OLD CODE, NEW TRICKS
OR, HOW I LEARNED TO LOVE LEGACY CODE AND YOU CAN, TOO.

By M. Scott Ford
Founder, Corgibytes
@mscottford
WHY WAS THIS MY FAVORITE PROJECT?
WHY WAS I SO DISSATISFIED?

• Product vs. Services?
• Desktop vs. Web?
• Startup vs. Enterprise?
• Hacker vs. Craftsman?
• Self Employed?
PRODUCT LIFE CYCLE

DEVELOPMENT

INTRODUCTION

MATURE

GROWTH
PRODUCT LIFE CYCLE

- DEVELOPMENT
- INTRODUCTION
- GROWTH
- MATURETY
- DECLINE
- OBSCURITY
PRODUCT LIFE CYCLE

DEVELOPMENT

INTRODUCTION

MAKING

GROWTH

MATURITY

DECLINE

OBSURITY
PRODUCT LIFE CYCLE

- DEVELOPMENT
- INTRODUCTION
- GROWTH
- MATURITY
- MAKING
- DECLINE
- OBSURITY

@mscottford
PRODUCT LIFE CYCLE

DEVELOPMENT

INTRODUCTION

MAKING

MENDING

GROWTH

MATUREY

DECLINE

OBSCURITY
MAKERS VS MENDERS
MAKERS

- Speed to Market
- Rapid Prototyping
- Minimum Viable Product
- Likes Experimenting
- Energized by Big Launch
MENDERS

- Repair Technical Debt
- Reduce Entropy
- Bug Fixes & Integrations
- Likes Stable & Steady
- Energized by Small Wins
DEVELOPER LANDSCAPE

HACKER  __________________  CRAFTSMAN
DEVELOPER LANDSCAPE

MAKING

HACKER

MENDING

CRAFTSMAN
DEVELOPER LANDSCAPE

MAKING

RAPID PROTOTYPING

HACKER

MENDING

CRAFTSMAN
DEVELOPER LANDSCAPE

MAKING

RAPID PROTOTYPING

SOLID

MENDING

HACKER

CRAFTSMAN

@mscottford
DEVELOPER LANDSCAPE

- HACKER
  - RAPID PROTOTYPING
  - FIRE FIGHTING
- CRAFTSMAN
  - SOLID
  - MENDING
DEVELOPER LANDSCAPE

HACKER  MAKING

RAPID PROTOTYPING  SOLID

FIRE FIGHTING  ????

MENDING  CRAFTSMAN

@mscottford
DEVELOPER LANDSCAPE

- Rapid Prototyping
- Solid
- Fire Fighting
- Software Remodeling
SOFTWARE REMODELING

activities that reduce software entropy
SOFTWARE ENTROPY
SOFTWARE ENTROPY

1. Using software means continuous modification.

2. Modification increases complexity, unless a conscious effort is made.


@mscottford
The longer it’s been since the last time a program has been modified, the more expensive it will be to make the next modification.
BULLDOZE VS REMODEL
WHEN TO REMODEL?
WHEN TO REMODEL?

FEATURES
YOU HAVE

@mscottford
WHEN TO REMODEL?

FEATURES YOU HAVE

FEATURES YOU NEED
WHEN TO REMODEL?

- FEATURES YOU HAVE
- REMODELING JUSTIFICATION
- FEATURES YOU NEED
REMODELING PRINCIPLES
REMODELING PRINCIPLES

Language Matters

@mscottford
How do you describe a LEGACY project?
WHAT COLOR IS YOUR FIELD?
• Talking positively about our work helps us feel good about it.

• Consider the following:

<table>
<thead>
<tr>
<th>NEGATIVE</th>
<th>POSITIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>Existing Code</td>
</tr>
<tr>
<td>Brownfield</td>
<td>Restore</td>
</tr>
<tr>
<td>Rescue</td>
<td>Remodel</td>
</tr>
<tr>
<td>Spaghetti Code</td>
<td>Revitalize</td>
</tr>
<tr>
<td>Antiquated</td>
<td>Vintage/Classic</td>
</tr>
</tbody>
</table>
LANGUAGE MATTERS

• Some terms used by the Agile community begin to break down when talking about existing projects.

• Consider the following:
  — Sprint vs. Iteration
  — Estimate vs. Forecast
REMODELING PRINCIPLES

Language Matters

Modern Techniques
USE MODERN TECHNIQUES

• Would a doctor treat you using only medical knowledge available in the year you were born?

• When working on an old house, would you limit yourself to only tools available the year it was built?

• You DO NOT have to limit yourself to old tools and techniques.
REMODELING PRINCIPLES

Language Matters
Modern Techniques
Respect The Past

@mscottford
RESPECT THE PAST

- Look at work as a form of software archaeology.
- Practice giving and receiving critiques.

“Regardless of what we discover, we understand and truly believe that everyone did the best job they could, given what they knew at the time, their skills and abilities, the resources available, and the situation at hand.”
How to give a good critique:

- Critique the code: never the author.
- Call attention to the good things, as well as opportunities for improvement.
- Make it clear that you are stating opinion and not fact.

Practice at [http://exercism.io](http://exercism.io)

- git blame

@mscottford
REMODELING PRINCIPLES

Language Matters
Modern Techniques
Respect The Past
Systems, Not Goals
SYSTEMS, NOT GOALS

- Goals work great for initial launch, not so well for maintenance.
  - Ex: Better to establish system where tests are added with every commit than set a “goal” of 100% test coverage.
  - Continuous Integration
  - Continuous Deployment
  - Automated Code Review
• Some of our favorite tools for creating maintenance systems:
  – Style Cops (*rubocop, FxCop*)
  – Linters (*jslint, csslint, xmlLint*)
  – Quality (*Code Climate, BitHound*)
  – Continuous Monitoring (*Honeybadger, Airbrake, New Relic*)
  – Continuous Deployment & Integration (*CodeShip, Jenkins, Travis CI, Circle CI*)
  – Chat-Ops (*Slack, Gitter, HipChat*)
QUESTIONS?