

PLANNING YOUR MIGRATION TO A FRAMEWORK

#PMFW, Process and Methodology

Brian Ghidinelli

www.ghidinelli.com

CF

CF.Objective()

Launch Day: May 3rd, 2008

It was a dark and stormy Saturday morning...

CF

CF.Objective()

Why switch to a framework?

- In-house framework hit ceiling
- Skills evolved; looking to continue learning
- Cost of bringing outside developers up to speed



Problems due to...

- Inexperienced developer?
 - Developing CF for 10+ years
- Original application not reliable?
 - In production for four years, CFC based
- Little or no testing?
 - Unit tests, Bug Parties



How I did the work

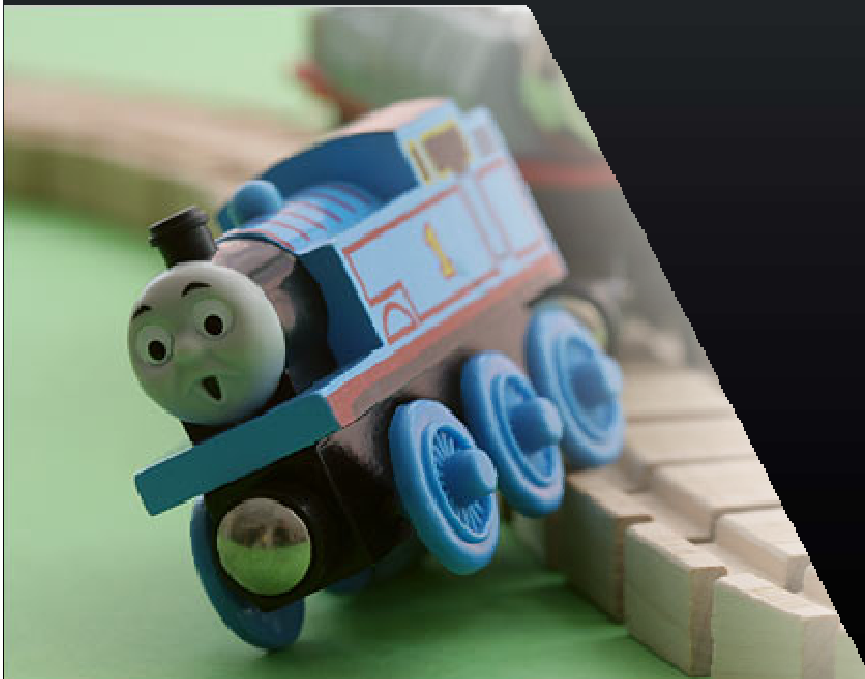
- Researched options, selected Transfer + Coldspring + Model-Glue
- Branched SVN, setup 2nd CF instance
- Converted API first and wrote tests
- Refactored as I learned to take advantage of framework

Where I finished

- Launched May 3rd (target: Jan 1)
- Managed not to lose a customer
- ~11 months effort in 8 month span
- Obtained solid understanding of OO, MVC and concrete knowledge of MG/CS/TR

What went wrong?

Or, how we turn a train wreck into a learning opportunity



Four sins of migration

1. Applications are more complex than you remember
2. Coding efficiency drops when learning
3. Misdirected or inadequate testing
4. “While you’re in there...”


#1: Underestimating application complexity

Or, don't trust your gut under any circumstances

Improving your estimates

- Treat migration like a consulting project
 - Your gut instincts are almost certainly wrong
- Inventory how you interact with database
- Look for user interface and custom tag libraries
- Identify external dependencies

Improving your estimates #2

- Look for duplicate or “stinky” code
- Document all non-standard views like Ajax, Flex, etc.
- Identify abstracted, re-usable code 
 - Even great code probably needs refactoring
- Beware of working solo!

#2: Historical returns are
no guarantee of future
productivity

Or, how to spend an entire day on
five lines of code

Regaining Productivity

- Budget 2x time to learn new paradigms for coding, debugging and make decisions
- Is existing code base a jump start?
- Obtain independent analysis of your estimates
- Include extra time for unit tests
- *Hire* a mentor to keep learning curve steep



#3: Inadequate testing

Or, the #!@#\$ site is broken!

How to get value from testing


- Decide how to test database early
- Decide on pragmatic test coverage %
- Focus on validating business logic
- Run at least one load test to avoid an “aha” moment
 - Selenium and OpenSTA support recording browser sessions – very straightforward!



#4: “While you’re in there...”

Or, how to *not* paint yourself into a tiny corner

Limiting scope creep

- Hold on to *backwards compatibility* at all costs; ability to run in parallel or roll back is priceless 
- You are switching to the framework to buy flexibility and speed so wait for it
- Migration itself will teach you a great deal; patience now will be rewarded later

Where am I today?

I'm still alive, but am I still in business?

Current status

- Test coverage improved but still < 50%
- Solved performance with DL360 G4s
- Accommodated 60% user growth
- Speed of innovation has increased
- Leverage MG/CS/TR communities as my “team”

Would I do it again?

No witty subtitle available at press
time

Q&A

When are you switching?

Why are you switching?