Building Evolutionary Architectures Workshop

Neal Ford

@neal4d
nealford.com

http://evolutionaryarchitecture.com
cHAnGe

technology driven

business driven
technology driven

business driven

everything changes all the time!

dynamic equilibrium
How is long term planning possible when things constantly change in unexpected ways?
Once I’ve built an architecture, how can I prevent it from gradually degrading over time?

Second-Order Effect

Second-Order Effect

Second-Order Effect
Second-Order Effect

governance

Evolutionary Architecture
Evolutionary Architecture

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

An evolutionary architecture supports guided incremental change across multiple dimensions.

Guided evolutionary computing fitness function:

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

Traveling Salesman Problem
Traveling Salesman Problem

fitness function = length of route

architectural fitness function:

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

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

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

Evolutionary Architecture

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

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

An evolutionary architecture supports guided incremental change across multiple dimensions.

guided architectural fitness function:

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

Guided architectural fitness function:

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

atomic

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

holistic

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

triggered

run based on a particular event: — developer executing a unit test.
Categories of Fitness Functions

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

executes constant verification of architectural aspect(s)
Categories of Fitness Functions

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

**dynamic**
- rely on a shifting definition based on extra context.

**automated**
- tests and other verification mechanism that run without human interaction.
Categories of Fitness Functions

Automated tests and other verification mechanisms that run without human interaction.

Manual must involve at least one human.

Categories of Fitness Functions

Temporal architects may want to build a time component into assessing fitness

Temporal break on upgrade
Categories of Fitness Functions

architects may want to build a time component into assessing fitness

break on upgrade overdue library update

---

Categories of Fitness Functions

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

---

Categories of Fitness Functions

architectural characteristic domain-specific
<table>
<thead>
<tr>
<th>Architectural Characteristic</th>
<th>Problem Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain-Specific</td>
<td>UAT</td>
</tr>
<tr>
<td>Unit</td>
<td></td>
</tr>
</tbody>
</table>

## Categories of Fitness Functions

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

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

packages/namespaces

Coupling Fitness Function

```java
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.dependOn(util);
    web.dependOn(util);
    JDepend.analyze();
    assertEquals("Dependency mismatch",
            true, JDepend.dependencyMatch(constraint));
}
```

Consumer Driven Contracts

[Source: martinfowler.com/articles/consumerDrivenContracts.html]
Use synthetic transactions to test production systems.
Fitness Function

- atomic
- holistic
- triggered
- continuous

System-wide Fitness Function

- Systemwide fitness function
- Individual fitness functions
Implementing Fitness Functions

Protecting architectural characteristics

Automating governance

maintainable?
Cyclomatic complexity < 50 for all projects

Naming conventions
Cyclomatic complexity < 50 for all projects

Naming conventions

maintainable?

(incoming/outgoing)
Controlled afferent/efferent coupling

Governing Code Quality

130

131

132
if state == "AL" then
doSOMethingForAL();
else if state == "GA" then
doSOMethingForGA();
else if ...
Governing Code Quality

```c
if state == "AL" then
doSomethingForAL();
else if state == "GA" then
doSomethingForGA();
else if ...
```
public class Controller {
    private final Service service;
    private final Service service2;
    public Controller(Service service, Service service2) {
        this.service = service;
        this.service2 = service2;
    }
}

java.lang.AssertionError:
Expected: Comply with rules
but: DENIED com.jdriven.fitness.packaging.by.feature.a -gt;

// Allow package / module a to acces b
a.use(b);
ArchUnit
https://www.archunit.org/
coding rules

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

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

gTest
public void classes_should_not_use_java_util_logging() {
    NO_CLASSES_SHOULD_USE_JAVA_UTIL_LOGGING.check(classes);
}

coding rules

ArchUnit
https://www.archunit.org/
interface rules

public interface {}
public class InterfaceRules {

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

        noClasses().that().areInterfaces().should().haveNameMatching("\"Interface\"");
    }

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

        noClasses().that().areInterfaces().should().haveNameMatching("\"Interface\"");
    }

    // Package by layer (horizontal slicing)

    // layer dependency

    // layer dependency
private JavaClasses classes;

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

@Test
public void should_not_access_controllers() {
    mcClasses().that().resideInPackage("..service..")
        .should().accessClassesThat().resideInAPackage("..controller..")
        .check(classes);
}

@Test
public void should_not_access_services() {
    mcClasses().that().resideInPackage("..service..")
        .should().accessClassesThat().resideInAPackage("..service..")
        .check(classes);
}

ArchUnit

https://www.archunit.org/

governance

Legality of Open Source Libraries
Fitness Function Katas

http://evolutionaryarchitecture.com/ffkatas/
Two Big Ideas

- fitness functions for evolvability

architectural quantum
Domain Driven Design

Bounded Context

Bounded Context + Continuous Delivery = microservices
Architectural Quantum

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

Why Quantum?
Why Quantum?

operational view of architecture
Why Quantum?

holistic

event

processor

module

Message Bus

Message Router

Message Dispatcher

Aggregator

event

processor

module

event

processor

module

event

processor

module

event

processor

module

"multiple dimensions"

useful for architectural analysis
Why Quantum?

The quantum is where architectural characteristics live.
Patterns encapsulate a well-known set of architectural characteristics.
Quantum: Large to Small

Monoliths

"Big Ball of Mud"
Monoliths

"Big Ball of Mud"

unstructured monolith

Monoliths

"Big Ball of Mud"

layered architecture

Monoliths

"Big Ball of Mud"

unstructured monolith

modular monolith


presentation
layered architecture

user interface

business rules

persistence

component

component

component

component

component

component

component

component

layered architecture

presentation

unstructured monolith

modular monolith

Monoliths

“Big Ball of Mud”

unstructured monolith

layered architecture

modular monolith

Monoliths

layered architecture

modular monolith

Monoliths

technical partitioning

layered architecture

modular monolith
Monoliths

technical partitioning

domain partitioning

layered architecture

modular monolith

distributed systems

ESB-driven SOA

ESB-driven SOA

Event-driven Architectures

Broker Base Architecture
Mediator Message Flow

you move...

process engine

notify

claims

adjustment

quote

customer

process process process process process process

notify

change

recalc

update

adjust

you moved

change

recalc

update

adjust

you move...

process engine

notify

claims

adjustment

quote

customer

process process process process process process

notify

change

recalc

update

adjust

you moved

change

recalc

update

adjust

you move...

process engine

notify

claims

adjustment

quote

customer

process process process process process process

notify

change

recalc

update

adjust

you moved

change

recalc

update

adjust
Microkernel

claims processing

Microkernel Quanta

Microkernel: UI
Microkernel: Quanta

Microkernel: Quanta

Microservices
What makes microservices so evolvable?

- extremely loose coupling
- transactions
- eventual consistency
What makes microservices so evolvable?

- extremely loose coupling
- eventual consistency
- bounded context
- small quantum

Microservices
Microservices

Service-based Architectures

1. Larger service granularity
Service-based Architectures

1. Larger service granularity
2. Database scope

Service-based Architectures

2. Use of service bus as integration hub

Reducing Quanta Size

quantum
Reducing Quanta Size

useful for architectural analysis

helps analyze coupling

Smaller Quanta Size

Evolutionary

Microkernel Redux
Last 10% Trap

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

what the user wants

80% 10% 10%

What Happened to the 4GLs?
What Happened to the 4GLs?

see also: Vendor King

Be careful of the *Last 10% Trap* when choosing tools & frameworks.
Exercise #2

identify the quanta

Building Evolutionary Architectures

EVOLUTIONARY DATA

[Image of the cover of the book]
Data & code are both abstractions based on the real world. Data & code are symbiotic.

**DB Evolution & Deployment**

- scripting all db changes incrementally
- db refactoring
- decouple db migration from app migration

**DbDeploy Pattern**
Continuous Integration for Databases

- Prepare environment
- Deploy app
- Create dbs, apply schema
- Add app reference data
- Run acceptance tests
For DB CI We Need To:

- start with a clean database
- apply changes incrementally
- use the same process everywhere
- be comprehensive in change management

#1: Baseline

- Create database
- Add metadata table
- Restore scheme & reference data to current production state

#2: Apply Deltas

- Run each delta in order
- Stop the line if one fails
- Record success in metadata table
#3: Run Tests

Baseline Database → Apply Deltas → Test!

Acceptance tests verify database scripts worked.

Shared-database Integration

Application A → Application B → Application C → Shared Data

Refactoring Databases
Move Column Refactoring

Original Schema

<table>
<thead>
<tr>
<th>Customer</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>FName</td>
<td>AccountID</td>
</tr>
<tr>
<td></td>
<td>Balance</td>
</tr>
<tr>
<td></td>
<td>transactionsHistory</td>
</tr>
</tbody>
</table>

Transition Period

<table>
<thead>
<tr>
<th>Customer</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>FName</td>
<td>AccountID</td>
</tr>
<tr>
<td></td>
<td>Balance</td>
</tr>
</tbody>
</table>

Move Column Refactoring

Ending Schema

<table>
<thead>
<tr>
<th>Customer</th>
<th>Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>FName</td>
<td>AccountID</td>
</tr>
<tr>
<td></td>
<td>Balance</td>
</tr>
</tbody>
</table>

Move Column Refactoring
Decouple DB Updates: the Expand/contract Pattern

Expand/Contract Pattern

Deploy new changes, migrate data, put in scaffolding code

Start

Implement the refactoring

Expand

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

Customer
Customer ID
Name
Starting State

Start
name = "Pramod Sudalage"
#1: integration points, legacy data

```
ALTER TABLE customer ADD firstname VARCHAR2(50);
ALTER TABLE customer ADD lastname VARCHAR2(50);
ALTER TABLE customer DROP COLUMN name;
```
#2: integration points, legacy data

```
ALTER TABLE Customer ADD firstname VARCHAR2(40);
ALTER TABLE Customer ADD lastname VARCHAR2(40);
UPDATE Customer set firstname = extractfirstname (name);
UPDATE Customer set lastname = extractlastname (name);
ALTER TABLE customer DROP COLUMN name;
```

#3: integration points, legacy data

```
ALTER TABLE Customer ADD firstname VARCHAR2(40);
ALTER TABLE Customer ADD lastname VARCHAR2(40);
UPDATE Customer set firstname = extractfirstname (name);
UPDATE Customer set lastname = extractlastname (name);
CREATE OR REPLACE TRIGGER synchrName
BEFORE INSERT OR UPDATE
ON Customer;
BEGIN
  IF NEW.Name IS NULL THEN
    NEW.firstname := NEW.firstname;
    NEW.lastname := NEW.lastname;
  END IF;
  IF NEW.Name IS NOT NULL THEN
    NEW.firstname := extractfirstname(NEW.Name);
    NEW.lastname := extractlastname(NEW.Name);
  END IF;
END;
```

Decentralized Data Management
Decentralized Data Management

Transactions are temporal coupling.

Decentralized Data Management

Limit transactional contexts.

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

Penultima

<home> <catalog> <item>

Evolving Routing

Routes

routes

RouteContext

routes origin workflow

Evolving Routing

Penultima
1. Identify dimensions

prioritization
1. Identify dimensions

prioritization

cost to implement & maintain

Mechanics

1. Identify dimensions

2. Define fitness function(s)
2. Define Fitness Function(s)

3. Use deployment pipelines and/or continuous fitness functions

Mechanics

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

Remove Needless Variables
Remove Needless Variables

Knight Capital
"bankrupt in 45 minutes"

SMARS

PowerPeg
Identify and remove needless variability.

Make Decisions Reversible

Blue/Green Deployments

Canary Releasing
Make as many decisions as possible reversible (without overengineering).

Prefer Evolvable over Predictable

...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

Build Anti-corruption Layers
Application

New

Library

Interface

implements

Library

Non Library

Interface

Application

Library

implements

Non Library

Application

Interface

Library

implements

Non Library

Application

Interface

Library

implements

Non Library
Strangler Pattern

make something new that obsoletes a small percentage of something old
put them live together
rinse, repeat
Mitigate External Change

The Business Case

The Trust Engineers

http://www.radiolab.org/story/trust-engineers/
Hypothesis Driven UX

Three Hypotheses

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

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

**Better Prioritization**
If we prioritize information according to user needs then we achieve better guidance because the user can see all relevant information at a glance.
Experiments to Perform

- More Listings
- Better Structure
- Better Prioritization

- Vision, strategy, business goals
- Ideation
- Portfolio of experiments
- Pivot
- Fold
- Double down

Move Fast & Fix Things
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.
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
Why

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

Why NOT?
Why NOT

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

Case Study: Consulting Judo

Demonstration defeats discussion!

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

-William Gibson

Move Fast without Breaking Things

Less Risk
New Capabilities

Untangling the Ball of Mud

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