Building Evolutionary Architectures

O'Reilly

Building Evolutionary Architectures

Support constant change

Neal Ford, Rebecca Parsons & Patrick Kua

ThoughtWorks:

NEAL FORD
Director / Software Architect / Team Wrangler

with Rebecca Parsons & Pat Kua

@neal4d
nealford.com
Rebecca Parsons

Pat Kua

Neal Ford

Photos by Martin Fowler:
http://martinfowler.com/albums/ThoughtWorkers/
What is Software Architecture?
Entire architectural Scope
Entire architectural Scope

- Auditability
- Data
- Security
- Requirements
- Performance
- Legality
- Scalability

Time
Once I’ve built an architecture, how can I prevent it from gradually degrading over time?
How is long term planning possible when things change unexpectedly?
Dynamic Equilibrium
“Architecture is the decisions that you wish you could get right early in a project.”

— Ralph Johnson

things that people perceive as hard to change.
What if we build architectures that expect change?
Definition:

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
Perspectives on Architecture

Technical Architecture
Evolvability of Architectures
Big Ball of Mud

coupling connections
dimensions:

classes
Layered Architecture

presentation layer
business layer
persistence layer
database layer

opportunities: 4
dimensions: 1
Microkernel

dimensions: 1
REST

dimensions : 1
Domain Perspective

presentation layer

business layer

persistence layer

database layer

dimensions: ♦
Microservices

Evolutionary architecture dimensions:
Definition: evolutionary architecture

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
Composability
Composability
Composability
Definition:

**evolutionary architecture**

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
Incremental Change

Components are **deployed**.

Features are released.

Applications consist of routing.
Incremental Change
Definition: evolutionary architecture

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
Perspectives on Architecture

Data Architecture
Perspectives on Architecture

Security Architecture
Perspectives on Architecture

Domain Architecture
Microservices

Domain Architecture
Fitness Functions

\( \omega \)

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

metrics

ω

tests
Definition:

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
Agenda

- Definition
- Incremental change
- Fitness functions
- Appropriate coupling
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.
Architecture Fitness Functions

metrics

tests
Architecture Fitness Functions

metrics

fitness functions

tests

...
Fitness Function

- atomic
- holistic
- batch
- continuous
Fitness Function

atomic

holistic

batch

continuous
Cyclic Dependency Function

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

Coupling Fitness Function

```java
protected void setUp() throws IOException {
    jdepend = new JDepend();
    jdepend.addDirectory("/path/to/project/util/classes");
    jdepend.addDirectory("/path/to/project/ejb/classes");
    jdepend.addDirectory("/path/to/project/web/classes");
}

public void testMatch() {
    DependencyConstraint constraint = new DependencyConstraint();
    JavaPackage ejb = constraint.addPackage("com.xyz.ejb");
    JavaPackage web = constraint.addPackage("com.xyz.web");
    JavaPackage util = constraint.addPackage("com.xyz.util");

    ejb.dependsUpon(util);
    web.dependsUpon(util);

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

batch  atomic

holistic  continuous
Fitness Function

atomic
holistic
batch
continuous
Fitness Function

- atomic
- batch
- holistic
- continuous
Holistic fitness functions must run in a specific (shared) context.
Consumer Driven Contracts

martinfowler.com/articles/consumerDrivenContracts.html
Fitness Function

- Atomic
- Holistic
- Batch
- Continuous
atomic monitoring

continuous logging
Fitness Function

holistic

atomic
continuous

batch
Fitness Function

atomic
holistic
batch
continuous
Fitness Function

- atomic
- batch
- continuous
- holistic
holistic resilience-ility?
Fitness Function

- Atomic
- Holistic
- Batch
- Continuous
System-wide Fitness Function
Fitness Function Fit
Guided Evolution
Agenda

- definition
- incremental change
- fitness functions
- appropriate coupling
Prerequisites
Deployment Pipeline
Deployment Pipeline
Incremental Change

\[ V \propto C \]

where
\[ c = \text{cycle time} \]
\[ v = \text{maximum speed of new generations} \]

Engine of evolutionary architecture
Agenda

definition
incremental change
fitness functions
appropriate coupling
Code Reuse (Over Time)
Code Reuse (Over Time)
The more *reusable* code is, the less *usable* it is.
Decentralized Data Management
Decentralized Data Management

Transactions are temporal coupling.
Decentralized Data Management

Limit transactional contexts.
Evolutionary Database Design

http://databaserefactoring.com/
Evolving Columns

Customer
- FirstName
- CustomerID <<PK>>
- Balance
- CheckNoAccounts {
  event = before delete
}

Account
- AccountID <<PK>>
- CustomerID <<FK>>
- CheckCustomerExists {
  event = before update | before insert
}

Original Schema
Transition

Customer

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FirstName</td>
<td></td>
</tr>
<tr>
<td>CustomerID &lt;&lt;PK&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Balance</td>
<td>{removal date = June 14 2007}</td>
</tr>
</tbody>
</table>

Account

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccountID &lt;&lt;PK&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>CustomerID &lt;&lt;FK&gt;&gt;</td>
<td></td>
</tr>
<tr>
<td>Balance</td>
<td></td>
</tr>
</tbody>
</table>

Transition Period

SynchronizeAccountBalance
{ event = on update | on delete | on insert, drop date = June 14 2007 }

CheckNoAccounts
{ event = before delete }

SynchronizeCustomerBalance
{ event = on update | on insert, drop date = June 14 2007 }

CheckCustomerExists
{ event = before update | before insert }
Expand/Contract Pattern

DB version 13
- app v205 deployed

DB version 14
- migrate db to v14
- app v230 deployed

DB version 14
- app v234 deployed

DB version 14 and 15
- app v234 deployed
- migrate db to v15
- app v248 deployed

DB version 15
- app v241 deployed
- app v248 deployed
Decentralized Governance
Decentralized Governance
Decentralized Governance
“Goldilocks” Governance

Choose technology stacks appropriate to problem scale.
Shift to Domain-centric Architectures
Shift to Domain-centric Architectures
Shift to Domain-centric Architectures
Incidently Coupled Teams

user interface

server-side

DBA
Conway’s Law

“organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations”

Melvin Conway, 1968

en.wikipedia.org/wiki/Conway%27s_law
Incidentally Coupled Teams

- user interface
- server-side
- DBA
Autonomous Teams

Orders

Shipping

Inverse Conway Maneuver

Catalog
Low Efferent Coupling between Teams

\[ \frac{n(n-1)}{2} \]
Architectural Quantum
Architectural Quantum
Architectural Quantum
Architectural Quantum

![Diagram showing cost and benefit against the number of quanta. The cost line slopes downward indicating increasing cost with increasing quanta, while the benefit line slopes upward indicating increasing benefit with increasing quanta.]
Utilizing Evolutionary Architecture
1. Choose Dimensions
Entire architectural Scope

- requirements
- auditability
- data
- security
- performance
- legality
- scalability
- time
- data
- auditability
- security
- scalability
- performance
- legality
- requirements
- time
Entire architectural Scope

- auditability
- data
- security
- requirements
- performance
- legality
- scalability

fitness functions

time
Entire architectural Scope

- Auditing
- Data
- Security
- Performance
- Legality
- Scalability
- Fitness functions
- Time
- Requirements
Utilizing Evolutionary Architecture
2. Identify Fitness Functions

atomic / automated / manual

holistic
Utilizing Evolutionary Architecture
3. Apply Incremental Change
Utilizing Evolutionary Architecture
Utilizing Evolutionary Architecture
Agenda

- Definition
- Incremental change
- Fitness functions
- Appropriate coupling
Why should a company decide to build an evolutionary architecture?
Predictable versus Evolvable

Scale

Cycle Time as a Business Metric

Isolating “-ilities” at the Quantum Level

Longer Lasting Useful Systems

Advanced Business Capabilities

Why should a company decide to build an evolutionary architecture?
Why should a company decide to build an evolutionary architecture?
Why would a company choose *not* to build an evolutionary architecture?
Can’t Evolve a Ball of Mud

Other Architectural Characteristics Dominate

Sacrificial Architecture

Planning on Closing the Business Soon
Predictable versus Evolvable

Scale

Cycle Time as a Business Metric

Isolating “-ilities” at the Quantum Level

Longer Lasting Useful Systems

Advanced Business Capabilities

Why should a company decide to build an evolutionary architecture?
Hypothesis and Data Driven Development
Move Fast and Fix Things

Move Fast & Fix Things

Anyone who’s worked on a large enough codebase knows that bad code can be an insidious problem. The more deeply an application grows in size and complexity, the more difficult it becomes to maintain. This is especially true when changes are made to the codebase over a long period of time, and it becomes difficult to keep track of who made what changes, and why.

At GitHub, we try to keep our codebase clean and efficient. We do this by regularly reviewing our own code and fixing any bugs or issues we find. This helps us maintain a clean codebase that is easy to understand, work on, and maintain.

As an example, a few weeks ago we released one of the most critical code changes in our application. This code change was complex and required significant effort to implement. However, we were able to complete the change in a timely manner because we had a clean codebase to work with.

We also use tools like Git to make sure our codebase is well-organized and easy to navigate. Git allows us to keep track of changes over time and helps us identify any potential issues before they become problems.

Merges in Git

We help all engineers to use the merging model that Git is known for. The merging model allows us to incorporate code changes from other branches into our main branch. This way, changes can be easily reviewed and merged into the main codebase.

We use a tool called Git Merge Manager to help us with merging code changes. This tool allows us to easily review and merge changes from other branches into our main codebase.

Because of this, we are able to incorporate changes quickly and efficiently. The merging model allows us to quickly incorporate changes from other branches into our main codebase, which helps us stay on top of changes and maintain a clean codebase.

To ensure that we are using our codebase effectively, we regularly review our code and make changes as needed. This helps us maintain a clean and organized codebase that is easy to work with.

In summary, moving fast and fixing things is an important part of maintaining a codebase. By regularly reviewing our code and using tools like Git, we are able to keep a clean and organized codebase that is easy to work with.

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.
Accuracy

The number of times that the candidate and the control agree or disagree. View mismatches
The number of incorrect/growts any.
**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, whilst one side of the merge had removed the file and the other side had changed the filemode; bug in git!
- Git incorrectly successfully merged files w/ 768 conflicts; fixed git shell script
- new library was skipping an entire step; bug found & fixed
Definition:

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
Agenda

- definition
- incremental change
- fitness functions
- appropriate coupling