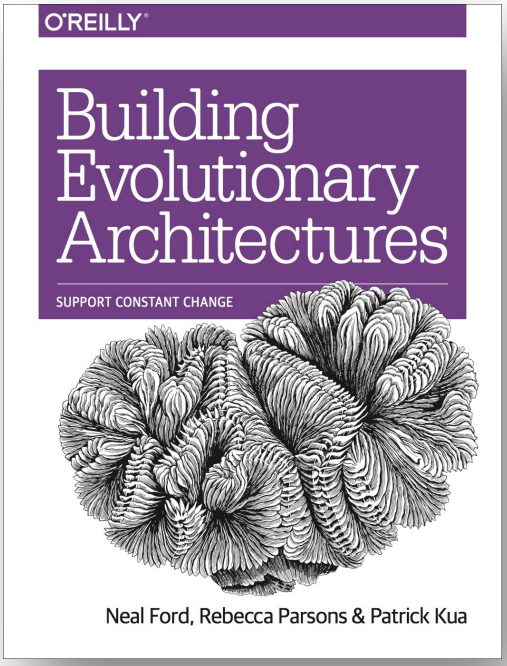


http://evolutionaryarchitecture.com

Building Evolutionary Architectures Workshop

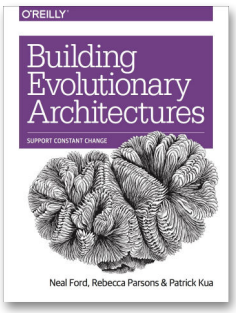
Neal Ford

@neal4d

nealford.com

bUiLdiNG eVoLuTiONaRy ARcHitECtuREs wOrkSHoP

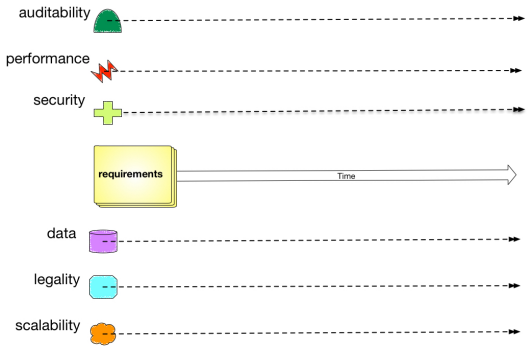
1



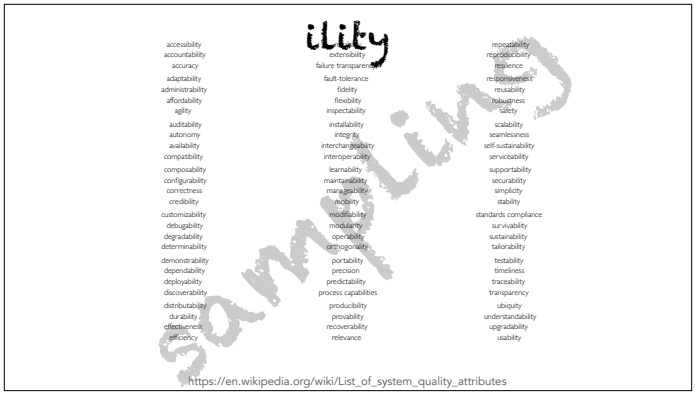
@neal4d
nealford.com
@rebeccaparsons
@patkua

2

3

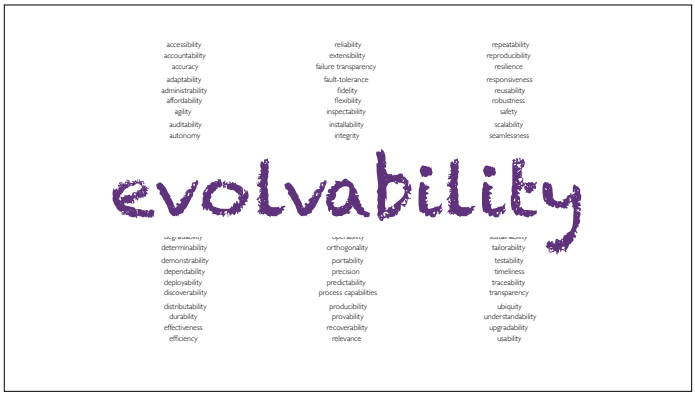


4



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

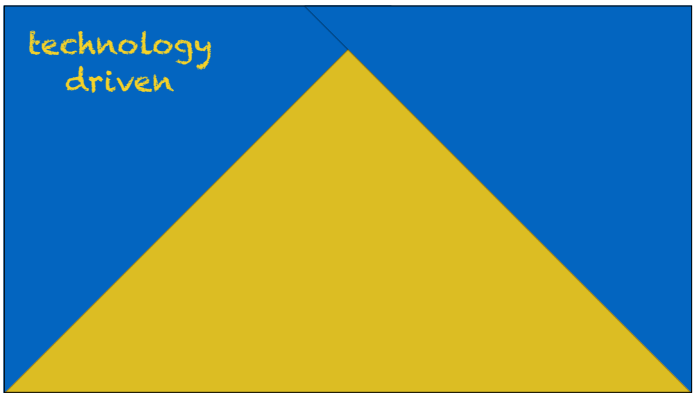
5



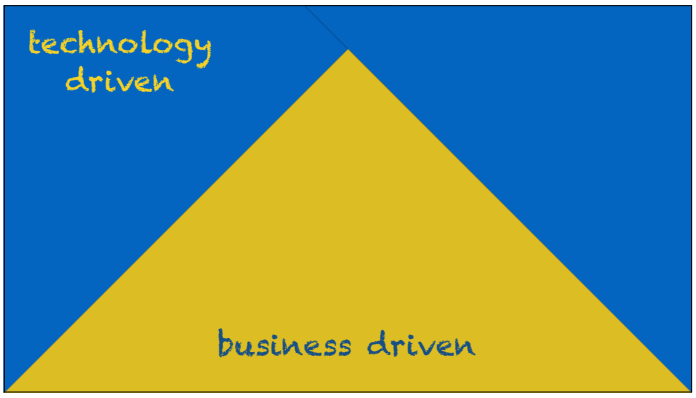
6

cHAnGe

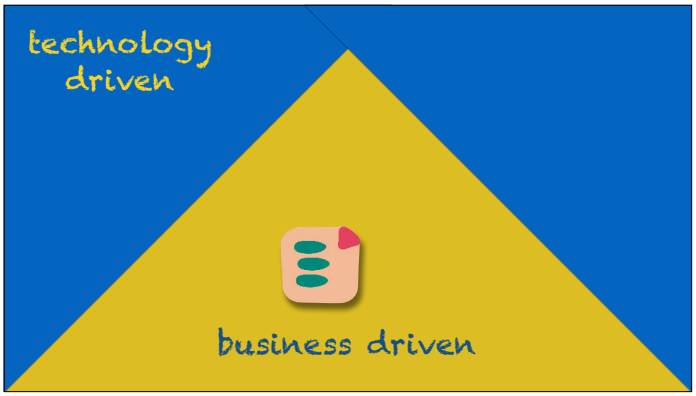
7



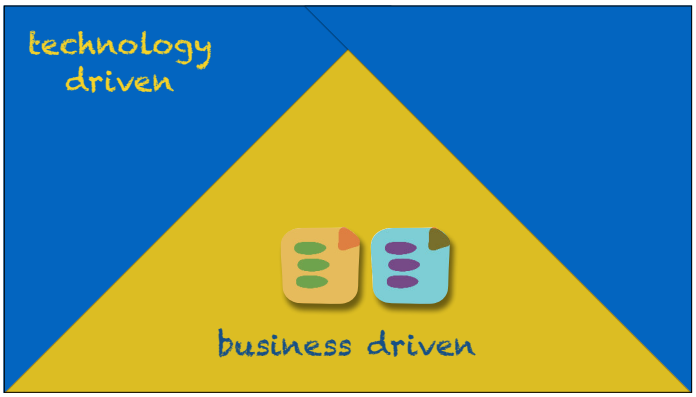
8



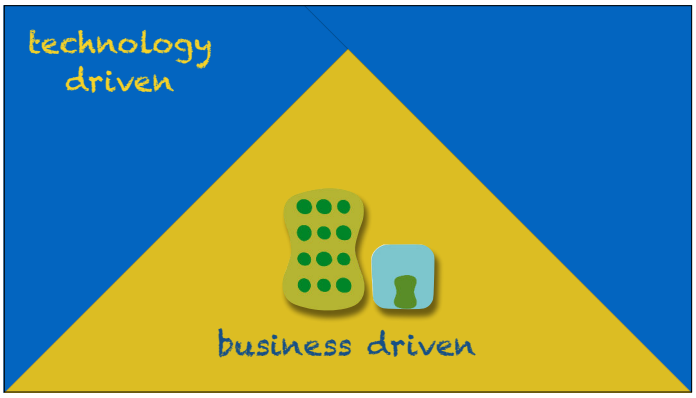
9


technology
driven
business driven

10


technology
driven
business driven

11


technology
driven
business driven

12

technology driven

business driven

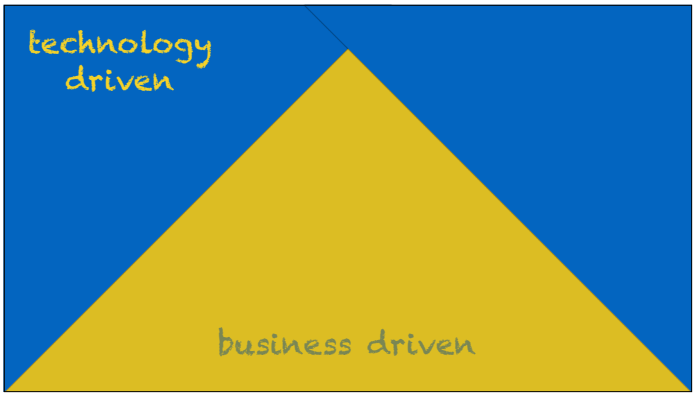
13

14

technology driven

business driven

15



16

everything chAnGEs
all the time!

17

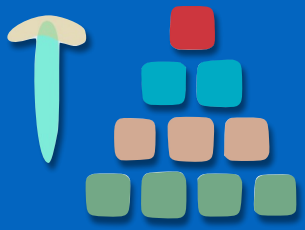
dYNaMic eqUiLiBRiuM

18

dYNaMic eqUiLiBRiuM

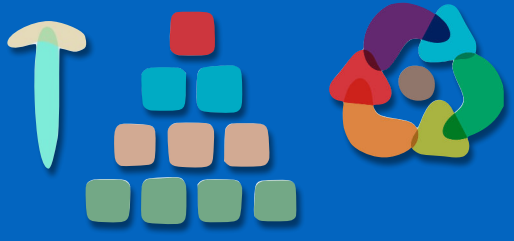
19

dYNaMic eqUiLiBRiuM

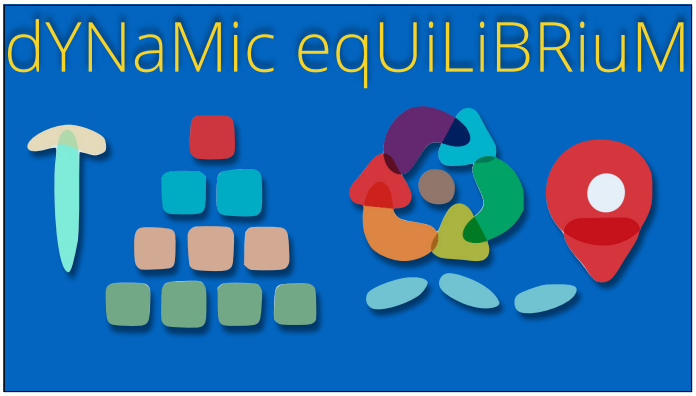


20

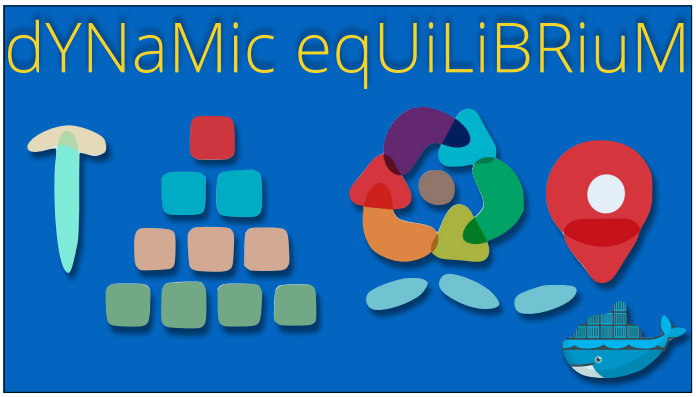
dYNaMic eqUiLiBRiuM



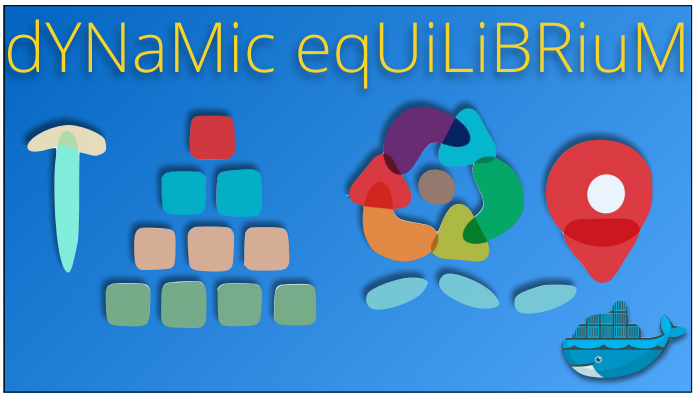
21


dYNaMic eqUiLiBRiuM

22


dYNaMic eqUiLiBRiuM

23

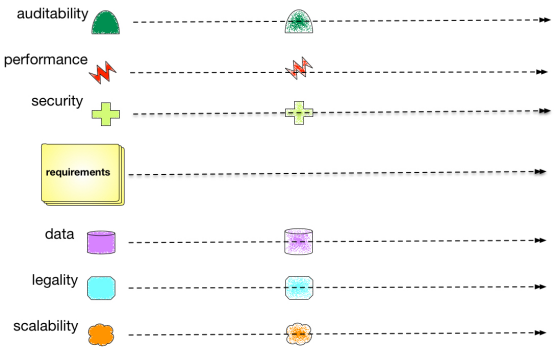

dYNaMic eqUiLiBRiuM

24

25

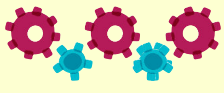
How is long term planning possible when things constantly change in unEXpECtEd **ways?**

26

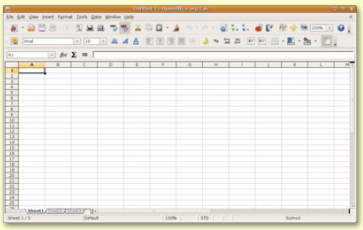


27

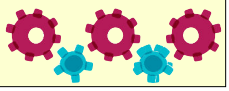
Penultima⬆e



EA Spreadsheet

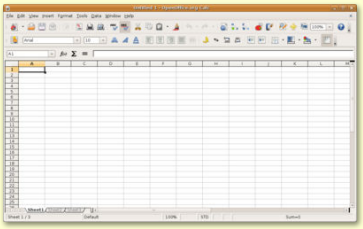


Penultima⬆e



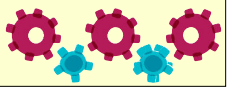
28

EA Spreadsheet



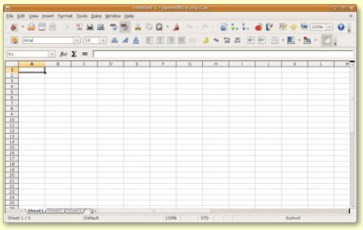
✓ definition

Penultima⬆e



29

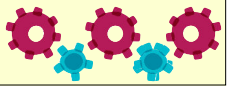
EA Spreadsheet



✓ definition

! verification

Penultima⬆e



30

31

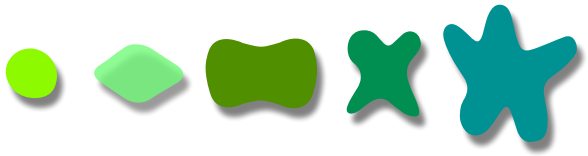
Once I've built an architecture, how can I prevent it from gradually dEgRAdiNg **over time?**

32

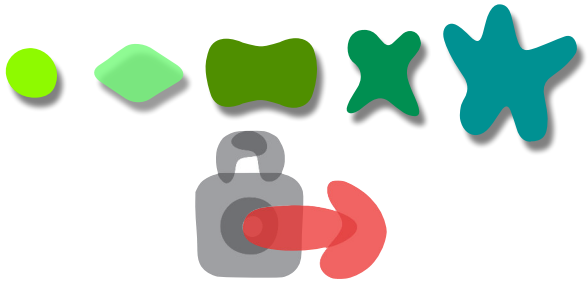
sEcONd-ORdeR eFfEcT

33

sEcONd-ORdeR eFfEcT

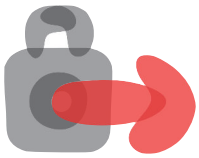


sEcONd-ORdeR eFfEcT



34

governance



35

Evolutionary Architecture

36

Evolutionary Architecture

Continuous Architecture?

37

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

38

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

39

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

Agile Architecture?

Adaptable Architecture?

40

Evolutionary Architecture

Continuous Architecture?

Incremental Architecture?

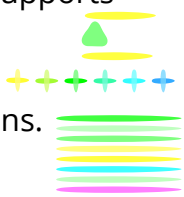
Agile Architecture?

Adaptable Architecture?

41

Evolutionary Architecture

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



42

Evolutionary Architecture

An evolutionary architecture supports
guided
incremental change
across multiple dimensions.

43

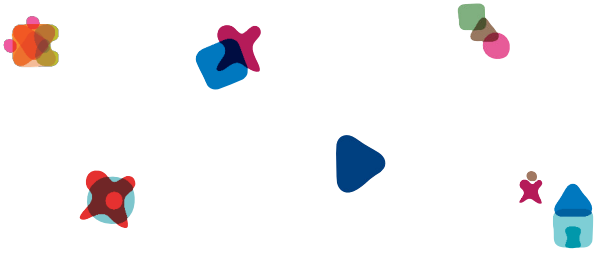
guided

evolutionary computing fitness function:

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

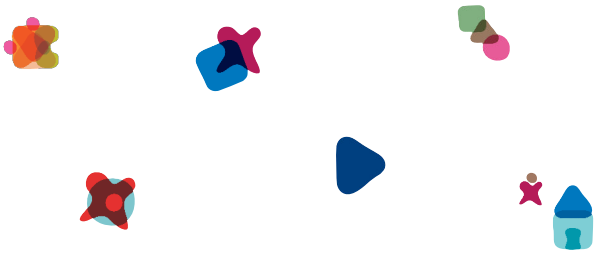
44

Traveling Salesman Problem



45

Traveling Salesman Problem



fitness function = length of route

46

47

guided

architectural *fitness function*:

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

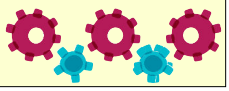
48

EA Spreadsheet

✓ definition

✓ verification

Penultimate



49

Evolutionary Architecture

An evolutionary architecture supports
guided
incremental change
across multiple dimensions.

50

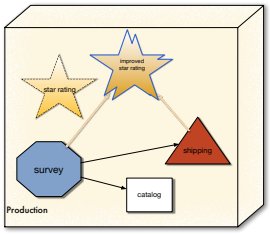
Evolutionary Architecture

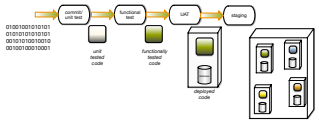
An evolutionary architecture supports
guided,
incremental change
across multiple dimensions.

51

incremental

52





Evolutionary Architecture

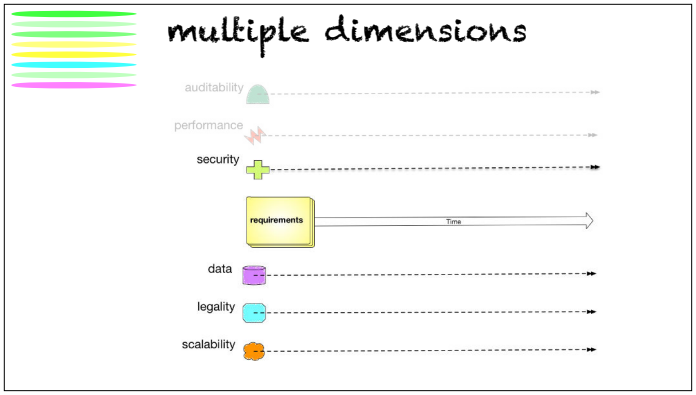
53

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

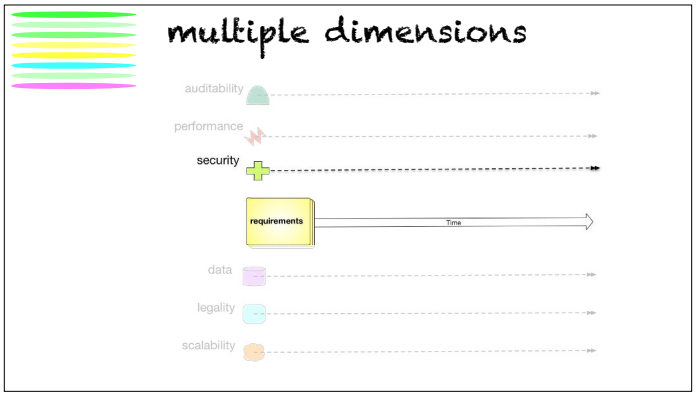
Evolutionary Architecture

54

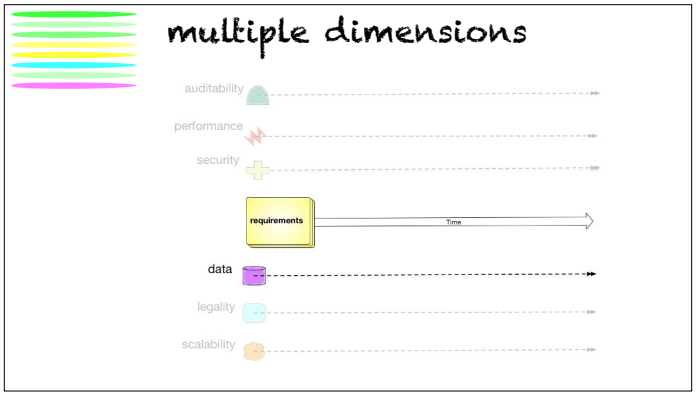
An evolutionary architecture supports
guided,
incremental change
across **multiple dimensions**


multiple dimensions
auditability
performance
security
requirements
Time
data
legality
scalability

55


multiple dimensions
auditability
performance
security
requirements
Time
data
legality
scalability

56


multiple dimensions
auditability
performance
security
requirements
Time
data
legality
scalability

57

Evolutionary Architecture

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

58

Evolutionary Architecture

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

59

Agenda Part 1

Definition

Incremental Change

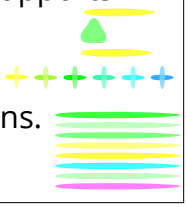
Guided Change via Fitness Functions

Structuring architecture for evolution

60

Evolutionary Architecture

An evolutionary architecture supports
guided
incremental change
across multiple dimensions.



61

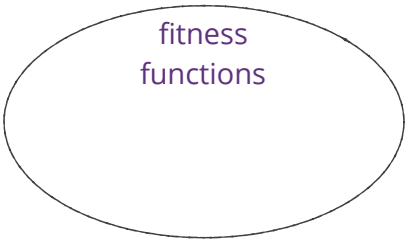
guided

architectural *fitness function*:

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

62

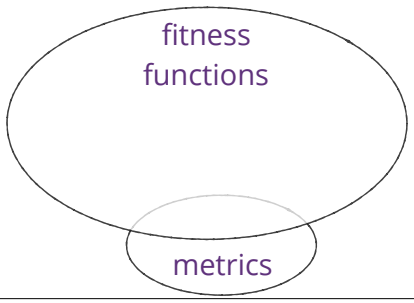
Fitness Functions



63

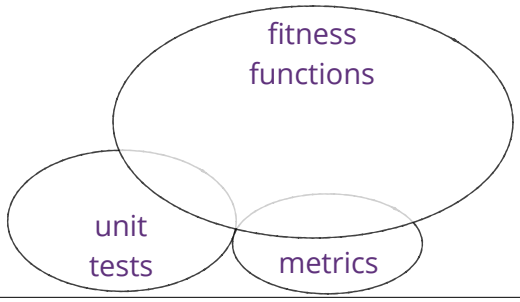
64

Fitness Functions



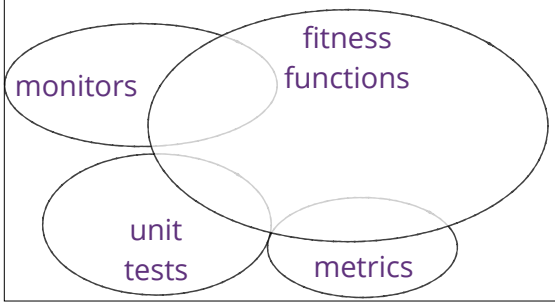
65

Fitness Functions

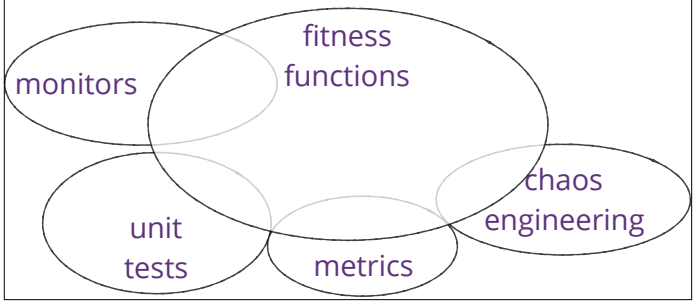


66

Fitness Functions

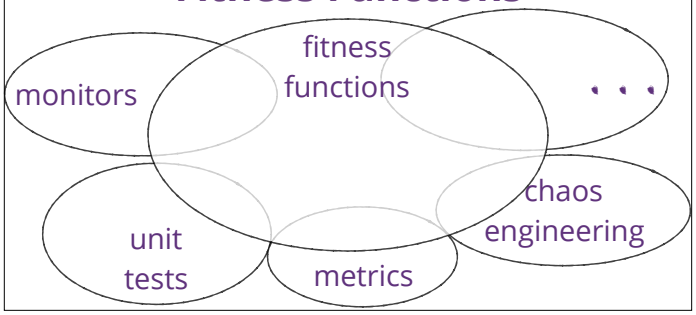


Fitness Functions



67

Fitness Functions

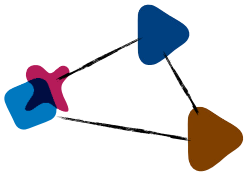


68

Cyclic Dependency Function

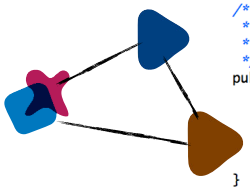
69

Cyclic Dependency Function



70

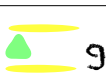
Cyclic Dependency Function



```
/**
 * 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

71



architectural *fitness function*:

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

72

Categories of Fitness Functions

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

73

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

74

Categories of Fitness Functions

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

75

Categories of Fitness Functions

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

76

Categories of Fitness Functions

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

77

Categories of Fitness Functions

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

executes constant verification of architectural aspect(s)

78

Categories of Fitness Functions

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

79

Categories of Fitness Functions

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

rely on a shifting definition based on extra context.

80

Categories of Fitness Functions

tests and other verification mechanism that run without human interaction.

81

Categories of Fitness Functions

automated

tests and other verification mechanism that run without human interaction.

manual

must involve at least one human.

82

Categories of Fitness Functions

temporal

architects may want to build a time component into assessing fitness

83

Categories of Fitness Functions

temporal

architects may want to build a time component into assessing fitness

break on upgrade

84

Categories of Fitness Functions

temporal

architects may want to build a time component into assessing fitness

break on upgrade

overdue library update

85

Categories of Fitness Functions

domain-specific

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

86

Categories of Fitness Functions

architectural characteristic

domain-specific

87

Categories of Fitness Functions

architectural characteristic

problem domain

domain-specific

unit

UAT

functional

88

Categories of Fitness Functions

intentional

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

89

Categories of Fitness Functions

intentional

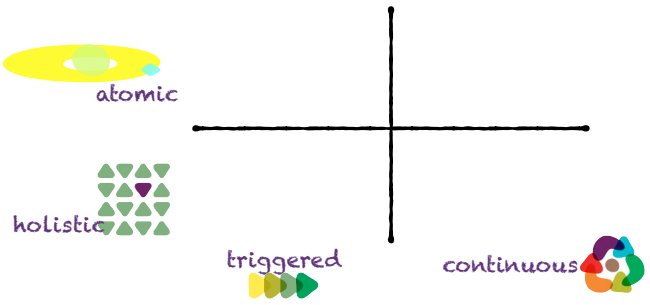
architects will define most fitness functions at project inception as they elucidate the characteristics of the architecture...

emergent

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

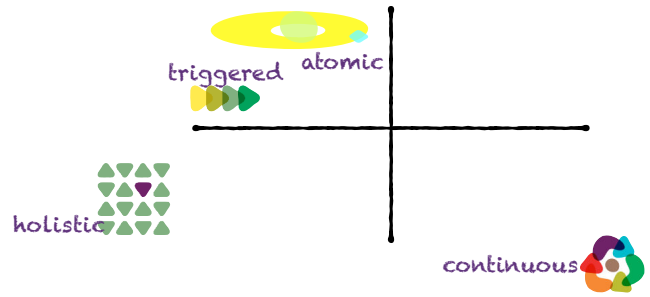
90

Fitness Function



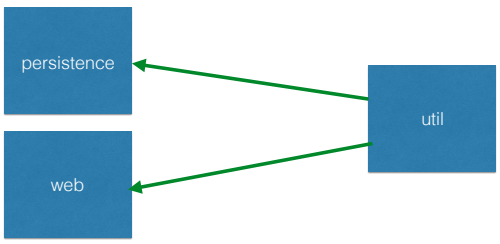
91

Fitness Function



92

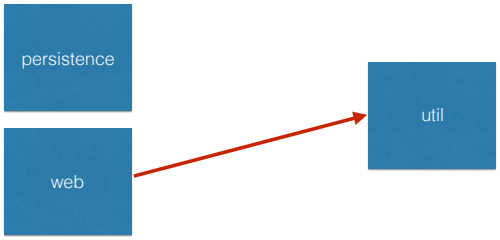
Directionality of Imports



packages/namespaces

93

Directionality of Imports



packages/namespaces

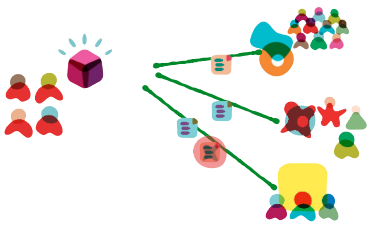
94

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));
    }
```

95

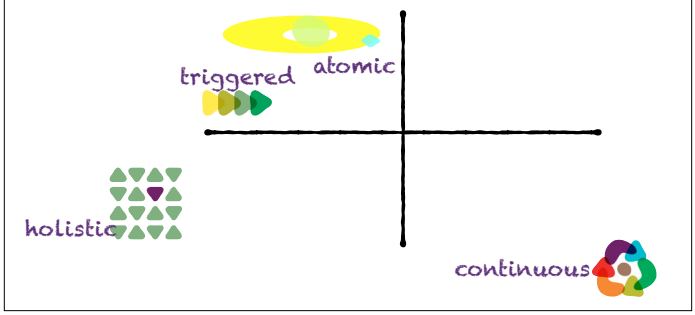
Consumer Driven Contracts



martinfowler.com/articles/consumerDrivenContracts.html

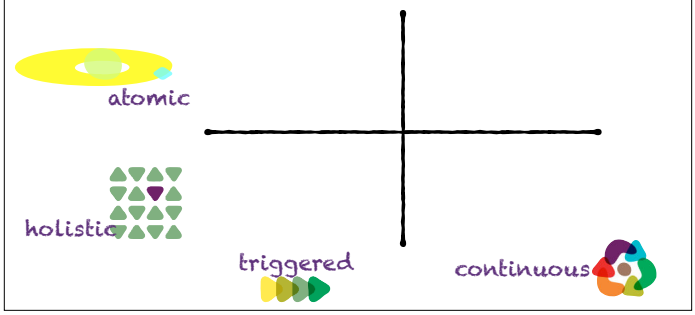
96

Fitness Function



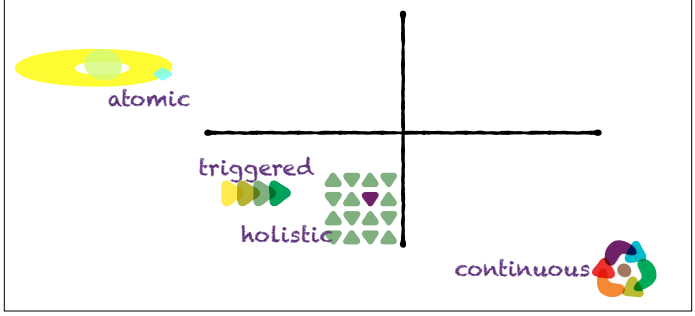
97

Fitness Function

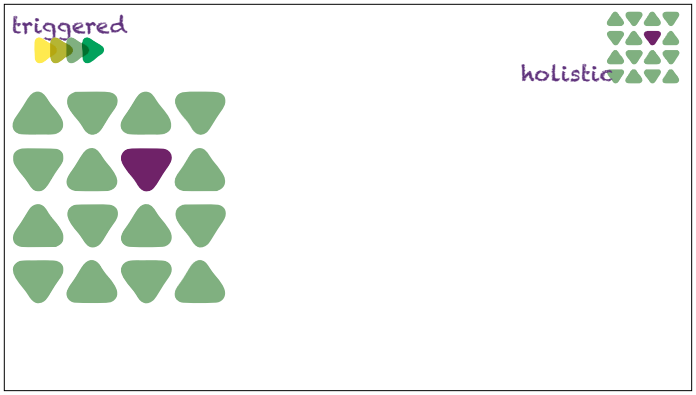


98

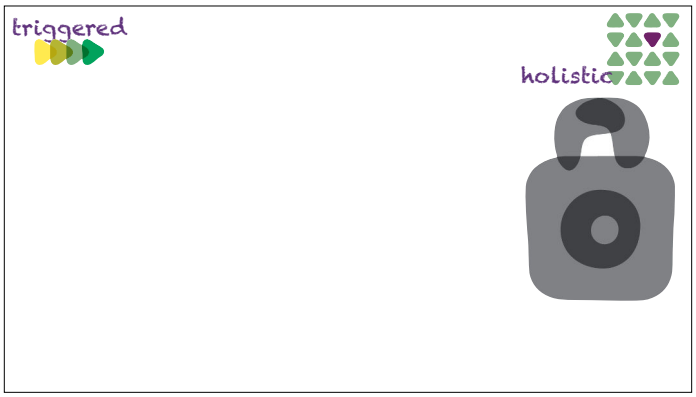
Fitness Function



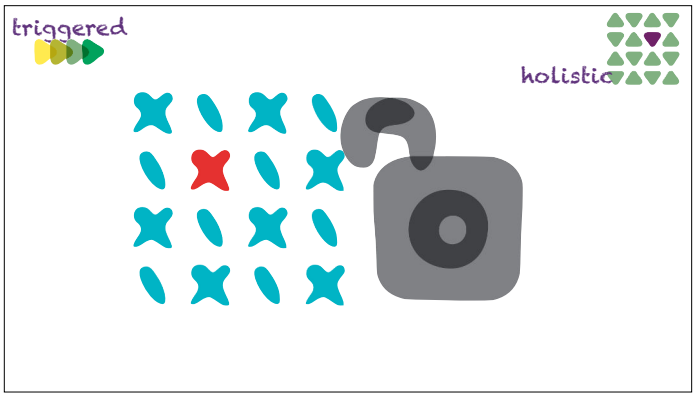
99


triggered
holistic

100

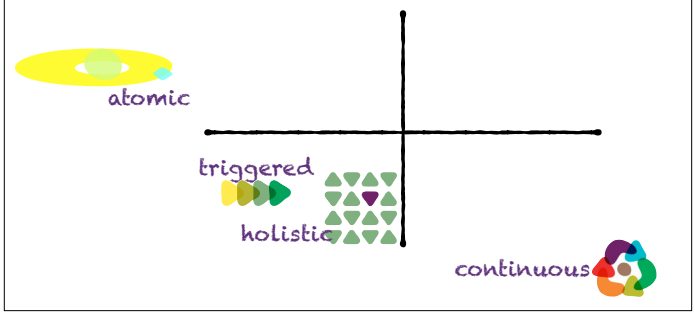

triggered
holistic

101


triggered
holistic

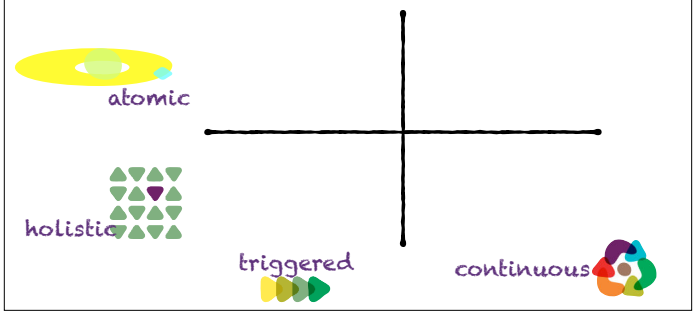
102

Fitness Function



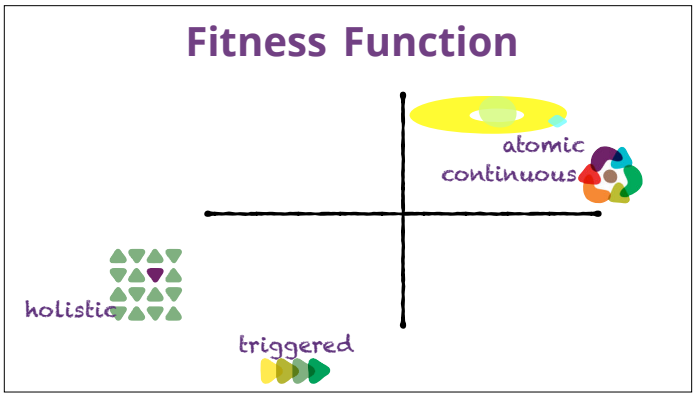
103

Fitness Function

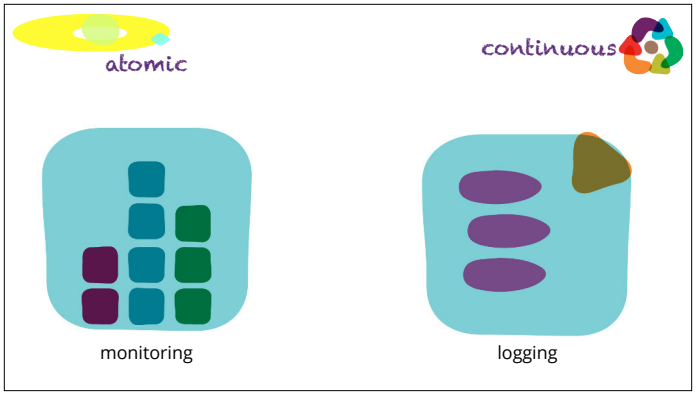


104

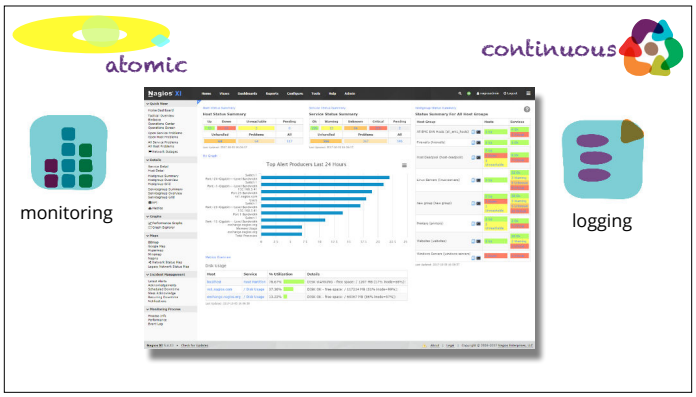
Fitness Function



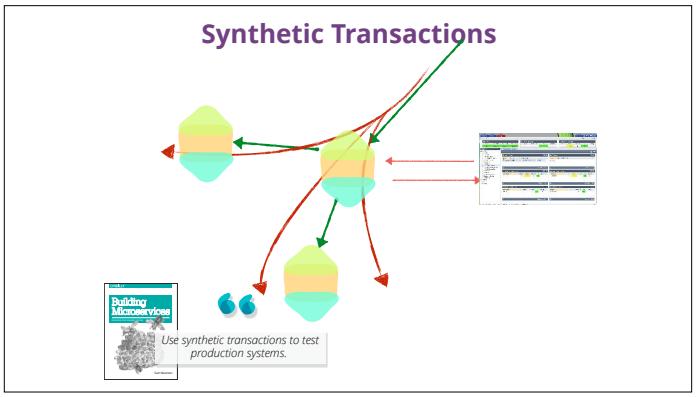
105


atomic
continuous
monitoring
logging

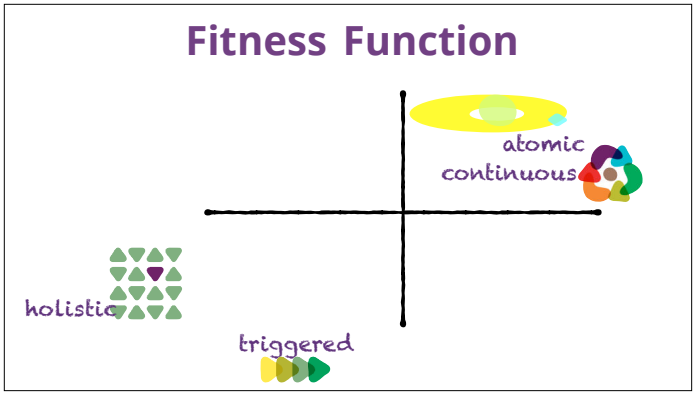
106


atomic
continuous
monitoring
logging

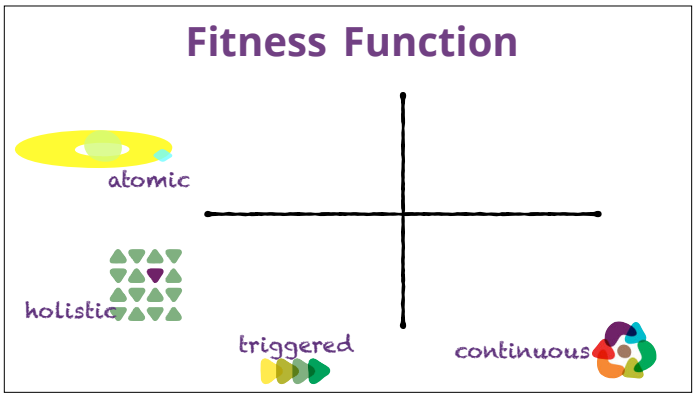
107


Synthetic Transactions
Building Microservices
Use synthetic transactions to test production systems.

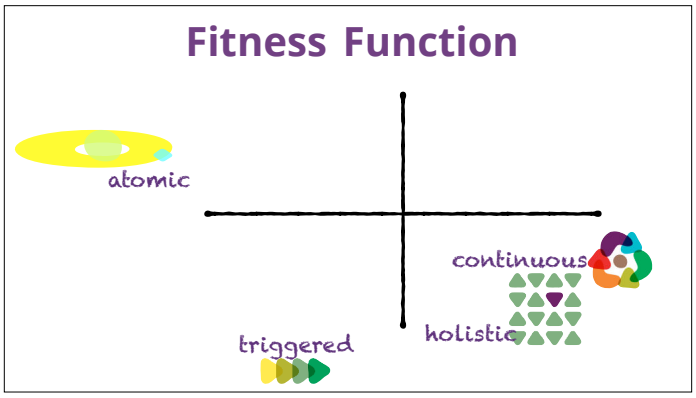
108


Fitness Function
atomic
continuous
holistic
triggered

109


Fitness Function
atomic
holistic
triggered
continuous

110


Fitness Function
atomic
continuous
holistic
triggered

111


amazon
web services
CHAOS
MONKEY

112


amazon
web services
CHAOS
MONKEY
SIMIAN
ARMY

113


chaos monkey
SIMIAN
ARMY

114


chaos gorilla
SIMIAN
ARMY

115


latency monkey
SIMIAN
ARMY

116


doctor monkey
SIMIAN
ARMY

117


janitor monkey
SIMIAN
ARMY

118

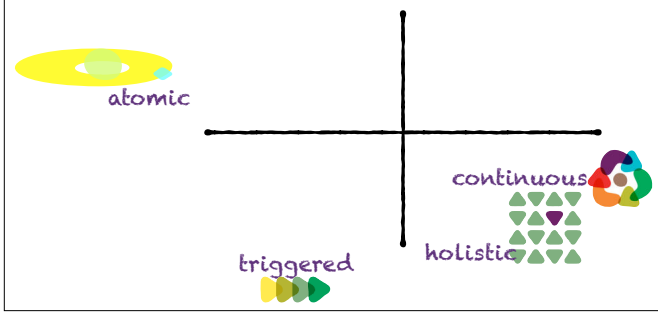

conformity monkey
SIMIAN
ARMY

119


security monkey
SIMIAN
ARMY

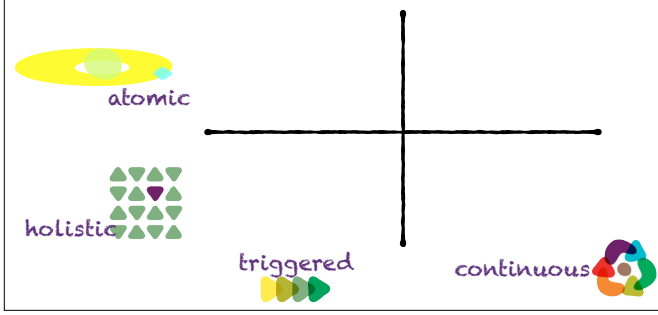
120

Fitness Function



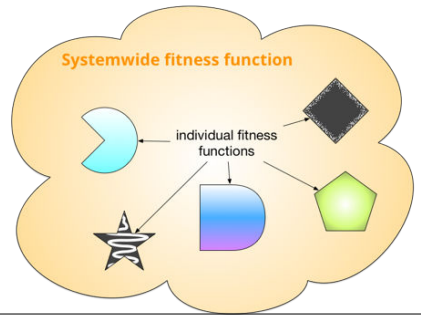
121

Fitness Function



122

System-wide Fitness Function



123

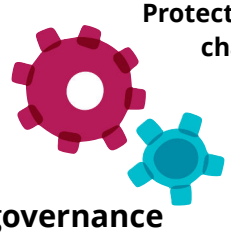
Implementing Fitness Functions

Protecting architectural characteristics

124

Implementing Fitness Functions

Protecting architectural characteristics



Automating governance

125

maintainable?

126

127

maintainable?

128

Cyclomatic complexity < 50 for all projects

maintainable?

129

Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

(incoming/outgoing)
Controlled afferent/efferent coupling

130

Cyclomatic complexity < 50 for all projects

Naming conventions

immutability

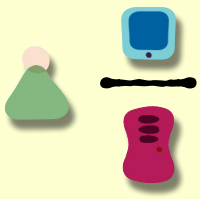
maintainable?

(incoming/outgoing)
Controlled afferent/efferent coupling

131

Penultimate

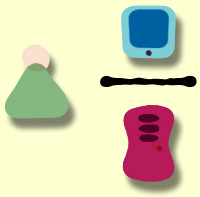
Governing Code Quality



132

Penultimate

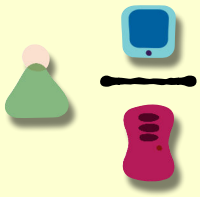
Governing Code Quality



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


Penultimate

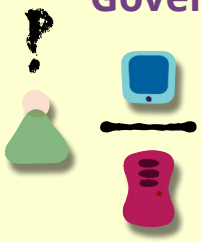
Governing Code Quality



```
if state ==    then
    doSom      L();
else if        " then
    doS        ;
else i
```

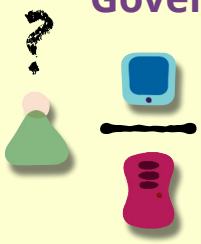
Penultimate

Governing Code Quality



Penultimate

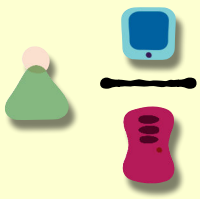
Governing Code Quality



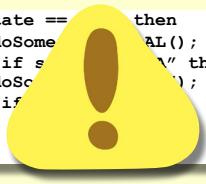
```
if state == "AL" then
    doSomethingForAL();
else if state == "GA" then
    doSomethingForGA();
else if …
```


Penultimate

Governing Code Quality

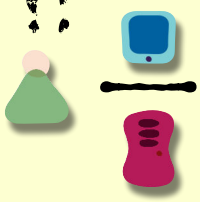


```
if state ==      then
    doSome      AL();
else if s       " then
    doSo        );
else if
```



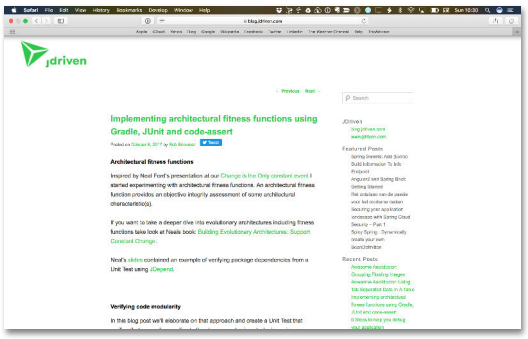
Penultimate

Governing Code Quality



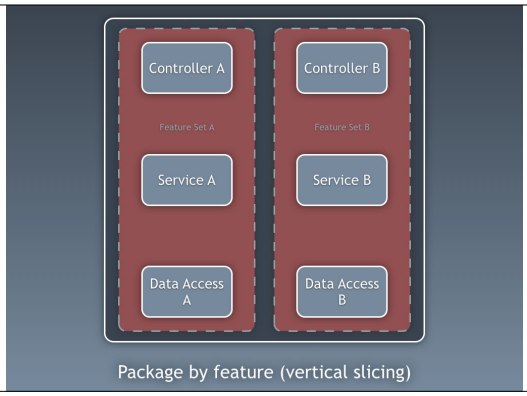
Strategy Design Pattern

139



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

140



141

```java
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());
    }
}
```

142

```java
public class ControllerA {

    private final ServiceA serviceA;
    private final ServiceB serviceB;

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

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

143

```java
public class ControllerA {

    private final ServiceA serviceA;
    private final ServiceB serviceB;

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

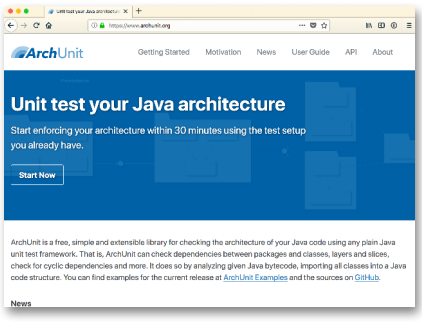
```
java.lang.AssertionError:
 Expected: Comply with rules
 but: DENIED com.jdriven.fitness.packaging.by.feature.a -&gt;
 com.jdriven.fitness.packaging.by.feature.b (by com.jdriven.fitness.packaging.by.feature.a.ControllerA)
```

```java
// Allow package / module a to acces b
a.mayUse(b);
```

144

ArchUnit

https://www.archunit.org/



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);
    }
}
```

145

ArchUnit

https://www.archunit.org/

coding rules

```
@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);
}
```

146

ArchUnit

https://www.archunit.org/

interface rules

```
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);
    }
}
```

147

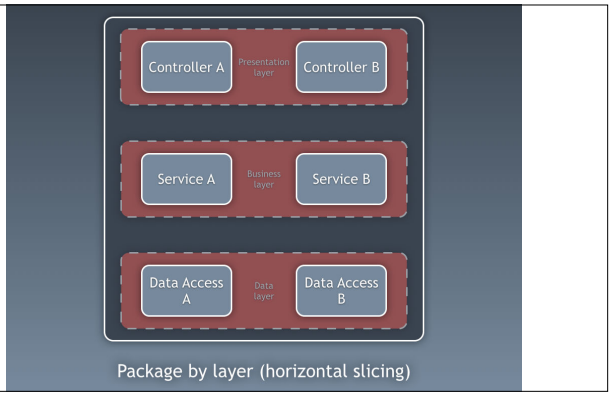
ArchUnit

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
```

ir

148



149

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

150

```java
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);
}
```

151

ArchUnit

https://www.archunit.org/

governance

```java
@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));
}
```

152

Legality of Open Source Libraries

Penultima⬆e

153

Legality of Open Source Libraries

Penultimate

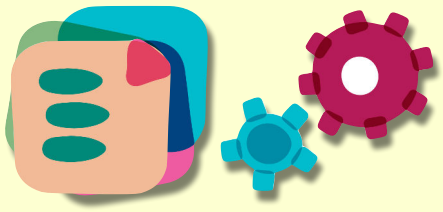
154

Legality of Open Source Libraries

Penultimate

155

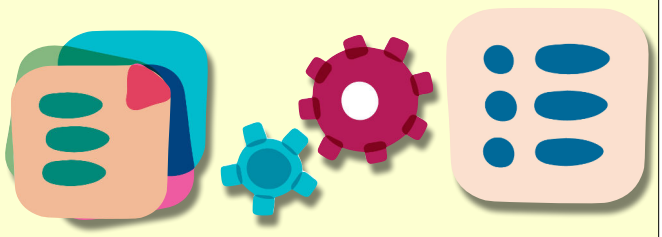
Legality of Open Source Libraries



Penultimate

156

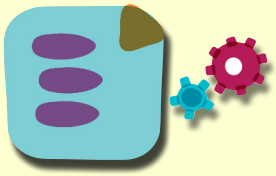
Legality of Open Source Libraries



Penultimate

157

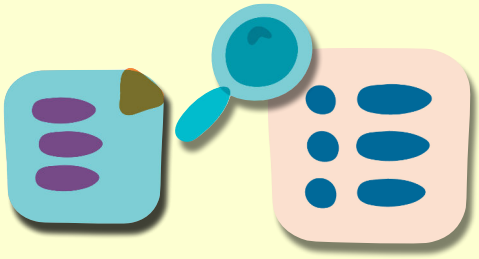
Legality of Open Source Libraries



Penultimate

158

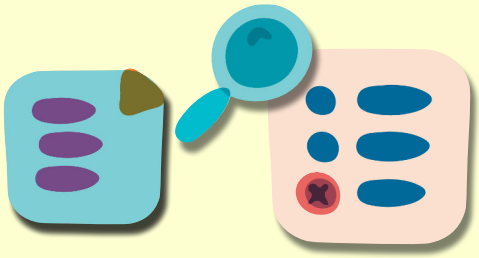
Legality of Open Source Libraries



Penultimate

159

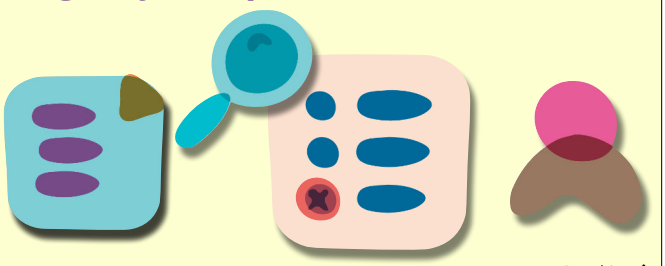
Legality of Open Source Libraries



Penultimate

160

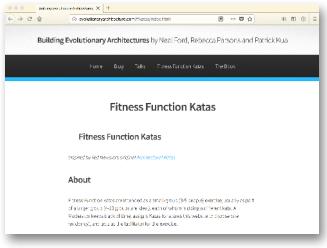
Legality of Open Source Libraries



Penultimate

161

Fitness Function Katas



http://evolutionaryarchitecture.com/ffkatas/

162

Two Big Ideas

163

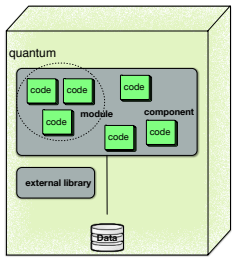
Two Big Ideas

fitness functions for evolvability

164

Two Big Ideas

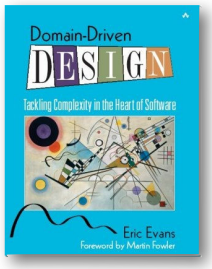
fitness functions for evolvability

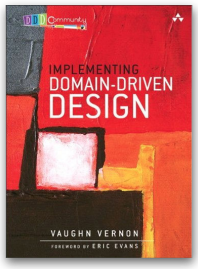


architectural quantum

165

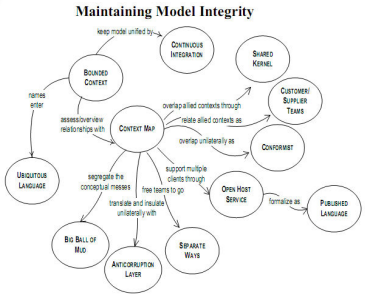
Domain Driven Design





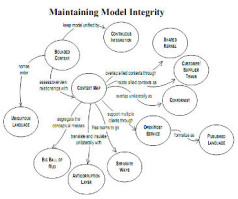
166

Bounded Context

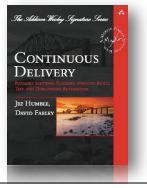


167

Bounded Context + Continuous Delivery = microservices



+



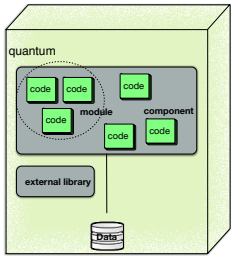
168

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.

169

Architectural Quantum



170

Why Quantum?

171

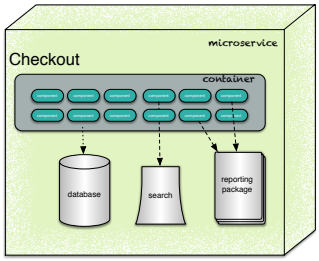
Why Quantum?

operational view
of architecture

172

Why Quantum?

operational view
of architecture



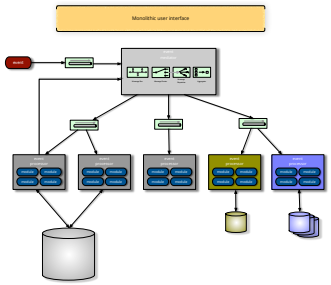
173

Why Quantum?

holistic

174

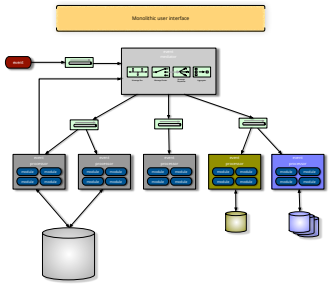
Why Quantum?



holistic

175

Why Quantum?



holistic

"multiple dimensions"

176

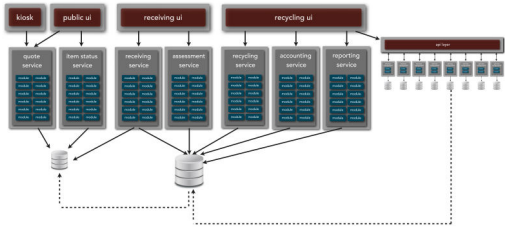
Why Quantum?

useful for architectural analysis

177

Why Quantum?

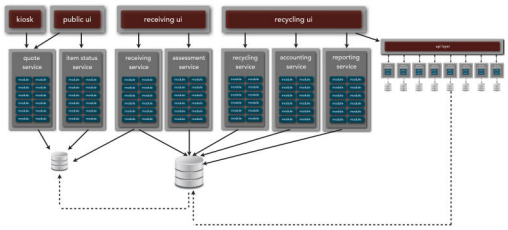
useful for
architectural analysis



178

Why Quantum?

useful for
architectural analysis

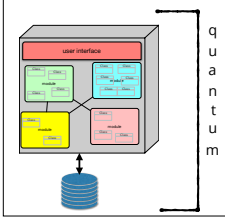


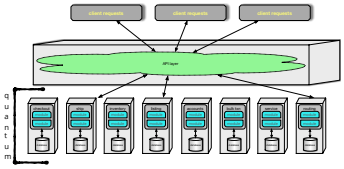
helps analyze
coupling

179

Why Quantum?

The quantum is where architectural characteristics live.



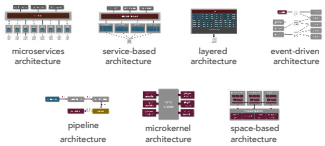


180

Architectural Patterns

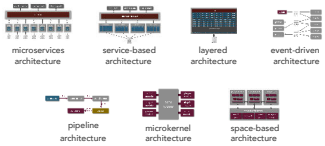
181

Architectural Patterns



182

Architectural Patterns

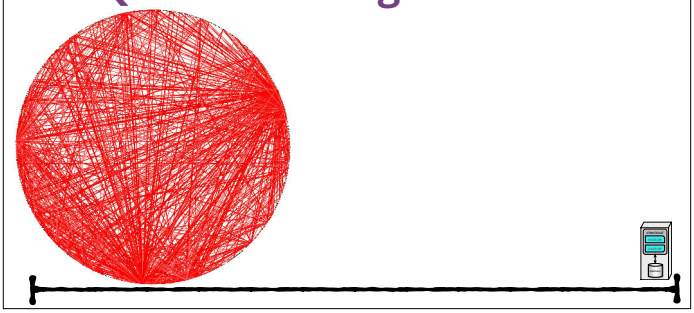


Patterns encapsulate a well-known set of architectural characteristics.

183

184

Quantum: Large to Small

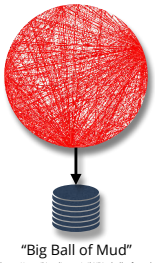


185

Monoliths

186

Monoliths

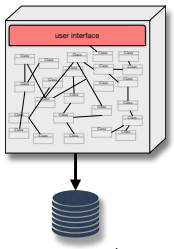


"Big Ball of Mud"

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

Monoliths

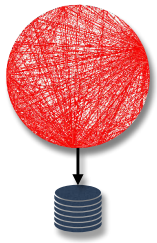
"Big Ball of Mud"
https://en.wikipedia.org/wiki/Big_ball_of_mud



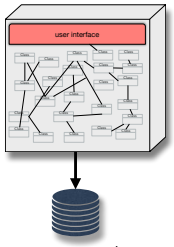
unstructured
monolith

187

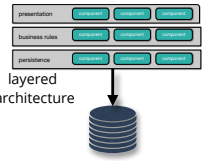
Monoliths



"Big Ball of Mud"
https://en.wikipedia.org/wiki/Big_ball_of_mud



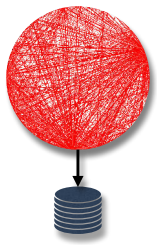
unstructured
monolith



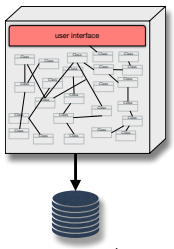
layered
architecture

188

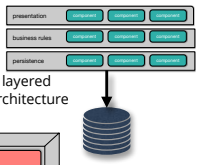
Monoliths



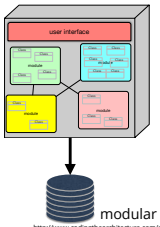
"Big Ball of Mud"
https://en.wikipedia.org/wiki/Big_ball_of_mud



unstructured
monolith



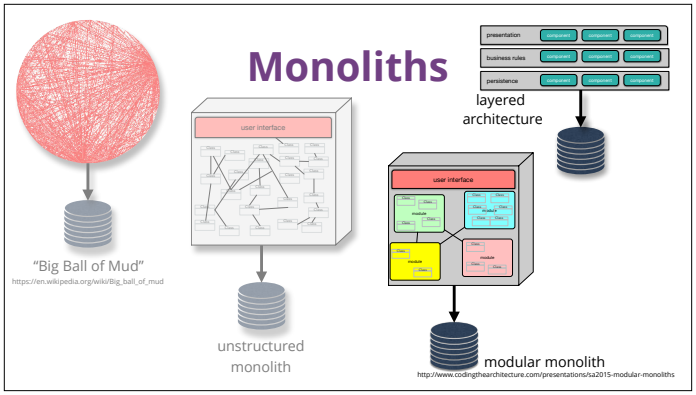
layered
architecture



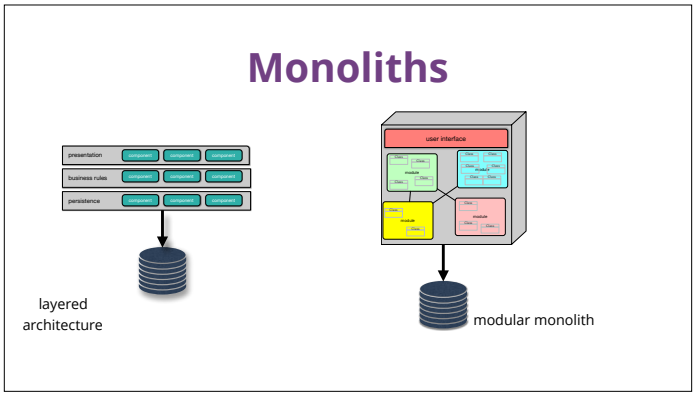
modular monolith
http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths

189

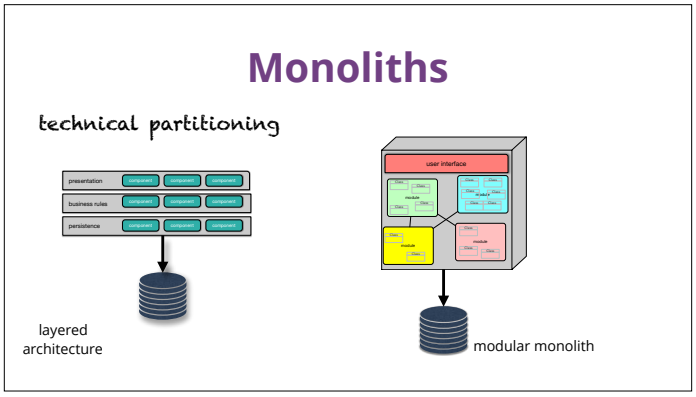
190


Monoliths
"Big Ball of Mud"
https://en.wikipedia.org/wiki/Big_ball_of_mud
unstructured
monolith
layered
architecture
modular monolith
http://www.codingthearchitecture.com/presentations/sa2015-modular-monoliths

191


Monoliths
layered
architecture
modular monolith

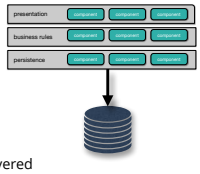
192


Monoliths
technical partitioning
layered
architecture
modular monolith

193

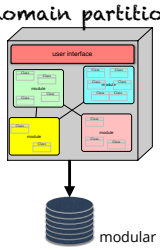
Monoliths

technical partitioning



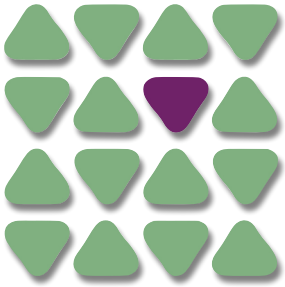
layered architecture

domain partitioning



modular monolith

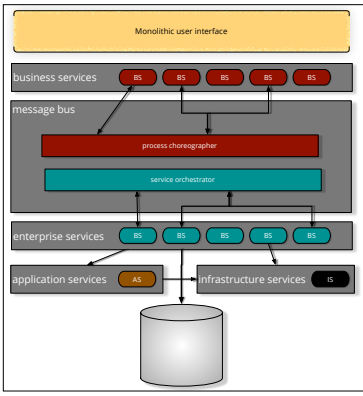
194

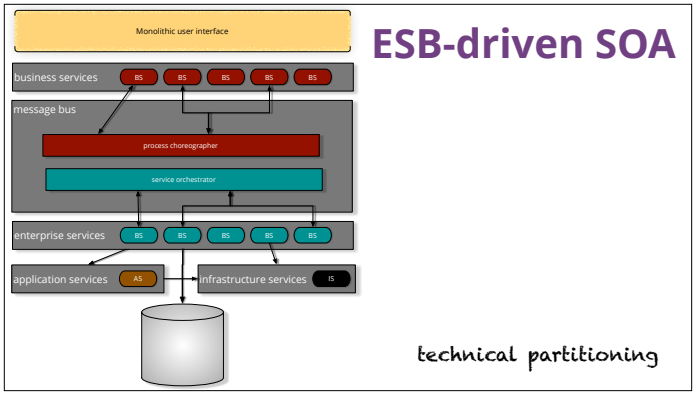


distributed systems

195

ESB-driven SOA





ESB-driven SOA

196

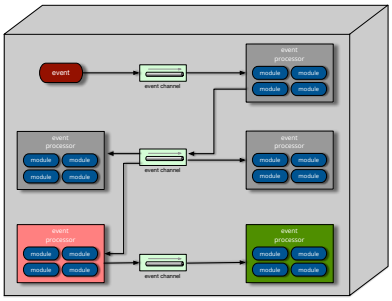
Event-driven Architectures

broker topology

mediator topology

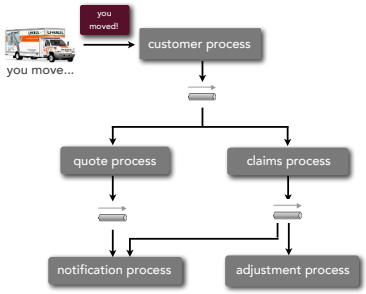
197

Broker Base Architecture



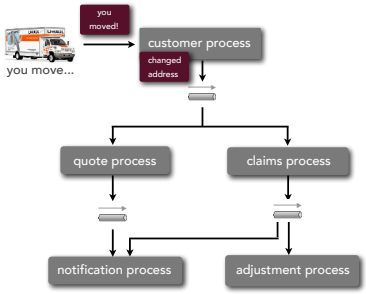
198

Broker Message Passing



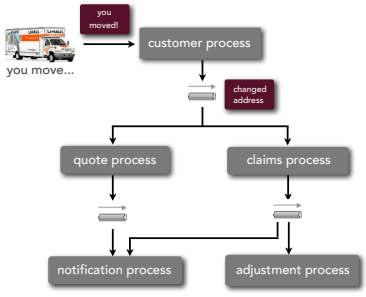
199

Broker Message Passing



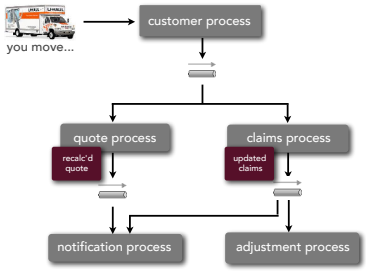
200

Broker Message Passing



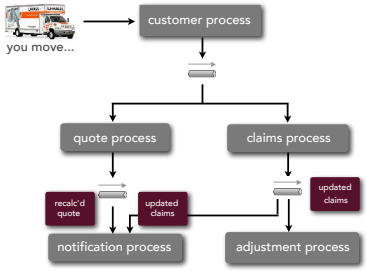
201

Broker Message Passing

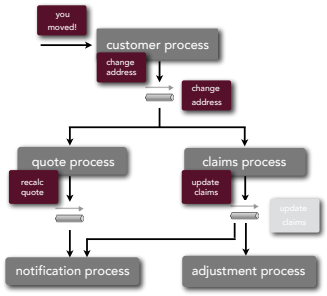


202

Broker Message Passing



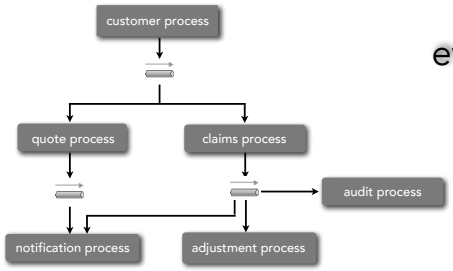
203



Error handling?

204

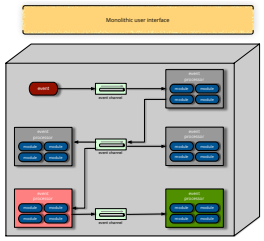
205



evolution

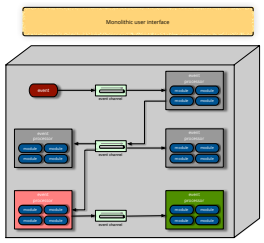
206

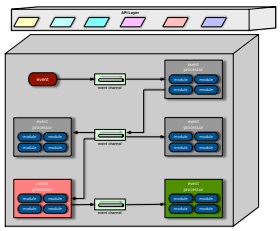
Broker: UI



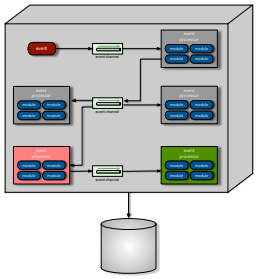
207

Broker: UI



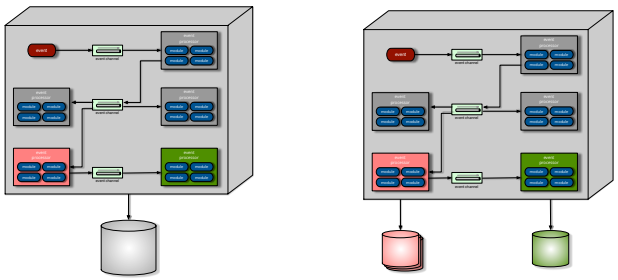


Broker: Data



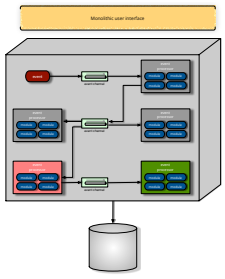
208

Broker: Data



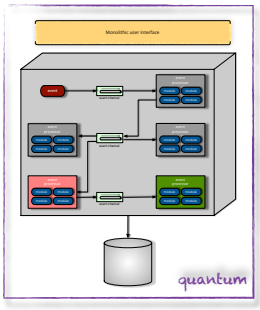
209

Broker: Quanta



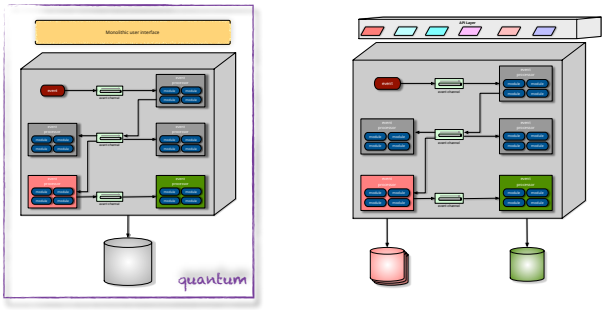
210

Broker: Quanta



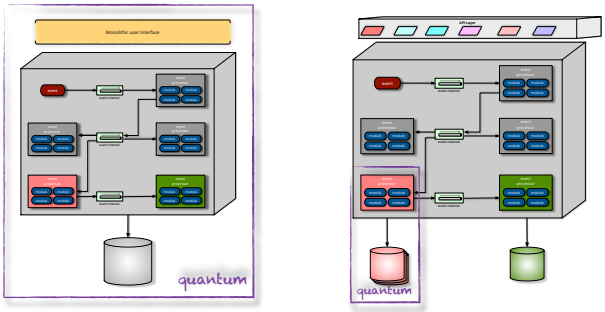
211

Broker: Quanta



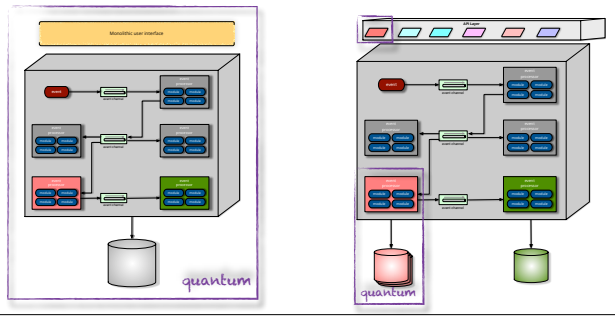
212

Broker: Quanta



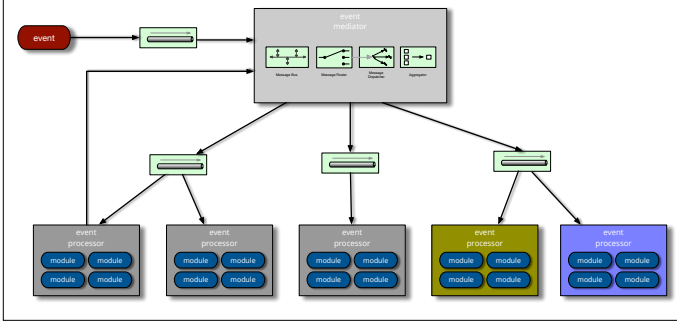
213

Broker: Quanta



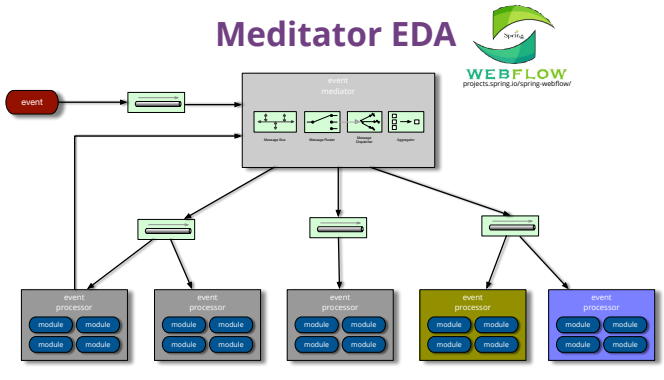
214

Meditator EDA

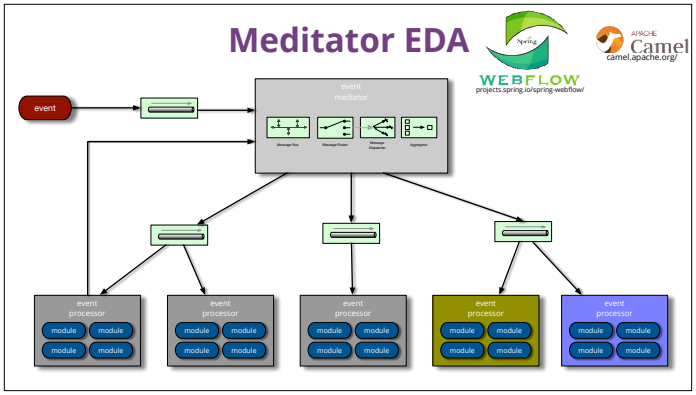


215

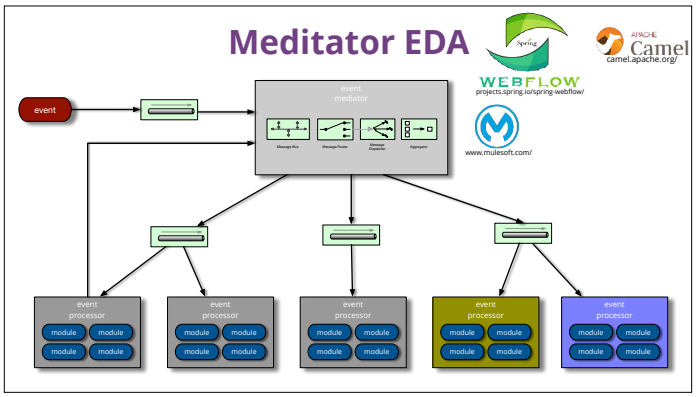
Meditator EDA



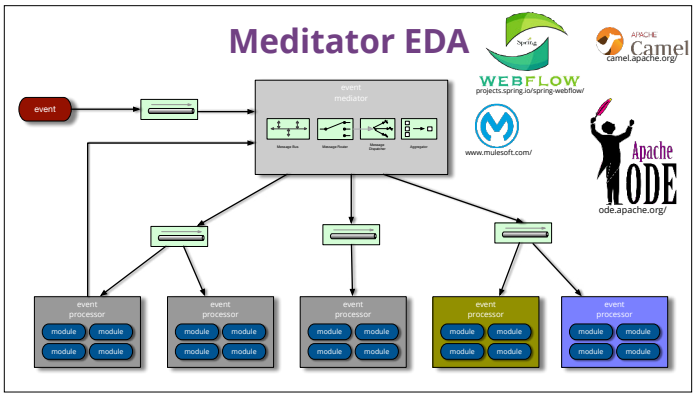
216


Meditator EDA
WEBFLOW
projects.spring.io/spring-webflow/
APACHE Camel
camel.apache.org/
event
event mediator
event processor
module

217

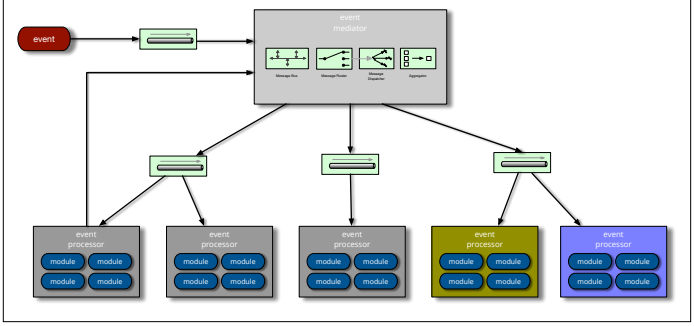

Meditator EDA
WEBFLOW
projects.spring.io/spring-webflow/
APACHE Camel
camel.apache.org/
www.mulesoft.com/
event
event mediator
event processor
module

218


Meditator EDA
WEBFLOW
projects.spring.io/spring-webflow/
APACHE Camel
camel.apache.org/
www.mulesoft.com/
Apache ODE
ode.apache.org/
event
event mediator
event processor
module

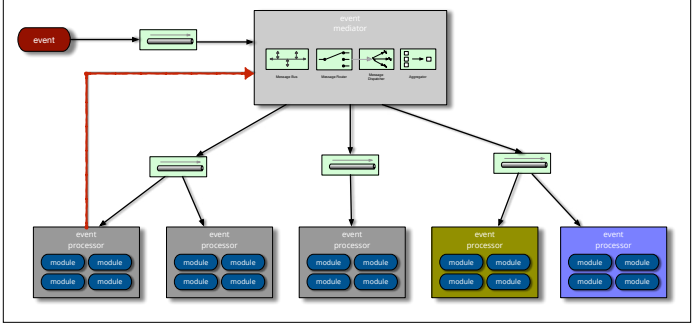
219

Meditator EDA



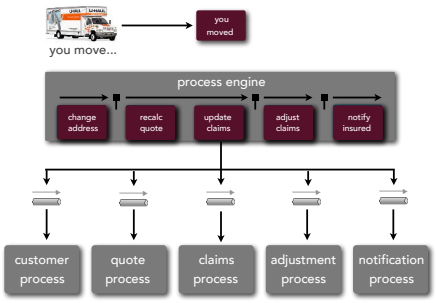
220

Meditator EDA



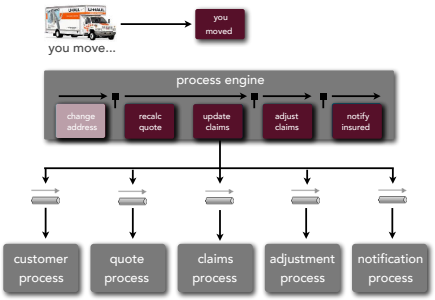
221

Mediator Message Flow



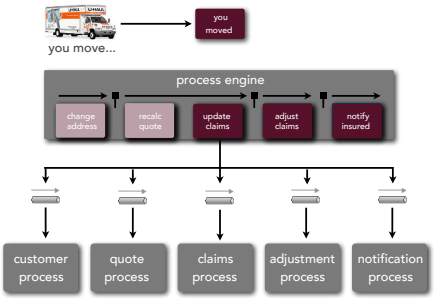
222

Mediator Message Flow



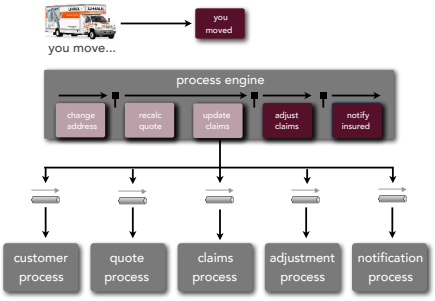
223

Mediator Message Flow



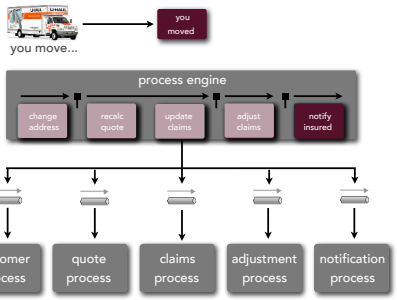
224

Mediator Message Flow



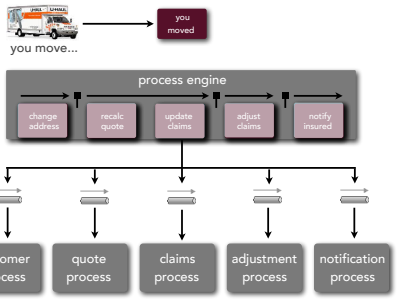
225

Mediator Message Flow

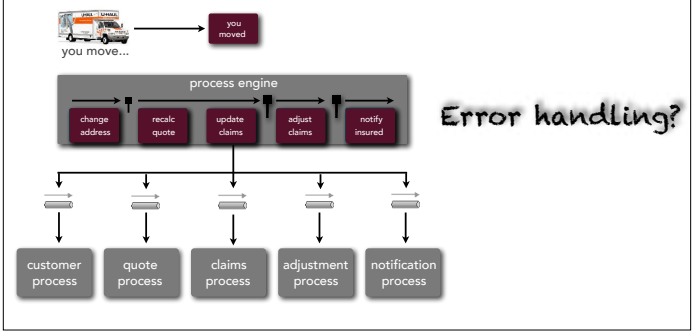


226

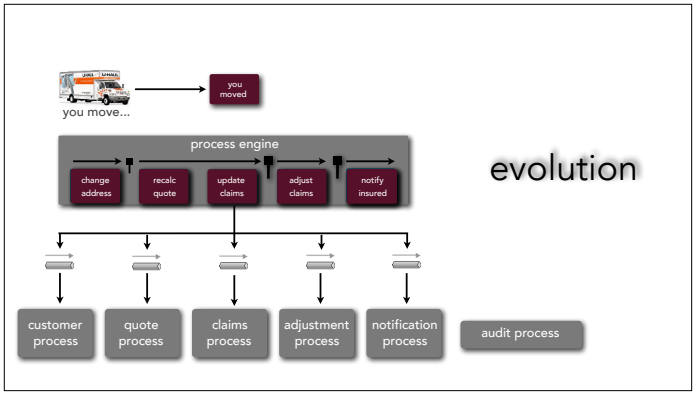
Mediator Message Flow



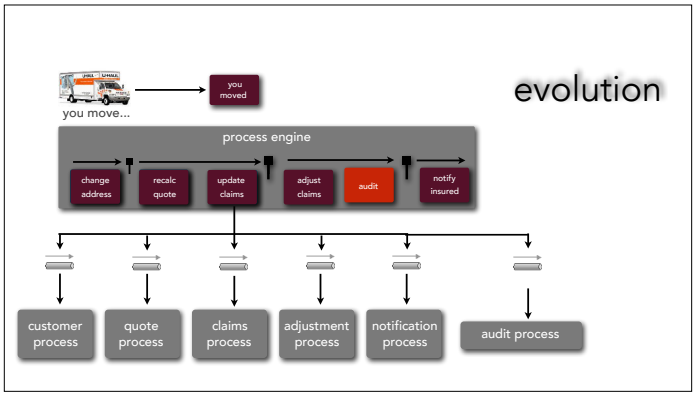
227



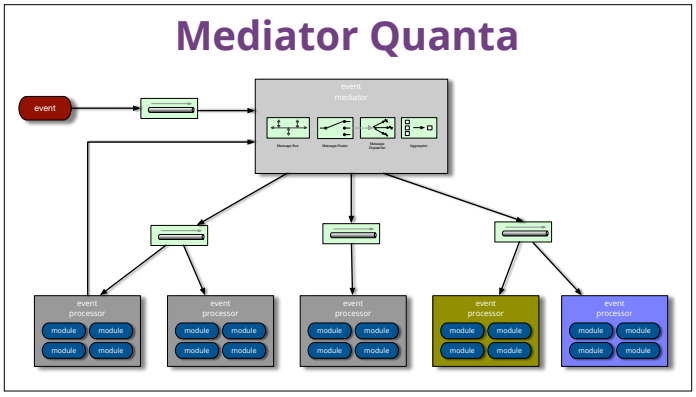
228


you move...
you moved
process engine
change address
recalc quote
update claims
adjust claims
notify insured
evolution
customer process
quote process
claims process
adjustment process
notification process
audit process

229

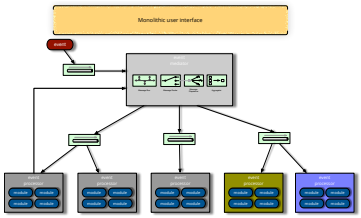

you move...
you moved
evolution
process engine
change address
recalc quote
update claims
adjust claims
audit
notify insured
customer process
quote process
claims process
adjustment process
notification process
audit process

230


Mediator Quanta
event
event mediator
event processor
module

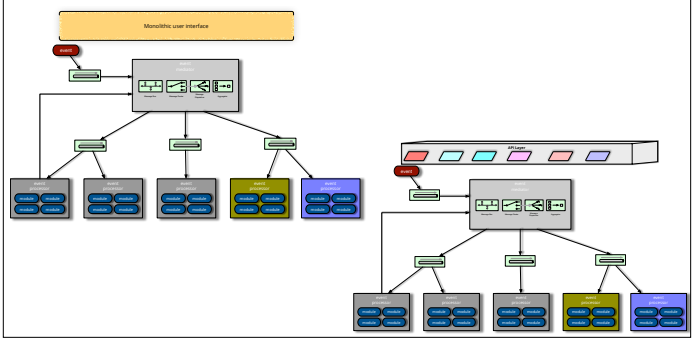
231

Mediator: UI



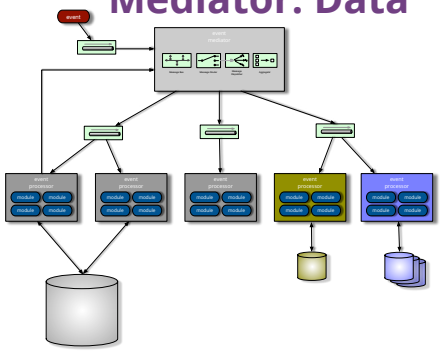
232

Mediator: UI



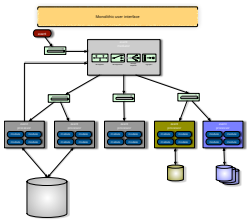
233

Mediator: Data



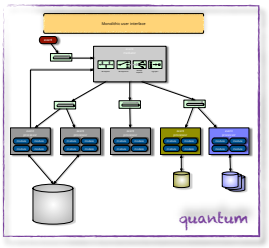
234

Mediator Quanta



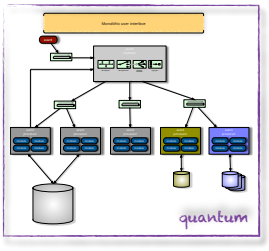
235

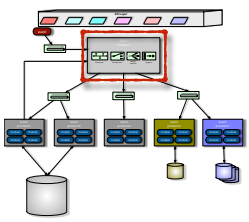
Mediator Quanta



236

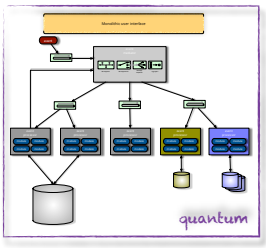
Mediator Quanta

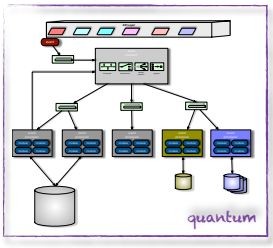




237

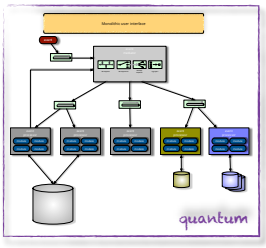
Mediator Quanta

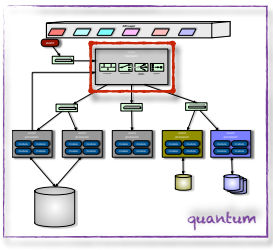




238

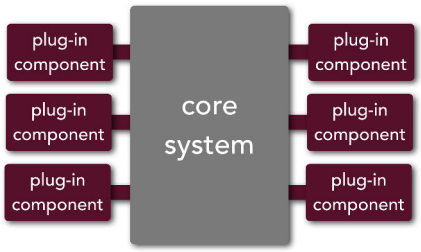
Mediator Quanta





239

Microkernel



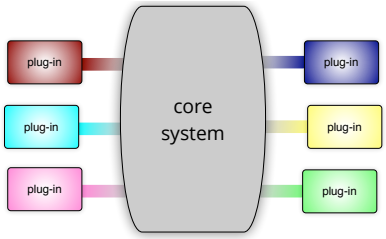
240

Microkernel

claims processing

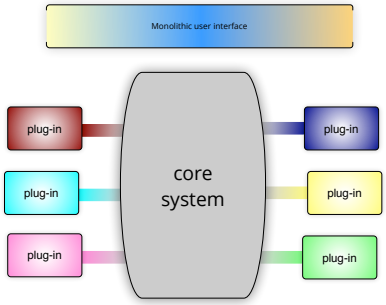
241

Microkernel Quanta



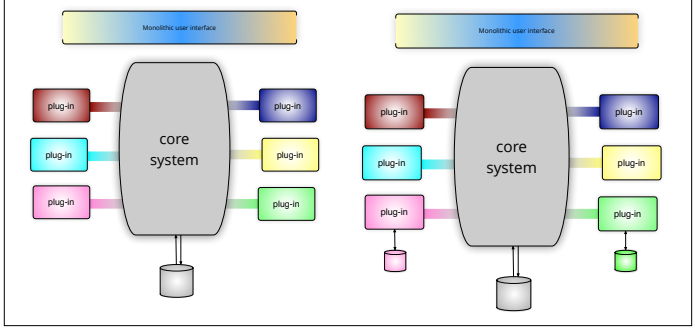
242

Microkernel: UI



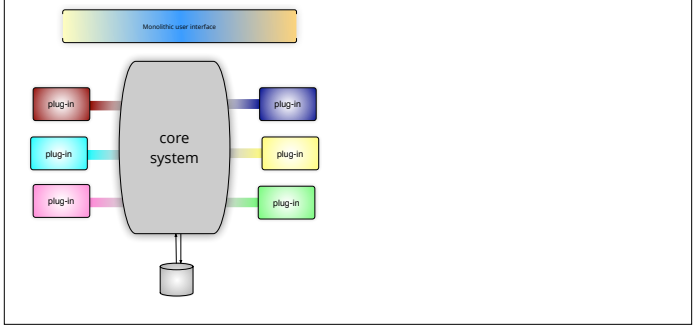
243

Microkernel: Data



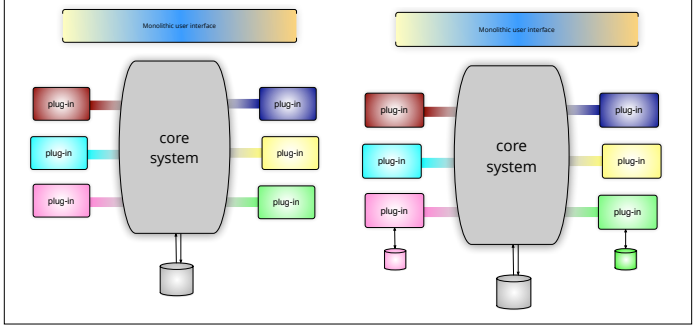
244

Microkernel: Data



245

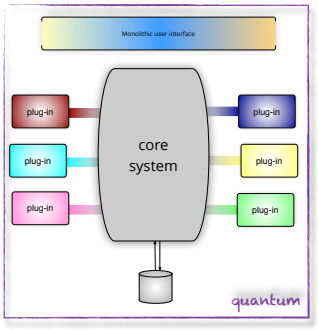
Microkernel: Data



246

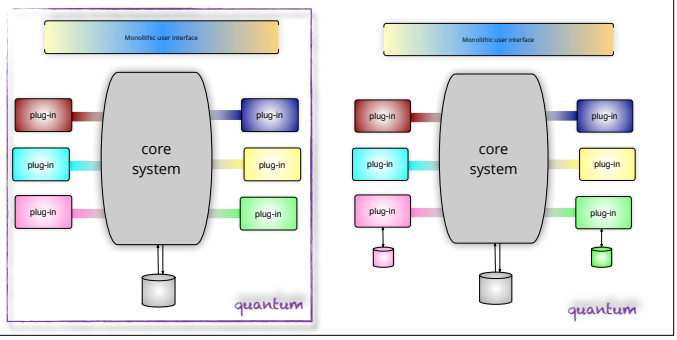
Microkernel: Quanta

247



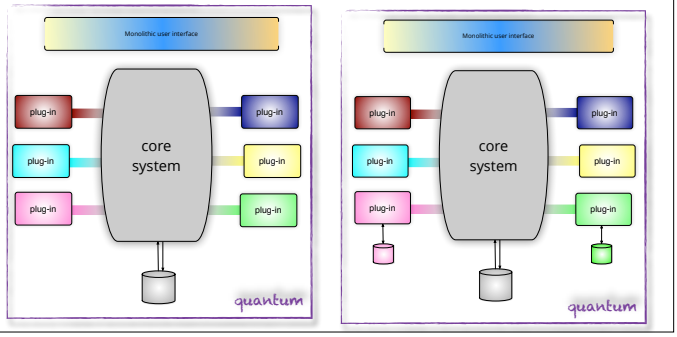
Microkernel: Quanta

248



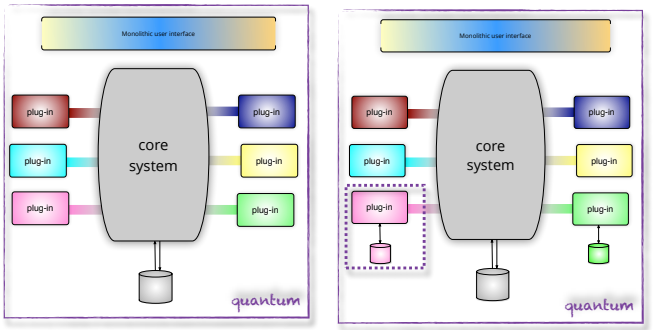
Microkernel: Quanta

249



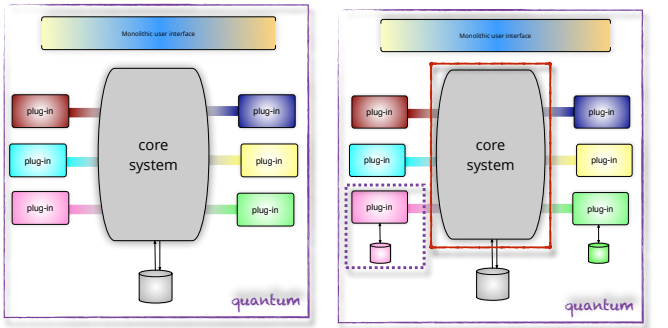
Microkernel: Quanta

250



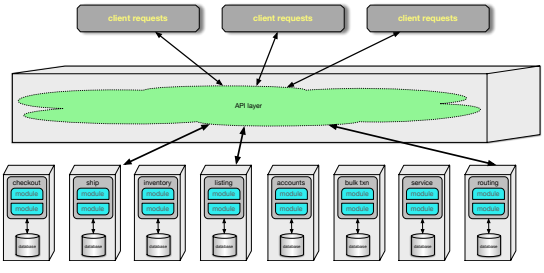
Microkernel: Quanta

251



Microservices

252



What makes microservices so evolvable?

253

extremely loose coupling

What makes microservices so evolvable?

254

extremely loose coupling

What makes microservices so evolvable?

255

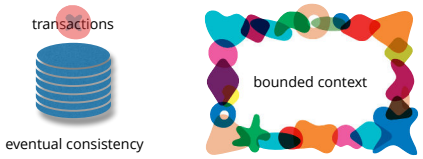
extremely loose coupling

eventual consistency

What makes microservices so evolvable?

256

extremely loose coupling



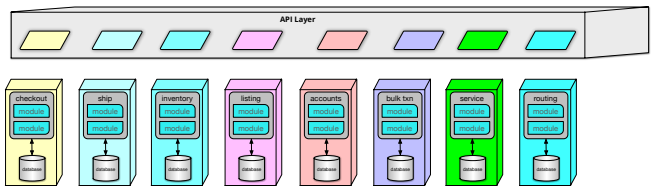
What makes microservices so evolvable?

257

extremely loose coupling

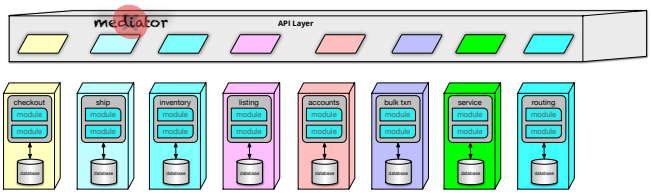
Microservices

258



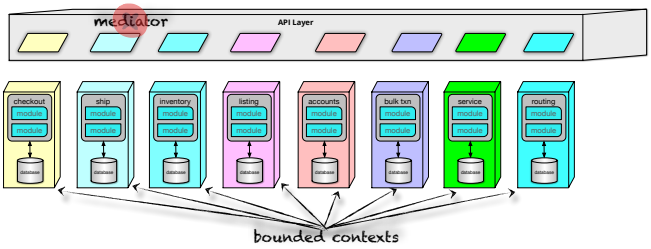
Microservices

259



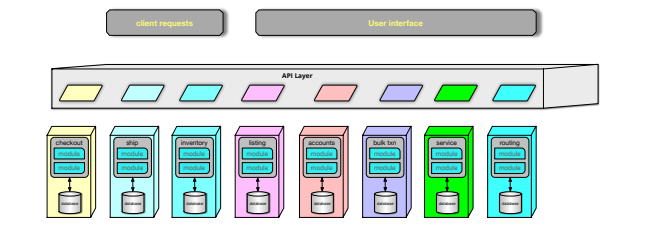
Microservices

260

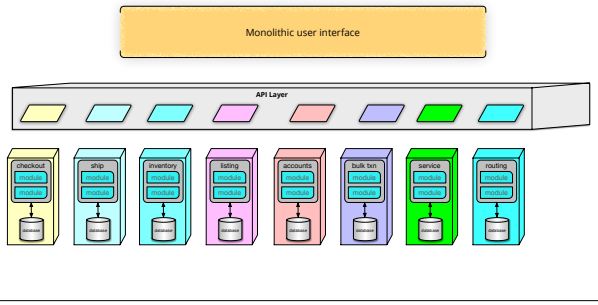


Microservices

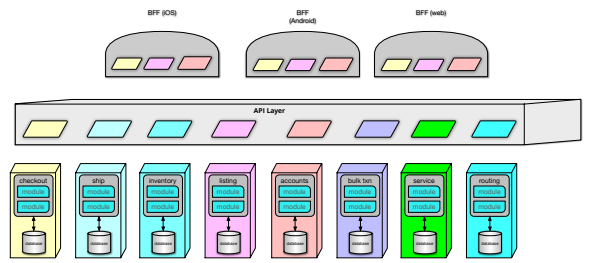
261



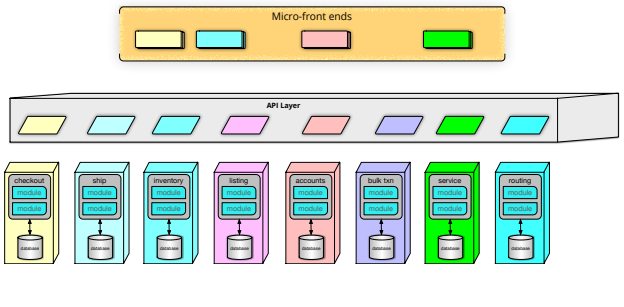
Microservices: Monolithic UI



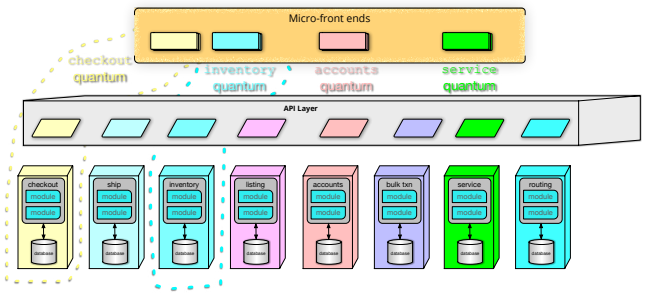
Microservices :BFF



Microservices

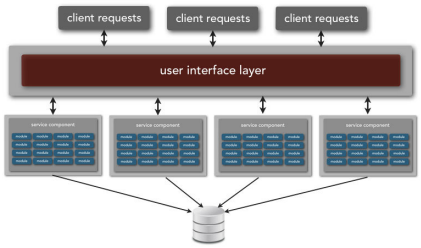


Microservices



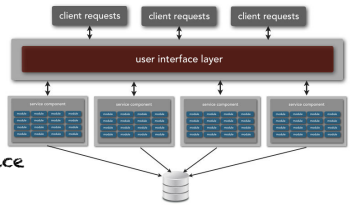
265

Service-based Architectures



266

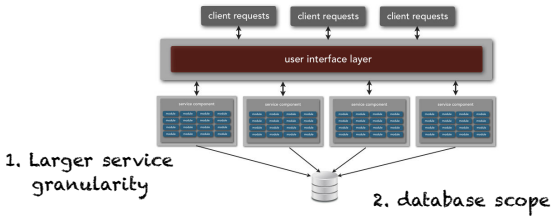
Service-based Architectures



1. Larger service granularity

267

Service-based Architectures



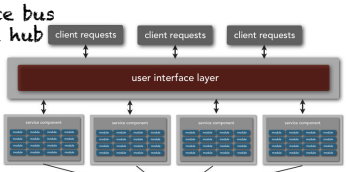
1. Larger service granularity

2. database scope

268

Service-based Architectures

3. use of service bus as integration hub

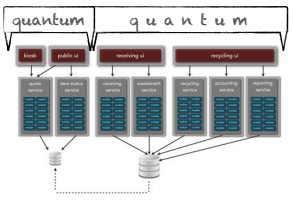


1. Larger service granularity

2. database scope

269

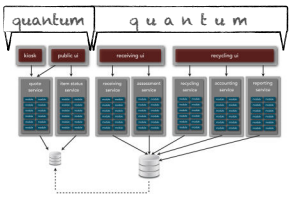
Reducing Quanta Size



270

Reducing Quanta Size

useful for architectural analysis



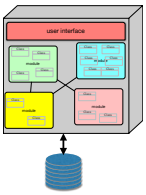
helps analyze coupling

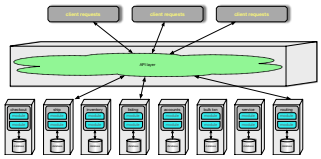
271

Smaller Quanta Size

≙

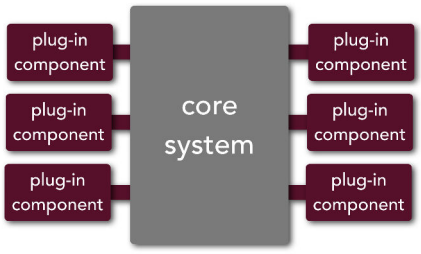
Evolutionary





272

Microkernel Redux



273

Last 10% Trap

274

Last 10% Trap

275

Last 10% Trap

276

Last 10% Trap

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

what the user wants

277

What Happened to the 4GLs?

278

What Happened to the 4GLs?

dBASE™

Microsoft® Visual FoxPro®

Borland® Paradox® for Windows NEW! 5.0

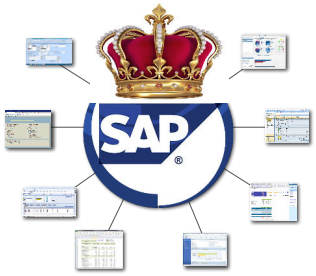
279

What Happened to the 4GLs?

DSL

280

see also: Vendor King



281

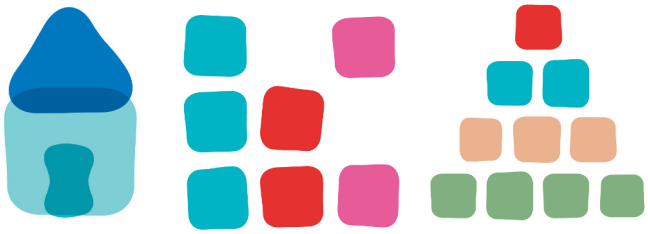
Be careful of the *Last 10% Trap* when choosing tools & frameworks.

282

Domain/Architecture Isomorphism

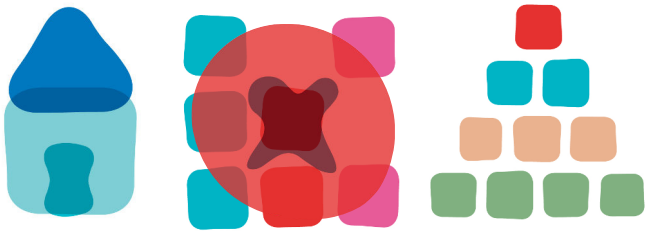
283

Domain/Architecture Isomorphism



284

Domain/Architecture Isomorphism



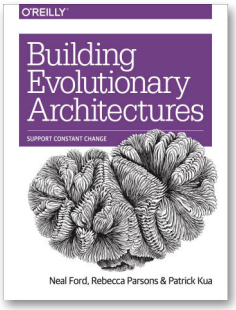
285

Exercise #2

identify the quanta

286

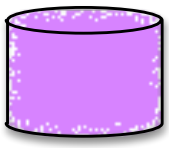
Building Evolutionary Architectures

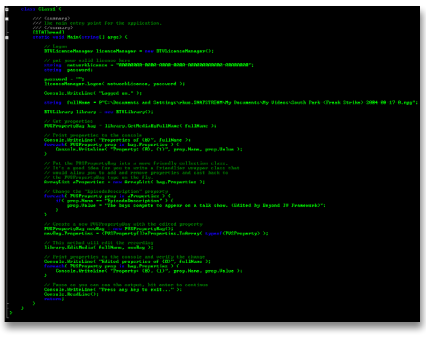


EVOLUTIONARY DATA



287

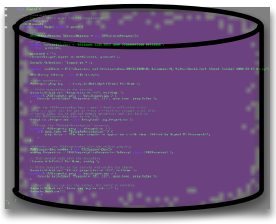




288

289

Data & code are both abstractions based on the real world.



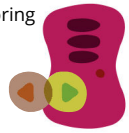
Data & code are symbiotic.

290

DB Evolution & Deployment

scripting all db changes incrementally

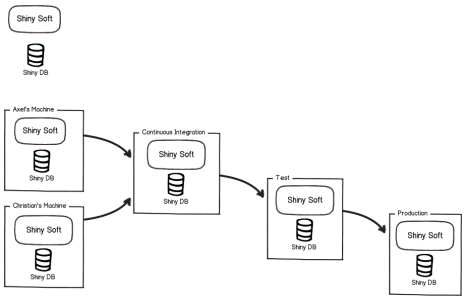
db refactoring

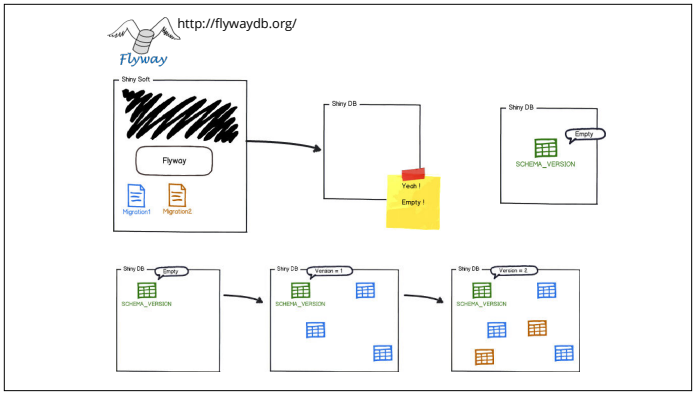


decouple db migration
from app migration

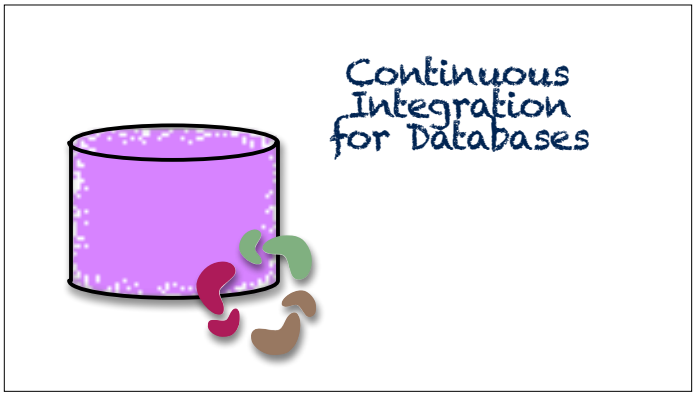
291

DbDeploy Pattern

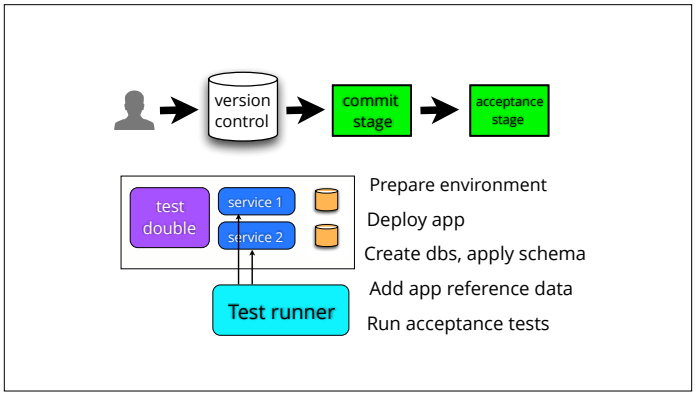



http://flywaydb.org/
Flyway
Shiny Soft
Flyway
Migration1
Migration2
Shiny DB
Yeah !
Empty !
Shiny DB
Empty
SCHEMA_VERSION
Shiny DB
Empty
SCHEMA_VERSION
Shiny DB
Version = 1
SCHEMA_VERSION
Shiny DB
Version = 2
SCHEMA_VERSION

292


Continuous
Integration
for Databases

293


version
control
commit
stage
acceptance
stage
test
double
service 1
service 2
Test runner
Prepare environment
Deploy app
Create dbs, apply schema
Add app reference data
Run acceptance tests

294

For DB CI We Need To:

295

start with a clean database

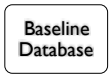
apply changes incrementally

use the same process everywhere

be comprehensive in change management

#1: Baseline

296



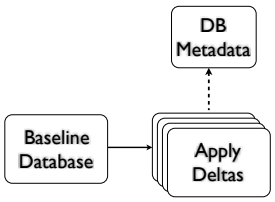
—Create database

—Add metadata table

—Restore scheme & reference data to current production state

#2: Apply Deltas

297

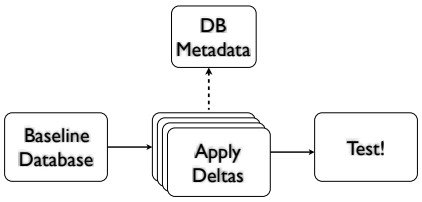


—run each delta in order

—stop the line if one fails

—record success in metadata table

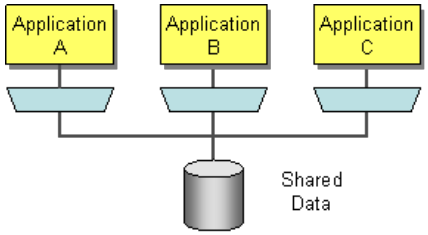
#3: Run Tests



acceptance tests verify database scripts worked

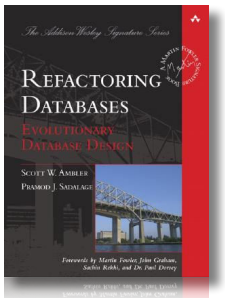
298

Shared-database Integration



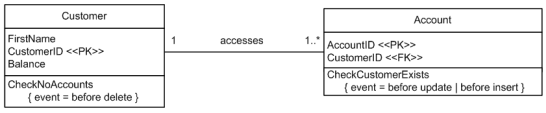
299

Refactoring Databases



300

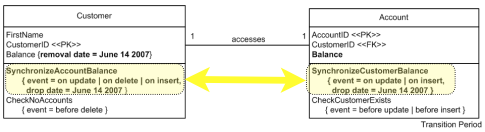
Move Column Refactoring



Original Schema

301

Transition Period

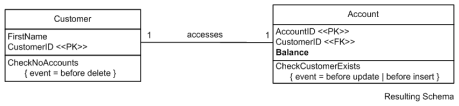


Transition Period

Move Column Refactoring

302

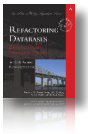
Ending Schema



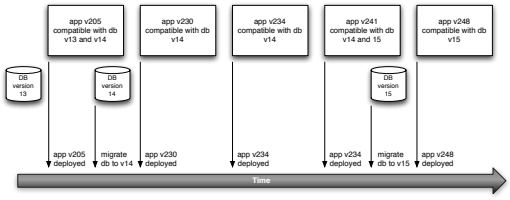
Resulting Schema

Move Column Refactoring

303

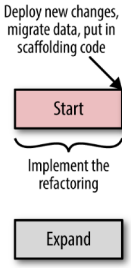


Decouple DB Updates: the Expand/contract Pattern



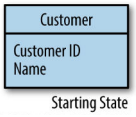
304

Expand/Contract Pattern



305

Change 'name' to 'firstname' & 'lastname'

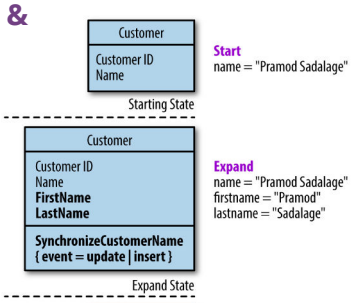


Starting State

Start
name = "Pramod Sadalage"

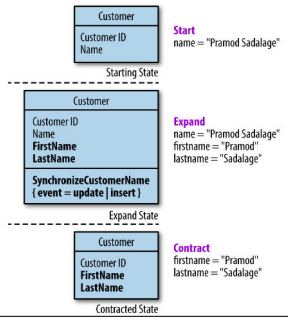
306

Change ‘name’ to ‘firstname’ & ’lastname’



307

Change ‘name’ to ‘firstname’ & ’lastname’



308

#1: ~~integration points, legacy data~~

```
ALTER TABLE customer ADD firstname VARCHAR2(60);
ALTER TABLE customer ADD lastname VARCHAR2(60);
ALTER TABLE customer DROP COLUMN name;
```

309

310

#2: ~~integration points~~, legacy data

```
ALTER TABLE Customer ADD firstname VARCHAR2(60);
ALTER TABLE Customer ADD lastname VARCHAR2(60);
UPDATE Customer set firstname = extractfirstname (name);
UPDATE Customer set lastname = extractlastname (name);
ALTER TABLE customer DROP COLUMN name;
```

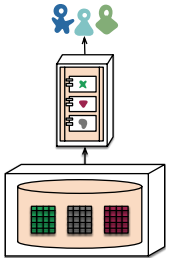
311

#3: integration points, legacy data

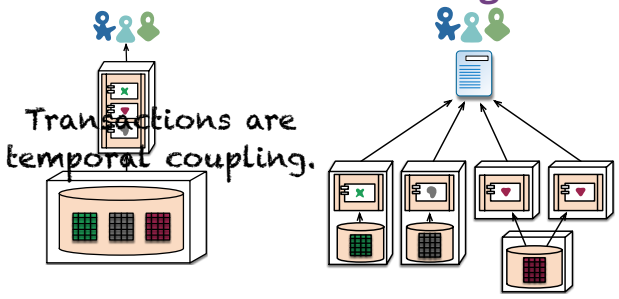
```
ALTER TABLE Customer ADD firstname VARCHAR2(60);
ALTER TABLE Customer ADD lastname VARCHAR2(60);

UPDATE Customer set firstname = extractfirstname (name);
UPDATE Customer set lastname = extractlastname (name);

CREATE OR REPLACE TRIGGER SynchronizeName
BEFORE INSERT OR UPDATE
ON Customer
REFERENCING OLD AS OLD NEW AS NEW
FOR EACH ROW
BEGIN
IF :NEW.Name IS NULL THEN
  :NEW.Name := :NEW.firstname||' '||:NEW.lastname;
END IF;
IF :NEW.name IS NOT NULL THEN
  :NEW.firstname := extractfirstname(:NEW.name);
  :NEW.lastname := extractlastname(:NEW.name);
END IF;
END;
```

312

Decentralized Data Management



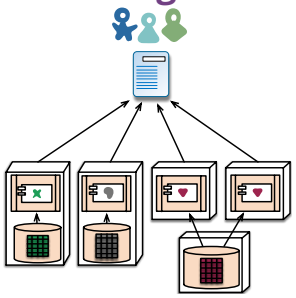
Decentralized Data Management



313

Decentralized Data Management

Limit transactional contexts.



314

Database transactions act as a strong nuclear force, binding quanta together.

315

Penultimate

Evolving Routing

< >

<home> <catalog><item>

316

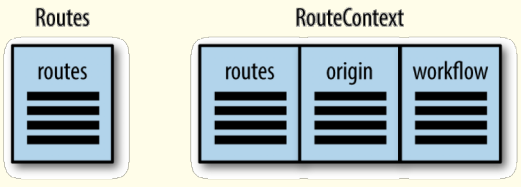
Penultimate

Evolving Routing

317

Penultimate

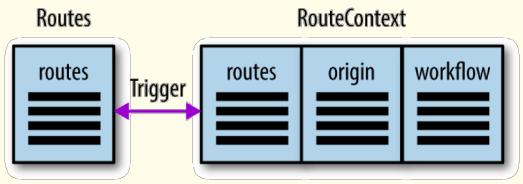
Evolving Routing



318

Penultimate

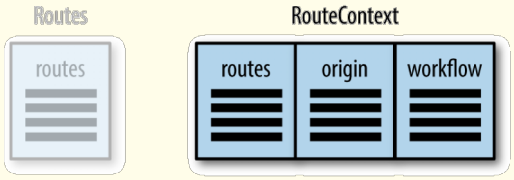
Evolving Routing



319

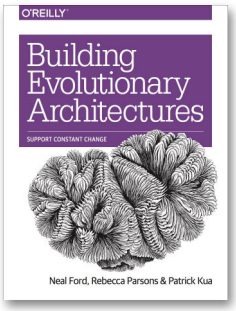
Penultimate

Evolving Routing



320

Building Evolutionary Architectures



BUILDING EVOLVABLE ARCHITECTURES

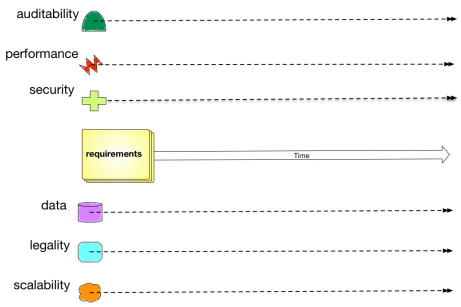
321

Mechanics

1. Identify dimensions

322

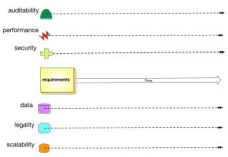
1. Identify dimensions



323

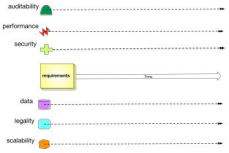
1. Identify dimensions

prioritization



324

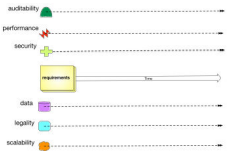
1. Identify dimensions



prioritization

cost to implement & maintain

1. Identify dimensions



prioritization

cost to implement & maintain

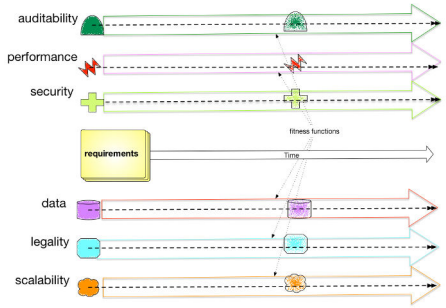
change variable “fit it” cost to constant maintenance cost

Mechanics

1. Identify dimensions

2. Define fitness function(s)

2. Define Fitness Function(s)



328

Mechanics

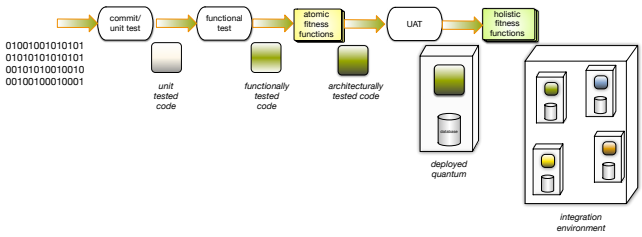
1. Identify dimensions

2. Define fitness function(s)

3. Use deployment pipelines and/or continuous fitness functions

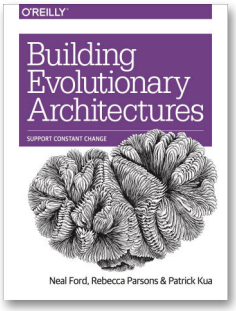
329

3. Use deployment pipelines and/or continuous fitness functions



330

Building Evolutionary Architectures



GUIDELINES FOR EVOLUTIONARY ARCHITECTURES



331

Remove Needless Variables

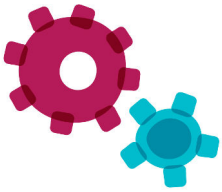
332

Remove Needless Variables

333

334

Remove Needless Variables

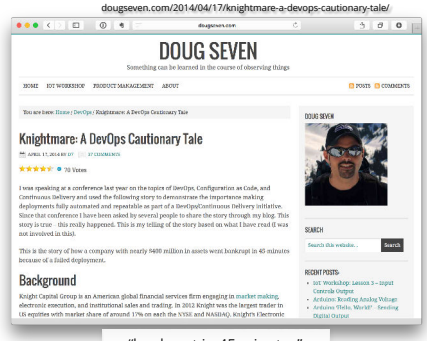


335

Knight Capital

336

dougseven.com/2014/04/17/knightmare-a-devops-cautionary-tale/

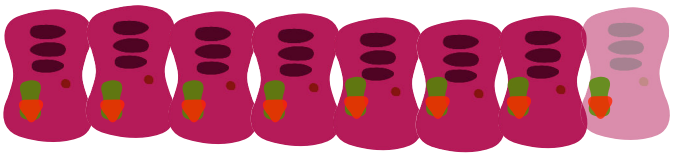


"bankrupt in 45 minutes"

337

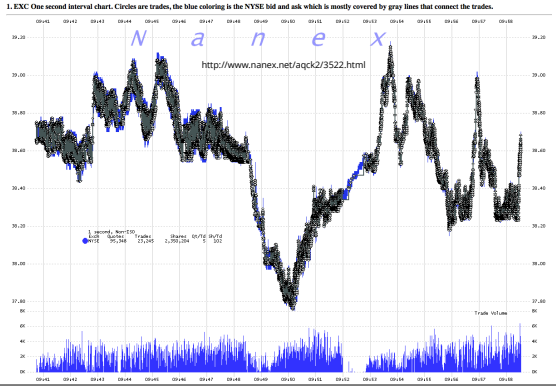
SMARS

PowerPeg

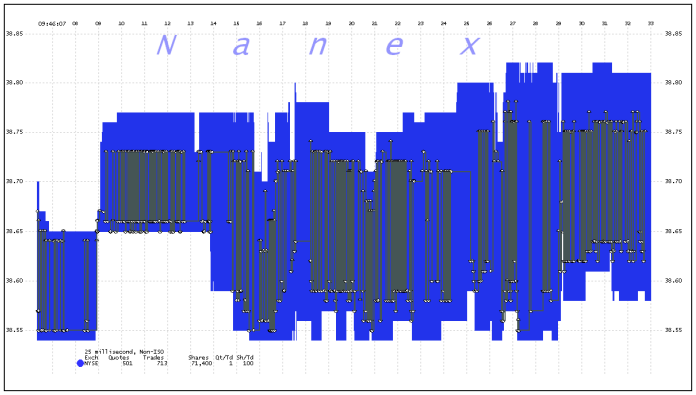


338

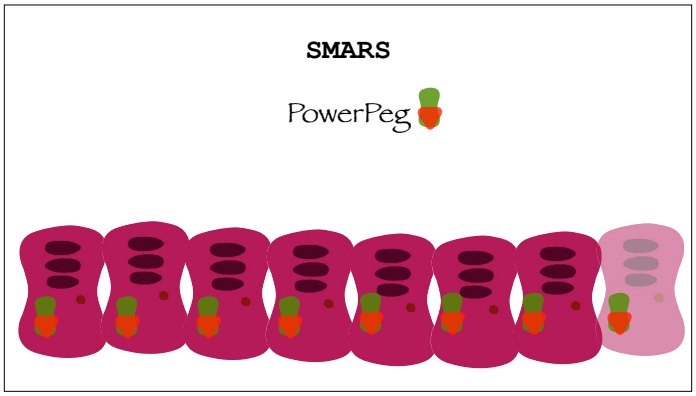
1. EXC One second interval chart. Circles are trades, the blue coloring is the NYSE bid and ask which is mostly covered by gray lines that connect the trades.



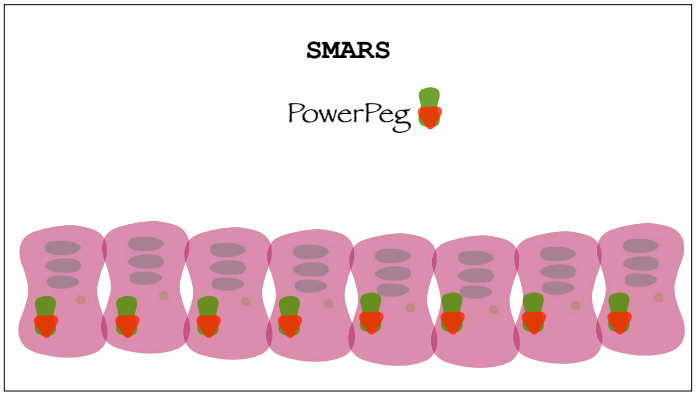
339



340



341



342

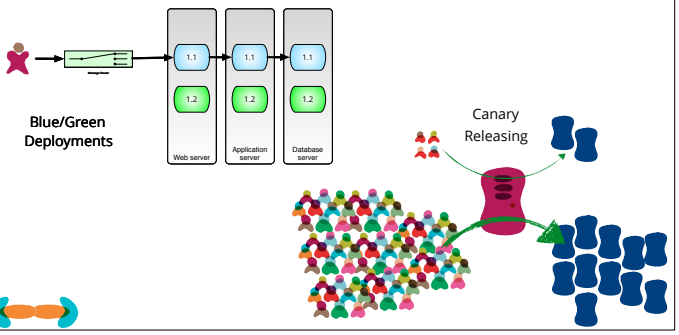
Identify and remove needless variability.

343

Make Decisions Reversible

344

Make Decisions Reversible



345

346

Make as many decisions as possible reversible

(without overengineering).

347

Prefer Evolvable over Predictable

348

...because as we know, there are known knowns; there are things we know we know.

We also know there are known unknowns; that is to say we know there are some things we do not know.

But there are also `unknown unknowns`*—the ones we don't know we don't know.*

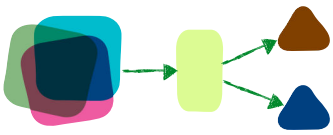
–former US Secretary of Defense Donald Rumsfeld

unknown unknowns

All architectures become iterative because of unknown unknowns; agile just recognizes this and does it sooner.

–Mark Richards

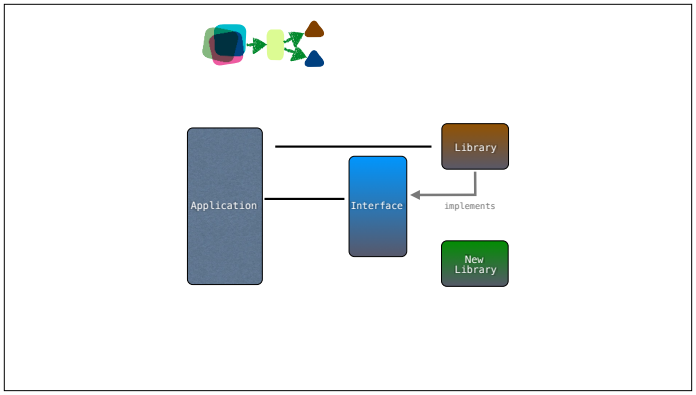
349



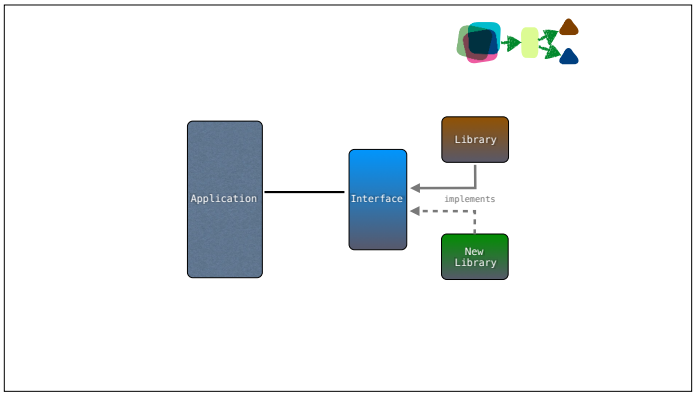
Build Anti-corruption Layers

350

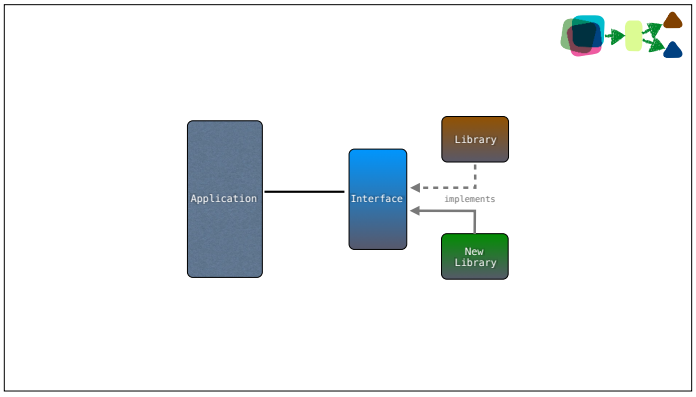
351


Application
Interface
Library
implements
New Library

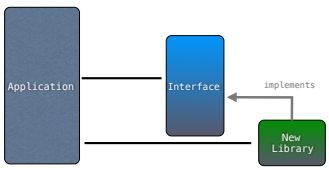
352


Application
Interface
Library
implements
New Library

353

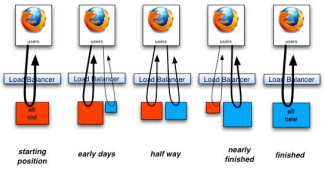

Application
Interface
Library
implements
New Library

354



355

Strangler Pattern



356

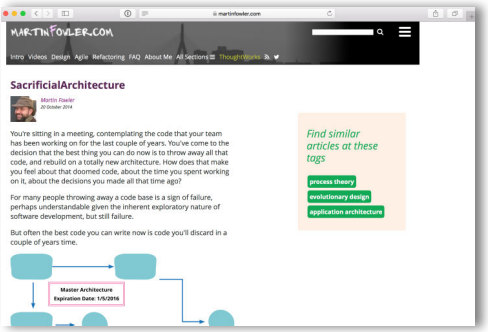
make something new that obsoletes a small percentage of something old

put them live together

rinse, repeat

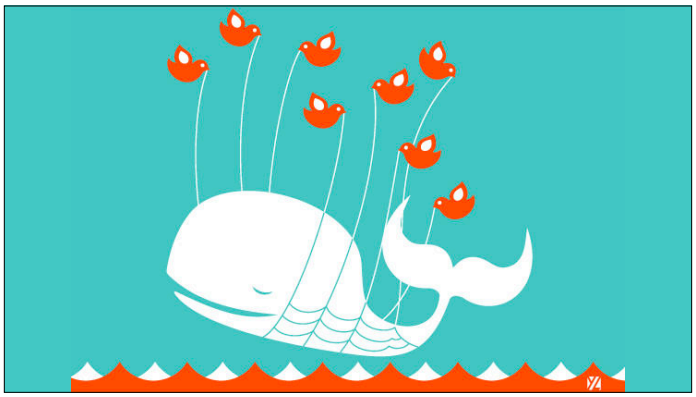
Build Sacrificial Architectures

357



https://martinfowler.com/bliki/SacrificialArchitecture.html

358



359

360

Mitigate External Change

361

The Business Case

362

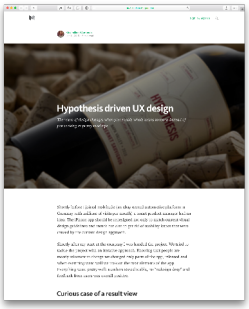
The Trust Engineers

RADIOLAB

http://www.radiolab.org/story/trust-engineers/

363

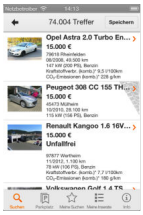
Hypothesis Driven UX



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

364

Hypothesis Driven UX



365

Three Hypotheses

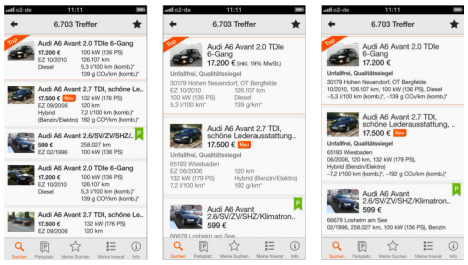
More Listings
If we provide more listings on the screen then we can provide better comparability and offer more diversity on our platform because users like to compare a lot of listings on the result page

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

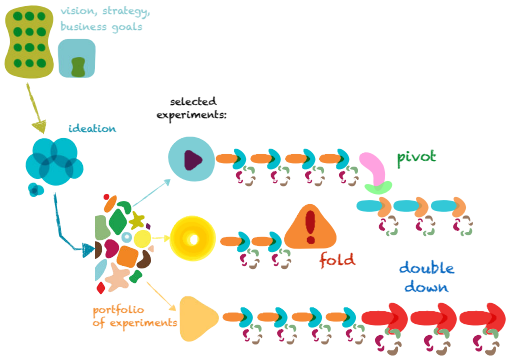
366

Experiments to Perform

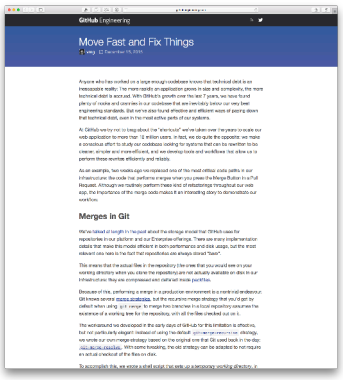


More Listings | Better Structure | Better Prioritization

367

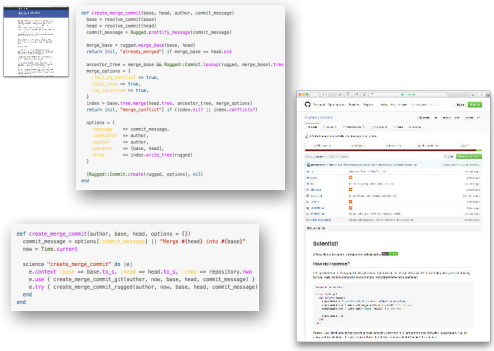


368



Move Fast & Fix Things

369



https://github.com/github/scientist

370

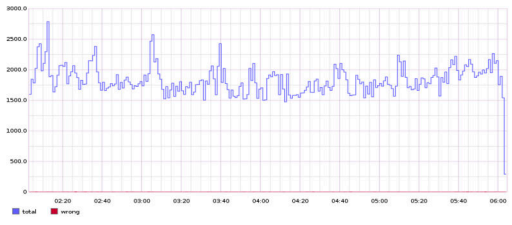
```
require "scientist"

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.

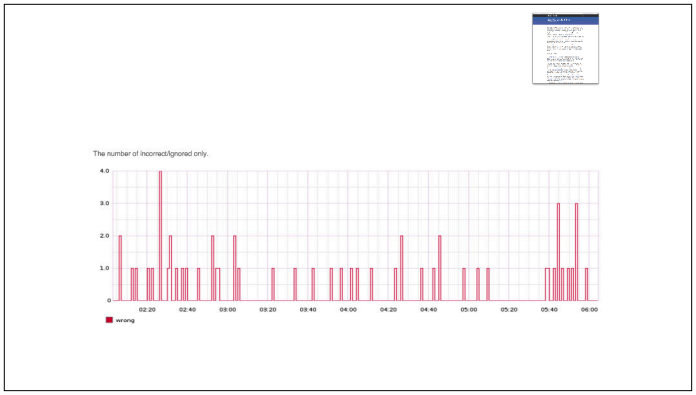
371

Accuracy

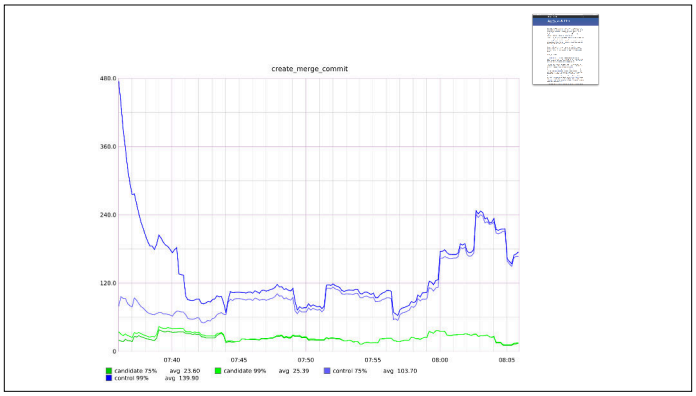
The number of times that the candidate and the control agree or disagree. View mismatches



372



373

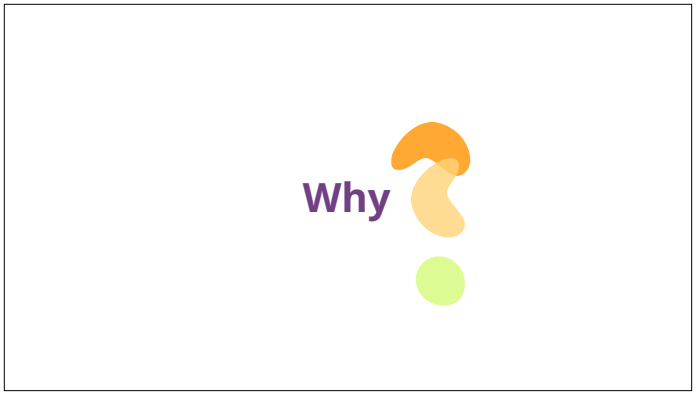


374

Bugs Found; Resolution

- faster conflict return because shell script exited immediately; replicated in library
- index write was causing O(n) problem; inlined into memory
- the ancestor had a file with a given filemode, while one side of the merge had removed the file and the other side had changed the filemode; bug in git!
- Github incorrectly successfully merged files w/ 768 conflicts; fixed git shell script
- new library was skipping an entire step; bug found & fixed

375

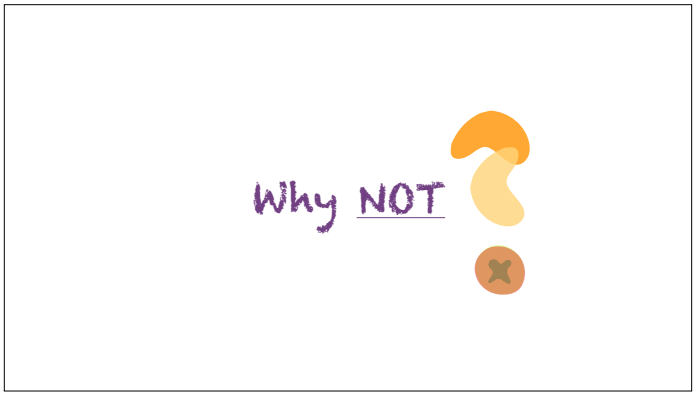


376

Why

- Predicable versus evolvable
- Scale
- Advanced business capabilities
- Cycle time as a business metric
- Isolating architectural characteristics at the quantum level

377



378

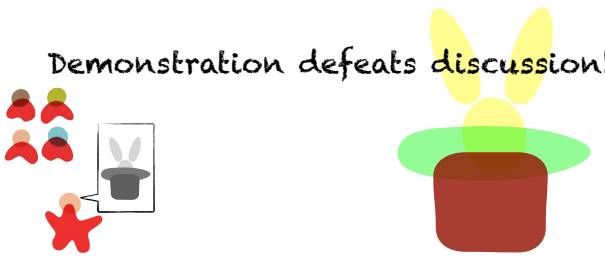
Why NOT

- Can't evolve a ball of mud
- Other architectural characteristics dominate
- Sacrificial architecture
- Planning on closing the business soon

379

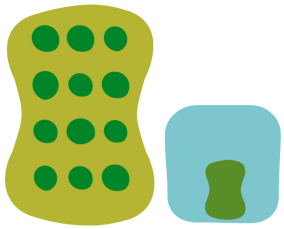
Case Study: Consulting Judo

Demonstration defeats discussion!



380

The Business Case



381

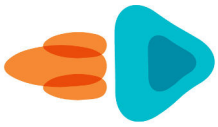
382

The future is already here—it's just not very evenly distributed.

–William Gibson

383

Move Fast without Breaking Things

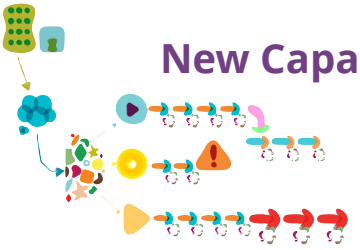


384

Less Risk

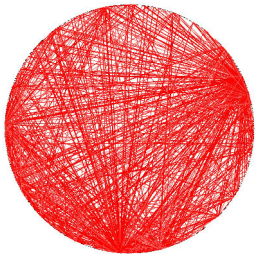
385

New Capabilities



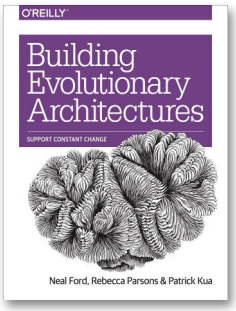
386

Untangling the Ball of Mud

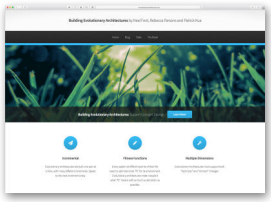


387

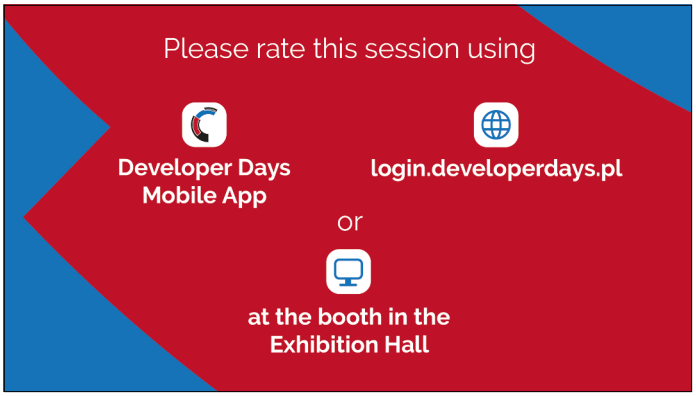
Building Evolutionary Architectures



For more information:



http://evolutionaryarchitecture.com


Please rate this session using
Developer Days Mobile App
login.developerdays.pl
or
at the booth in the Exhibition Hall