Web Software Evolution and Change Management on the Cloud

LOG 8430

Marios Fokaefs
Software Lifecycle

- Design
- Development
- Delivery
- Maintenance
Changes, changes, changes...

- Externally Stimulated
  - New Features
  - Increased Demand
  - Competition

- Internally Motivated
  - Improve Design
  - Fix Bugs
  - Improve Performance
Web Software Ecosystem

- Resource providers
  - VM costs
  - Data transfer/storage
  - Service costs

- Software provider
  - New prices
  - New clients
  - Improved performance

- Competitors
  - Functionality/quality friction
  - Price/costs friction

- Clients
  - New features
  - Improved performance

- Migration/Switching costs
  - New prices
  - Backwards incompatibility

- Evolution costs
  - Development effort
  - Resource costs

- Adaptation costs
  - New prices

©Marios Fokaefs - LOG8430
How do we make decisions on software change?

• Software change is not simply a technical problem.
• **RQ1**: How do we capture and take into account the different interests of various stakeholders?
  • It’s a social problem.
• **RQ2**: How do we consider the independent motives of our business partners and competitors?
  • It’s a business problem.
• **RQ3**: How do we extract the financial transactions between business entities from their technical interactions during change?
  • It’s an economic problem.
• **Thesis**: Software change is a **socio-technical** problem with **business and economic** extensions.
Web Service Evolution

A Game-Theoretic and Economic Approach
We need to change our service to offer more features.
The Research Problem

• Technical (RP1)
  • Tools for supporting client adaptation to new service versions
    • for the two prevalent service platforms, WS* (RP1a) and REST (RP1b)

• Economic Decision Making (RP2)
  • Tools to guide
    • clients on whether to adapt or migrate
    • providers on whether to change and whether to support their clients
Background

• Technical
  • Impact analysis (Wang and Capretz 2009)
  • Lack of (tool) support for service clients (Chow and Notkin 1996, Xing and Stroulia 2005a)

• Economic Decision Making
  • Cost/Value driven decisions (Boehm and Sullivan 2000, Ozkaya, Kazman and Klein 2007, Tansey 2008)
  • Decisions constrained within a single system
Thesis

• *Service system evolution can and should be, not only* **technically**, *but also* **economically** *conscious through support from* **CASE tools**.

• WSDarwin attempts to support and prove this statement.
the contribution

The WSDarwin Framework

- Economic model
- Game-theoretic model
- Decision-support system

- Generate WADL service interfaces
- Compare (cross-vendor) service-interface versions

- Compare WSDL service interfaces
- Adapt Java clients
- Test Java clients
WSDarwin for WS-* services

• Structured and machine-readable service interface specifications (WSDL)
  ➔ We can compare the specifications to infer additional information about evolution or similarity between services/version of services.
  ➔ We can auto-generate client proxies from the specification on any language.

• WSDL is verbose.
  ➔ We need a more abstract specification for more efficient tools.
WSDarwin for WS-* services

Service Interface v1 ➔ Translate into WSMeta ➔ Compare Interfaces ➔ Identify Refactorings ➔ Service Interface Delta

Test ➔ Adapt Client Proxy

Service client proxy v1 ➔ Service client proxy v2

Undo

©Marios Fokaefs - LOG8430
Service Interface Comparison

<table>
<thead>
<tr>
<th>CT</th>
<th>Input</th>
<th>O</th>
<th>Add</th>
<th>PT</th>
<th>operand1: int</th>
<th>operand2: int</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>result: int</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CT</th>
<th>Input</th>
<th>O</th>
<th>Add</th>
<th>PT</th>
<th>operand1: int</th>
<th>operand2: int</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>result: int</td>
<td></td>
</tr>
</tbody>
</table>

AddDelta

MatchDelta

ChangeDelta

writeToDB:boolean

WSDarwin for WS-* services

Service Interface v1
Service Interface v2

Translate into WSMeta

Compare Interfaces

Identify Refactorings

Service Interface Delta

Test

Adapt Client Proxy

Undo

Service client proxy v1
Service client proxy v2

©Marios Fokaefs - LOG8430
Service Client Adaptation

The client invokes the old stub

Client

The change is transparent to the client.

Client Stub V1

WSDarwin

Client Stub V2

Service V1

The old stub is transformed to invoke the new stub

Service V2

WSDarwin for WS-* services

comparison

Service Interface v1 → Translate into WSMeta → Compare Interfaces → Identify Refactorings → Service Interface Delta

adaptation

Test

Adapt Client Proxy

Undo

Service client proxy v1 → Service client proxy v2

©Marios Fokaefs - LOG8430
WSMeta: A Service Interface meta-model

WSMeta mimics the structure of the Java client proxies produced by Apache Axis2, which facilitates the adaptation process.

- WSDL/WSMeta → Operation-centric services
- WADL → Data-centric services

Replacing references with containment results in larger files but faster reference resolution.

WSDarwin for WS-* services - Overview

- WS-* services always come with a WSDL interface specification, representing the information necessary for service invocation.
- Auto-generated native source code follows the structure of the interface.
- WSDarwin uses the WSDL specification and generated proxy to infer more information about the evolution of a service and adapt clients to the new version.
- Comparison: 10x faster than generic methods for large interfaces (2.5K elements) (AWS handbook)
- Parsing+Comparison+Adaptation in under 10 minutes. (ICWS2014)

WSDarwin for REST services

REST Services do not come with structured machine-readable interface specifications.
WSDarwin for REST services

• REST services are not published through a structured, machine-readable specification.
  ➔ The specification and the evolution may be under-documented.

• A machine-readable specification can help us automate tasks like
  ➔ Client generation
  ➔ Service comparison and mapping

• For service mapping (and migration), we cannot trust structure or semantics.
  ➔ But we can trust data!
Service-interface & Client-proxy Generation

- Generate service interface
- Generate client proxy
- Test service
- Infer Schema
- Merge service elements
- Identify
  1. Enums
  2. Variable resource IDs

Web service source code

Request URL

provider

client

©Marios Fokaefs - LOG8430
URL Analysis and Resources w. Variable IDs

Resource with variable ID
- Path component in the same position but with different ID in different URLs.
- Common prefix.
- Minimum common suffix.
- Last components are not considered variable IDs
Schema Inference

• Type Resolution (using regular expressions)
  • string, long, int, short, byte, double, float, anyURI, email, dateTime, date
  • Batch processing gives confidence (e.g. type is string in 7 out of 10 URLs → 70% confidence)

• Enumerations
  • If values appear more than once → indication
  • AND if string values are in capital → recommendation
  • User eventually applies the refactoring to simple type with enumerations.
  • E.g. gender = {MALE, FEMALE}
Cross-vendor Service Mapping

manual steps | automated steps

API 1 ➔ Invoke service ➔ JSON/XML response

Manually mapped input and requests

API 2 ➔ Invoke service ➔ JSON/XML response

Compare parameters by value

Many-to-many parameter mapping

WSDarwin for REST services - Overview

Problem

• REST services do not come with a WADL interface specification, which limits the automatic support REST clients.
• Auto-generating native source code tools also exist for WADL and REST.
• A method for auto-generating WADL interfaces increases automation and reduces the provider-consumer integration overhead.
• Mapping data and structure from different services can help migration.

Solution

• WADL Generation: 1 request/resource, plus 1 additional request for every variable ID; more than 90% parameters correctly generated. (ongoing evaluation)
• Mapping: 63% precision (85%), 78% recall (83%), 6-7x faster than human. (WSE2013)
Service Evolution Economics
Decisions, decisions, decisions...

**When?** stimulus
- Changes from the environment.

**Why?** motivation
- Mainly business and financial motives.

**What?** plan
- Decide on a particular change.

**How?** execution
- Design the implementation of the change.
Service Ecosystems
Technical and Business transactions

Client
Proxy
Service
Interface
Provider1
ClientX
Service
App
Adapter
o-n
Service
Interface
Provider2

price
adaptation cost
adaptation cost
Value

price
adaptation cost
evolution cost
Value

2019-10-07
©Marios Fokaefs - LOG8430
Service Evolution Economic Model

\[ \text{Value of service} = f(\text{expected return, development effort, client requirements}) \]

\[ \text{Evolution/Adaptation cost} = f(\text{wage, development effort}) \]

- \textit{Price} as an optimization over the best and most probable market share.
- \textit{Market share} as an optimization over the highest and most probable price.
Game Theory 101

Normal Form (Simultaneous)

<table>
<thead>
<tr>
<th></th>
<th>P2</th>
<th>Not Confess</th>
<th>Confess</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>Not Confess</td>
<td>-1, -1</td>
<td>-3, 0</td>
</tr>
<tr>
<td></td>
<td>Confess</td>
<td>0, -3</td>
<td>-2, -2</td>
</tr>
</tbody>
</table>

Extensive Form (Sequential)

©Marios Fokaefs - LOG8430
Service Evolution Game

The setting
• Competitive market: Many providers – Many Clients
• Perfect Information game: Every player knows what the other player has done
• Rational players
• 2-stage game: Providers play first and then the clients react

The providers can:
1. Retain the status quo of the service (SQ)
2. Evolve the service (E)
3. Evolve the service and support the client (S)

The clients can:
1. Adapt to the new version of their current provider’s service (A)
2. Leave the provider and switch to another service (L)
Utilities

What do providers and clients get (or lose) from evolution?

Value for the client from a specific provider

Client

Utility  \(\equiv\)  Value  \(\equiv\)  price  \(\equiv\)  Cost

Adaptation costs

Evolution Costs

Provider

Utility  \(\equiv\)  Clients  \(\times\)  price  \(\equiv\)  Cost

Adaptation Costs for supporting current clients
Nash Equilibrium-Backwards induction
Calculate utilities for all game states, solve the client game first, upload the market division to the provider game, and find the Nash equilibrium for the N-player general-sum simultaneous game.

### First stage: Providers

<table>
<thead>
<tr>
<th></th>
<th>Provider1</th>
<th>Provider2</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQ</td>
<td>U1,U2</td>
<td>U1,U2 U1,U2</td>
</tr>
<tr>
<td>E</td>
<td>U1,U2</td>
<td>U1,U2 U1,U2</td>
</tr>
<tr>
<td>S</td>
<td>U1,U2</td>
<td>U1,U2 U1,U2</td>
</tr>
</tbody>
</table>

### Second stage: Clients

![Game tree diagram](image)
Service Evolution Economics Evaluation: The Cloud Ecosystem

• 3 Providers
  • Amazon
  • Google
  • Microsoft
• 5 (randomly generated) clients
  • Initial Market Division
    Amazon = \{A,B,C\}
    Microsoft = \{D\}
    Google = \{E\}
• Real data
  • Wages, Current and Future Prices, Initial Market Division
• Generated data
  • Efforts, Expected Returns for Clients

The scenarios discussed here are fantastic and there is very little connection to reality.
Service Evolution Economics Evaluation:
Final decision

Nash Equilibrium #1
(Provider Welfare)

Nash Equilibrium #2
(Ecosystem Welfare)
Economics-aware Cloud Resource Scalability

Extending cloud elasticity services for performance and economic optimization.
What is scaling (elasticity)?

• capacity = 600 req / VM / hour
Is scaling up/out always profitable? (CLOUD2016)

- price = 0.12 $ / request
- cost = 0.6 $ / VM / hour
- capacity = 600 req / VM / hour

Let’s drop the extra request!
Scaling through economic optimization

\[
\max \Pi = p \times x - W \times C_W - x \times c_n \\
\text{s.t. } x \geq \lambda \times A \\
W \times d_u \geq x \geq W \times d_l
\]

- \( \Pi \): service profit
- \( p \): nominal revenue per request
- \( x \): service throughput per time unit
- \( W \): number of servers
- \( C_W \): server cost per time unit
- \( c_n \): associated costs per request
- \( \lambda \): arrival rate per time unit
- \( A \): availability rate (SLA)
- \( d_u, d_l \): upper/lower capacity of server in requests per time unit

<table>
<thead>
<tr>
<th></th>
<th>elasticity</th>
<th>optim</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>53.47%</td>
<td>64.22%</td>
</tr>
<tr>
<td>Availability</td>
<td>100%</td>
<td>97%</td>
</tr>
<tr>
<td>Response time</td>
<td>32.53ms</td>
<td>35.06ms</td>
</tr>
<tr>
<td>revenue</td>
<td>922.20$</td>
<td>890.81$</td>
</tr>
<tr>
<td>cost</td>
<td>1.88$</td>
<td>1.78$</td>
</tr>
<tr>
<td>ServProf</td>
<td>920.33$</td>
<td>889.03$</td>
</tr>
</tbody>
</table>

Still not good... What if we raise the price instead of dropping requests? And wait before we scale?
Extended scaling economics (SEAMS 2016)

- price = 0.12 $ / request
- cost = 0.6 $ / VM / hour
- capacity = 600 req / VM / hour
- true value = price – cost / req

\[ v = \frac{\text{price}}{\text{capacity}} - \frac{\text{cost}}{\text{request}} \]

- At 1,500 requests: \( v = 0.02 \ $ / \text{request} \)
- At 1,700 requests: \( v = 0.02 \ $ / \text{request} \)
Adaptation process

Monitor → CPU → Analyze

- **U_Black**: \(v' \geq 0\)
  - \(\#S++\)
- **U_Gray**: \(v \geq 0\)
  - \(\#S\)
- **White**: \(v \leq 0\)
  - \(\#S--\)
- **L_Gray**: \(p' = \pi c'\)
- **L_Black**: \(p' = c'\)

Results

- This is not a profit-maximization strategy but loss-minimization.
- How can we bound price increases to minimize the risk of losing clients?

potentially lower prices while minimizing losses.

<table>
<thead>
<tr>
<th></th>
<th>High Thresholds/High-Fixed Price</th>
<th>Low Thresholds/Low-Fixed Price</th>
<th>Low Thresholds/High-Fixed Price</th>
<th>High Thresholds/Low-Fixed Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>62.65</td>
<td>62.65</td>
<td>60.12</td>
<td>62.74</td>
</tr>
<tr>
<td>Response time</td>
<td>125.85</td>
<td>125.85</td>
<td>125.95</td>
<td>121.45</td>
</tr>
<tr>
<td>Cost/Request</td>
<td>0.0032</td>
<td>0.0032</td>
<td>0.0033</td>
<td>0.0035</td>
</tr>
<tr>
<td>Price/Request</td>
<td>0.0040</td>
<td>0.0028</td>
<td>0.0028</td>
<td>0.0040</td>
</tr>
<tr>
<td>Total Cost</td>
<td>78.56</td>
<td>82.65</td>
<td>82.65</td>
<td>86.27</td>
</tr>
<tr>
<td>Total Revenue</td>
<td>98.69</td>
<td>69.08</td>
<td>69.14</td>
<td>98.76</td>
</tr>
</tbody>
</table>

Avg True Value

- This is not a profit-maximization strategy but loss-minimization.

How can we bound price increases to minimize the risk of losing clients?

potentially lower prices while minimizing losses.
**Web Software Ecosystem**

\[ \Pi_s = p_s \times \lambda - CW \times W - c_n \times \lambda \]

\[ \Pi_A = \gamma \times \lambda - p_s \times \lambda - c_n \times \lambda \]

\[ U = \alpha \times \frac{\gamma_u - \gamma}{\gamma_u - \gamma_l} + \beta \times \frac{R_u - R}{R_u - R_l} \]
Multi-objective optimization

- Find Pareto optimal results – more than one.
- Pick one that satisfies a goal.
- Set the goals as combination of weights ($w_S + w_A + w_U = 1$)
- Pick the result which maximizes the total profit over the given goal.
- Service and User goals “agree”
- App needs non-zero weight to “survive”

<table>
<thead>
<tr>
<th></th>
<th>Service</th>
<th>App</th>
<th>User</th>
<th>Ecosystem</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td></td>
<td>▼</td>
<td></td>
<td></td>
</tr>
<tr>
<td>app reven</td>
<td></td>
<td>▼</td>
<td>▲</td>
<td>▲</td>
</tr>
<tr>
<td>price</td>
<td></td>
<td>▼</td>
<td>▼</td>
<td></td>
</tr>
<tr>
<td>service</td>
<td>▲</td>
<td>▼</td>
<td></td>
<td></td>
</tr>
<tr>
<td>app</td>
<td>▼</td>
<td>▼</td>
<td>▼</td>
<td>▼</td>
</tr>
<tr>
<td>user</td>
<td>▼</td>
<td>▼</td>
<td>▼</td>
<td>▼</td>
</tr>
</tbody>
</table>
DevOps: From philosophy to culture, from practice to vision
...that DevOps would have a single definition!
DevOps according to...everybody!

- **Wikipedia**: “DevOps is a software development methodology...”
- **Atlassian**: “DevOps is a set of practices...”
- **Amazon**: “DevOps is the combination of cultural philosophies, practices, and tools...”
- **Microsoft**: “DevOps is the union of people, process, and products...”
- **RedHat**: “DevOps is an approach to culture, automation, and platform design...”
- **IBM**: “DevOps is an approach to lean and agile software delivery...”
What is DevOps?

A: A process
B: A culture
C: A philosophy
D: A set of tools
What is DevOps?

A: A process
B: A culture
C: A philosophy
D: A set of tools

©Marios Fokaefs - LOG8430
DevOps as a philosophy
As a philosophy, DevOps is a... (continuous) process!
But what does it mean to be continuous? ...and how can we be?

Without automation, manual tasks will slow down the process, may introduce errors and will eventually break continuity.

Without integration, there will be gaps in communication, understanding, interoperability and automation is almost impossible.
Time for cooking!

• Automation needs integration!
• Abstractions are easier to integrate with each other.

• Abstractions can be easier to be communicated to stakeholders.
• It is easier for abstractions to understand each other.

• A workflow involving abstractions can be automated.
• Models can specify abstractions! (pretty well actually!)
And into the oven!

• So, models are good for:
  • Integrating knowledge and expertise (functionality, performance, security, business value)
  • Communicating this knowledge to the stakeholders in a language they understand (unique for every class of stakeholders).

• But, models alone do not implement automation!

• We need tools!

• Tools can:
  • Seamlessly connect the abstractions and automate the workflow.
  • Hide the knowledge so as to minimize the learning curve and the need for extended understanding.
DevOps as a culture
To begin with a story...

**FIGURE 5.** DevOps team formation. (a) Traditional horizontal teams. (b) Vertical teams in DevOps. In DevOps, each team is responsible for a specific service and contains people with different skills, such as development and operations skills. The team members cooperate from the project’s start to create more value for the particular service’s end users.
DevOps is the people

• We cannot consider DevOps without the people. (at least not now)

• The collusion of developers and IT staff/sys admins exists for a long time now.

• However, now it becomes more necessary and relevant due to the nature of software, its velocity, volume, and volatility.

• We need people who would understand the multiple facets of software and have the necessary skills to function in heterogeneous teams.

• $122,969-$133,378 avg US salary for DevOps Engineer ($92,172 for IT professionals), 74,834 LinkedIn job postings for DevOps Engineers worldwide.
DevOps in practice
A plethora of tools!
But without foundations?!

• The tools emerged before the concept of DevOps itself.
• Partly, because the tools tackled specific parts or phases of DevOps.
• For this reason, the landscape is fragmented rather than integrated.
• This also led to misconceptions and the disagreements wrt a universal definition:
  • “DevOps is Continuous Integration.”
  • “You don’t do DevOps, unless you use Docker.”
• We are in need of reengineering.
  • We need to redefine the existing tools in the new context of DevOps.
  • We need to reengineer the tools so that they fit in the overall paradigm of DevOps.
  • We need to develop new integrated tools properly tailored to DevOps.
DevOps future outlooks
Something is definitely missing

• Adaptivity and self-adaptive systems
  • Control Theory?!

• Quality
  • Trade-offs!

• Models

• Training

• Business
  • Ecosystem and stakeholders
  • Game theory?!

• “Proper” machine learning
The two cycles of DevOps
DevOps is Software Engineering 2.0

- Continuous Integration
- Continuous Delivery
- Continuous Deployment
- Continuous Utility
- Continuous Software Engineering
- Continuous Sustainability
- Continuous Profitability

Operations

Development

Business
Future ideas and projects
I have some positions for summer interns and master students.
CRSNG / BRPC - BOURSES DE RECHERCHE EN MILIEU UNIVERSITAIRE (1er cycle)

- https://www.polymtl.ca/aide-financiere/bourses/crsng-brpc-bourses-de-recherche-en-milieu-universitaire-1er-cycle
1. Management of data in IoT systems (Smart Buildings)

• Use of Lambda architecture.
• Use of the Tomcat-Kafka-Spark-Cassandra topology
• Improve throughput, response time for analytics.
• Provide elasticity to manage performance and security, but maintain cost and energy consumption.
• What is the impact of location, motion, network connection, data schema and communication protocol in the quality and economics of the system?
2. Multi-objective autoscaler

- Create an autonomic management system for cloud and web software systems.
- Manage the resources of the system, its state and its architecture.
- Maintain performance, security and economics at the same time.
- Build a self-adaptive framework; a modular and pluggable architecture to support current and future analyses and self-adaptive actions.
- Use Docker, Kubernetes, OpenStack and build multi-objective optimizers.
3. SAM: A system administration minion

- Combine self-adaptive systems, autonomic management systems, cognitive computing and intelligent assistants (chatbots).
- Use IBM Watson and other cognitive platforms.
- Build a chatbot that would interact with a DevOps team to monitor, visualize, notify, and act upon the system.
4. BizDevOps and FinTech

• Build management and self-adaptation solutions for banking and financing applications.
• Elasticity in banking applications.
• Decision-support systems for marketing and banking software solutions.
5. Root cause analysis for Technical Debt

- Technical Debt is now detected only by self-admission comments in the code and the documentation.
- Is it possible to detect Technical Debt instances if they are not admitted by the developers?
- Low quality is often an indicator for Technical Debt and Refactoring can be used to address TD.
- Can we use the history of metrics and refactorings to detect the first occurrence of a TD instance?