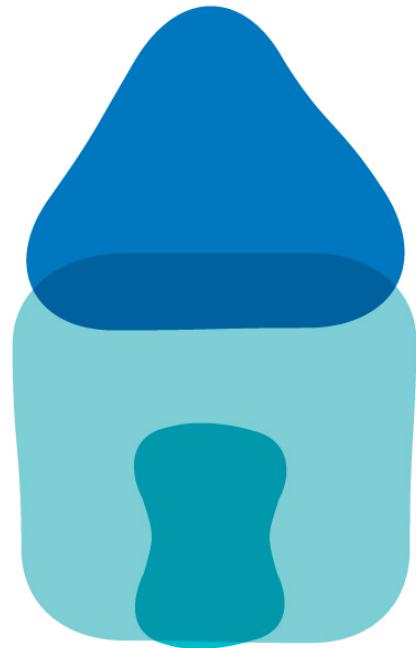
Smaller Quanta Size

△

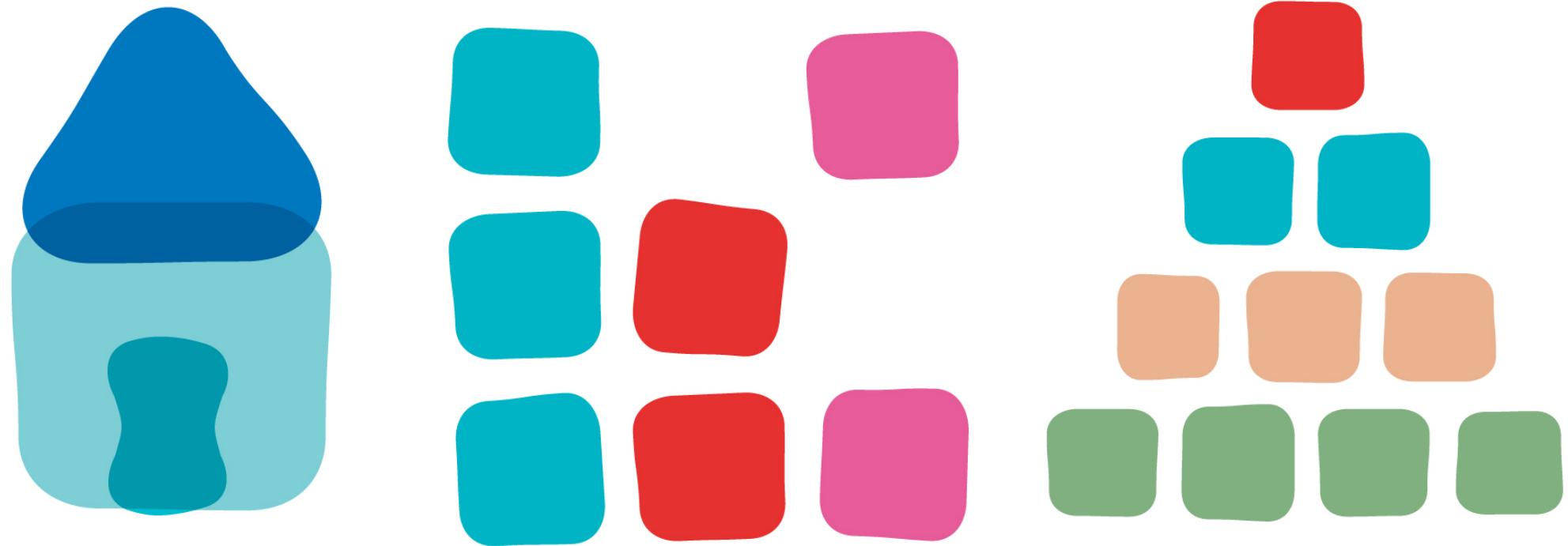
Evolutionary



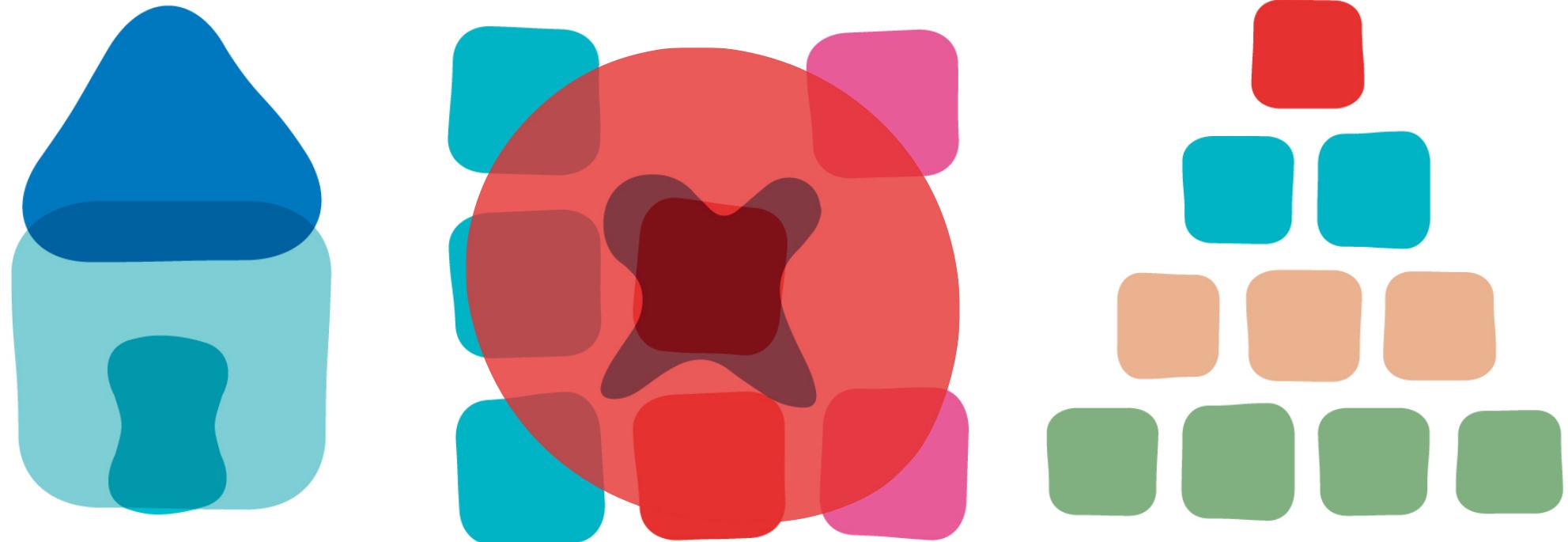
Domain/Architecture Isomorphism



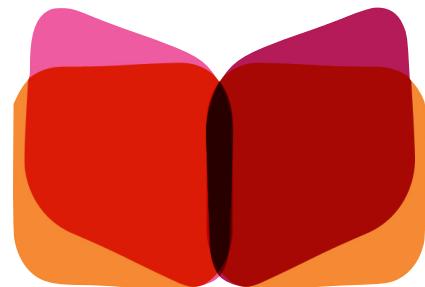
Domain/Architecture Isomorphism



Domain/Architecture Isomorphism



Agenda Part 1



Definition



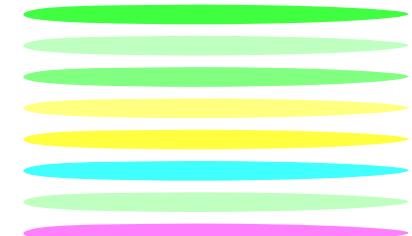
Guided Change via Fitness Functions



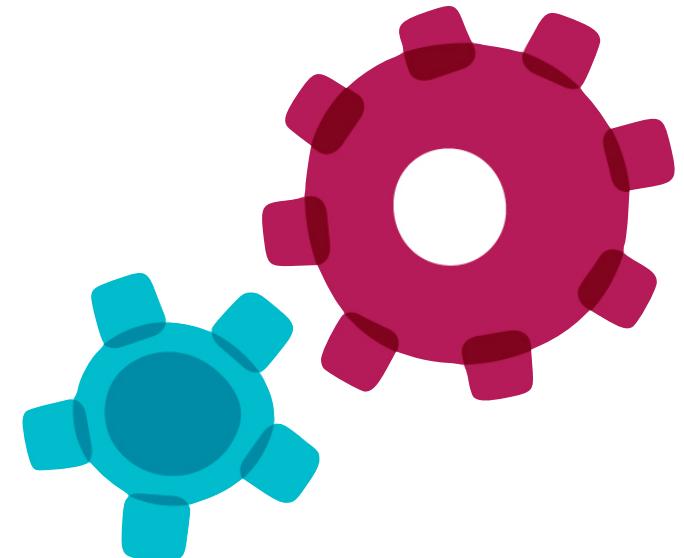
Structuring architecture for evolution

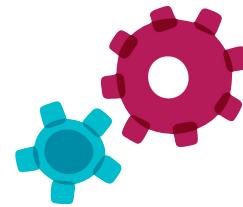


Incremental Change



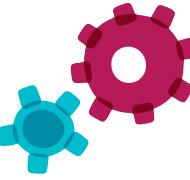
Putting evolutionary architecture into Practice





Mechanics

1. Identify dimensions



1. Identify dimensions

auditability



performance



security



requirements

Time

data

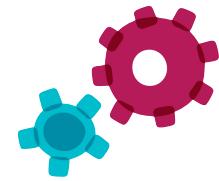


legality



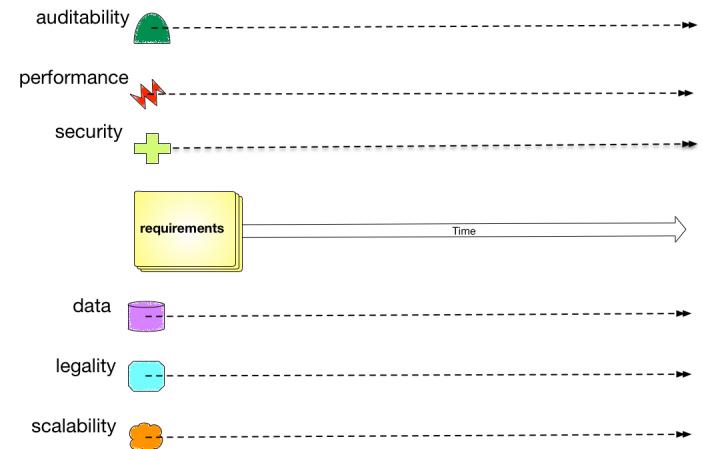
scalability

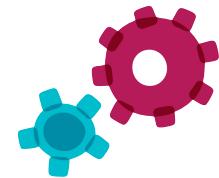




1. Identify dimensions

prioritization

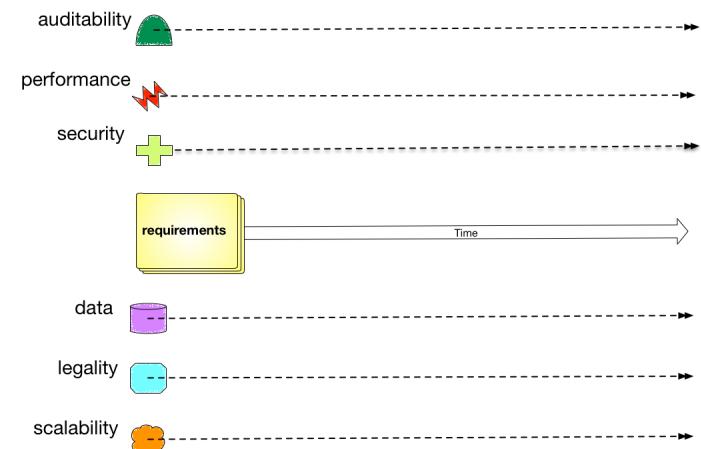


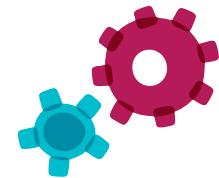


1. Identify dimensions

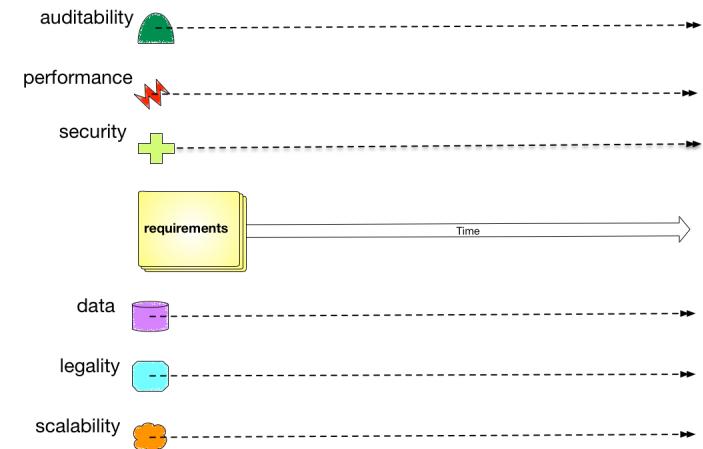
prioritization

cost to implement & maintain





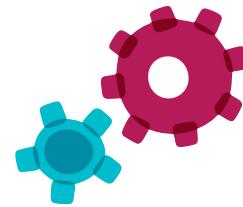
1. Identify dimensions



prioritization

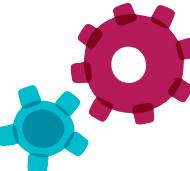
cost to implement & maintain

change variable “fit it” cost to constant
maintenance cost

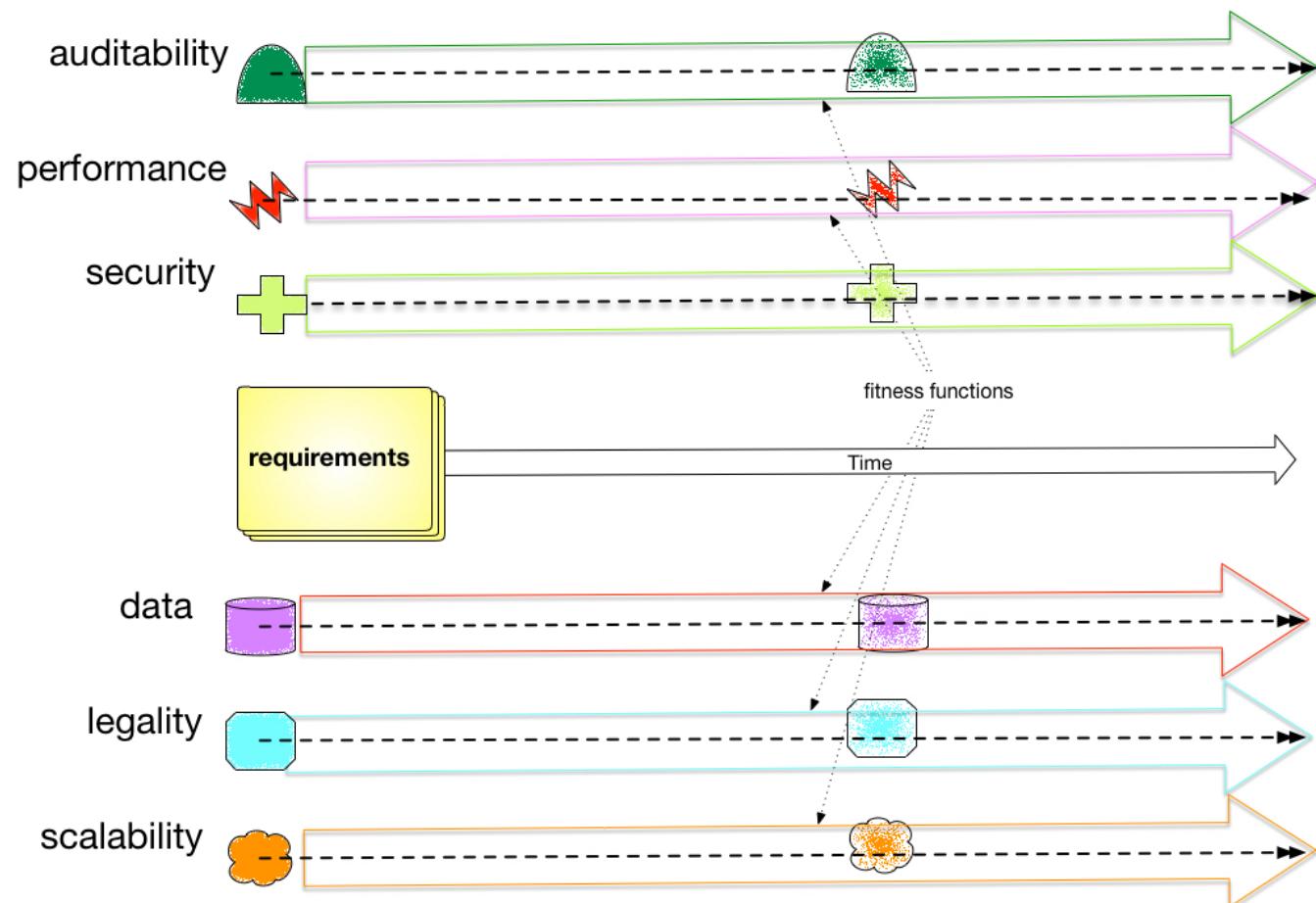


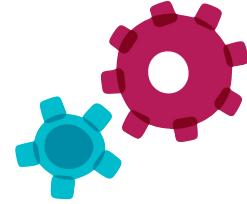
Mechanics

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



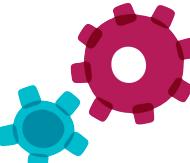
2. Define Fitness Function(s)



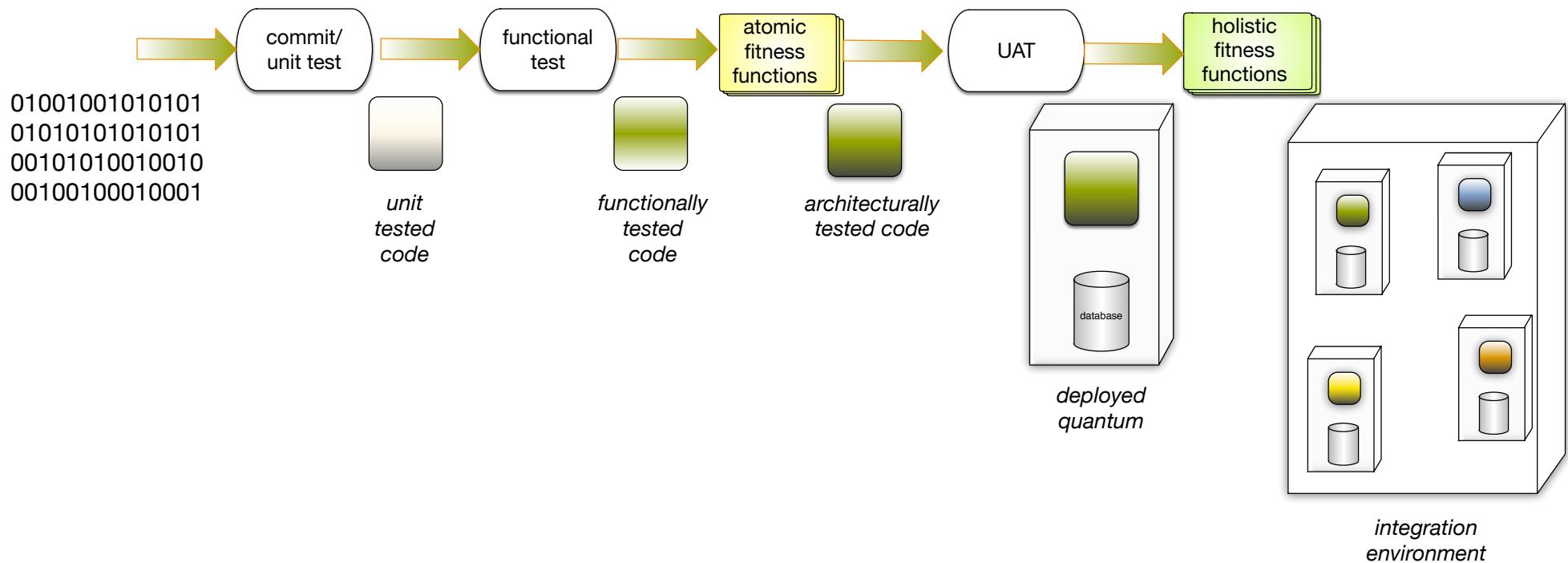


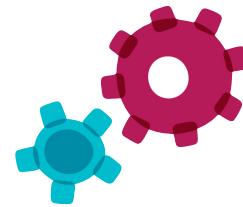
Mechanics

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



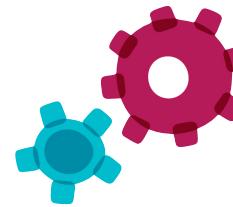
3. Use deployment pipelines and/or continuous fitness functions





Mechanics

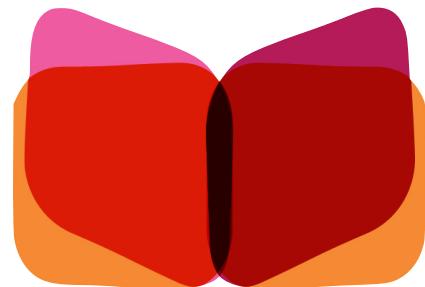
1. Identify dimensions
2. Define fitness function(s)
3. Use deployment pipelines and/or continuous fitness functions



Mechanics

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

Agenda Part 1



Definition



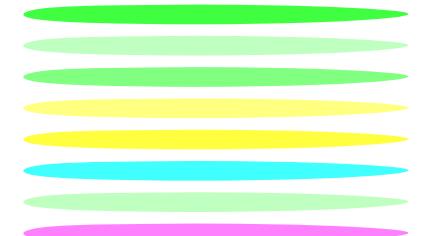
Guided Change via Fitness Functions

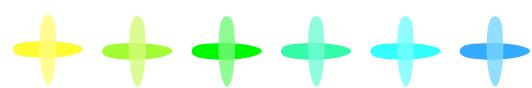


Structuring architecture for evolution

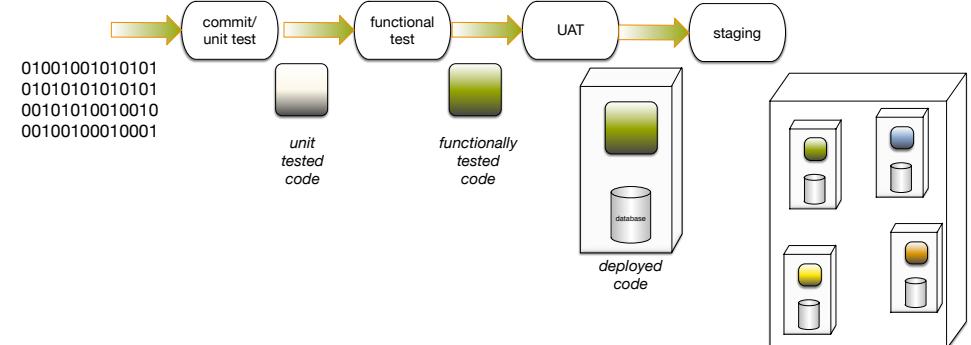
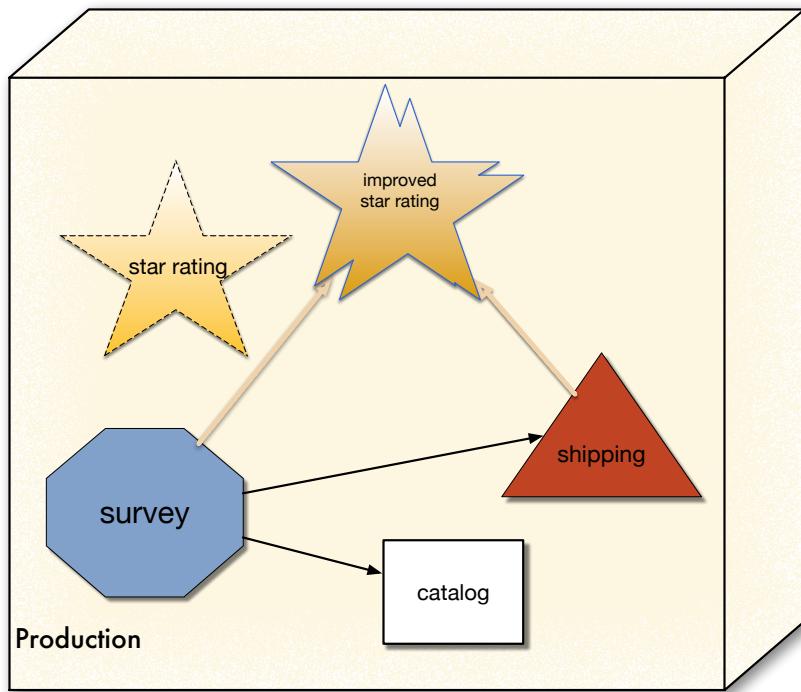


Incremental Change

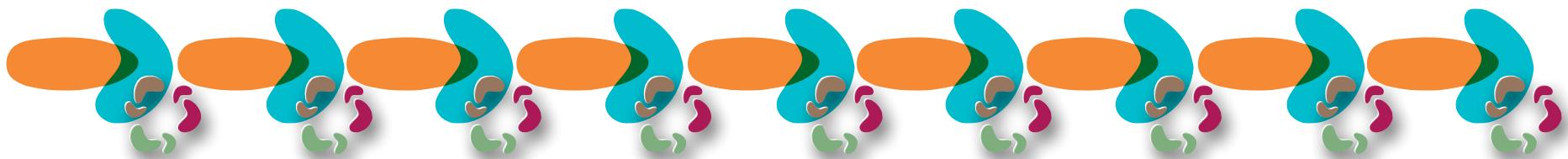




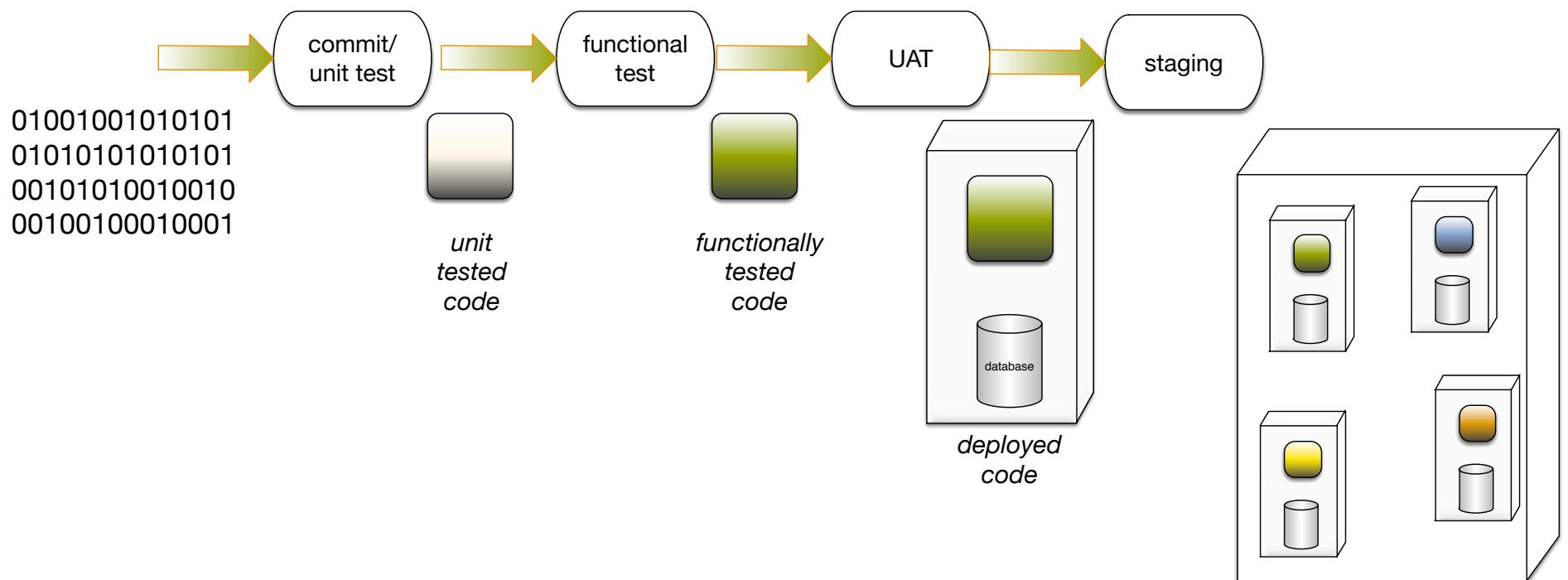
incremental



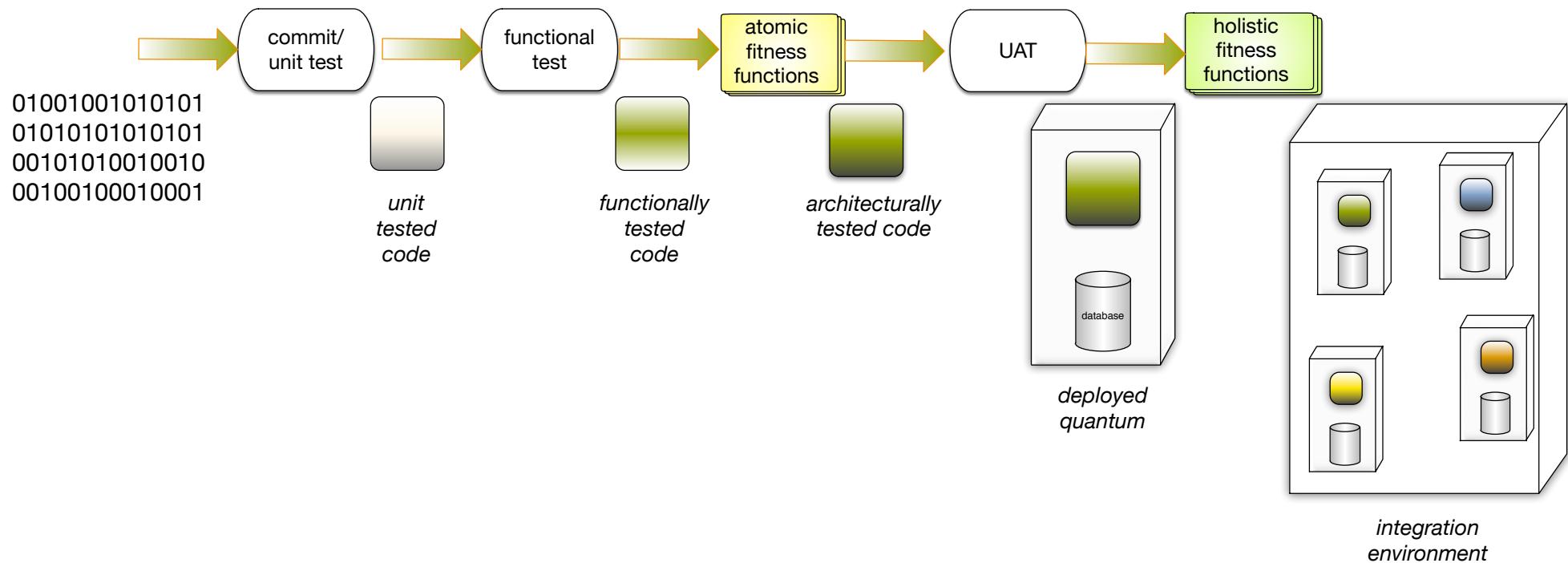
Deployment Pipelines



Deployment Pipeline

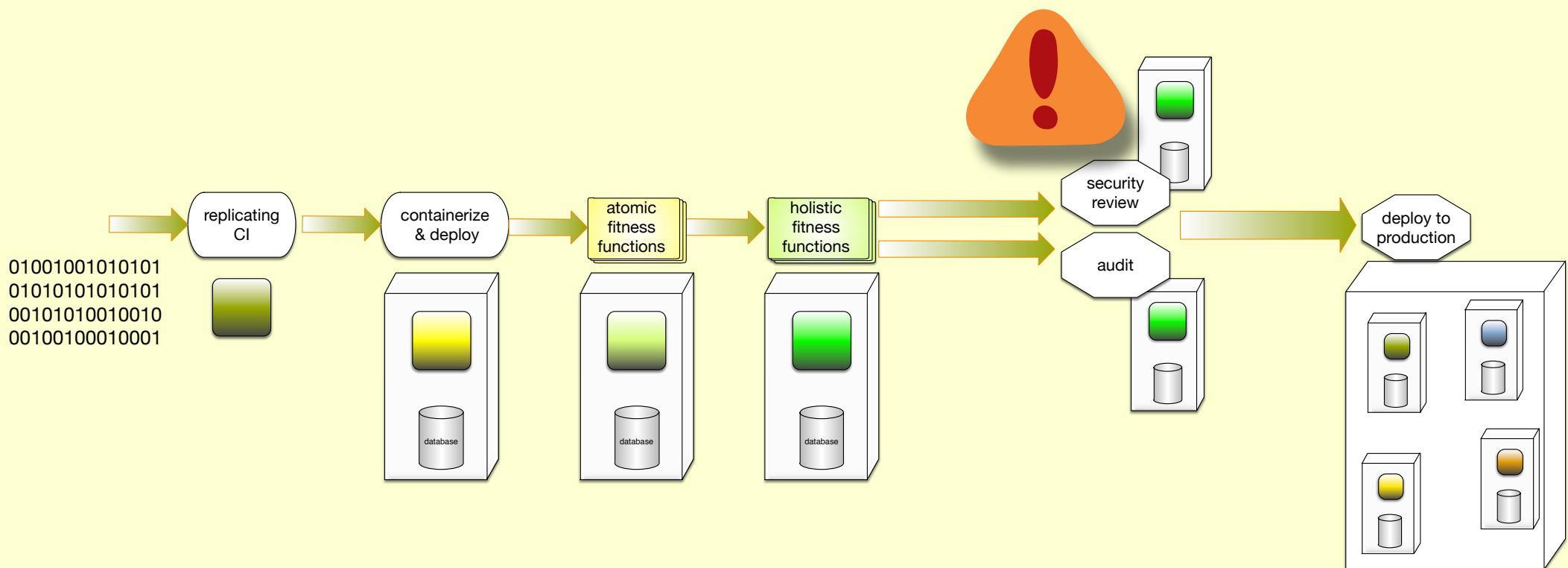


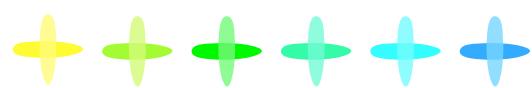
Deployment Pipeline + Fitness Functions



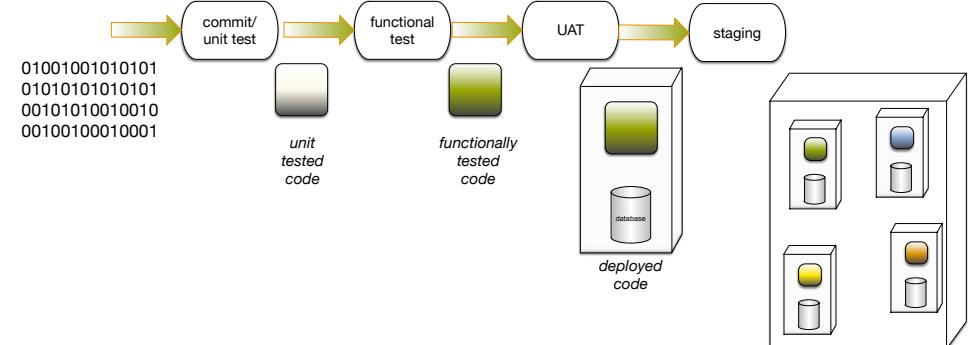
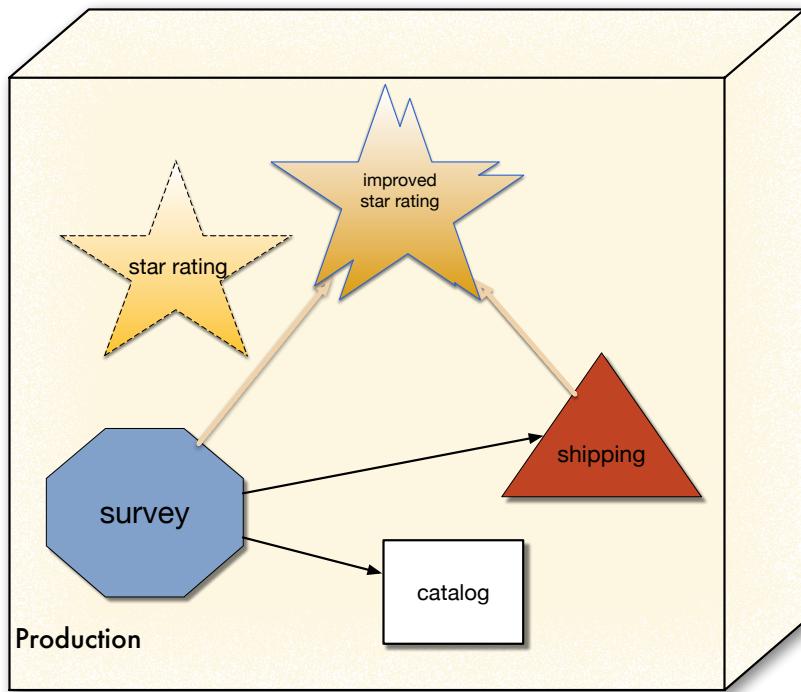


Penultima[↑]e Deployment Pipeline

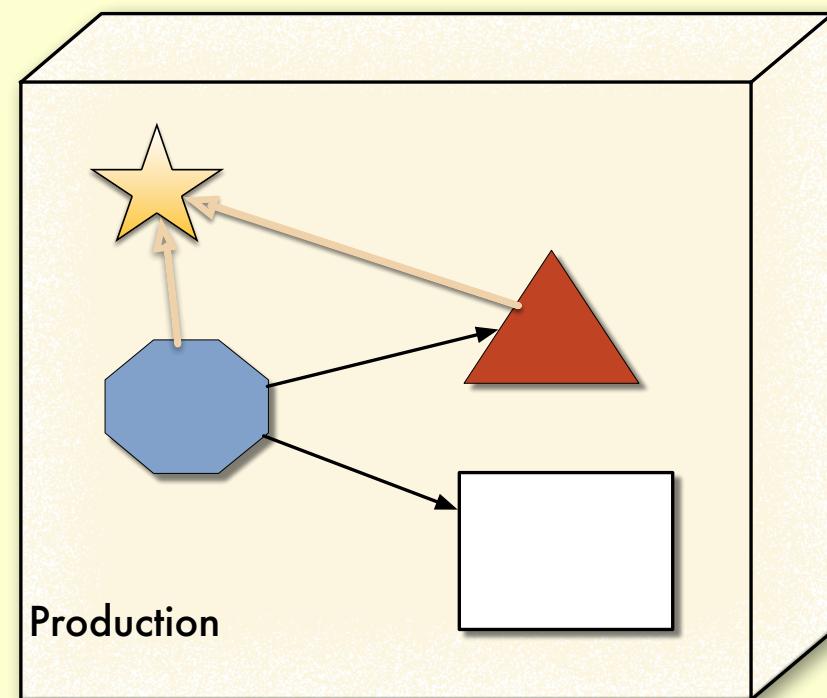
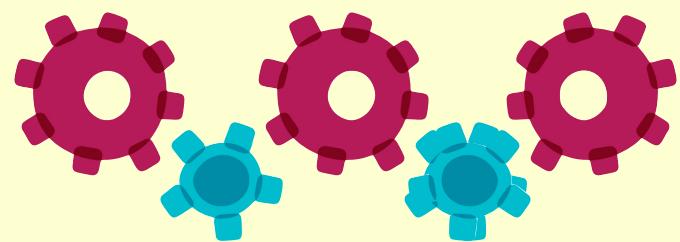




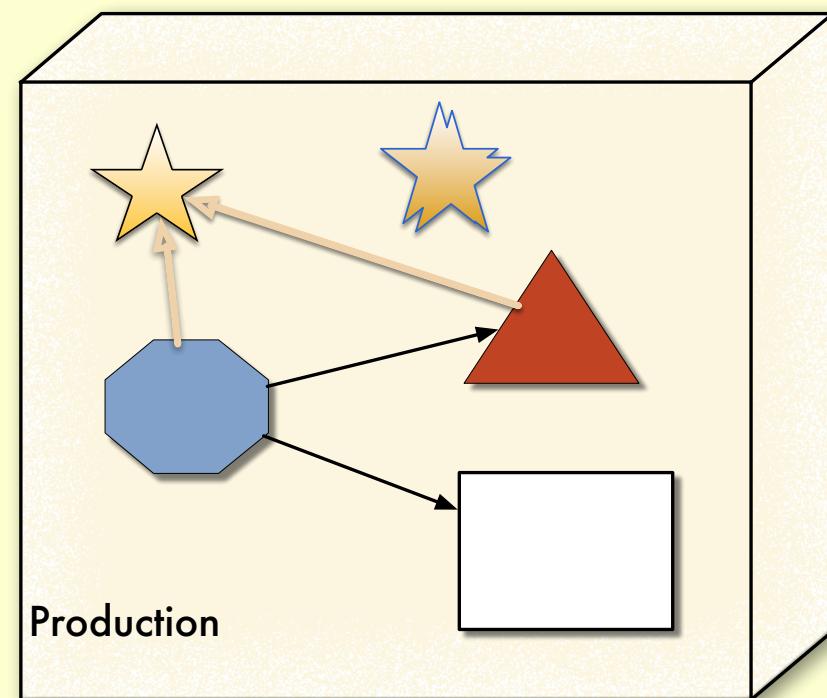
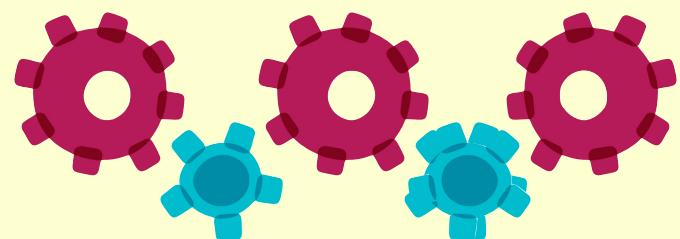
incremental



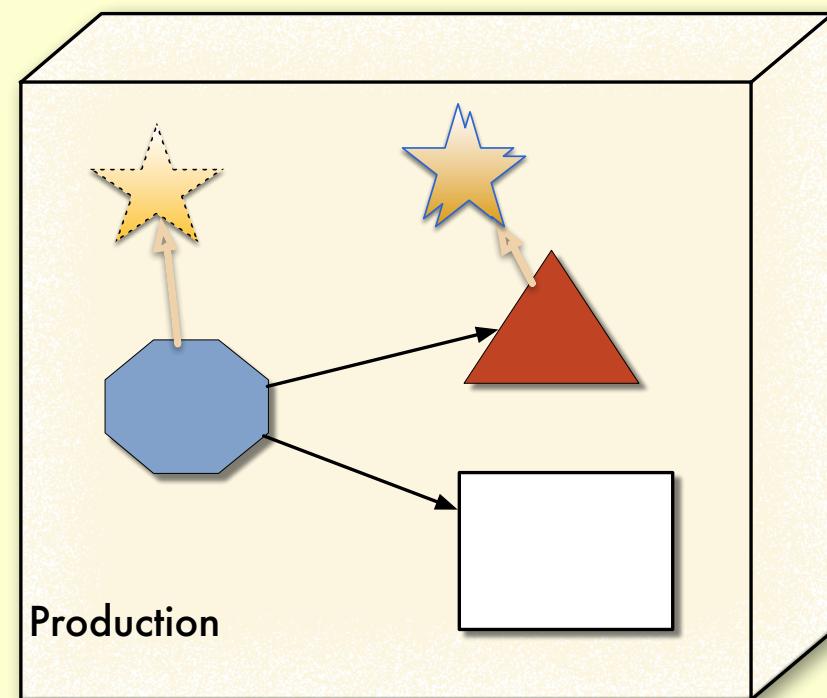
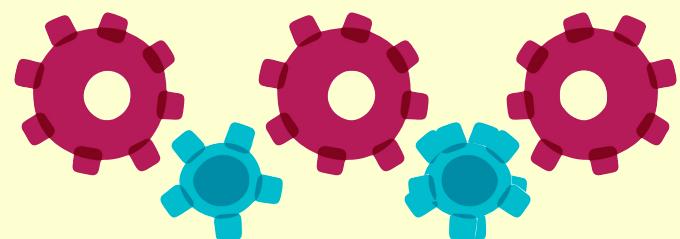
Penultima ↑ e



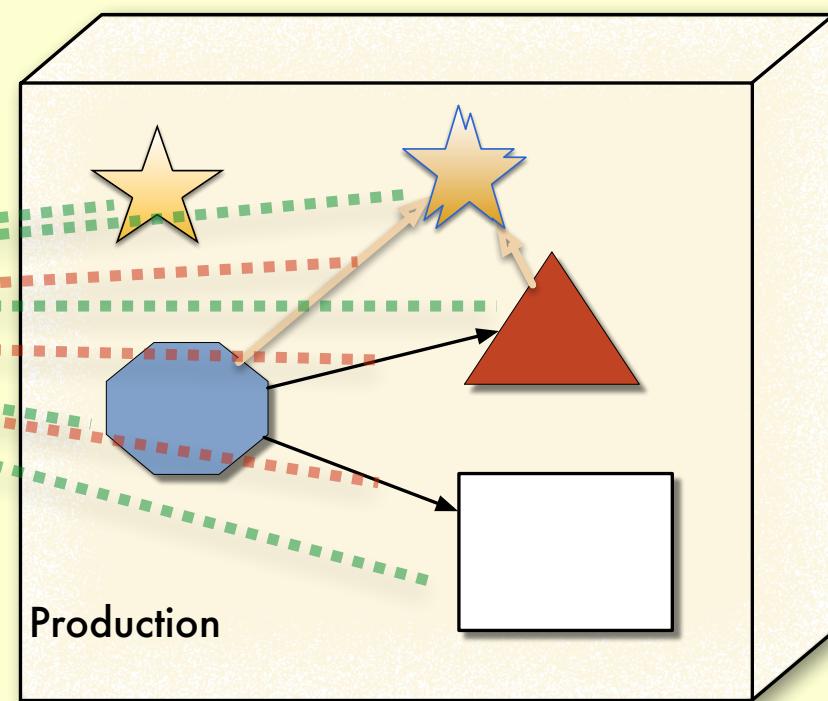
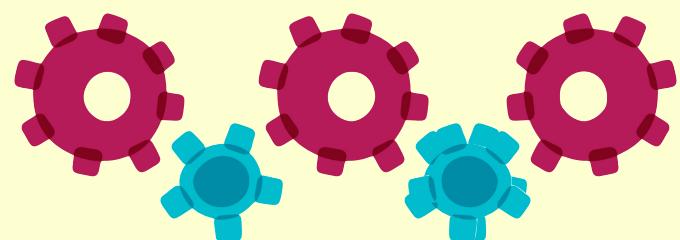
Penultima ↑ e



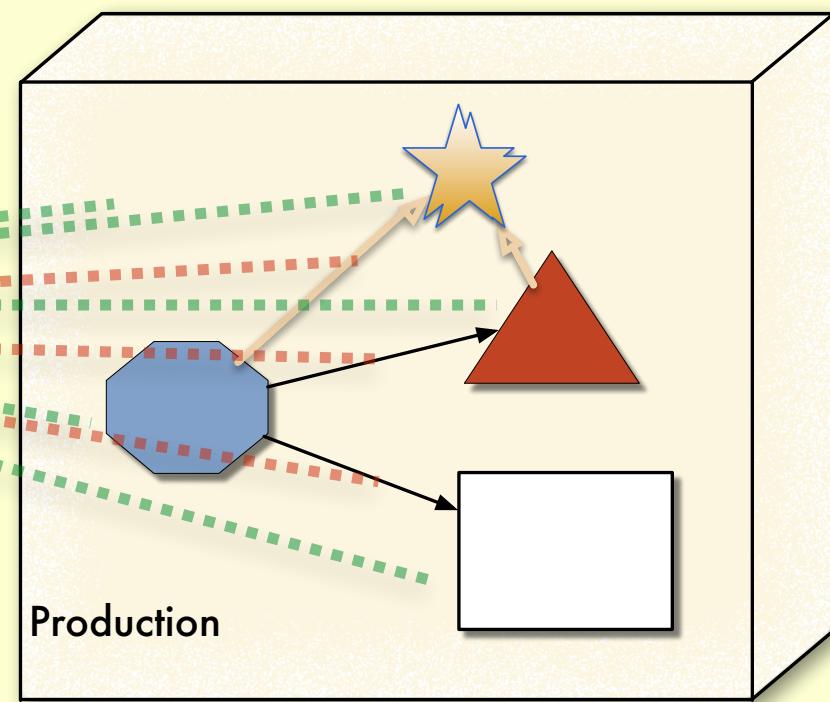
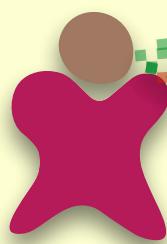
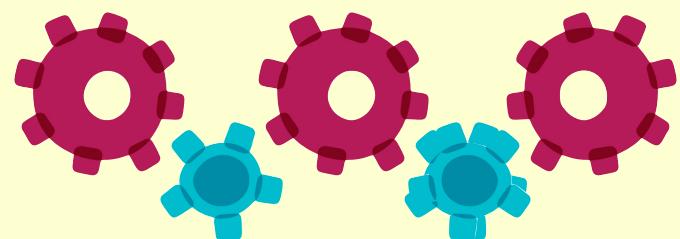
Penultima ↑ e



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". The author is listed as "vmg" and the date is "December 15, 2015". The post content discusses technical debt and how GitHub has approached it over the years. It includes a section titled "Merges in Git" which details the storage model of GitHub repositories and the challenges of merging branches. The post concludes with a note about a shell script used to handle merges.

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.prettyify_message(commit_message)

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

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

A screenshot of the GitHub repository page for "scientist". The repository has 100 stars, 96 forks, and 16 contributors. It contains 71 commits, 1 branch, 6 releases, and is licensed under MIT. The README.md file contains a section titled "Scientist!" with a note about changing permissions and a code snippet for refactoring. The code snippet shows the use of the "use" and "try" blocks to handle permissions.

<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

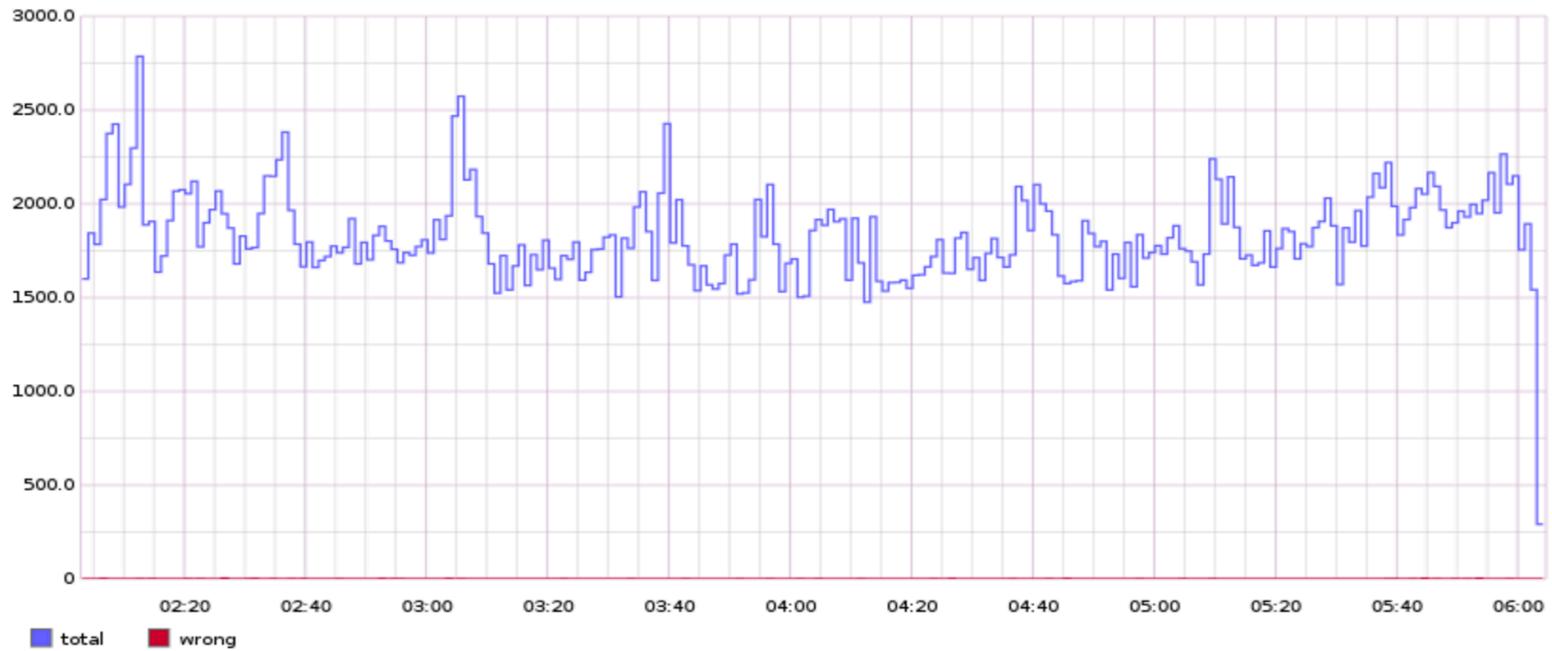
    experiment.run
  end
end
```

- 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.



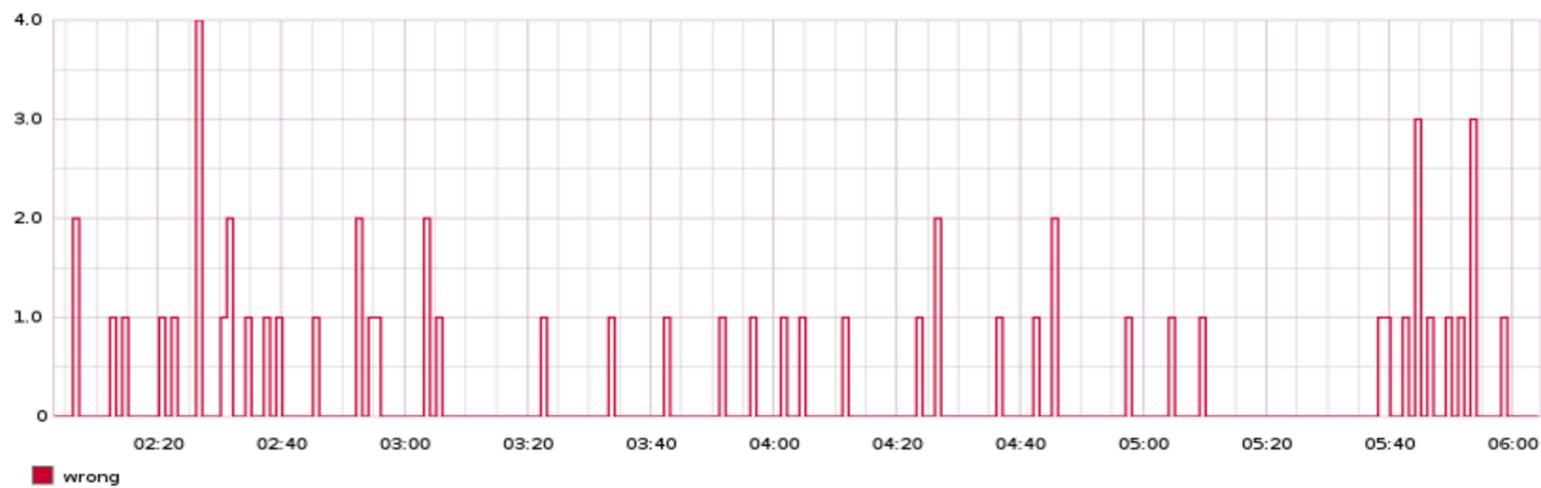
Accuracy

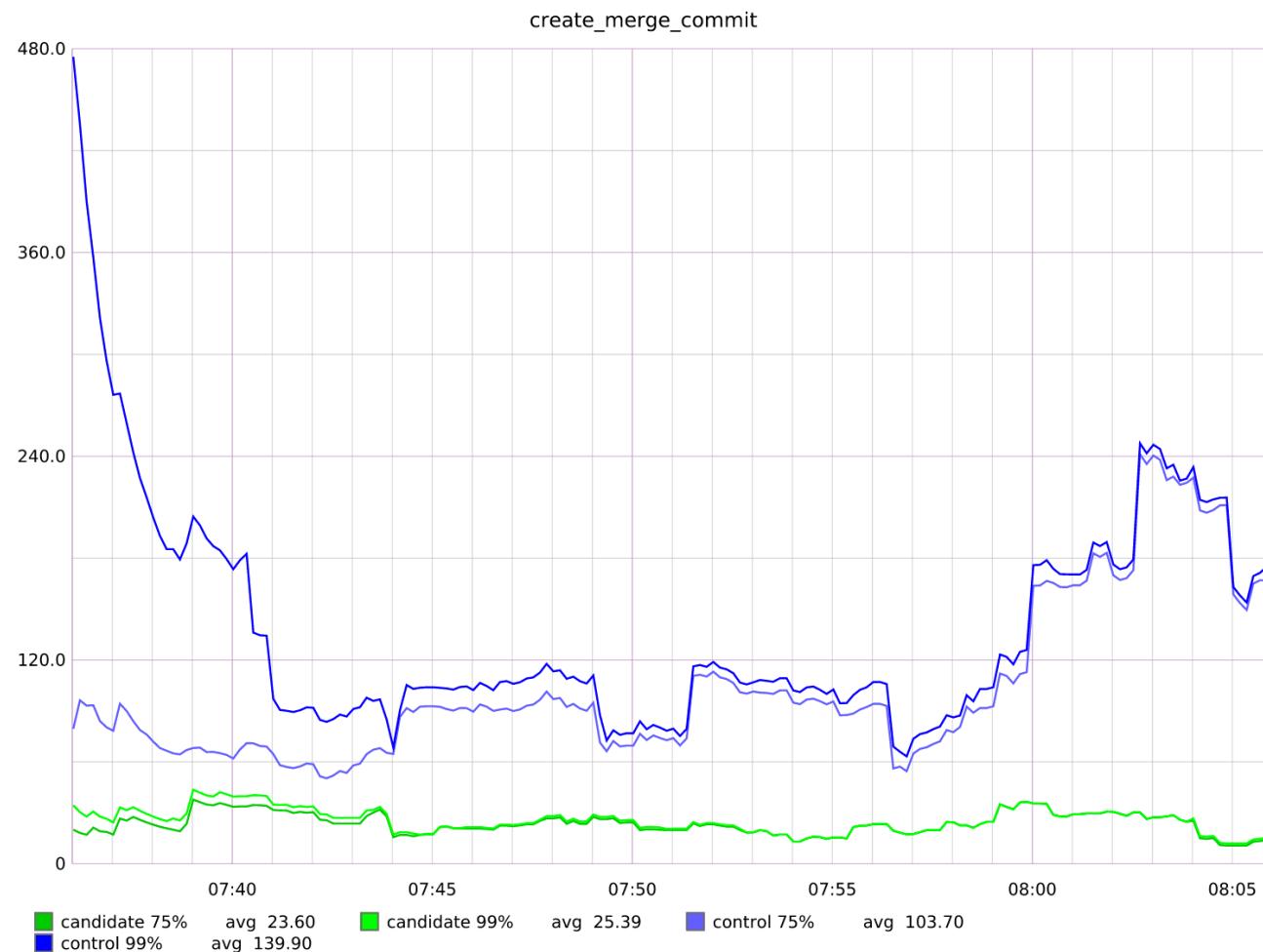
The number of times that the candidate and the control agree or disagree. [View mismatches](#)



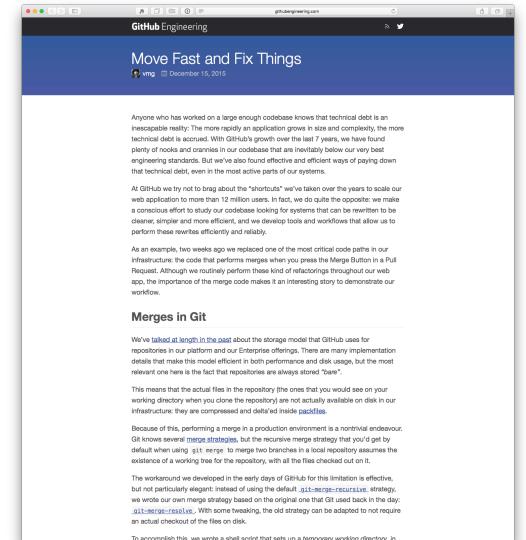


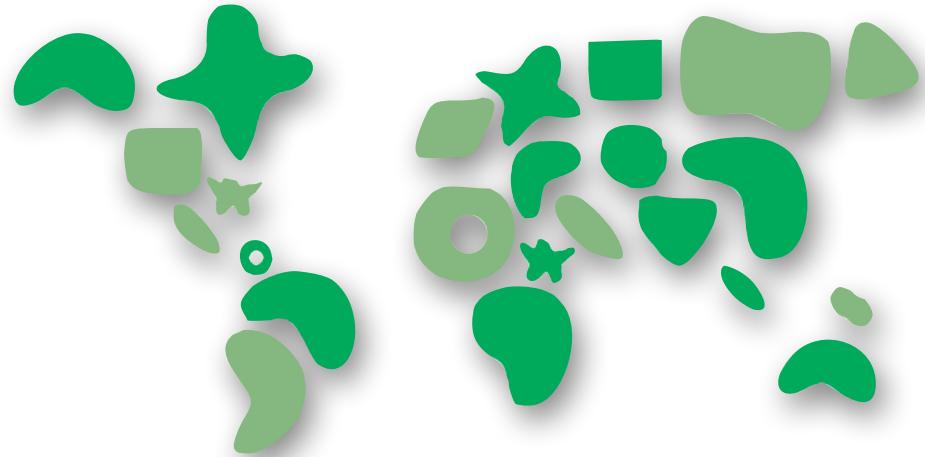
The number of incorrect/ignored only.



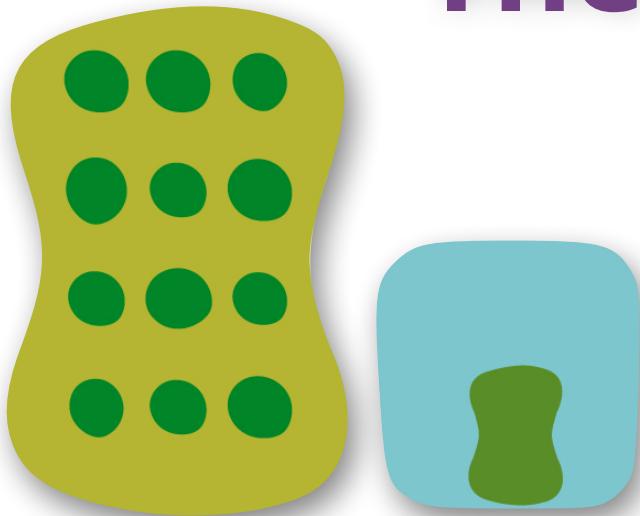


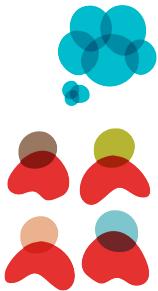
4 days
24 hours/
no mismatches or slow cases
> 10,000,000
comparisons





The Business Case





barryoreilly.com

BARRY O'REILLY

experimental by design, free thinker, curious, i use a lot of post-its

PRESENTATIONS LEAN ENTERPRISE CONSULTING ABOUT HOME

OCT 21 2013
10 COMMENTS

AGILE, CONTINUOUS DELIVERY, INNOVATION, LEAN, LEAN ENTERPRISE

[LINK](#)

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 to explain what we had observed. 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 constructing more sophisticated experiments, and tuning, evolving 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 Development¹) is thinking about the development of new ideas, products and services – even organizational change – as a series of experiments to determine whether an expected outcome will be achieved. The process is iterated upon until a desirable outcome is obtained or the idea is determined to be not viable.

We 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 code will execute and even how the customer will use it.

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 succeed if the code passes the test. Ultimately, product or service development is a process to test a hypothesis about system behaviour 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 expect 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 improved 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, over simply becoming a feature factory by 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

The screenshot shows a Medium article page. At the top, there's a navigation bar with icons for back, forward, search, and a user profile. The title 'Hypothesis Driven UX' is displayed prominently. Below the title, the author's name 'Maximilian Wambach' and the publication date 'Jan 11, 2015 - 5 min read' are shown. The main content features a large image of a wine bottle lying on its side, surrounded by many wine corks. The wine bottle has a label that includes the word 'HYPOTHESIS'. The text on the page reads:

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

Netzbetreiber 14:13

74.004 Treffer Speichern

Opel Astra 2.0 Turbo En... >
15.000 €
79618 Rheinfelden
08/2008, 49.500 km
147 kW (200 PS), Benzin
Kraftstoffverbr. (komb.)* 9,5 l/100km
CO₂-Emissionen (komb.)* 228 g/km

Peugeot 308 CC 155 TH... >
15.000 €
45473 Mülheim
10/2010, 28.100 km
115 kW (156 PS), Benzin

Renault Kangoo 1.6 16V... >
15.000 €
Unfallfrei
97877 Wertheim
11/2012, 1.100 km
78 kW (106 PS), Benzin
Kraftstoffverbr. (komb.)* 7,7 l/100km
CO₂-Emissionen (komb.)* 180 g/km

Volkswagen Golf 1.4 TS

Suchen Parkplatz Meine Suchen Meine Inserate Info

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

o2-de 11:11

6.703 Treffer

Top Audi A6 Avant 2.0 TDle 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 TDle 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 € 132 kW (176 PS)
EZ 09/2006 120 km

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

More Listings

o2-de 11:11

6.703 Treffer

Top Audi A6 Avant 2.0 TDle 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..
599 €
66679 Losheim am See
Suchen **Parkplatz** **Meine Suchen** **Meine Inserat** **Info**

Better Structure

o2-de 11:11

6.703 Treffer

Top Audi A6 Avant 2.0 TDle 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, schöne Lederausstattung, ..
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..
599 €
66679 Losheim am See
Suchen **Parkplatz** **Meine Suchen** **Meine Inserat** **Info**

Better Prioritization

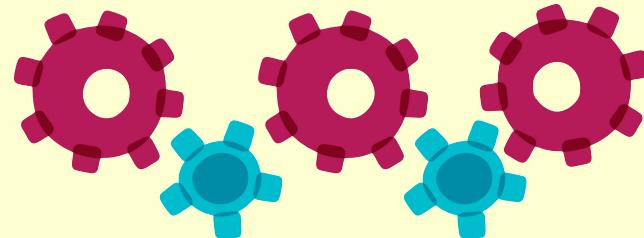
Evolutionary Architecture

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

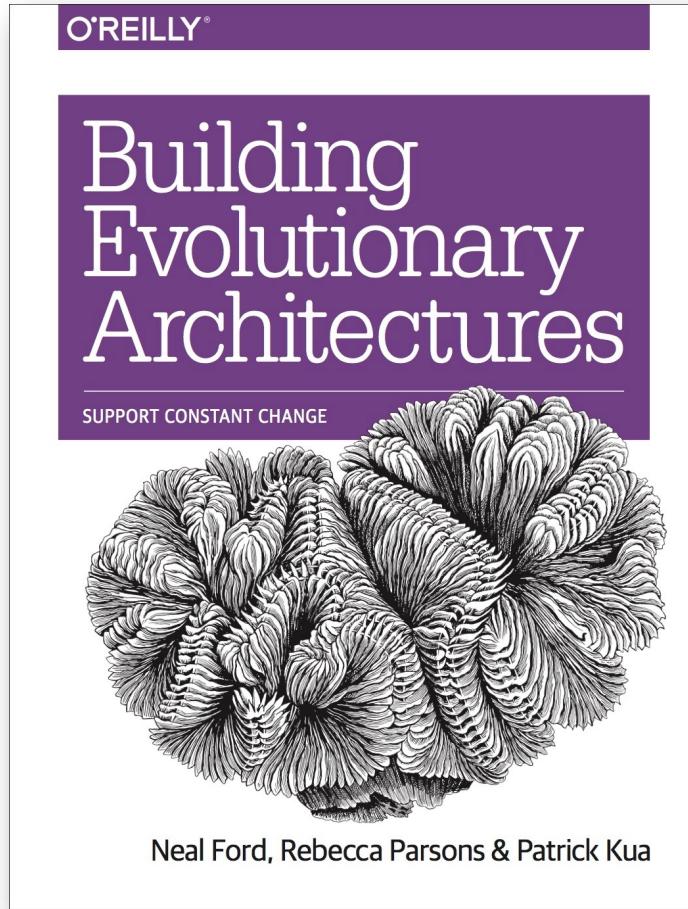


Agenda Part 2

The Evolution of
Penultima↑e



bUiLdiNG eVoLuTiONaRy



ARcHiEcTuREs

For more information:

A screenshot of a website for 'Building Evolutionary Architectures'. The header features the book's title and authors. Below the header is a large, blurred photograph of green grass. A navigation bar with links to Home, Blog, Talks, and The Book is visible. Three circular icons represent 'Incremental', 'Fitness Functions', and 'Multiple Dimensions'. Each icon has a brief explanatory text below it. At the bottom right of the page is a 'Learn More' button.

Building Evolutionary Architectures by Neal Ford, Rebecca Parsons and Patrick Kua

Home Blog Talks The Book

Building Evolutionary Architectures: Support Constant Change [Learn More](#)

Incremental
Evolutionary architectures are built one part at a time, with many different increments. Speed to the next increment is key.

Fitness Functions
Every system at different points of their life need to optimize to be "fit" for its environment. Evolutionary architectures make it explicit what "fit" means with as much automation as possible.

Multiple Dimensions
Evolutionary Architectures must support both "technical" and "domain" changes

<http://evolutionaryarchitecture.com>