Eating an Elephant

Iterative Maintenance & Modernization of a Legacy System
Working with a Legacy System is...

1. Difficult
2. Under-appreciated
3. Opportunity to innovate
What is *Legacy*?
If your software is used, it is legacy.

However... it is valuable because it is used

Don’t forget the end user has value in the remaining functionality of the legacy system.

A legacy system has a reputation

- Sometimes a very good reputation
- Don’t dismiss the credibility it has

Thomas Mullen - Writing code for other people: cognitive psychology of chunking

Michael Feathers - RailsConf talk (Working effectively with Legacy code)

Taylor Jones - Working with Legacy Code

What is Legacy?

"written years ago with outdated techniques, yet continues to be useful" …

"large software systems that we don’t know how to cope with but are vital to our organization"

"any systems that cannot be modified to adapt to constantly changing business requirements and their failure can have a serious impact on business"


“A major portion of the time spent coding and designing is taken up in learning and understanding the application code.”

“The majority of the development cost is spent maintaining the existing code.”

“Most software tasks are to extend/mend existing software…”

If you don’t know where you are, a map won’t help.

~Watts Humphrey
Maintenance, Modernization and Migration (in context)
Maintenance includes Migration

ISO 14764-2006 and other IEEE standards (like ISO/IEC 12207) place migration as a departure from the maintenance cycle...

But in my experience (working on small teams), it should be part of the maintenance cycle.

A structured legacy to SOA migration process and its evaluation in practice
Migration as a Structured Process

The iterative model looks like a way to categorize maintenance activities into phases of migration.
Extended Iterative Maintenance Lifecycle Using eXtreme Programming

Migration as Maintenance

Looking at the big picture, we see the incorporation of migration planning into the cycling of maintenance process.

Spahn, N. (2016), When can we migrate? A model for approaching legacy system migration
https://hal.archives-ouvertes.fr/hal-01687747
What about the System at UCSB?

Working with a legacy system is difficult
Enterprise Resource Planning Tool

History:
Development began in the late 90’s
Built to meet a need:
  Replacing a non-y2k compliant system
Commercial framework (Graphical IDE)

Architecture:
Distributed systems
End-to-end proprietary language
Commercial backups to attached storage
Enterprise Resource Planning Tool

History:
Development began in the late 90’s
Replacing a non-y2k compliant system
Commercial framework (Graphical IDE)

Architecture:
Distributed systems
End-to-end proprietary language
Commercial backups to attached storage

4 years ago:
I was hired onto the project
Graduate studies in Software Engineering
Learned all that I could from the Chief Architect/Designer & end users
Enterprise Resource Planning Tool

3 years ago:
New manager was hired
Phenomenal programmer
Quick learner with good ideas

2 years ago:
Iterative Migration Model
- Scrum practice: maintenance which incorporates development strategies
- Success at every structured phase
- Melding migration into maintenance
Working with a Legacy System is: **Difficult**

1. **20 years of code**
   - Developed by various people

2. **Commercial Software product**
   - Updates regularly (older versions no longer supported)
   - Desktop compatibility issues

3. Difficult to find skilled help

4. *Significant changes are costly!*

What have you done with the system?

Working with a legacy system is under-appreciated
Automation

Backups & Monitoring

Backup: off-site managed storage

Monitoring:
- age of backups across the distributed systems
- Responsiveness of servers across infrastructure
Consistency

We only have vanilla Ansible scripts to:

- Report on server configurations
- Surface anomalies
Active maintenance

We know a language that you don’t know...

We have become fluent in writing and debugging this legacy language so that we can:

- Correct defects
- Create new functionality
Create new functionality...

In the *legacy language*.

- Using the limited data types
- Language nuances
- Within the paradigm of the existing Software Framework

In *NodeJS*:

- Creating APIs to be consumed
- Building prototypes
Building Prototypes...

Leveraging Campus SSO

NodeJS + React + Firebase

- Serverless (almost) architecture
Working with a Legacy System is:

**Under-Appreciated**

Prototypes don’t always get to production
Working with a Legacy System is: Under-Appreciated

Prototypes don’t always get to production

Most of the changes that we make to the system are never noticed by users
Working with a Legacy System is: **Under-Appreciated**

Prototypes don’t always get to production

Most of the changes that we make to the system are never noticed by users

*Incremental changes* for a better user experience
Working with a Legacy System is: Under-Appreciated

Prototypes don’t always get to production

Most of the changes that we make to the system are never noticed by users

*Incremental changes* for a better user experience

Image: https://www.bluecoda.com/blog/shift-3-year-redesign-amazon-model
What are you working on now?

Working with a legacy system provides an opportunity to innovate.
APIs and services

NodeJS

Replace:

‘one of a kind’ services:
don’t touch it, *it might break*

Create:

Testable, reproducible:

**micro-services**
Replace: fragile services

What was:

- Older version of a Desktop OS
- Legacy server software not supported
- Proprietary database driver
- Convoluted codebase:
  - Legacy language
  - Not in the style of the rest of the app
Replace: fragile services

What was:
- Older version of a Desktop OS
- Legacy server software not supported
- Proprietary database driver
- Convoluted codebase:
  - Legacy language
  - Not in the style of the rest of the app

Is becoming:
- Lightweight express application
- Deployable to any modern OS
- Leveraging open source code
  - Testable
  - Reliable
  -Inspectable
Create: future-facing tools & prototypes

UCPATH
- Interface to aide in title code changes
- Employee ID mapping service utilized within existing framework

APIGEE
- Looking to be a producer/consumer of the API service

Overhaul of the existing web interface:
- Single page web application
- Making use of the existing auth system

Image: https://inhabitat.com/wp-content/blogs.dir/1/files/2013/05/Golden-Gate-Bridge-Pavilion-Jensen-Architects-3.jpg
Working with a Legacy System provides an opportunity to innovate.

Some prototypes evolve into production services.

Change is slow, but it is taking place.

The goal is maintain usability to meet business needs. Maintain, modernize or migrate with end users.
Discussion...