Eating an Elephant

Iterative Maintenance &
Modernization of a Legacy System

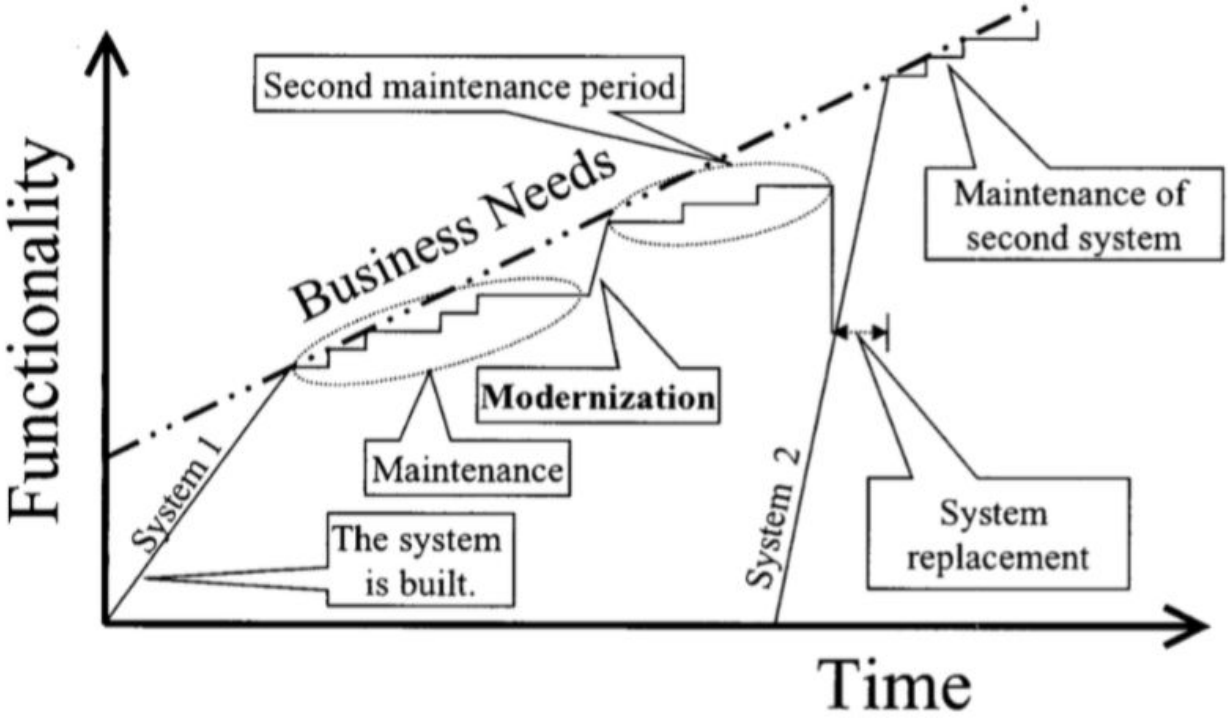
Working with a Legacy System is...

1. Difficult
2. Under-appreciated
3. Opportunity to innovate

image-source: http://getdrawings.com/image/golden-gate-bridge-drawing-55.jpg

What is *Legacy*?

If your software is used, it is legacy.



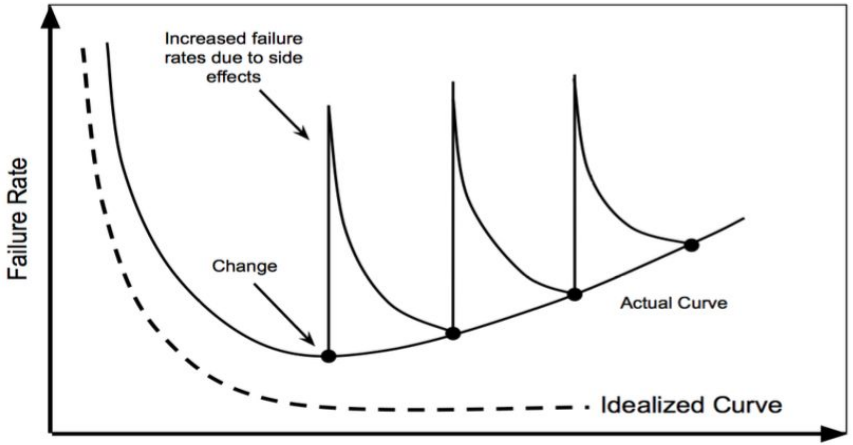
Comella-Dorda, S.; Wallnau, K. C.; Seacord, R. C. & Robert, J. E. (2000),
A Survey of Black-Box Modernization Approaches for Information Systems., in 'ICSM' , IEEE Computer Society, , pp. 173-183 .

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

Pressman, R. (1994), *Software Engineering, a Practitioner's Approach (European Edition)* , McGraw Hill , New York .

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”written years ago with outdated techniques, yet continues to be useful” …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

K. Bennett, “Legacy systems: Coping with success,” IEEE Software, vol. 12, no. 1, pp. 19–23, 1995.

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

M. L. Brodie and M. Stonebraker, Migrating Legacy Systems. Gateways, Interfaces, and the Incremental Approach. Morgan Kaufmann, 1995.

“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…”

T. Mullen, “Writing Code for Other People: Cognitive Psychology and the Fundamentals of Good Software Design Principles. OOPSLA 2009

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

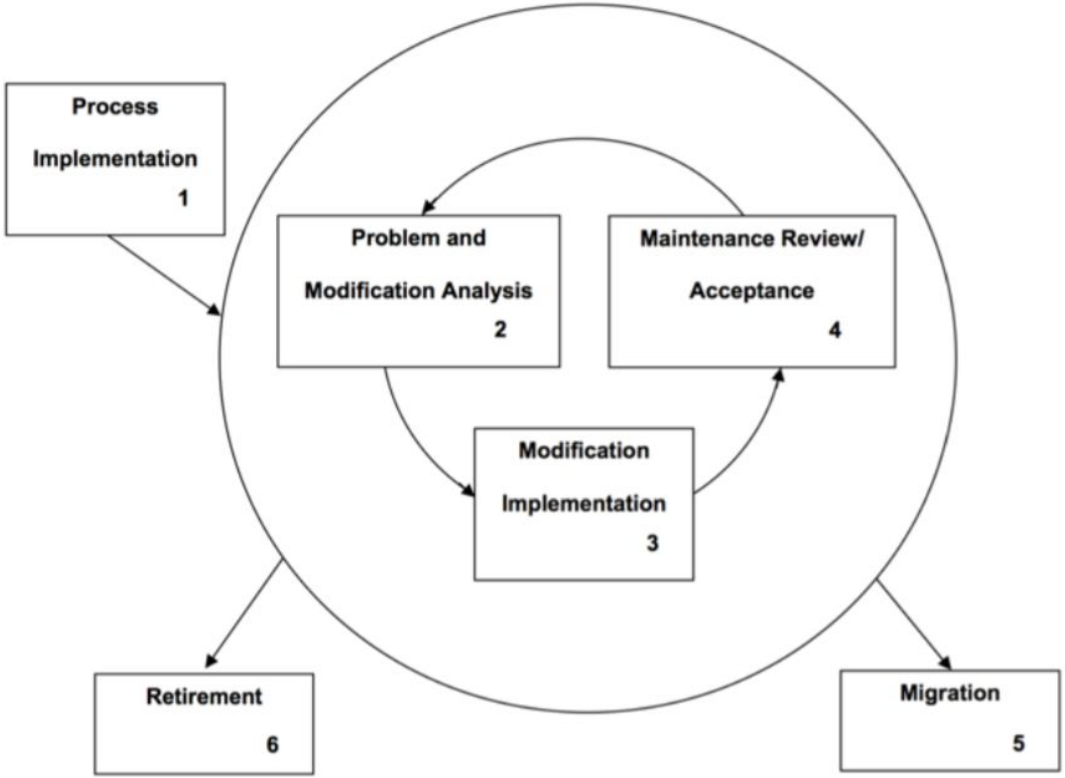
~Watts Humphrey

Maintenance, Modernization and Migration *(in context)*

Maintenance includes Migration

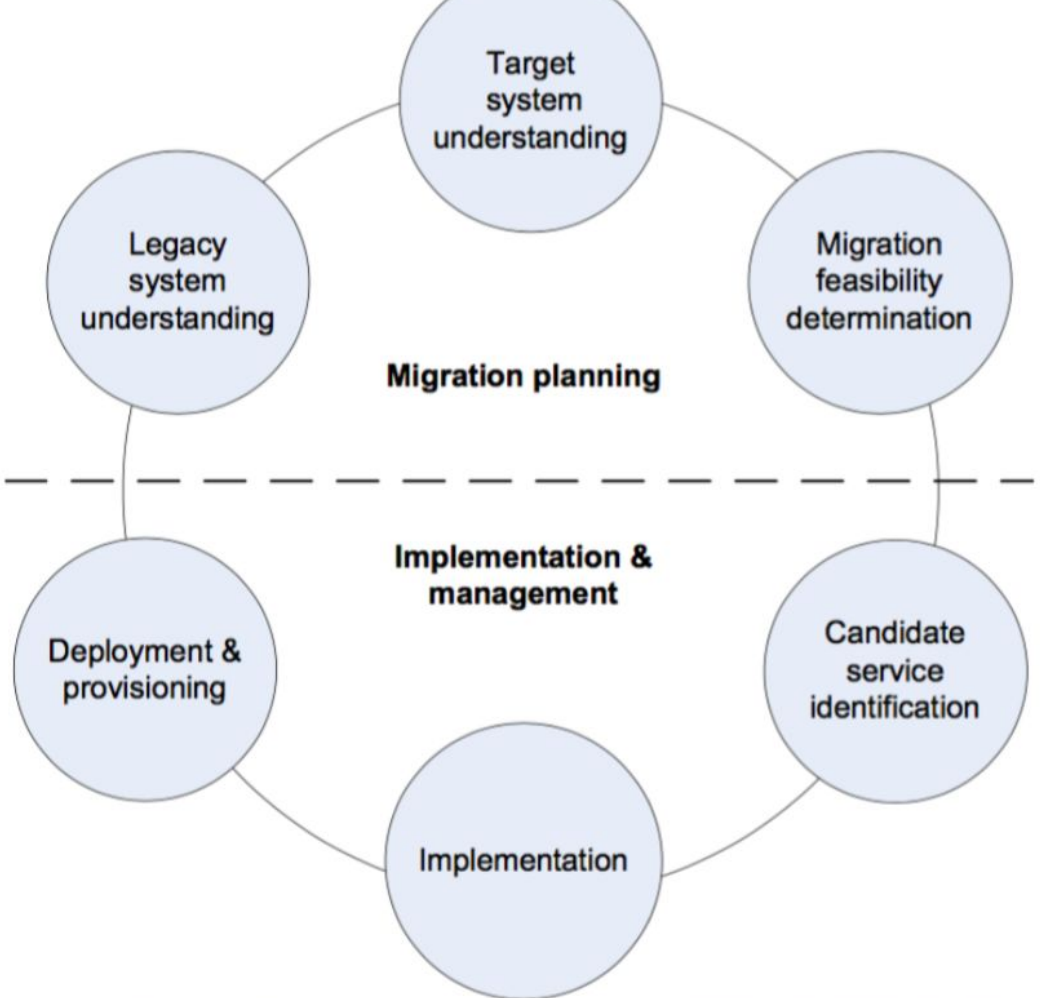
ISO 14764-2006 and other IEEE standards (like ISO/IEC 12207) place migration as a departure from the maintenance cycle…

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



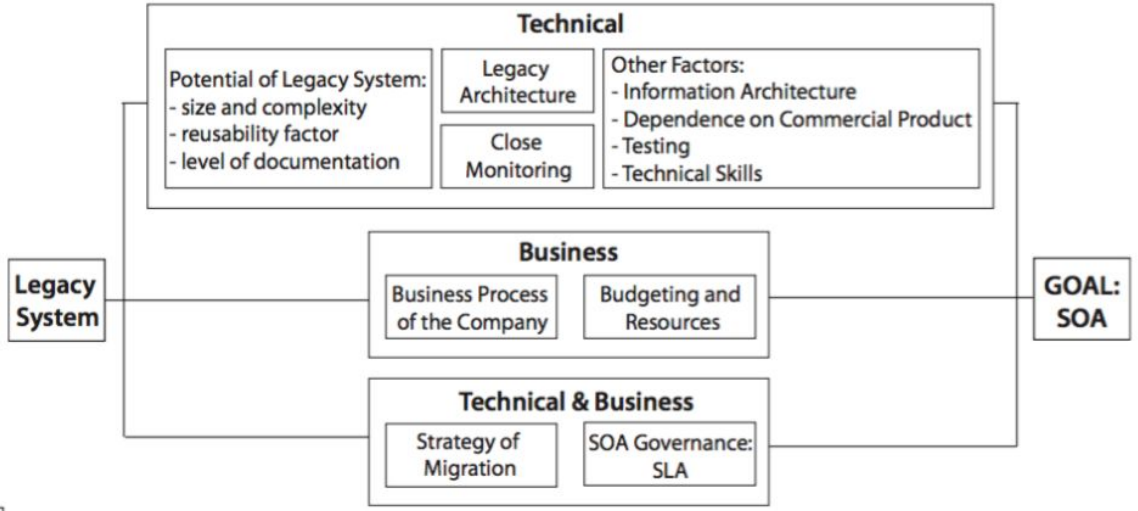
International Standards Organisation (ISO) (2006), *Standard 14764 on Software Engineering - Software Maintenance* , ISO/IEC .

A structured legacy to SOA migration process and its evaluation in practice



Khadka, R.; Saeidi, A.; Jansen, S. & Hage, J. (2013), A structured legacy to SOA migration process and its evaluation in practice in Anca Daniela Ionita; Grace A. Lewis & Marin Litoiu, ed., 'MESOCA' , IEEE, , pp. 2-11 .

Success Factors model for migrating legacy systems



Propositions - Factors in Literature:

Potential of Legacy System
Strategy of Migration
SOA Governance: SLA
Business Process of the Company
Budgeting and Resources

Findings - Factors in Case Studies:

Potential of Legacy System
Strategy of Migration
SOA Governance: SLA
Business Process of the Company
Budgeting and Resources
Legacy Architecture
Close Monitoring
Other Factors:
- Information Architecture
- Dependence on Commercial Product
- Testing
- Technical Skills

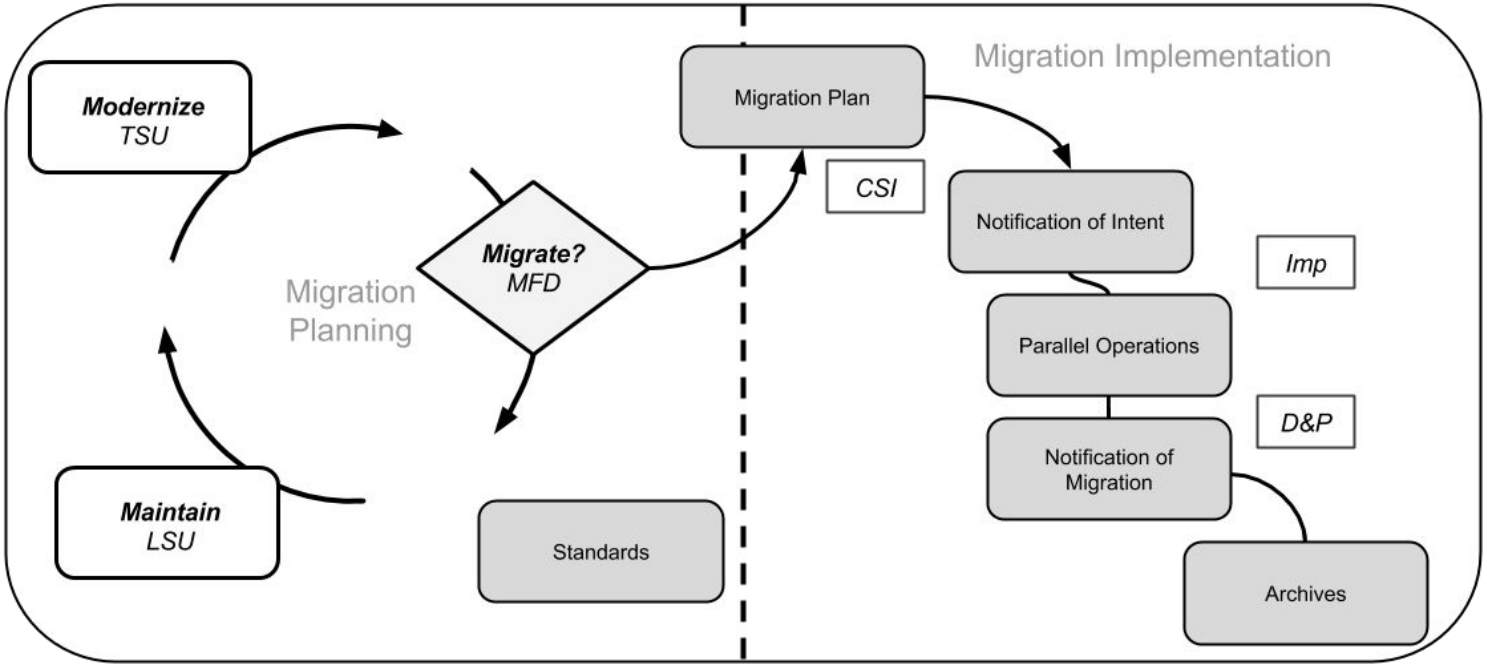
Conclusions- Generalized Factors mentioned by all Case Studies:

Potential of Legacy System
Strategy of Migration
SOA Governance: SLA

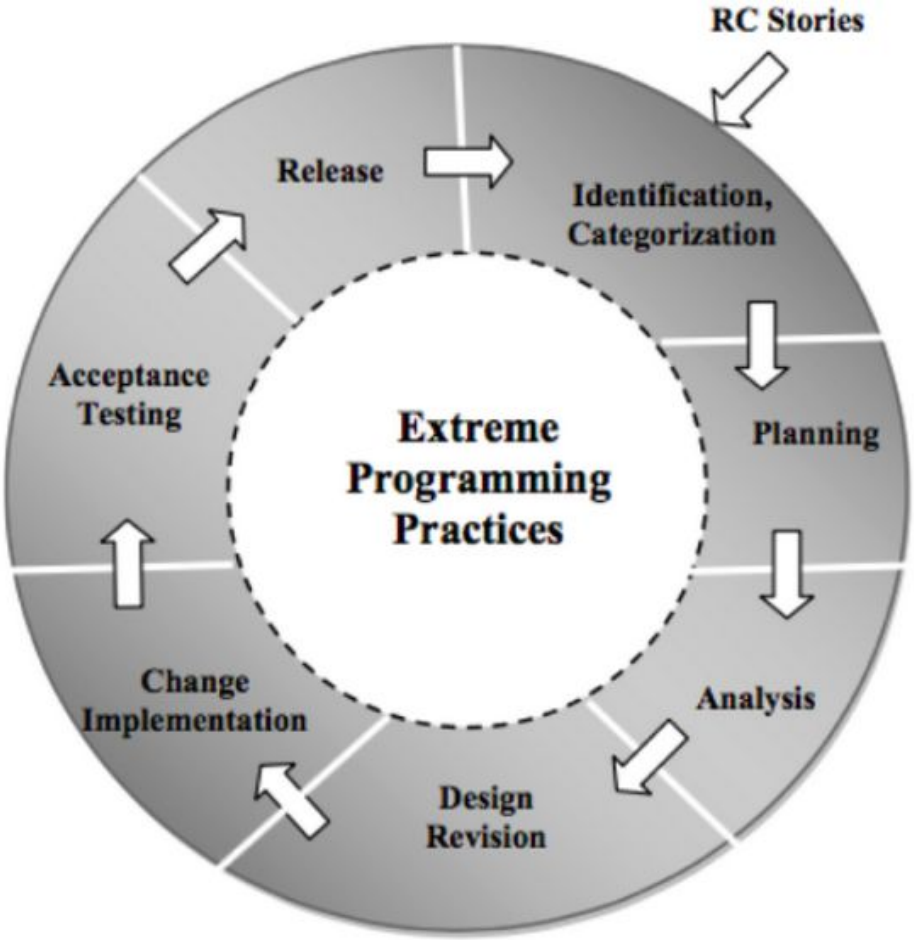
M. Galinium and N. Shahbaz, “Success factors model: Case studies in the migration of legacy systems to service oriented architecture,” in Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on, pp. 236 – 241, IEEE, 2012.

Migration as a Structured Process

The iterative model looks like a way to categorize maintenance activities into phases of migration.



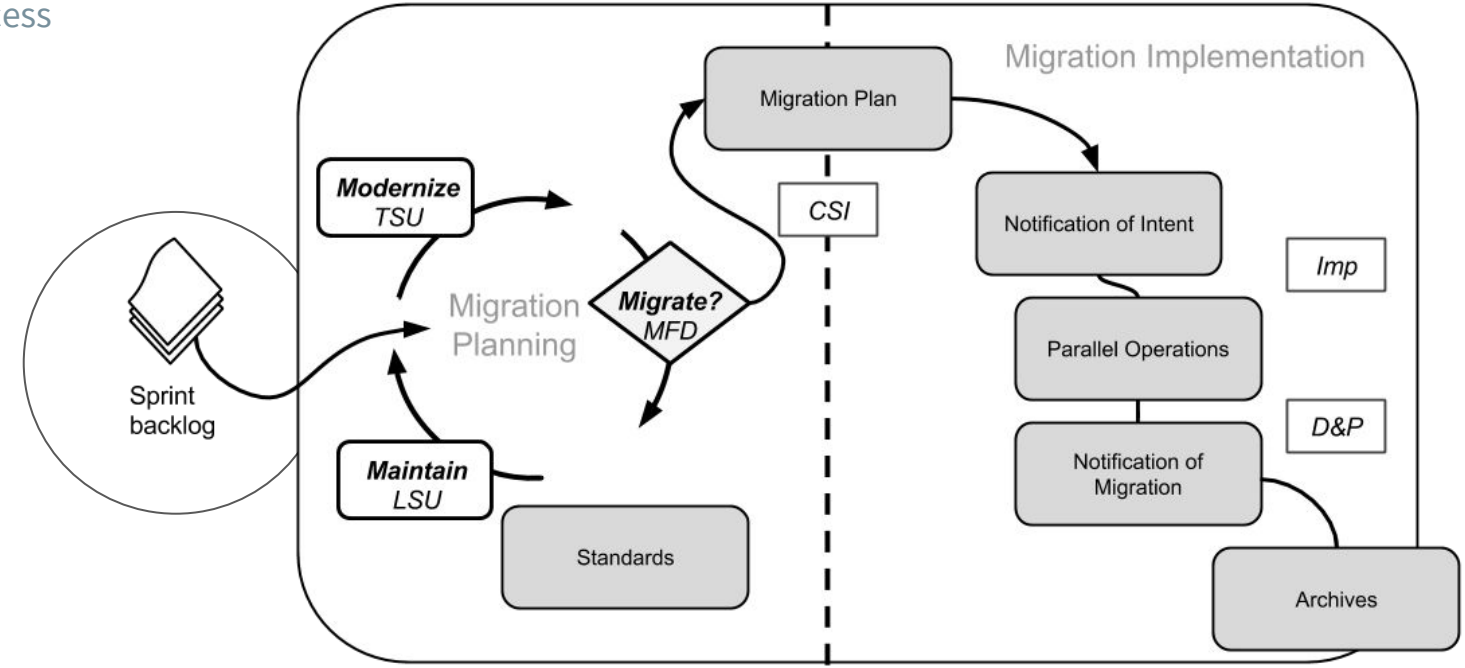
Extended Iterative Maintenance Lifecycle Using eXtreme Programming



Choudhari, J. & Suman, U. (2014), 'Extended iterative maintenance life cycle using eXtreme programming.', ACM SIGSOFT Software Engineering Notes 39 (1), 1-12.

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

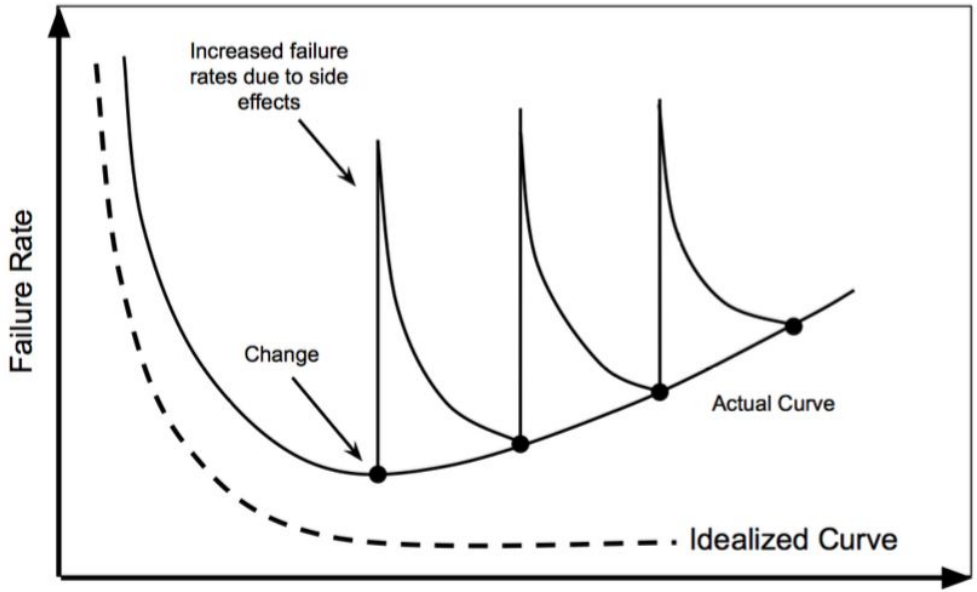
2 years ago:

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!*



Pressman, R. (1994), *Software Engineering, a Practitioner's Approach (European Edition)* , McGraw Hill , New York .

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*:

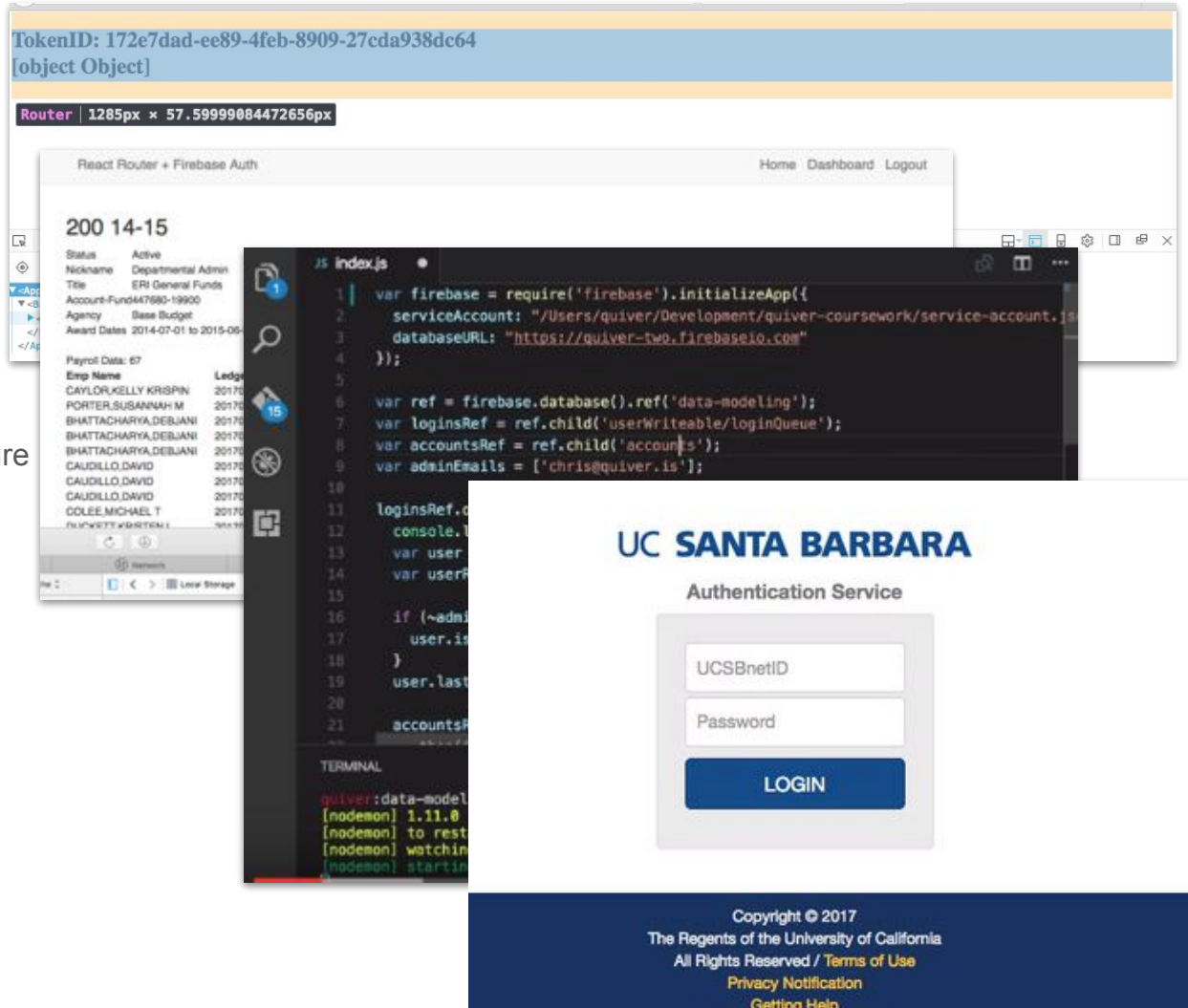
- Creating APIs to be consumed
- Building prototypes

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

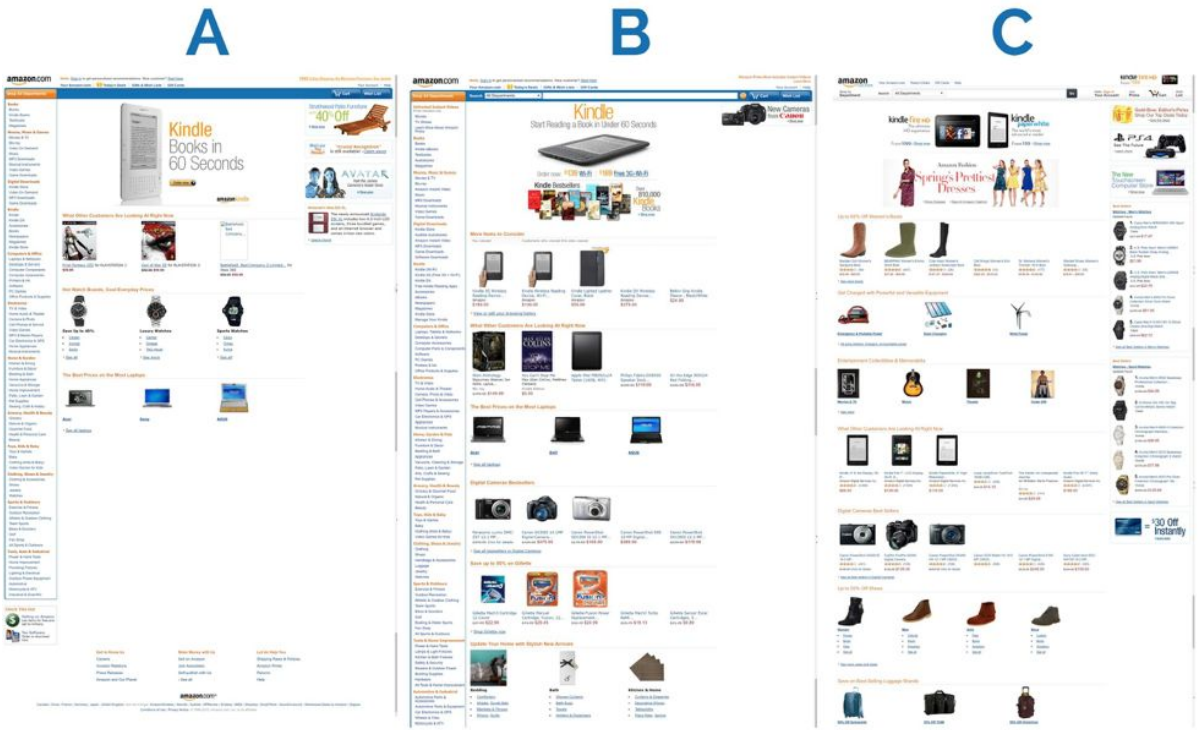


Image: https://www.bluecoda.com/blog/shift-3-year-redesign-amazon-model

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

image-source: http://justfunfacts.com/wp-content/uploads/2016/03/golden-gate-bridge-drawing.jpg

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

image-source: http://justfunfacts.com/wp-content/uploads/2016/03/golden-gate-bridge-drawing.jpg

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

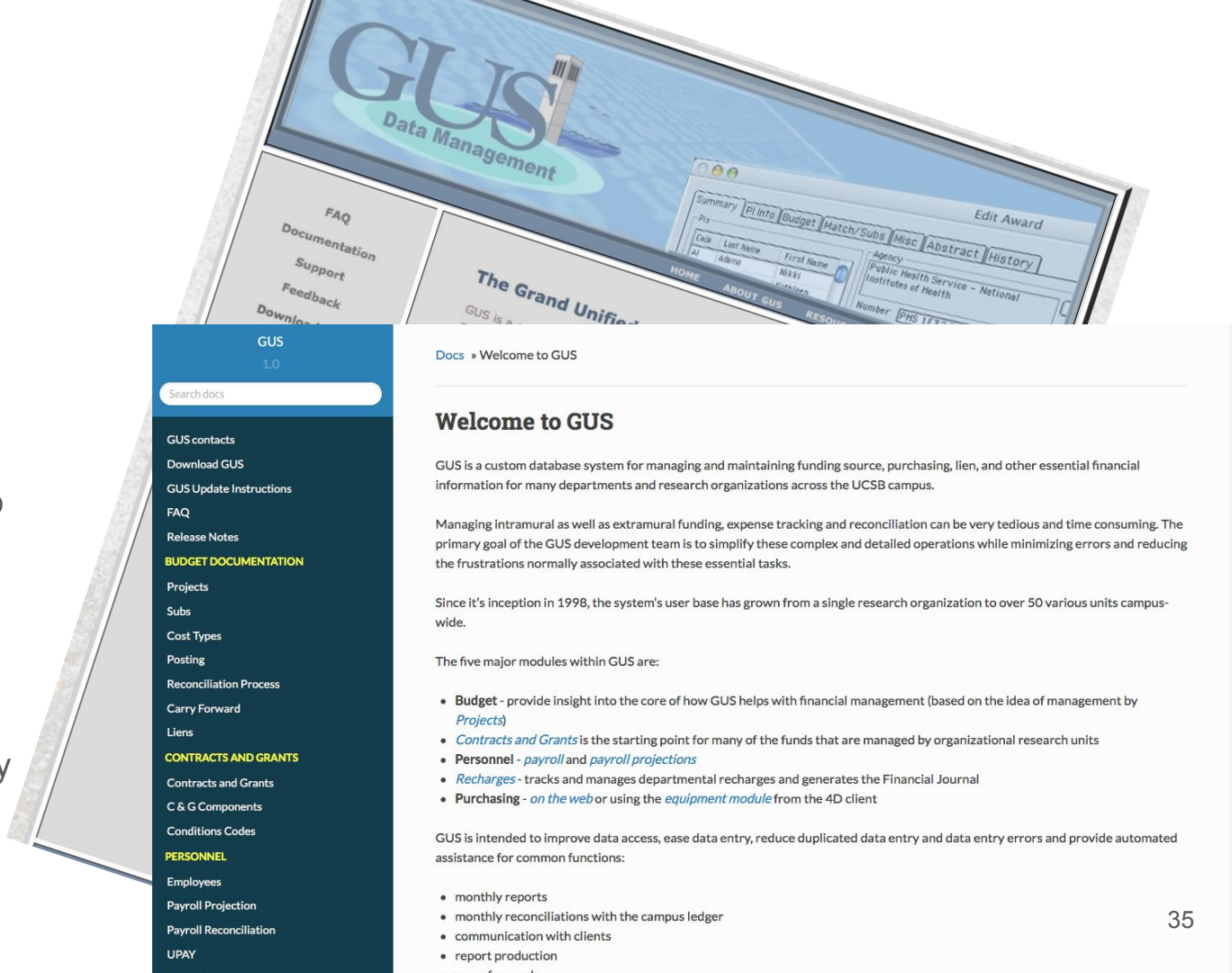
Image: https://inhabitat.com/wp-content/blogs.dir/1/files/2013/05/Golden-Gate-Bridge-Pavilion-Jensen-Architects-3.jpg

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...

Noah Spahn
noah.spahn@ucsb.edu