Principles and Techniques of Evolutionary Architecture

Rebecca Parsons
Chief Technology Officer
ThoughtWorks

Agenda

Why should I care?

Definition of Evolutionary Architecture

Principles of evolutionary architecture

Techniques of evolutionary architecture

How to achieve an evolutionary architecture in practice

Why Should I Care?

Expectations for pace of change are increasing rapidly

Business model lifetimes are shortening

It’s harder to predict the future

If you miss your minute of fame...

Evolutionary Architecture supports guided, incremental change across multiple dimensions.

Evolutionary Architecture supports guided, incremental change across multiple dimensions.

Evolutionary Architecture supports **guided**, incremental change across multiple dimensions.

Evolutionary Architecture
supports guided,
incremental change
across multiple dimensions.

Evolutionary Architecture supports guided, incremental change across **multiple dimensions**.

Principles of Evolutionary Architecture

Last responsible moment

Architect and develop for evolvability

Postel’s Law

Architect for testability

Conway’s Law

Last Responsible Moment

Delay decisions as long as you can, but no longer

Maximizes the information you have

Minimizes technical debt from complexity

Decide early what your drivers are, and prioritize decisions accordingly

Evolutionary, neither emergent nor based on guesswork

Architect for Evolvability

Sensible breakdown of functionality

Consider data lifecycle and ownership

Appropriate coupling

Lightweight tooling and documentation

Develop for Evolvability

Software internal quality metrics focusing on ease of change

Find hotspots and focus efforts there

Measure continually, focusing on trends

Reversability

Postel's Law

Be conservative in what you send

Be liberal in what you receive

Only validate what you need

Holds for any information exchange

Use version changes when a contract must be broken

Architect for Testability

Aiming towards testability produces a well-architected system

Messaging infrastructure used for messaging, not business logic

Business sensible components

Testing at many levels, including contract

Build pipelines support the volume

Conway's Law

Organizations design systems reflecting their communication structures

Broken communications imply complex integration

Silos often result in broken communication

If you don't want your product to look like your organization, change your organization (or your product)

Techniques

Database Refactoring

Continuous Delivery

Choreography

Contract Testing

Database Refactoring

De-compose big change into series of small changes
Each change is a refactoring/migration pair (or triple if you include access code)
Changes compose in the same way functions compose
And of course, version control the changes
And apply in the various environments during promotion

Continuous Delivery

Automate environments and configurations

Automate builds and deployments and use continuous integration

Automate testing at all levels

Deployment should be boring!!

Just because you CAN release at any time doesn't mean you HAVE to

Choreography

As opposed to orchestration

Scripted outcomes and vision

Individuals perform to the vision without a conductor

Distributes authority about interactions

Introduces new kinds of failure scenarios

Contract Testing

Acceptance tests at the systems' interface

Documents assumptions made

Maximizes parallel independent work

Used in conjunction with Postel's Law

One traditional role of Enterprise Architect

Evolutionary Architecture

Define your architectural fitness function

Delay your decisions as long as you can

Understand various forms of technical debt

Implement evidence based re-use

Create and maintain the testing safety net

Thank you!

http://rebeccaparsons.com
http://www.thoughtworks.com
@rebeccaparsons