Accepting Application Ownership

UNIVERSITY OF NOTRE DAME

Jeremy Friesen

Digital Library Frameworks Specialist

University of Notre Dame

jfriesen@nd.edu

@jeremyfriesen

github.com/jeremyf

ndlib.github.io

Presentation at goo.gl/G6oN89

From my 2013 Open Repositories lightning talk.

Bad Luck Brian meme, created @ quickmeme.com

Let’s write some users stories to figure this out

“A Blank Page; A Fresh Start.” https://www.flickr.com/photos/rozabbotts

A Use Case

As a **developer** at Notre Dame

I want to enjoy working on applications

So that I have energy to keep working on applications

Well...

As a **strategist** at Notre Dame

I want to grow our institutional academic services

So that I can help meet the ever growing demands of the academy

Fine...

As a **developer** at Notre Dame

I want more developers on staff

So that we can meet the ever growing demands of the academy.

Actually…

As a **strategist** at Notre Dame

I want our existing institutional applications to have a low cost of ownership

So that I need not keep begging, pleading, and groveling for more resources from the higher ups

How About This…

As a **developer** at Notre Dame

I want to commit to owning the health of our applications

So that I can understand the understanding the state of our existing applications

A Use Case

As a **strategist** at Notre Dame

I want to hold you to that

So we can move on with this presentation

We created several apps, many of them Hydra applications.

These are some observations…

“the messy comp room: right” https://www.flickr.com/photos/blakespot

Where we were

The apps were expensive to maintain.

And brittle when we revisited them.

Used without permission from http://www.gluesociety.com/art/itwasntmeanttoendlikethis

What made these apps expensive? and brittle?

"Time is Money" https://www.flickr.com/photos/phphoto

What Made Them Expensive?

Slow Tests…

impede
iterative
change

“History of UK speed enforcement” https://www.flickr.com/photos/brizzlebornandbred

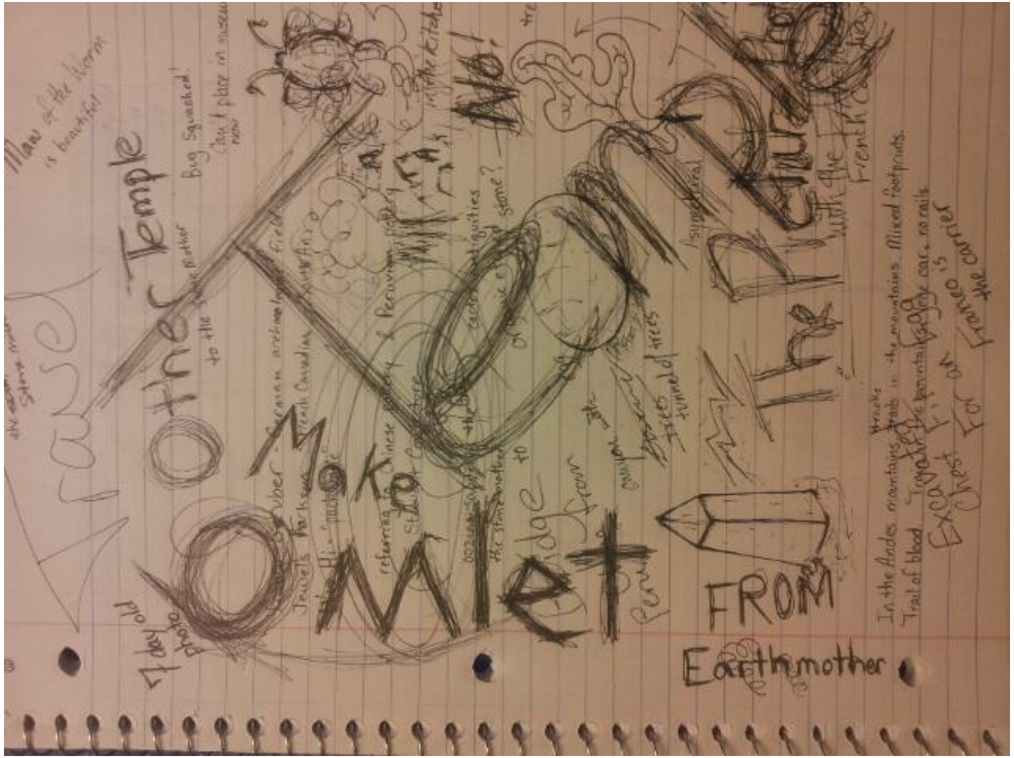
What Made Them Expensive?

Or worse…

Untested code…

Because we are blind to what we own

“Edge of the earth” https://www.flickr.com/photos/supercake



Inconsistent styles and idioms…

created higher code-orientation cost

GenCon 2012 “Cthulhu Dark” Character Journal by Jeremy Friesen

What Made Them Expensive?

New gets priority over old...

and time between changes is high

“Jealousy” https://www.flickr.com/photos/lukesaagi

What Made Them Expensive?

Ongoing development of dependencies…

creates an ever increasing upgrade cost

“Abandoned Boat Meets Abandoned Bridge on Summerland Key” https://www.flickr.com/photos/1stpix_diecast_dioramas

We needed to find a different way.

“1939 Church Road, St George, Bristol BS5” https://www.flickr.com/photos/brizzlebornandbred

Conjecture the First

If changes are slow or painful to make, the overall ownership cost will increase faster than the ownership benefit.

Therefore ensure that changes can be made quickly and painlessly.

Conjecture the Second

If setting aside our code and coming back to it later is expensive, then we should never step away from that code.

Maybe we can create tooling that revisits the code on our behalf.

Testing must be fast…

So we can keep coding

"History of UK speed enforcement" https://www.flickr.com/photos/brizzlebornandbred

All code must be tested…

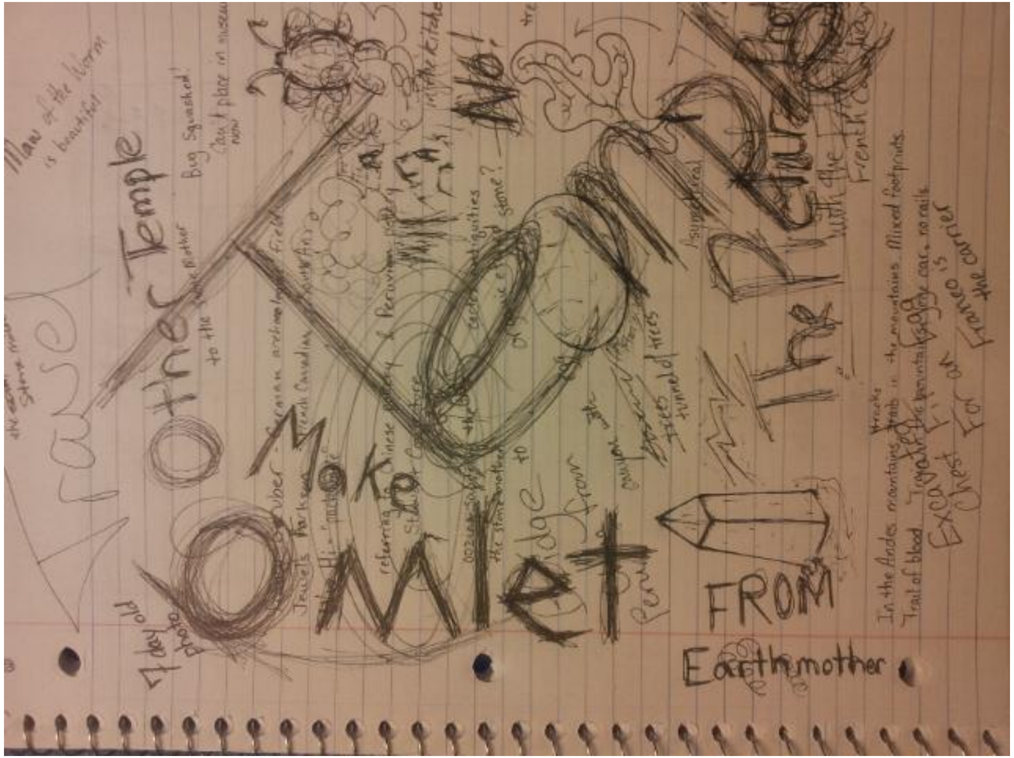
So we acknowledge the existence of the code…

And accept ownership of that code.

“Edge of the earth” https://www.flickr.com/photos/supercake

Adhere to a style guide…

So that orienting to the code is easier



GenCon 2012 "Cthulhu Dark" Character Journal by Jeremy Friesen

Lowering Ownership Cost

Drop in on your old projects to check up on them…

So you know if they are aging poorly

“Jealousy” https://www.flickr.com/photos/lukesaagi

Insulate against changes in your dependencies…

by adopting well known patterns for dependency survival

“Abandoned Boat Meets Abandoned Bridge on Summerland Key” https://www.flickr.com/photos/1stpix_diecast_dioramas

That's great…

but how do convert your philosophical diatribe to something actionable

"Amitabha" https://www.flickr.com/photos/h-k-d

Look at what you can control and set some goals

I'll use Sipity as an example

"001_365_01.01.2013" https://www.flickr.com/photos/plnaugle

On a developer's machine the test suite must complete in 30 seconds or less.

"History of UK speed enforcement" https://www.flickr.com/photos/brizzlebornandbred

Code coverage must be 100% or the build is considered broken

See git.io/hKRe for relevant commit

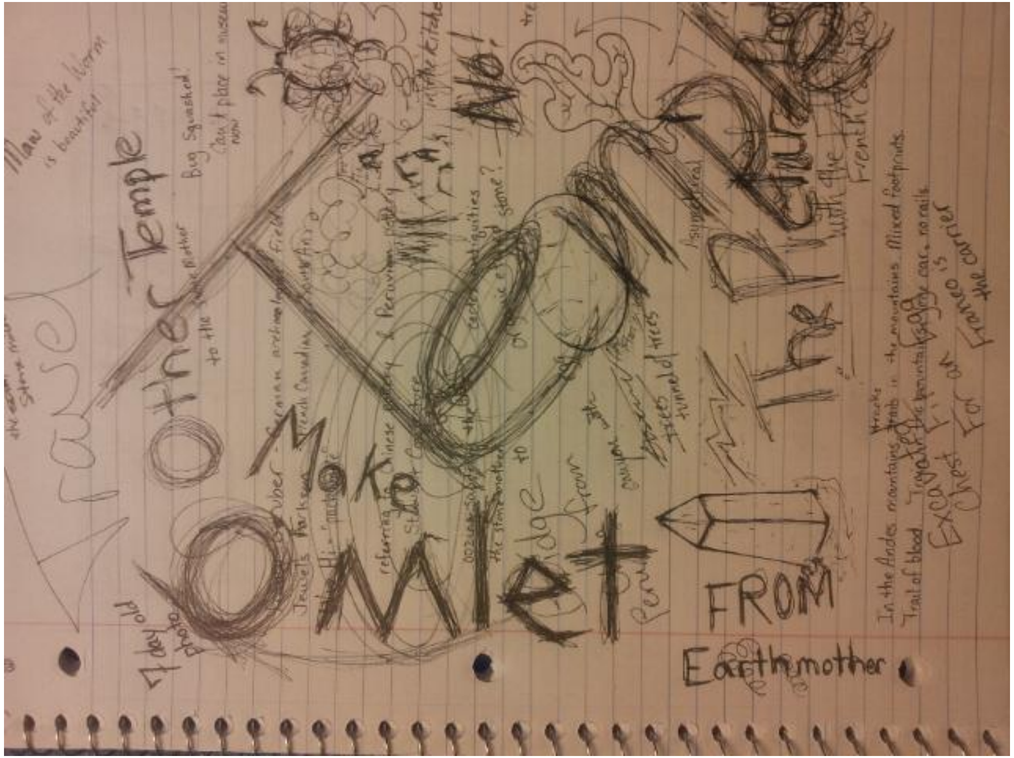
"Edge of the earth" https://www.flickr.com/photos/supercake

Less Philosophy…More Actionable

If Rubocop detects a violation the build is broken

Code review can focus on solutions not styles

github.com/bbatsov/rubocop



GenCon 2012 “Cthulhu Dark” Character Journal by Jeremy Friesen

Run an occassional "Enduring Commitment" sprint to revisit the old apps

"Jealousy" https://www.flickr.com/photos/lukesaagi

Know when your dependencies have changed…

by leaning on gemnasium.com

“Abandoned Boat Meets Abandoned Bridge on Summerland Key” https://www.flickr.com/photos/1stpix_diecast_dioramas

You might say "But I have legacy code"

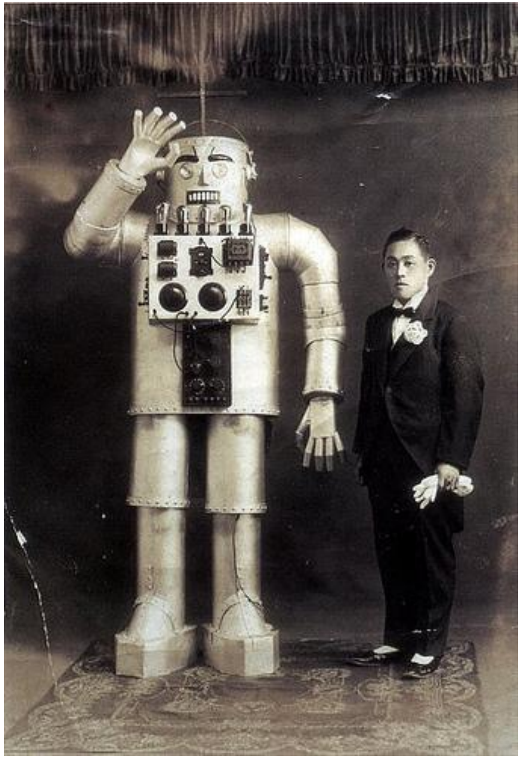
So do we

Set some S.M.A.R.T. goals for ownership

"001_365_01.01.2013" https://www.flickr.com/photos/plnaugle

Rubocop allows you to skip known violations, but not allow new ones:

```
rubocop --auto-gen-config
```



“japanese robot” https://www.flickr.com/photos/31472241

Determine your current code coverage

And don’t let it decrease

“Samhain tablecloth” https://www.flickr.com/photos/hexeimhollerbusch/

You know your application's state

And if you don't

Create a task to determine that state

"Magnifying Glass" https://www.flickr.com/photos/auntiep

Commit to Owning your App

We created the Commitment gem. As part of our test suite it runs:

- rubocop to enforce Ruby styles
- scss-lint to enforce SCSS styles
- jshintrb to enforce JS/Coffeescript styles
- simplecov to generate then enforce code coverage
- brakeman to enforce no known vulnerabilities

github.com/ndlib/commitment

Write a Commitment Contract

We are exploring the usage of the Ruby Contracts gem to define and clarify interfaces:

http://egonschiele.github.io/contracts.ruby/

Software development is complicated, and we should bring to bear any tooling that we can to help us with keeping our code healthy.

I recommend reading “Extreme Programming Explained” by Kent Beck and Cynthia Anders; It illuminates numerous development strategies and their pitfalls, yet combined they form a strong lattice of support.

A book on my “to read list”:

“Your Code as a Crime Scene: Use Forensic Techniques to Arrest Defects, Bottlenecks, and Bad Design in Your Programs” by Adam Tornhill

https://pragprog.com/book/atcrime/your-code-as-a-crime-scene

In conclusion

As a strategic developer

I want to, and as a professional should,

- Accept ownership of my apps
- Improve my ownership practices

So that I can rise to the challenges ahead of me

“001_365_01.01.2013” https://www.flickr.com/photos/plnaugle

Thank You

Jeremy Friesen

Digital Library Frameworks Specialist

University of Notre Dame

jfriesen@nd.edu

@jeremyfriesen

github.com/jeremyf

ndlib.github.io

Presentation at goo.gl/G6oN89