演进式架构：数字化世界"进化论"

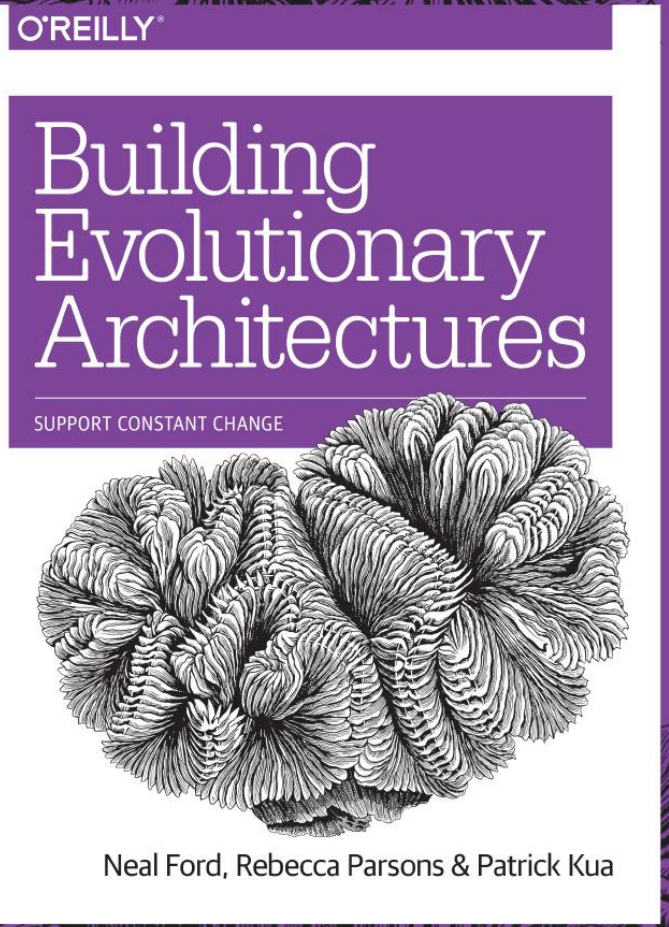
肖然

ThoughtWorks®

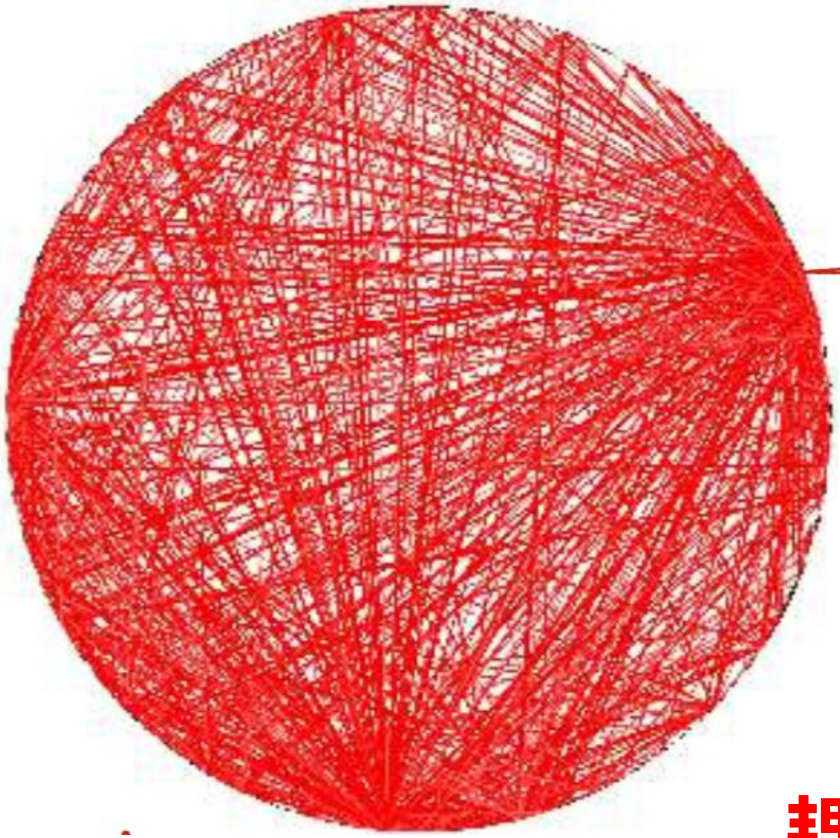
SUPPORT CONSTANT CHANGE

with the new book by

NEAL FORD,
DR. REBECCA PARSONS
& **PATRICK KUA**

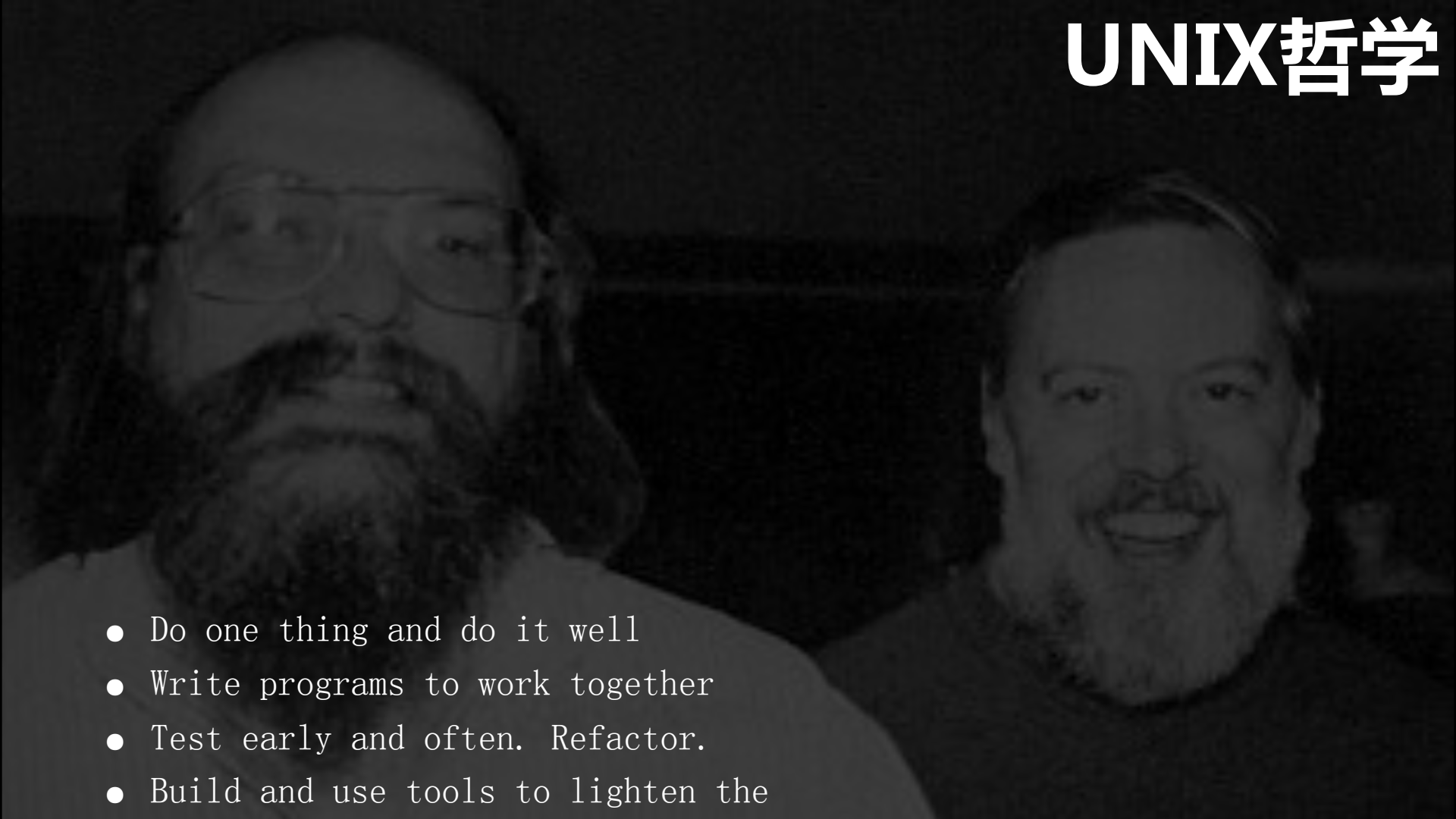



https://www.flickr.com/photos/swisscan/2308034084/
架构到底什么？

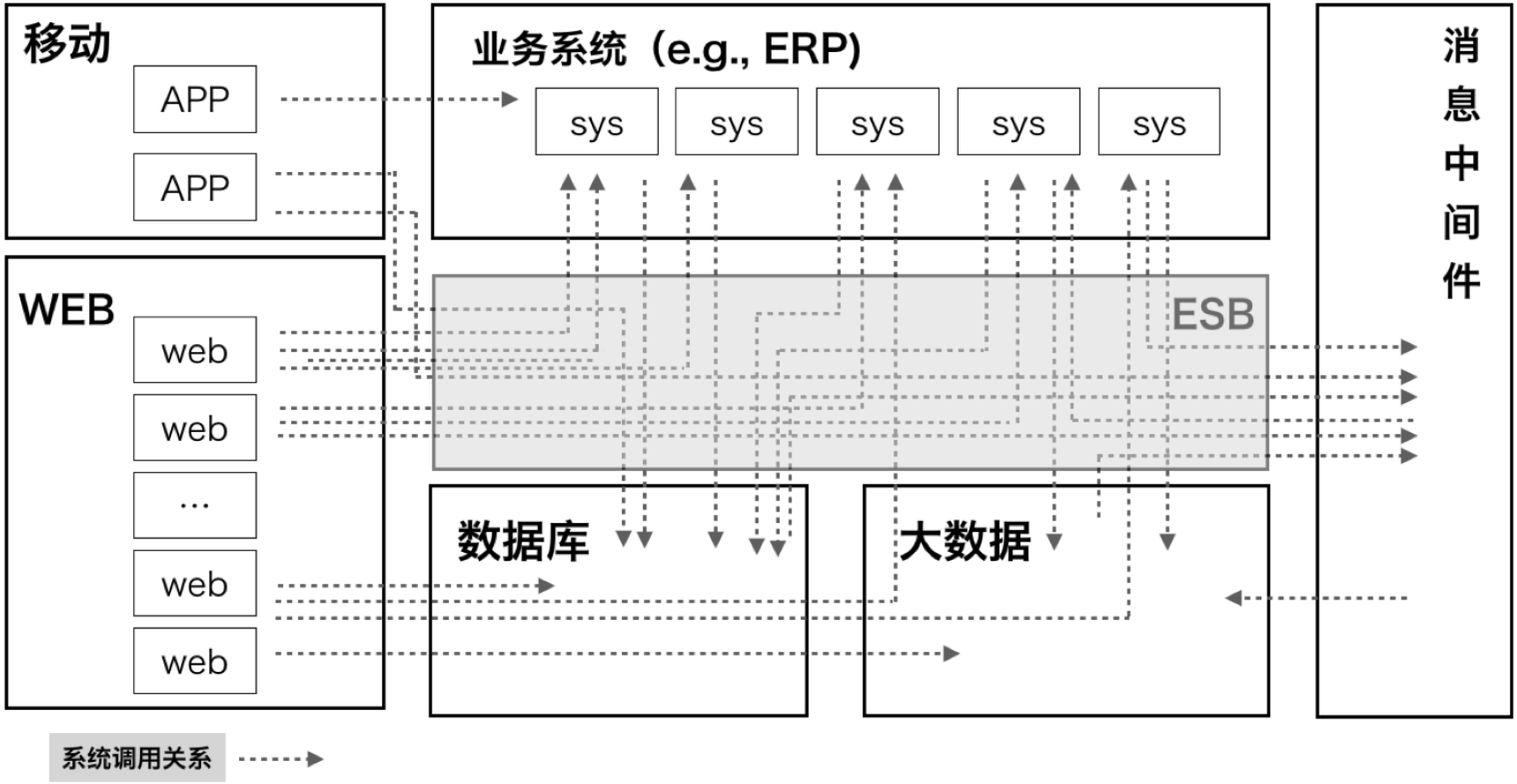


耦合关系

UNIX哲学

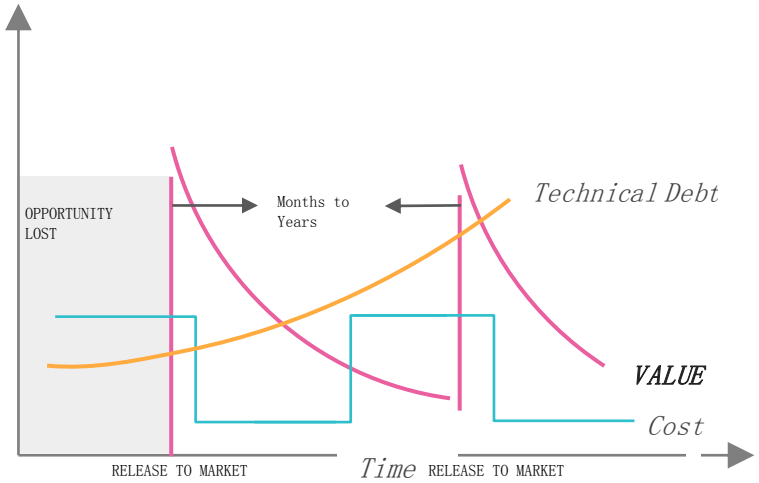


- Do one thing and do it well
- Write programs to work together
- Test early and often. Refactor.
- Build and use tools to lighten the

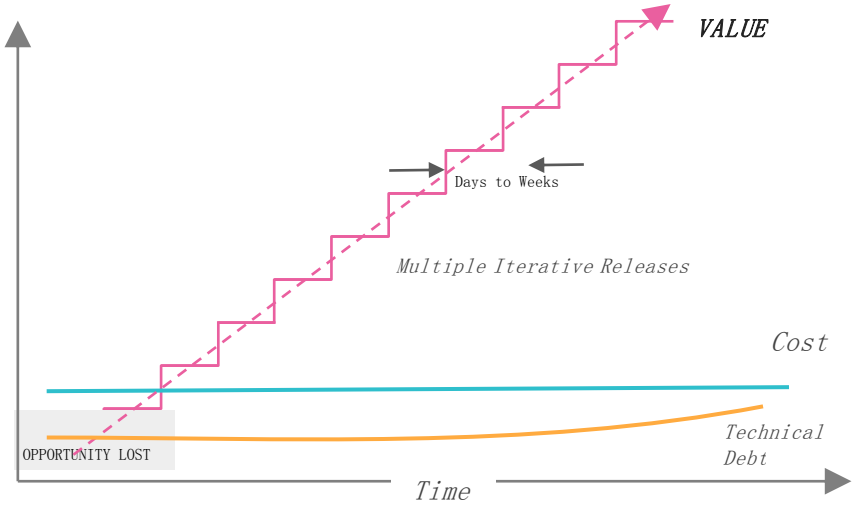

移动
APP
APP
业务系统（e.g., ERP)
sys
sys
sys
sys
sys
消息中间件
WEB
web
web
...
web
web
ESB
数据库
大数据
系统调用关系

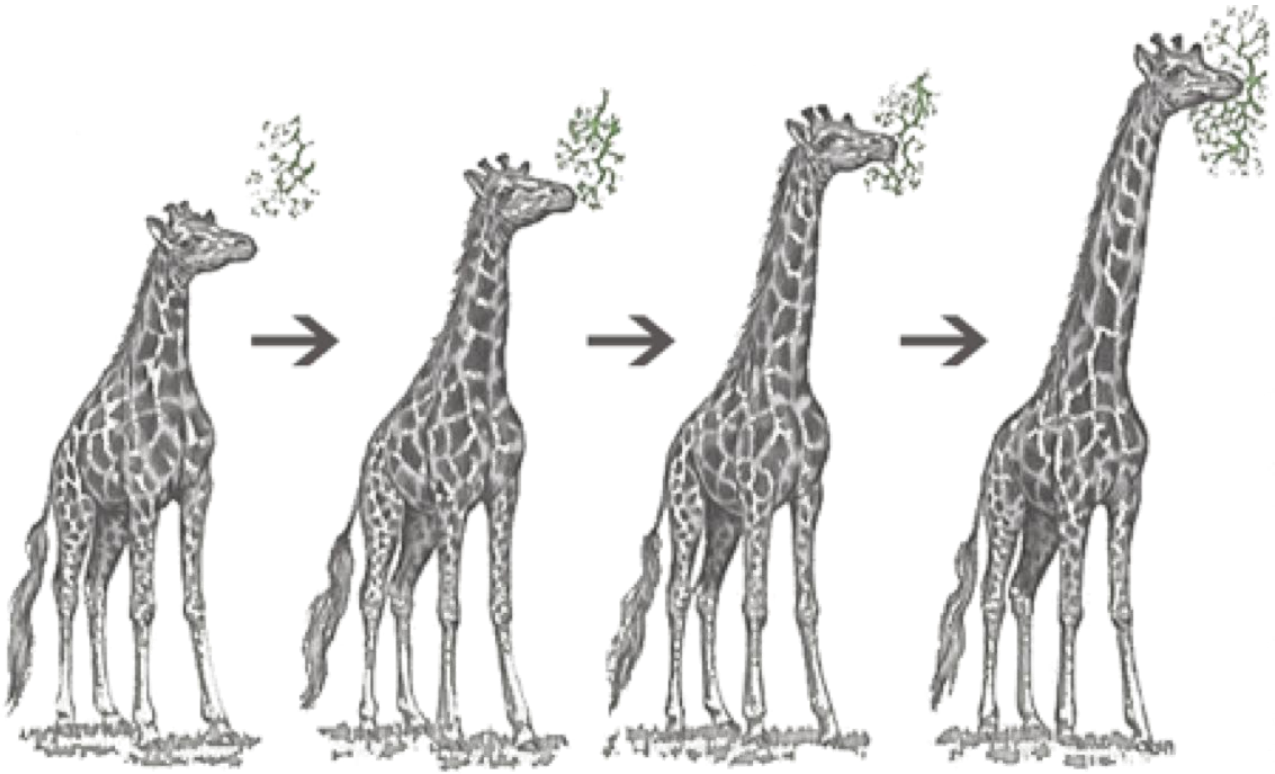
架构管理

我们现实的演进



我们期望的演进





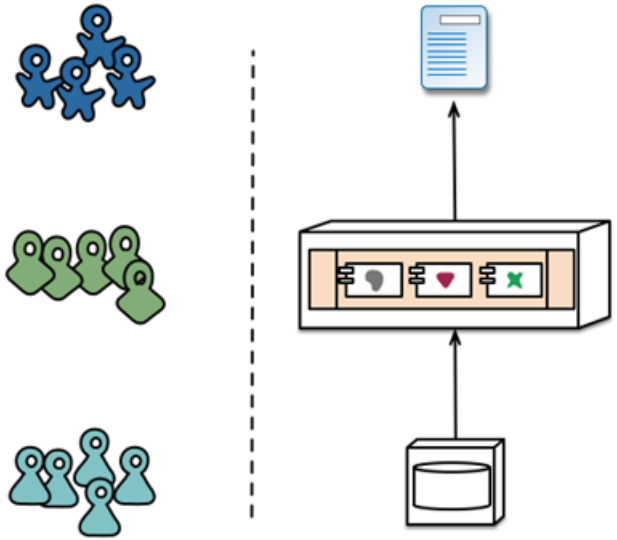
适者生存

拉馬克在*1809*年出版的《动物哲学》（*Philosophie Zoologique*）

演进式架构
EVOLUTIONARY ARCHITECTURE

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

在多维度上，刻意引导下的增量改变作为第一原则。



Any organization that designs a system will produce a design whose structure is a copy of the organization's communication structure.

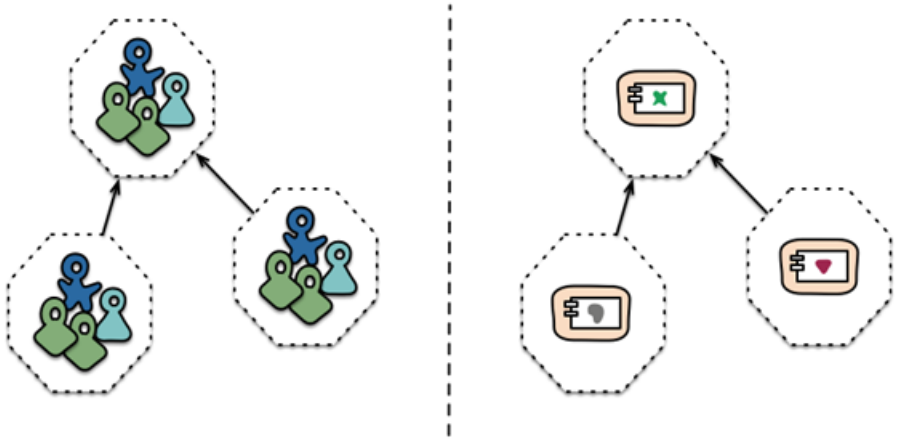
-- Melvyn Conway, 1967

FROM:

隔离的功能性团队，只对隔离的应用架构负责。

TO:

跨职能的团队，对业务能力端到端负责。



多维度

Multiple Dimensions

领域架构

安全架构

… …

evolvability

Table 1-1. Partial list of "-ilities"

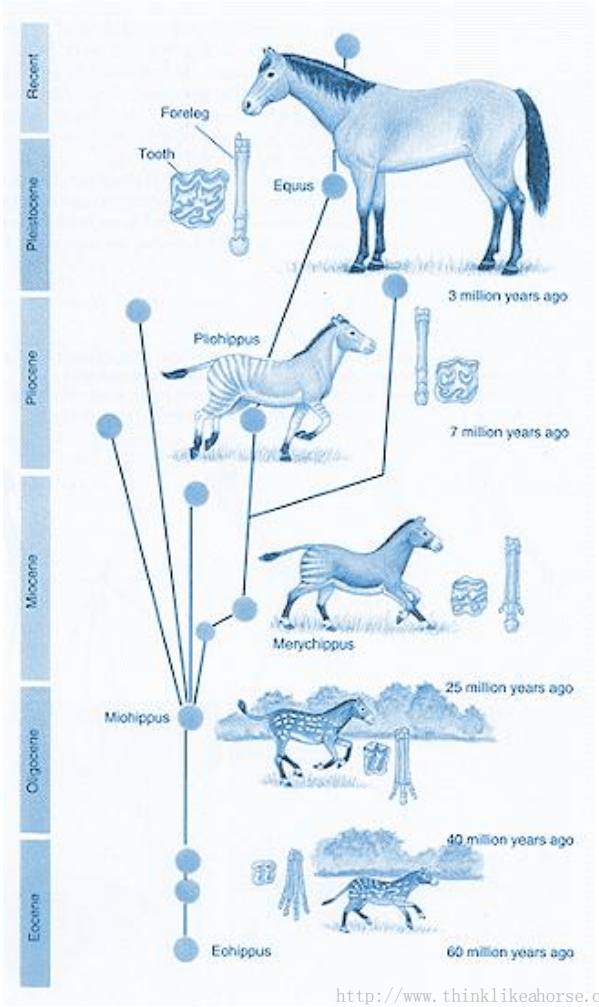
accessibility	accountability	accuracy	adaptability	administrability
affordability	agility	auditability	autonomy	availability
compatibility	composability	configurability	correctness	credibility
customizability	debugability	degradability	determinability	demonstrability
dependability	deployability	discoverability	distributability	durability
effectiveness	efficiency	usability	extensibility	failure transparency
fault tolerance	fidelity	flexibility	inspectability	installability
integrity	interoperability	learnability	maintainability	manageability
mobility	modifiability	modularity	operability	orthogonality
portability	precision	predictability	process capabilities	producibility
provability	recoverability	relevance	reliability	repeatability
reproducibility	resilience	responsiveness	reusability	robustness
safety	scalability	seamlessness	self-sustainability	serviceability
securability	simplicity	stability	standards compliance	survivability
sustainability	tailorability	testability	timeliness	traceability

增量变化

Incremental Change

$$V \propto C$$

where
c = cycle time
v = maximum speed of new generations



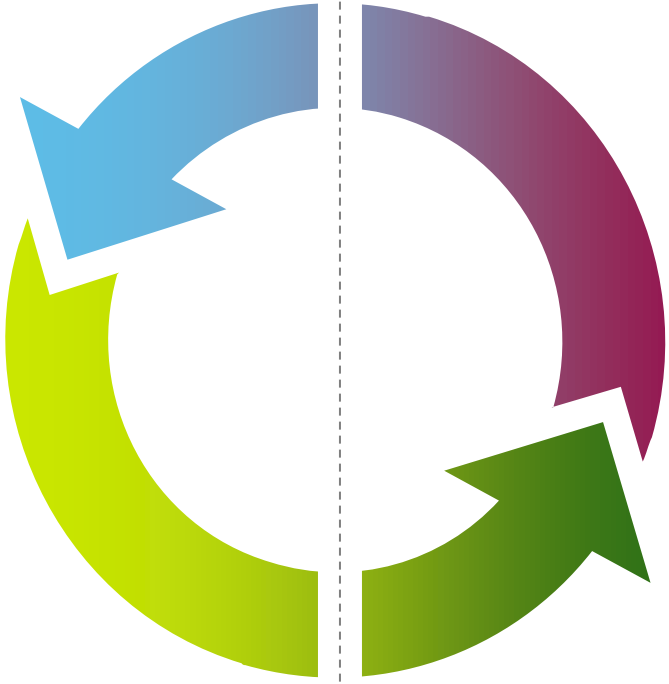
刻意引导

Guided

度量标准

- Performance
- Security
- Recoverability
- … …

Architects



Dev Team

验证测试

- Unit tests
- Functional tests
- Performance tests
- … …

适应度方程

Fitness Function

http://evolutionaryarchitecture.com

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

Depend on Dependencies

Categories: atomic, triggered, manual

Requirements:

- Make sure that no dependency updates break functionality past the developers' staging environment.

Additional Context:

- All affected projects use Java, Ruby, and Javascript.

Degrade Gracefully

Categories: holistic, continuous

Requirements:

- the enterprise architects have defined thresholds for acceptable performance degradation under scale, and the architecture should maintain performance and scale within those parameters

Additional Context:

- the architecture support good incremental change and modern DevOps practices

适应度方程

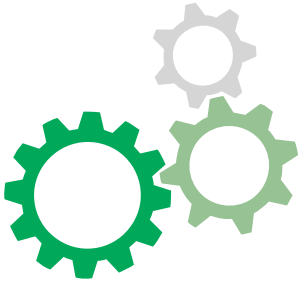
Fitness Function

Atomic
原子的

Holistic
全局的

Batch
批次的

Continuous
持续的

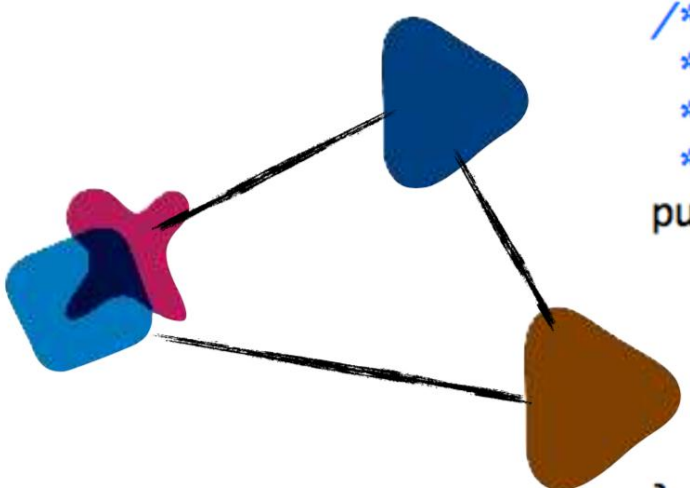


适应度方程

Fitness Function

Batch
批次的

Atomic
原子的



```
/**
 * 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());
```

适应度方程

Fitness Function

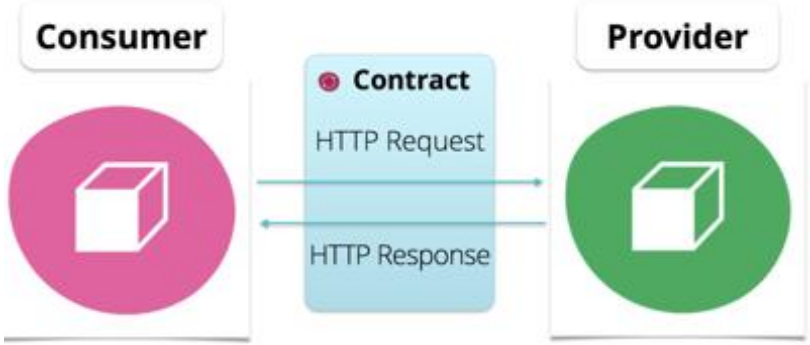
Batch
批次的

Holistic
全局的

```
PactNet.Mocks.MockHttpService.Models;
espace myHR.Tests.Pacts

public class myCoreUserContactsPacts : UserApiPactsBase<ContactsFixture>
{
    [Fact]
    public async Task should_get_user_contacts()
    {
        mockmyCoreApi
            .Given("user has a contact")
            .UponReceiving("a GET request for user contacts")
            .With(
                new ProviderServiceRequest
                {
                    Method = HttpVerb.Get,
                    Path = $"/users/{id}/contacts",
                })
            .WillRespondWith(
                new ProviderServiceResponse
                {
                    Status = 200,
                    Body = new[]
                    {
                        new
                        {
                            name = "Zhang BA",
                        }
                    }
                });
        await myCoreApiClient.GetContacts(id);
        mockmyCoretApi.VerifyInteractions();
    }
}
public class ContactsFixture : ServiceFixture
{
    public ContactsFixture()
    {
        mockService("myHR", "myCoreApi.Contacts", 10853);
    }
}
```

```
{
  "provider": {
    "name": "myCoreApi.Contacts"
  },
  "consumer": {
    "name": "myHR"
  },
  "interactions": [
    {
      "description": "a GET request for user contacts",
      "provider_state": "user has a contact",
      "request": {
        "method": "get",
        "path": "/users/af04e9ec-4511-47c4-9632-99a87b8e39d8/contacts"
      },
      "response": {
        "status": 200,
        "body": [
          {
            "name": "Zhang ba",
          }
        ]
      }
    }
  ],
  "metadata": {
    "pactSpecificationVersion": "1.1.0"
  }
}
```



适应度方程

Fitness Function

Continuous
持续的

Atomic
原子的

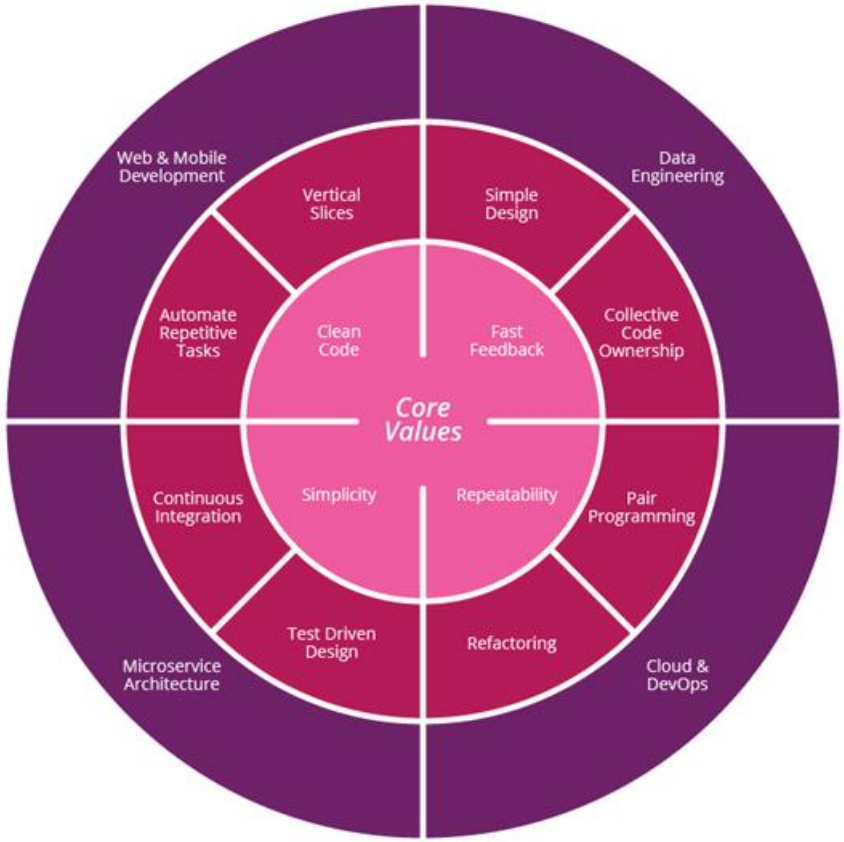
适应度方程

Fitness Function

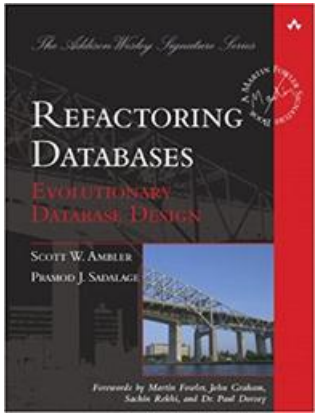
Continuous
持续的

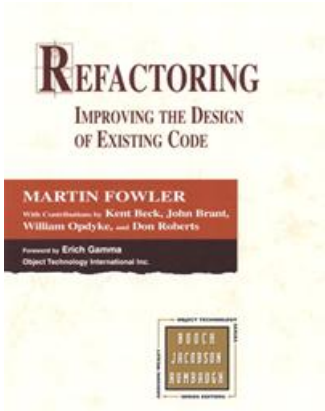
Holistic
全局的

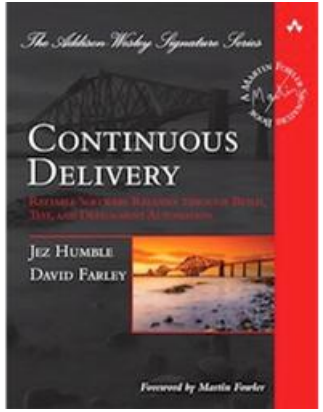
工程卓越 —— 演进的基础


Web & Mobile Development
Vertical Slices
Data Engineering
Simple Design
Automate Repetitive Tasks
Clean Code
Fast Feedback
Collective Code Ownership
Core Values
Continuous Integration
Simplicity
Repeatability
Pair Programming
Microservice Architecture
Test Driven Design
Refactoring
Cloud & DevOps


O'REILLY
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman


The Addison Wesley Signature Series
Refactoring Databases
Evolutionary Database Design
Scott W. Ambler
Pramod J. Sadalage


Refactoring
Improving the Design of Existing Code
MARTIN FOWLER
With Contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International Inc.
BOOCH
JACOBSON
RUMBAUGH


The Addison Wesley Signature Series
Continuous Delivery
Jez Humble
David Farley
Foreword by Martin Fowler

没有任何实践可以代替交流沟通

肖然
rxiao@thoughtworks.com

架构应接未来变化
IAS 2018