Evolutionary Architectures

with Rebecca Parsons & Pat Kua
Rebecca Parsons

Pat Kua

Neal Ford

Photos by Martin Fowler:
http://martinfowler.com/albums/ThoughtWorkers/
“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.
accessibility
correctness
credibility
customizability
debuggability
degradability
determinability
demonstratability
dependability
deployability
discoverability
distributability
durability
effectiveness
efficiency
exemplarity
extensibility
failure transparency
fault-tolerance
fidelity
flexibility
inspectability
installability
integrity
interchangeability
interoperability
learnability
maintainability
manageability
modifiability
modularity
modularity
reusability
on-modifiability
portability
precision
proportability
provability
recoverability
reliability
repeatability
reproducibility
resilience
responsiveness
reusability
robustness
safety
scalability
seamlessness
self-sustainability
serviceability
supportability
securability
simplicity
stability
standards compliance
survivability
sustainability
tailorability
testability
timeliness
traceability
transparency
ubiquity
understandability
upgradeability
usability

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

Technical Architecture
Perspectives on Architecture

Data Architecture
Perspectives on Architecture

Security Architecture
Perspectives on Architecture

Domain Architecture
Microservices

Domain Architecture
Evolvability of Architectures
Big Ball of Mud

coupling connections

dimensions: 16

classes
Layered Architecture

- Presentation layer
- Business layer
- Persistence layer
- Database layer

opportunities: 4
dimensions: 1
Opportunities vs Dimensions
Opportunities vs Dimensions

opportunities for evolution = L - (2 x L°)

L : # of layers
L° : # of open layers

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:

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.
Fitness Functions

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

ω

metrics

tests
Definition:

Evolutionary architecture

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
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());
}
```

clarkware.com/software/JDepend.html
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

batch

holistic

continuous
Fitness Function

atomic

holistic

batch

continuous
Fitness Function

- holistic
- batch
- atomic
- continuous
atomic

monitoring

continuous

logging
Use synthetic transactions to test production systems.
Use correlation IDs to track down nasty bugs
Fitness Function

- atomic
- continuous
- holistic
- 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
Architecture is abstract until operationalized.

nealford.com/memeagora/2015/03/30/architecture_is_abstract_until_operationalized.html
Prerequisites
Deployment Pipeline
Deployment Pipeline
Incremental Change

\[ V \propto C \]

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

increases technical debt

mitigates technical debt
Decentralized Data Management
Decentralized Data Management

Transactions are temporal coupling.
Decentralized Data Management

Limit transactional contexts.
Evolutionary Database Design

http://databaserefactoring.com/
Evolving Columns

Original Schema

Customer

- FirstName
- CustomerID <<PK>>
- Balance

CheckNoAccounts
  { event = before delete }

1 accesses 1..*

Account

- AccountID <<PK>>
- CustomerID <<FK>>

CheckCustomerExists
  { event = before update | before insert }
### Transition

**Customer**
- **FirstName**
- **CustomerID <PK>**
- **Balance** {removal date = June 14 2007}

**Account**
- **AccountId <PK>**
- **CustomerID <FK>**
- **Balance**

**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

- **app v205** compatible with db v13 and v14
- **app v230** compatible with db v14
- **app v234** compatible with db v14
- **app v241** compatible with db v14 and 15
- **app v248** compatible with db v15

- **DB version 13**
- **migrate db to v14**
- **DB version 14**
- **app v234 deployed**
- **app v234 deployed**
- **app v234 deployed**
- **migrate db to v15**
- **app v234 deployed**

**Time**
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

![Graph showing cost and benefit against the number of quanta.](image-url)
Utilizing Evolutionary Architecture
1. Choose Dimensions

“-ilities”

evolutionary change

testable
Utilizing Evolutionary Architecture
2. Identify Fitness Functions

atomic

automated / manual

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

incremental change

fitness functions

appropriate coupling
Why should a company decide to build an evolutionary architecture?
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 would a company choose *not* 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

Anyone who has worked on a large enough codebase knows that debt can build in an insidious way. The more repository isolation is a feature of the project, the more technical debt is introduced. At GitLab, over the last 6 years, we’ve built plenty of features and tools that have inevitably added new, often times, hidden layers of debt. We’ve also been very effective in reducing and removing the effect of this debt in our users’ lives.

At GitLab, we like to bring about the “first results” that we take over the next development cycle in our workflow. With the release of GitLab 11.6, we’ve introduced several new features that are being rolled out to our users. These new features include improved security, increased efficiency, and advanced user experience enhancements that are set to perform in a more effective and reliable manner.

As an example, we recently released one of the most critical code changes in our workflow and it shows that performance improvements can be achieved without compromising reliability. Although our team is under pressure to deliver features through our web app, the importance of the merge code makes it an interesting story to demonstrate our workflow.

Merges in Git

We’ve learned a lot about the merge model that Git uses for development in our codebase. The main reasons are factors that impact the performance and efficiency of the code we use. The benefit of the merge model is that it allows us to maintain a clean and consistent codebase.

In this sense, the merge model is more flexible and allows us to develop code in a modular way. The primary advantage of the merge model is that it allows us to develop code in a modular way. The primary advantage of the merge model is that it allows us to develop code in a modular way.

Because of this, performing merges in a pull request is a relatively straightforward. Git has two main merge strategies: one is called a "fast-forward" merge, and the other is called "merge". The fast-forward merge strategy simply takes the changes from the other branch and applies them to the local branch. The "merge" strategy is used when the changes are not preserved.

The work we’ve done with GitLab has been a significant improvement for our development process. With the new features we’ve released, our team has seen a significant increase in productivity. We believe that our development process is now more efficient and reliable. To summarize, we’ve seen a significant increase in productivity and reliability with our new development process. We believe that our development process is now more efficient and reliable.
```python
def create_merge_commit(base, head, author, commit_message)
    base = resolve_commit(base)
    head = resolve_commit(head)
    commit_message = Rugged.pretty_message(commit_message)
    merge_base = Rugged.merge_base(base, head)
    return (null, "already merged") if merge_base == head

    ancestor_tree = merge_base AA Rugged.Commit.tree(merge_baseorElse)
    merge_options = {
        fail_on_conflict: true,
        update_index: true,
        update_trees: true,
    }
    index = base.tree.merge(head.tree, ancestor_tree, merge_options)
    return (null, "merge conflict") if (index.conflicts == null || index.conflicts.length == 0)

    options = {
        message: commit_message,
        author: author,
        committer: author,
        #base: base,
        index: index.write_trees(ragged)
    }
    (null, Commit.create(ragged, options)), kill
```

https://github.com/github/scientist
- 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 guesses 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:

**evolutionary architecture**

An evolutionary architecture supports incremental, guided change as a first principle across multiple dimensions.
http://evolutionaryarchitecture.com
nealford.com/videos

www.oreilly.com/software-architecture-video-training-series.html

nealford.com/books
Agenda

definition
incremental change
fitness functions
appropriate coupling