Lessons Learned in a Continuously Developing Service Oriented Architecture

Wednesday, November 20, 13

Welcome to “Lessons Learned in a Continuously Developing Service Oriented Architecture!”

This talk is not about SOA, it’s not going to get super technical either... the goal of this talk is to talk about lessons learned in constant iteration towards an SOA; with a little SOA sprinkled in at the end.

Lesson 1

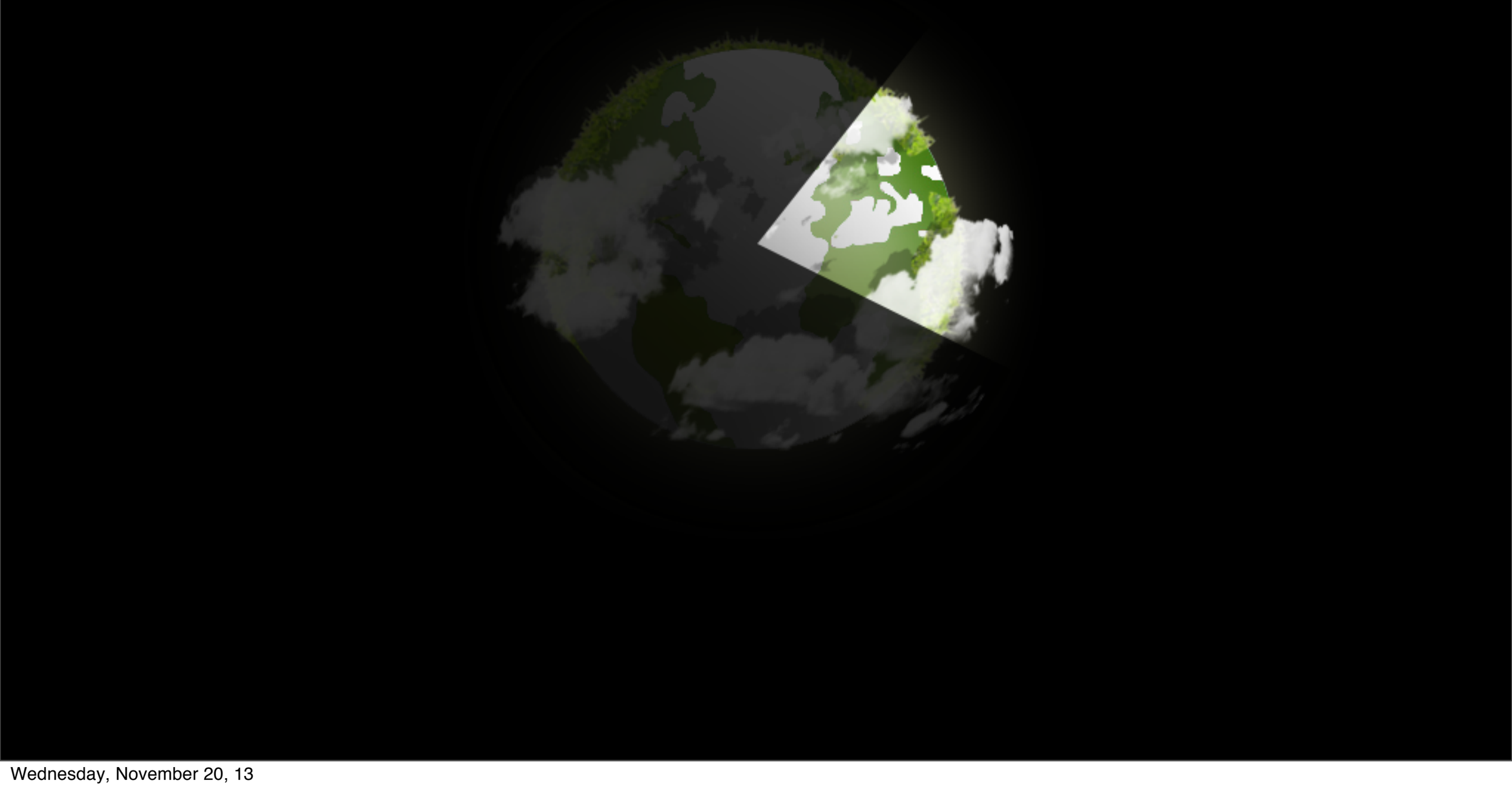
Approach Problems Honestly

Assess the situation for what it is.

Knowledge.

...things I know

...things **I know** I don't know



...things I **don't know** I don't know

The first lesson we learned is to accept the fact that our first attempt at new problem areas or domains will ALWAYS be naive to an extent and will never be the best we're capable of. As soon as we accept that, we can go on to function as developers should.

You have to be willing to make the best decisions you can with the information you have at the time and you leave yourself open to make improvements later.

Assess the Situation

Summary of the current organizational environment at NC State.

We have a central IT unit on campus that is responsible for core services such as email, ERP, human resources, student information systems, etc.

Most anything below the enterprise level is delegated to be picked up by colleges, departments, or not at all.

So you'll find pods of developers scattered throughout our University that may be in complete isolation or possibly lightly working with other groups.

Tendencies of Small Teams

This may not speak to every small team. Certainly, smaller development teams are more agile to change.

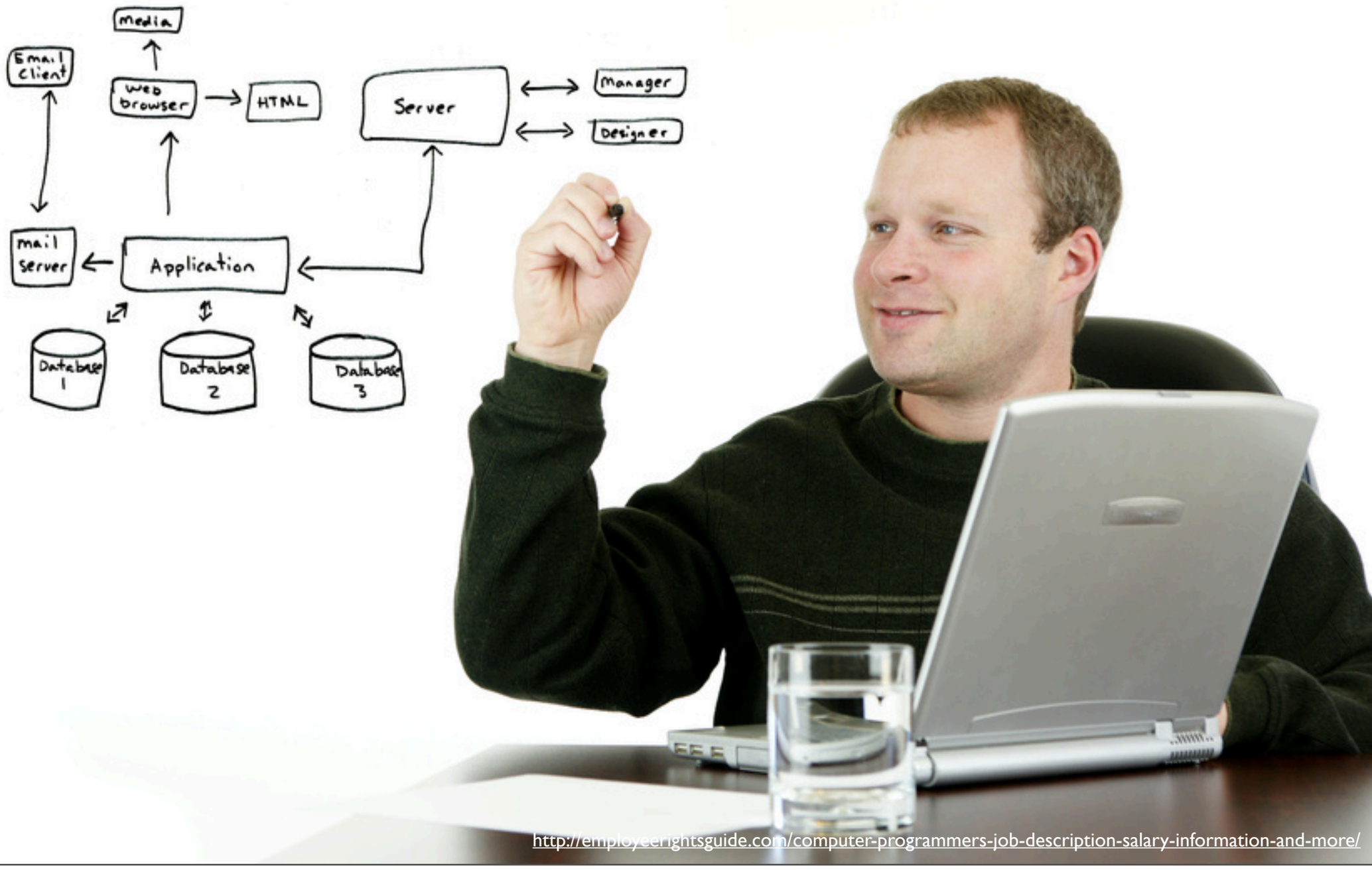
These are primarily observations I've made about teams I've been a part of.

Sub-optimal Skillset Distribution

Everyone's expected to be a DBA, Designer, Developer, and Business Analyst.

Small teams can also begin to transform into a group of independent contractors that sometimes interact on shared tasks. Silo-ed developers don't grow as rapidly as collaborative developers. There just isn't an opportunity for cross-training.

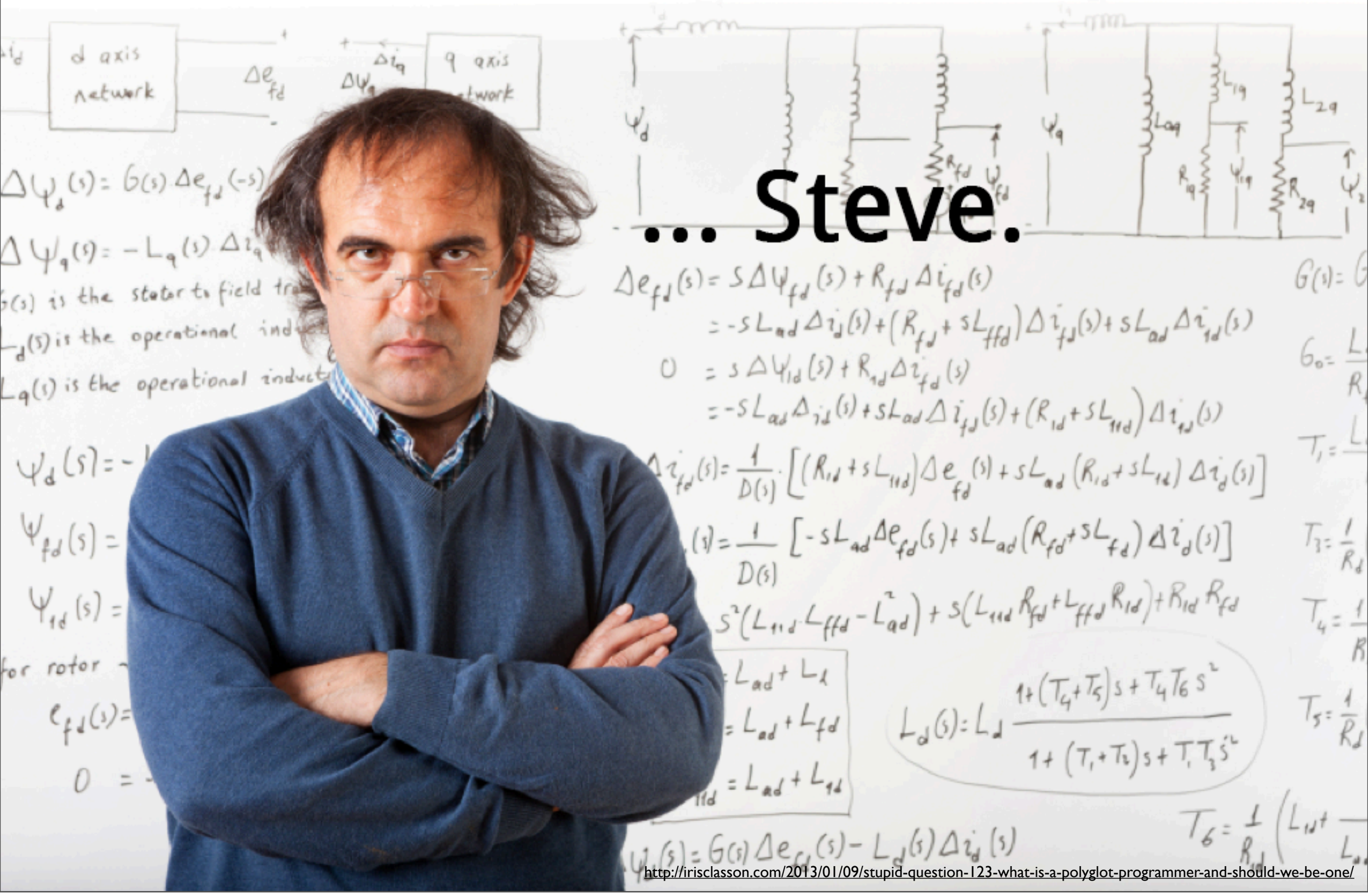
Alex



http://employeerightsguide.com/computer-programmers-job-description-salary-information-and-more/

Alex has just joined our mock group. Alex has been immediately assigned 3 projects from an employee who has recently left the organization. Alex doesn't know much about these projects and nobody else seems to have the full picture.

This means lots of meetings between Alex and the stakeholders these applications support.



Wednesday, November 20, 13

Steve has been programming since before Alex was born.

Steve is the group's senior developer and, as such, as accrued the majority of the team's domain expertise in the products they support. Steve is a critical member of this group's success and if he ever left, he'd be taking most of the group's experience with him; possibly crippling the group.

http://faildesk.net/2012/08/24/it-horror-stories-scumbag-programmers/

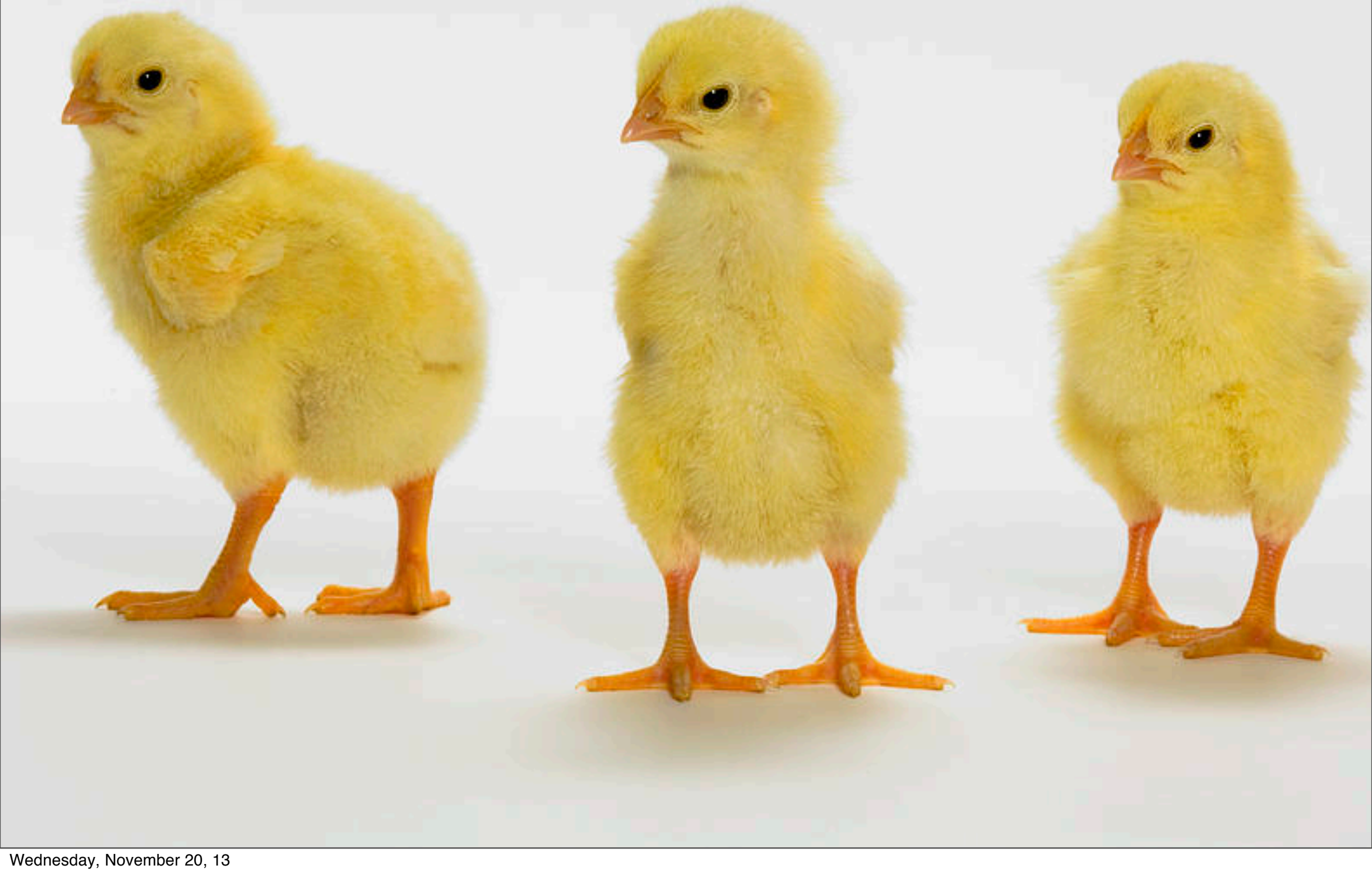
Wednesday, November 20, 13

Alex is replacing this guy.

Guy spent 3 years of “startup hopscotch” before he joined the group.

The REAL danger with this guy is that he can “talk the talk” with management. This particular person KNOWS that he’s writing bad code and that he’s leaving nails for the next developer to step on.

The fact that he’s fast and “gets the job done” makes him a stealth assassin for management and in the wake of his departure, Alex and Steve begin to smell the smells left behind.



Then, you have the baby chicks. These might be your student developers or interns of some type.

They don’t always know what they’re doing isn’t the “best practice”, but they have best intentions and you can’t get upset with them because best intentions are exactly what you want in a group.

If you put baby chicks on a keyboard and expect Twitter to pop out, then it’s not the chicks’ problem when they fail.

Little Long-term Vision

Smaller pods of developers don't always have the long-term 10-yr plan to think about.

This may cause them to focus on the implementation at hand without a lot of forethought. This will scale to a point, relative to the amount of resources you have.

Focusing solely on day-to-day tasks can give a development team tunnel vision and causes them to miss the important breakthroughs; to miss the opportunities for greater insight.

http://postgradproblems.com/8-things-you-didnt-know-about-the-mighty-morphin-power-rangers/

Wednesday, November 20, 13

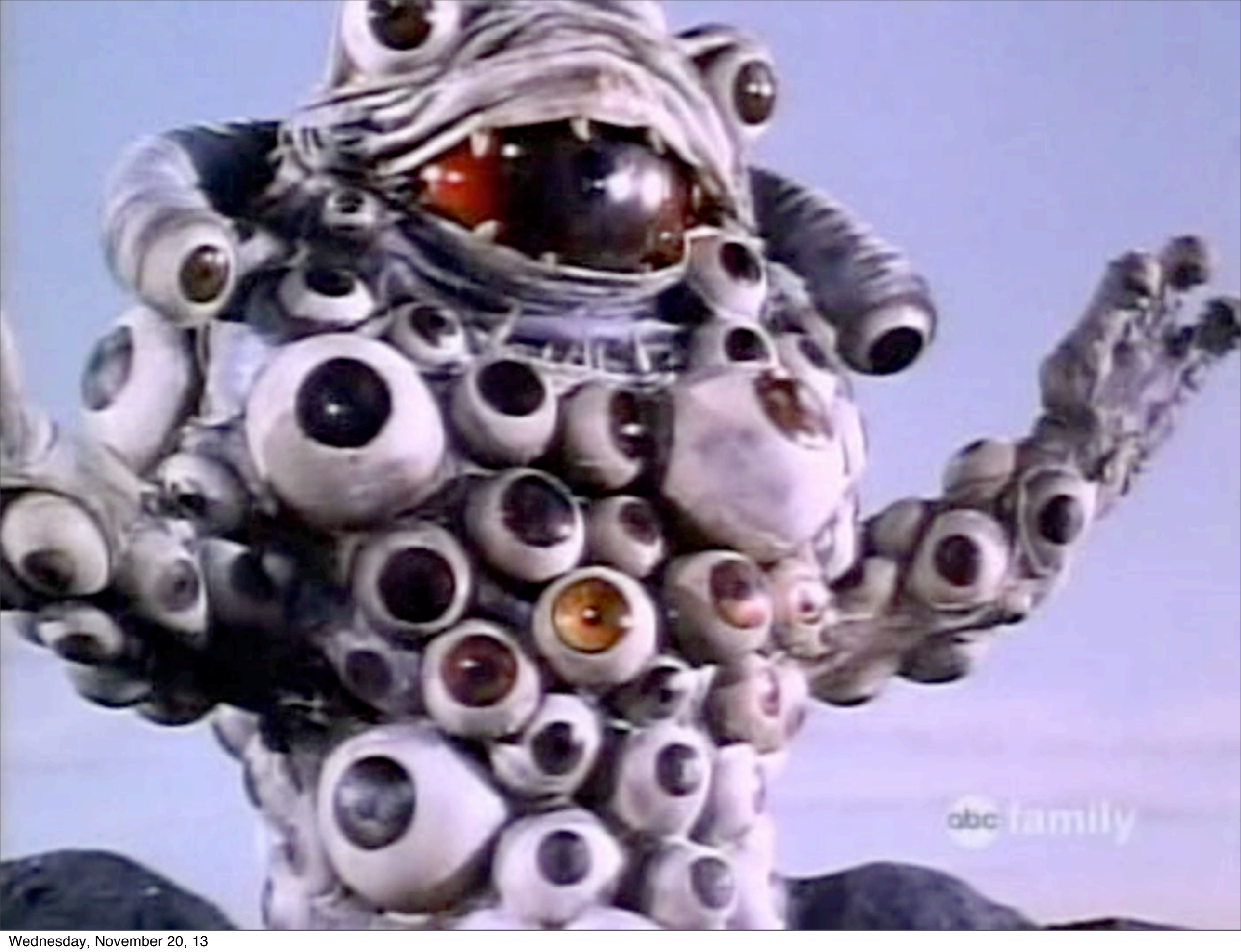
Take for instance, the Power Rangers. You could not find a more motivated group. However, their downfall is that they only see 30 minutes into the future.

Every episode of the power rangers was the same...

The show starts...

The bad guys send in the putty patrol and a monster...

They fight the putty patrol and defeat the monster...



Monster grows to some insurmountable size... and...

Wednesday, November 20, 13

The Power Rangers assemble the Megazord...

The problem is that over time, the Megazords collect rust and you're 20 seasons into Power Rangers: Jungle Jurassic Park and nobody knows what's going on anymore.

This is how unaccounted development and proliferation of web applications can become a problem. If you approach every problem with a new web-app (no matter how similar the business case is) you WILL end up with a pile of un-maintainable applications that a new developer is not going to get anything from and want to rewrite applications constantly.

This is NOT sustainable. If your group is headed down this path, STOP IT!

Sometimes Poor Overall Organization

Developers like all of the things.

Project Management

Project management on small teams can look like a post-it blizzard, email, issue trackers, stone tablets, and more. The key is to find something that truly works and buy into it completely.

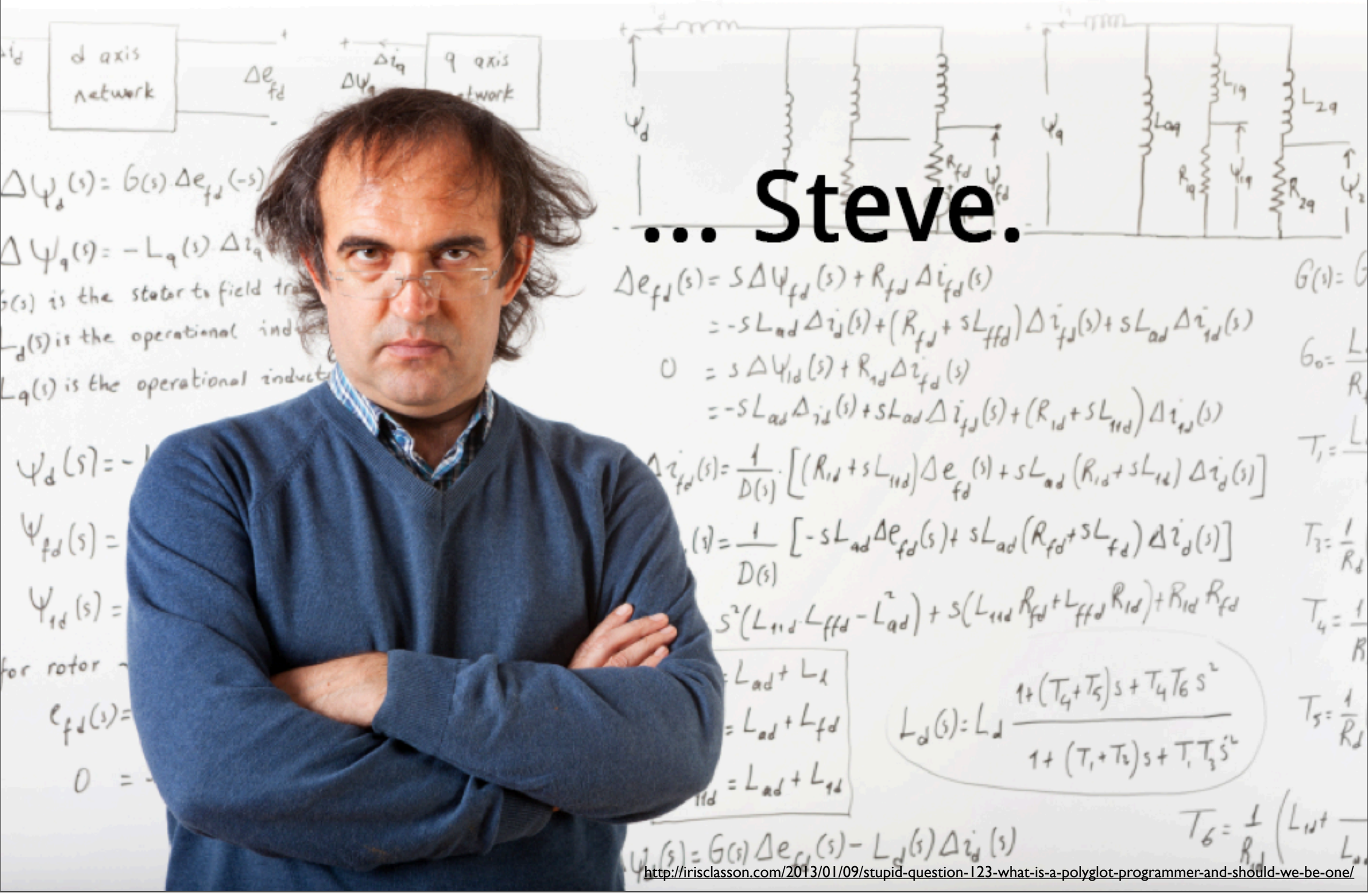

Trello
asana:


github:issues
Atlassian
JIRA

Be deliberate in tech choices.

We need to be able to say WHY we choose the technologies we do. This are ALL legitimate and well supported products and we NEED to be able to make informed decisions on why we use MySQL over redis, Laravel over ZF2, AngularJS over Backbone. Simply “picking one” demonstrates a naive approach to problem solving but, that said, CAN be useful for a young development team in creating stability in development environment.

Why use a phillips screwdriver over a flat-head?



Wednesday, November 20, 13

The reality of the situation is that unaccounted development will create difficult-to-maintain applications that next-generation developers are going to want to rewrite instead of gain insight from.

Why are we here today?

My primary goal is to discuss how our small team is working towards a Service Oriented Architecture and the lessons we're learning in the process that we believe are allowing us to function more efficiently in a resource-constrained environment.

Lesson 2

There is no silver bullet

Unless you make your own.

"There is NO right answer. There is NO silver bullet. However, there is ALWAYS a set of right answers to be selected from. You just have to get them out."

When we started this process, nobody really knew what the end result was going to be. However, stalling for THE right answer was an obvious blocker to productivity and would cost a lot of time and money in practice.

We had to recognize early-on that flexibility as a development group was going to be key to our success.

So, what was our problem?

We didn't know there was a problem.

For a long time, we didn't see any problem in the way we were managing day-to-day development. The job was getting done, so we thought no better.

As the team grew in responsibility (not necessarily resources), development velocity began to slow down and tasks that should by all rights take a few hours are taking days.

Projects are lingering for a long time and failing slowwwwly.

We found duplication everywhere.

We identified several important service offerings where there were multiple implementations solving parts of the same problem.

Content Management

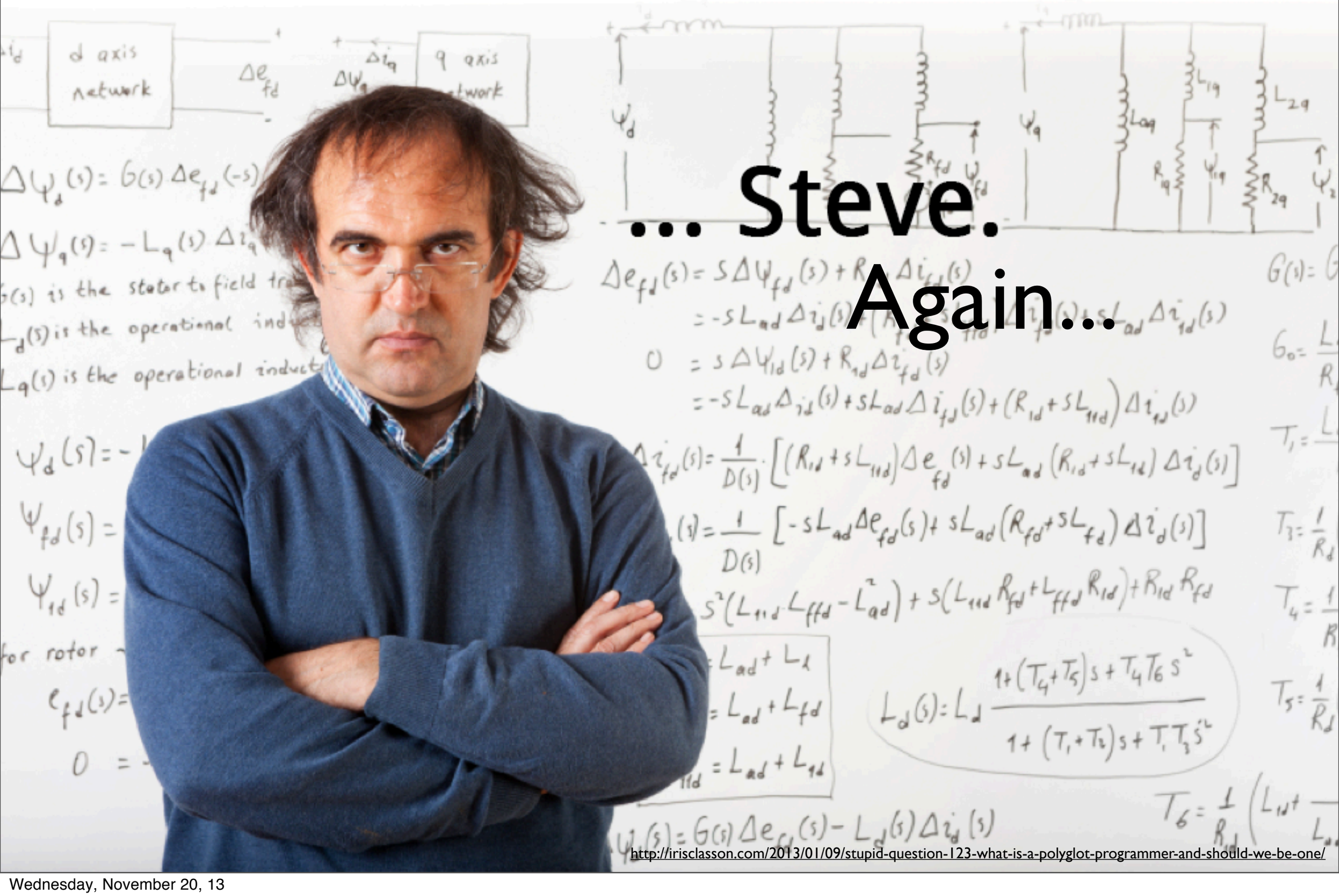
End-users are allowed to maintain HTML sites using whatever client they choose:

* SSH + VIM ... yep.
* Notepad++ & SFTP (Expandrive)
* Dreamweaver
* Contributes

All this means that we're also expected to support all of these technologies JUST for content management.

Application Development

* Access to directory information
* Large file storage and distribution
* Payment Card Industry Interfaces
* Multiple smaller applications that provide basic data collection and reporting



Wednesday, November 20, 13

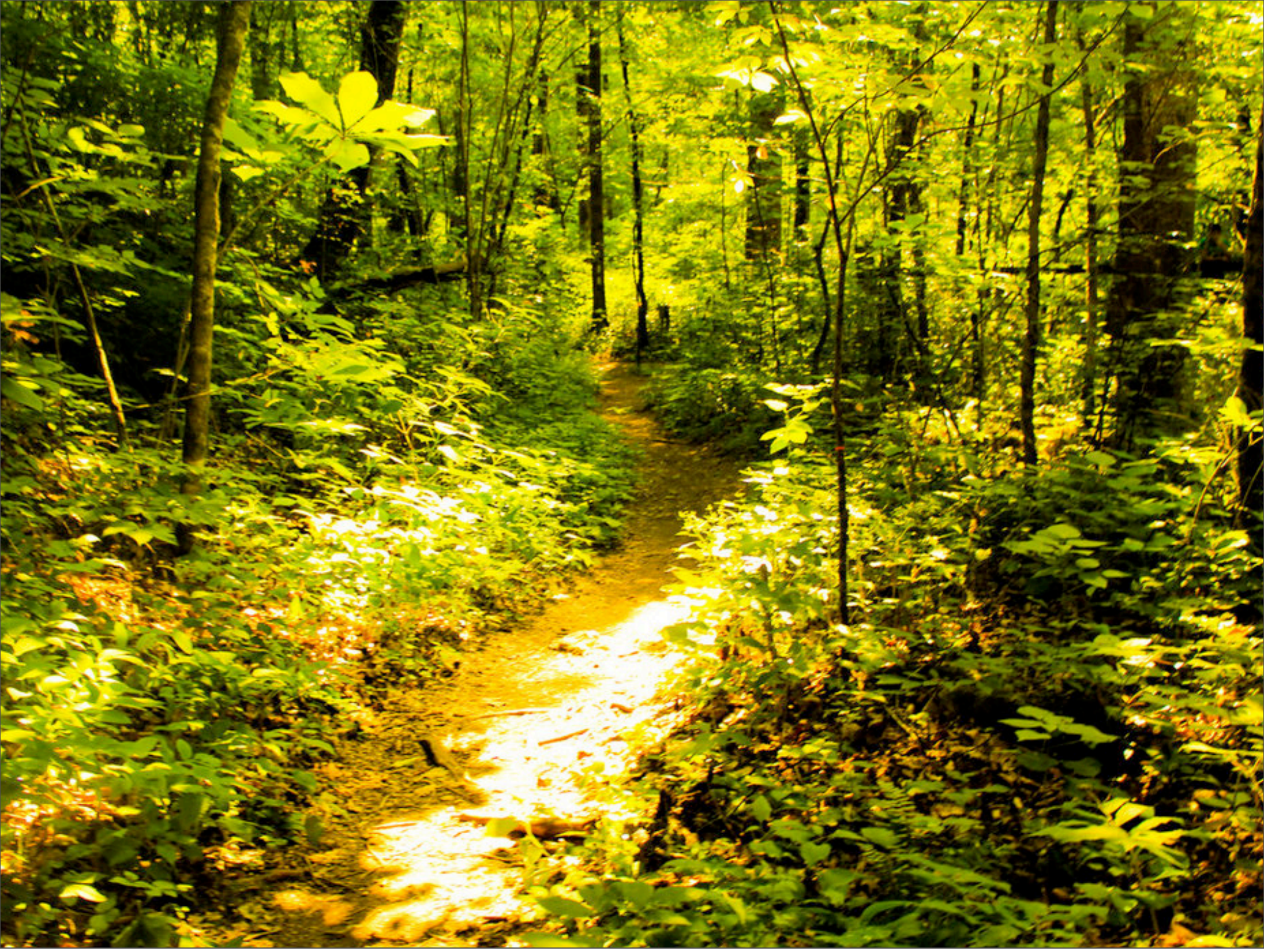
---We were in risk of losing our domain expertise---

Stability and **Agility**

We want to find stability in the areas we know how to solve so that we can spend resources on addressing future goals in a more agile fashion. This will lead to breakthroughs that improve our entire application suite.

We want a greater sense of stability in the maintenance of “legacy” applications.

Feather Analogy; Open-ness to try new methods to arrive at solid solutions.



So we’re down the right path!

We put a halt on feature development across many of our applications and committed to no new development for the duration of our infrastructure planning and migration.

This gave us the affordance to focus on the long-term problem; mitigating the distraction of day-to-day fires.

WordPress

Content-based Service Offering consolidated in Wordpress

Most of our content owners / managers are not technical staff. Because of that, Wordpress has a lower barrier to entry than other CMSs we assessed.

For basic content needs, Wordpress has been great so far. However, there is the issue of tying dynamic data sources into our Wordpress-based sites. This influenced part of our decision toward an SOA.

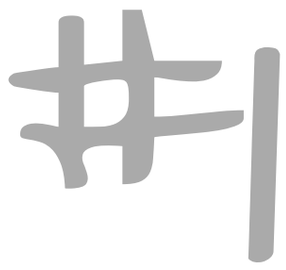
Lesson 3

Set some ground rules

Develop a set of best practices for your team.

Develop a set of best practices to form an organic record of how you do software development.

Apply them in new development and be prepared to constantly iterate on the things that just aren’t fitting.



Make iterative advances.

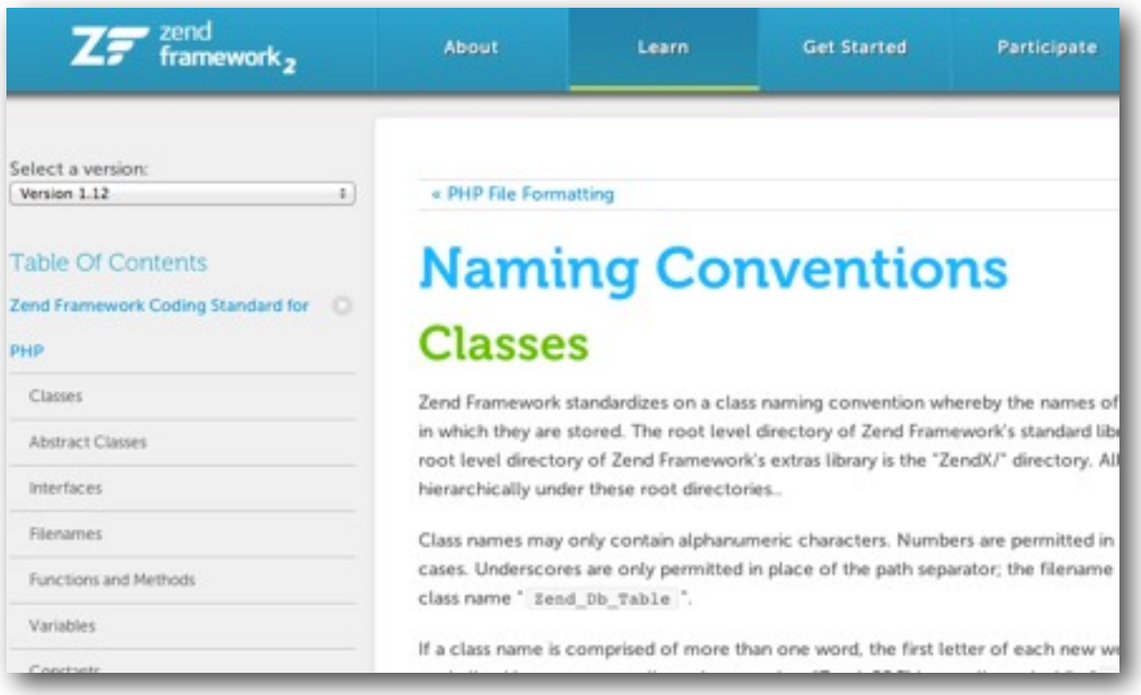
Make iterative advances in development projects, applying best practices as you go; always with an eye towards the next iteration.

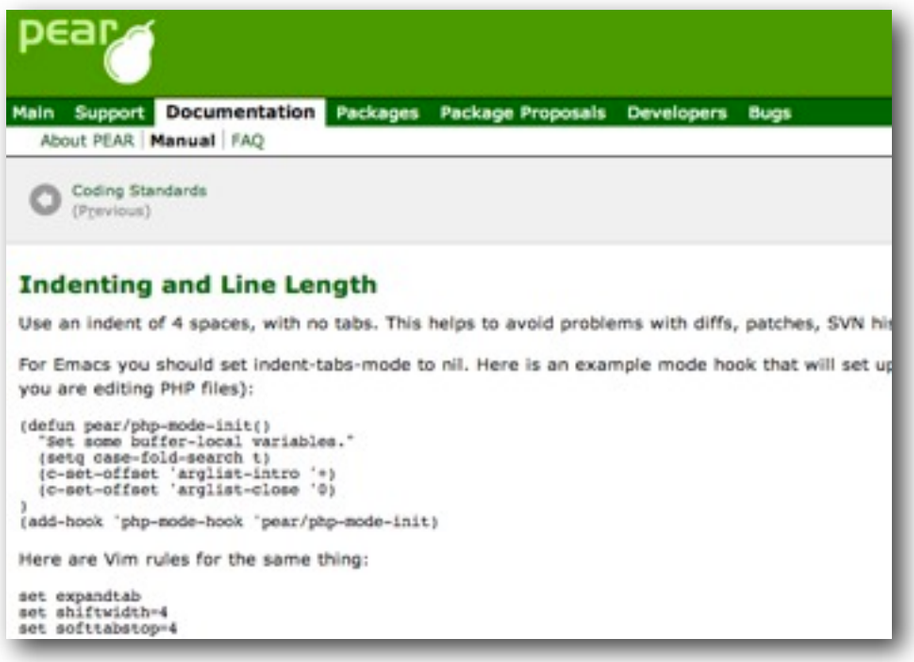
#2 Don't expect drastic change.

Do not try to change the world overnight. There are real habits built up that will take time to change.

#3 Adopt a Coding Standard

A coding standard sets the baseline for what can be expected as far as code quality. Applications suddenly begin to seem like they were written by a team, rather than a group of individual contractors.





Pick a standard or write your own flavor of an existing. It doesn't matter.

We went with PSR (PHPFIG), which is an RFC style standard that looks something like ... (next slide)

4.3. Methods

Visibility MUST be declared on all methods.

Method names SHOULD NOT be prefixed with a single underscore to indicate protected or private visibility.

Method names MUST NOT be declared with a space after the method name. The opening brace MUST go on its own line, and the closing brace MUST go on the next line following the body. There MUST NOT be a space after the opening parenthesis, and there MUST NOT be a space before the closing parenthesis.

A method declaration looks like the following. Note the placement of parentheses, commas, spaces, and braces:

```php
<?php
namespace Vendor\Package;

class ClassName
{
    public function fooBarBaz($arg1, &$arg2, $arg3 = [])
    {
        // method body
    }
}
```

5.1. `if`, `elseif`, `else`

An `if` structure looks like the following. Note the placement of parentheses, spaces, and braces; and that `else` and `elseif` are on the same line as the closing brace from the earlier body.

```
<?php
if ($expr1) {
    // if body
} elseif ($expr2) {
    // elseif body
} else {
    // else body;
}
```

#30 Adopt a web services Standard.

Adopt a {blank} standard. Codify how your organization DOES what it does.

We knew that we were going to be writing a bunch of web services that represented centralized service offerings we were providing.

We also had a goal of providing these services for consumption by other groups on campus. Soooo...

We didn't answer this question alone. We assembled a cross-organizational working group to talk about the problem and establish a standard for web services on campus.

First order of business... REST or SOAP.

NC State Interop Group is formed.

RESTful URLs and Actions

The constraints of REST describe the segregation of logical resources within a web service. These resources are acted upon using HTTP requests using the standard HTTP methods / verbs: GET, POST, PUT, PATCH, and DELETE.

Resource Naming

- Resource names MUST be represented as nouns.
- Resource names MUST use their plural form.
- Resource fields MUST use `snake_case` in all RESTful interactions. **Note: This excludes libraries that consume the service. Libraries are subject to the coding standards for the language they're written in.**
- There is no requirement for resources to map one-to-one with underlying models. The idea is to abstract away technical detail from the API consumer.

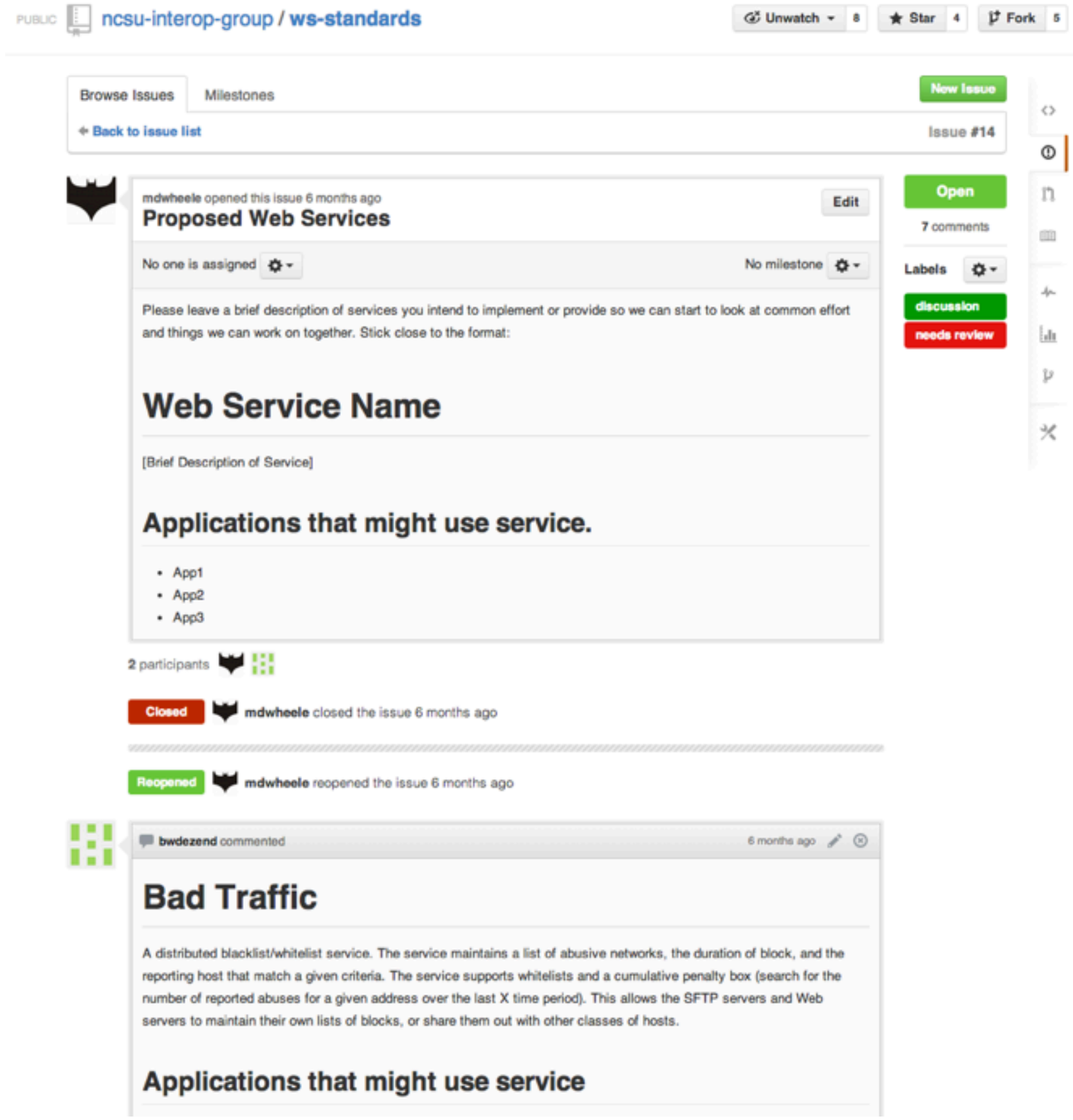
Filtering

- APIs MUST use a unique query parameter for every resource field that implements filtering.
- Filters MUST be mapped to the query string and MUST NOT be included in the resource mapping.
- Multiple filters on one resource field MUST be separated by comma.

Examples

- `GET students?major=mae` - Retrieves a list of MAE students.
- `GET students?major=mae,csc` - Retrieves a list of MAE and CSC students.

Immediate Benefits



We have a group of people to contribute towards the evolution of this standard and to start creating some really awesome services that no single group would realistically be able to provide.

Oops... I think we broke it.

It was maybe 48 hours after our group first approved WSSR-1 and we were writing a service that had a completely valid use-case to bend the standard.

That's okay. It means we didn't have a full understanding of the problem set up-front.

We expected this, remember?

#4 Establish a "Definition of Done"

"A common understanding of when a development task is finished."

This can transform a team from claiming best practices have prohibitive overhead costs. You simply accept these new practices as part of the "definition of done".

For example, for a feature to be complete, it must: be specified and documented, be coded, unit & acceptance tested, reviewed by the team, and then merged into the repository. Only then can it be marked complete.

Notice there is no backlog item to write tests or meeting to do a code review. It's simply accepted as part of the feature itself. More about this later in "Testing".

* Should only require tasks under teams control
* Avoid having too many states; to do, doing, and done are enough
* Avoid tasks that are susceptible to getting “stuck”

Following these guidelines, you also pear down tasks into manageable, distributable chunks.

Keep in mind, our team’s DoD may not fit your team. In all of this, you have to find something that fits the team that you have.

Lesson 4

Buy into established practices

Unless you make your own.

Version Control

Version control is a common practice in development teams today. 10 years ago, only larger software companies consistently used version control systems. I was at a conference earlier this year and the question asked wasn't who IS doing version control, but who WASNT (3 hands in a room of 200+)

Somewhere along the line, we decided that the rules of software development did not apply to what we were producing.

Version control is, in my opinion, the single-most impactful service we've bought into.

* Source control lays the foundation for best practice and is the beginning of a multiplicative return on investment
* Serves as an authoritative record to the state of an application and creates an open point of reference for a development team
* It's not **just** about being able to rollback

http://git-scm.com/book

So how do I get started?

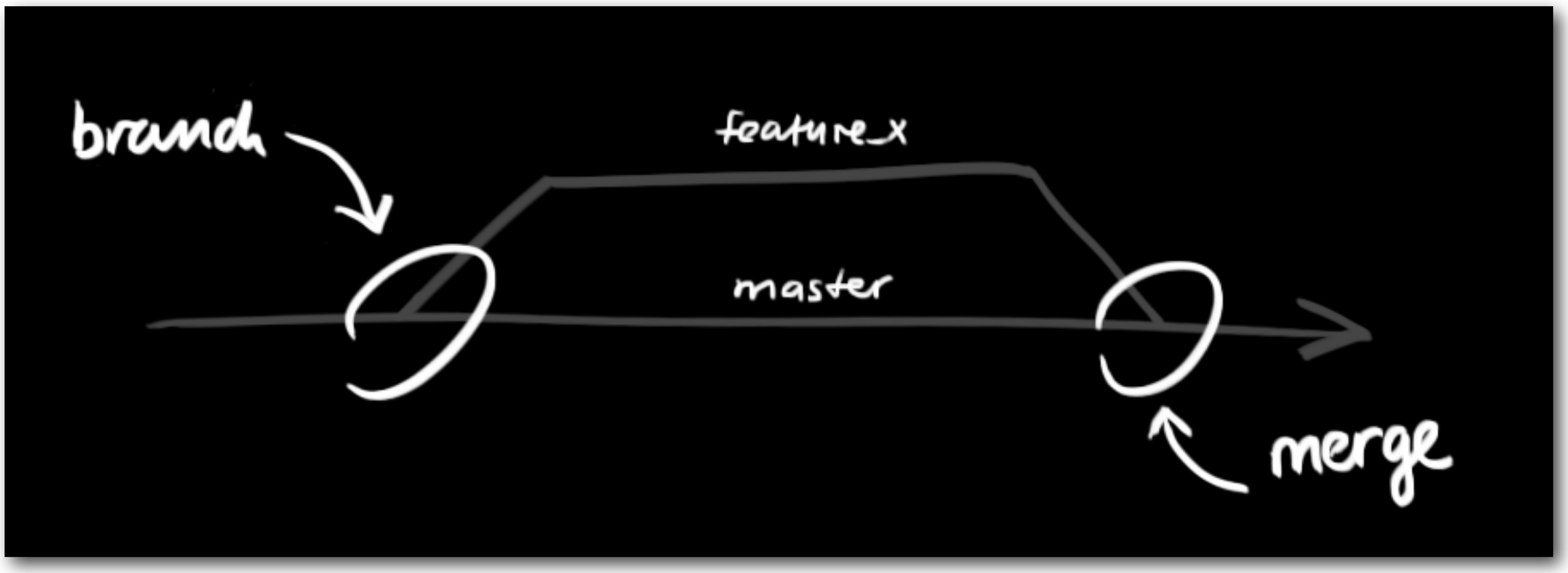
Git is free. It comes installed with OSX and most Linux distributions. Git is also available for Windows through a cyg-win'y terminal called Git-bash.

Command-line tools are scary! Get over it. While there ARE GUI applications for git, github, and others; you will get more out of learning the CLI.


github
SOCIAL CODING


Atlassian
Bitbucket

A better development workflow appears.



A starter workflow with git might look something like this:

1) A ‘master’ branch of development
2) When a feature is to be implemented, a new branch called ‘feature-awesome’ is created
3) When the feature is completed (DoD), the branch is merged back into ‘master’

What does Github do?

* Everything in `master` is deployable
* To work on something new, create a descriptive branch off of `master`
* Commit to that branch locally and regularly push your work to the server
* When you need feedback, help, or think it's ready, submit a "pull request"
* After code review, the branch is merged into master and the code is deployed immediately

This is what Github does every day. They claim to each commit deployable code 23-25 times a day.

That means that Github.com is deployed 24 * ~80 a day; that's 1,920 times...

Source Commits Network **Pull Requests (23)** Fork Queue Issues (290) Wiki (98) Graphs Branch: master

Open **josh** wants someone to merge 11 commits into `master` from `charlock-linguist` **#1497**

Discussion Commits <> 11 Diff >= 9

josh opened this pull request about 16 hours ago

Charlock+Linguist

Use Charlock to detect if blobs are binary or plain text. The encoding is then passed along to pygments.rb when the blob is rendered.

/cc **@brianmario**

Open

+ **63** additions

- **40** deletions

All Pull Requests

josh, **brianmario**, and **tmm1** are participating in this pull request.

josh added some commits about 16 hours ago

24825bd		Use charlock for blob binary and encoding detection
bb0b945		Merge branch 'master' into charlock-linguist

brianmario started a discussion in the diff about 16 hours ago

vendor/internal-gems/linguist/linguist.gemspec View full changes

```
...  ...  @@ -6,8 +6,9 @@ Gem::Specification.new do |s|
 6    6     s.files = Dir['lib/**/*']
 7    7     s.executables << 'linguist'
 8    8
 9         -  s.add_dependency 'escape_utils', '0.2.3'
10         -  s.add_dependency 'mime-types',   '1.16'
11         -  s.add_dependency 'pygments.rb',  '~> 0.2.0'
      9    +  s.add_dependency 'charlock_holmes', '~> 0.6.0'
```

1

brianmario repo collab about 16 hours ago

I'd change this to 0.6.2 - I yanked 0.6.1 since it had that bug

brianmario commented August 17, 2011

❤

bleikamp commented August 17, 2011

⚡

defunkt commented August 17, 2011

kneath added some commits August 17, 2011

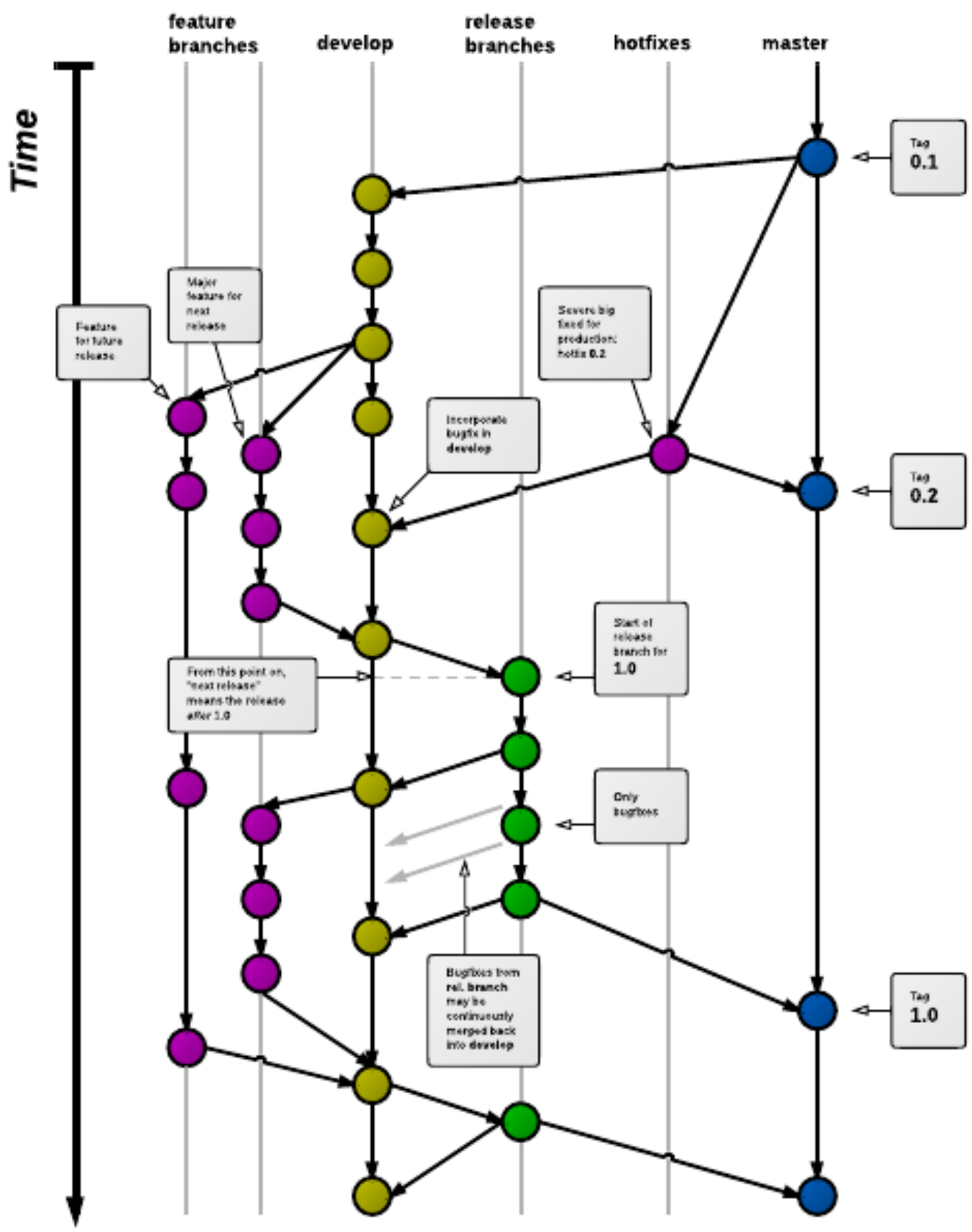
d0a5311  Org n00b notice needs new words.

kneath referenced this pull request from a commit August 17, 2011

a9ab4eb  Merge pull request #1431 from github/orgselector

Merged  **kneath** merged commit `a9ab4eb` into `master` from `orgselector` August 17, 2011

Closed  **kneath** closed the pull request August 17, 2011


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Major feature for next release
Feature for future release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
From this point on, "next release" means the release after 1.0
Start of release branch for 1.0
Only bugfixes
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Pick something that fits.

Code Reviews

Code reviews can be a touchy subject, depending.

Developers sometimes tie the code they write a little too closely to “who they are” as a person, in my opinion.

* Code review as part of a version control workflow creates a unique opportunity for pushing quality of code that goes into an application
* Reviews are a great time to enforce the current iteration of best practices your team has adopted
* It can foster discussion on how best to approach a new feature or bug-fix

* Don’t make it personal
* Don’t make inquisitiveness an attack
* Always... always assume best intentions

Just because you don’t know something or didn’t quite hit the mark does not mean you don’t know; you just don’t know, yet. Without code review, you’d never get the input that what you’re doing wasn’t quite right. While it may work, it’s not the best you can do.

And if you’re on the other side of that coin and you’re frustrated and say, “What were they thinking?!”; take a step back and be happy that your co-worker is about to find mentorship in you. Be that mentor.

Code review is a very sensitive subject that should be carefully approached, but definitely adopted. It’s highly dependent on the personality types you have on your team.

It’s important to have a team that can separate technical ability from self-worth as a member of the team.

ncsu-multiauth/ncsu-multiauth.php

outdated diff ✖

```
...  ...  @@ -8,114 +8,252 @@ Author: Dustin Wheeler
234  +    {
235  +        $plugin_data = get_plugin_data(__FILE__);
236  +        $updater = new Auto_Update($plugin_data['Version'], "http://www.webtools.ncsu.edu/wp-updates
237  +        add_filter('pre_set_site_transient_update_plugins', array($updater, 'check_update'));   // D
238  +        add_filter('plugins_api', array($updater, 'check_info'), 10, 3);    // Define alternate resp
239  +    }
240  +
241  +
242  +    /**
243  +     * Disable the password fields for non-admins, to prevent password changes
244  +     *
245  +     * @param   mixed   $flag   Unused
246  +     *
247  +     * @return  bool
248  +     */
249  +    public function disable($flag)
```

2

mdwheele 5 months ago

Do we want to make a distinction between local and WRAP users in allowing the capability to change passwords themselves? It's not that we're dis-allowing them access to change passwords, it's that it makes no sense for WRAP. However, local users would probably want to be able to.

mafields 5 months ago

The password fields are actually accessible to local accounts, so they can change their passwords. The password recovery function is currently disabled though. We can change that.

Add a line note

Testing...

The fact of the matter is... it's 2013. We've been through this with version control already. Do we really need to wait 10 more years before we buy into commonly accepted best practices in testing our applications?

I went to a conference earlier this year... Chris Hartjes.


Tests

“If there are no tests, it’s broken.”

This simplifies why to buy into testing our applications. It should simply be a part of everyone’s Definition of Done.

The next big feature might as well not even be released if we’re not going to test it. It’s like sending an airplane out of the factory with no wings.

Difference is, with software, things can seem to function and be horribly broken inside.

Unit, Functional, TDD, BDD... bunch of fads.

Unit Test

A method by which individual units of source code are tested to determine if they are fit for use.

Is your app testable in isolation?

(some apps resist the testing)

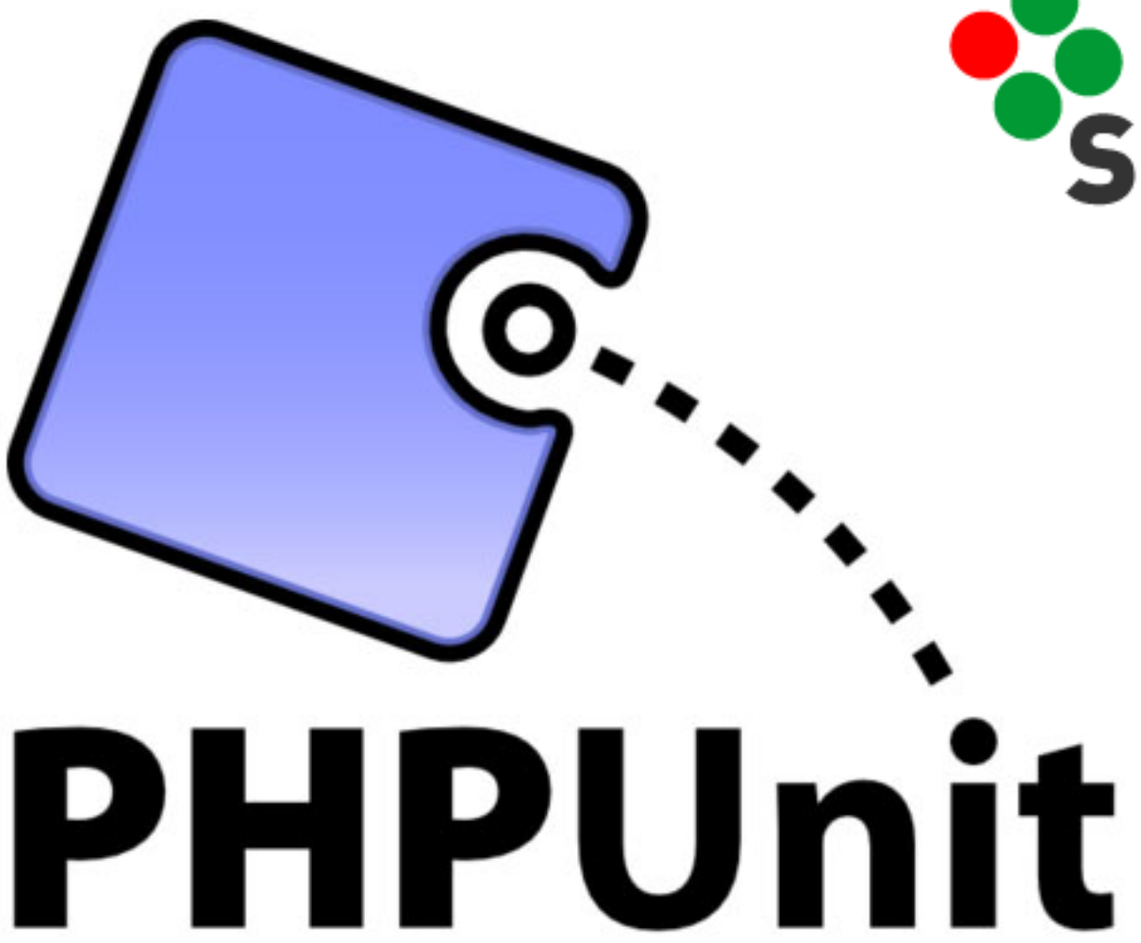
```
public class SuperMan implements SuperHero {

    private Cape cape = new Cape();
    private Spandex costume = new Spandex();

    @Override
    public void fly() {
        // Do something exciting with cape and spandex.
    }

}
```


This isn't testing in isolation.

```
public class SuperMan implements SuperHero {

    public SuperMan(Cape cape, Spandex costume) {
        this.cape = cape;
        this.costume = costume;
    }

    @Override
    public void fly() {
        // Do something exciting with cape and spandex.
    }

}
```


This becomes important when you start considering tests failing. If tests on YOUR code is passing and the app is breaking, that means you have an uncovered dependency that failed... which is GREAT!

Consider only using packages with tests for that very reason.



SimpleTest
Unit Testing for PHP

and more...

http://en.wikipedia.org/wiki/List_of_unit_testing_frameworks

file | 17 lines (14 sloc) | 0.278 kb

Open | Edit | Raw | Blame | History | Delete

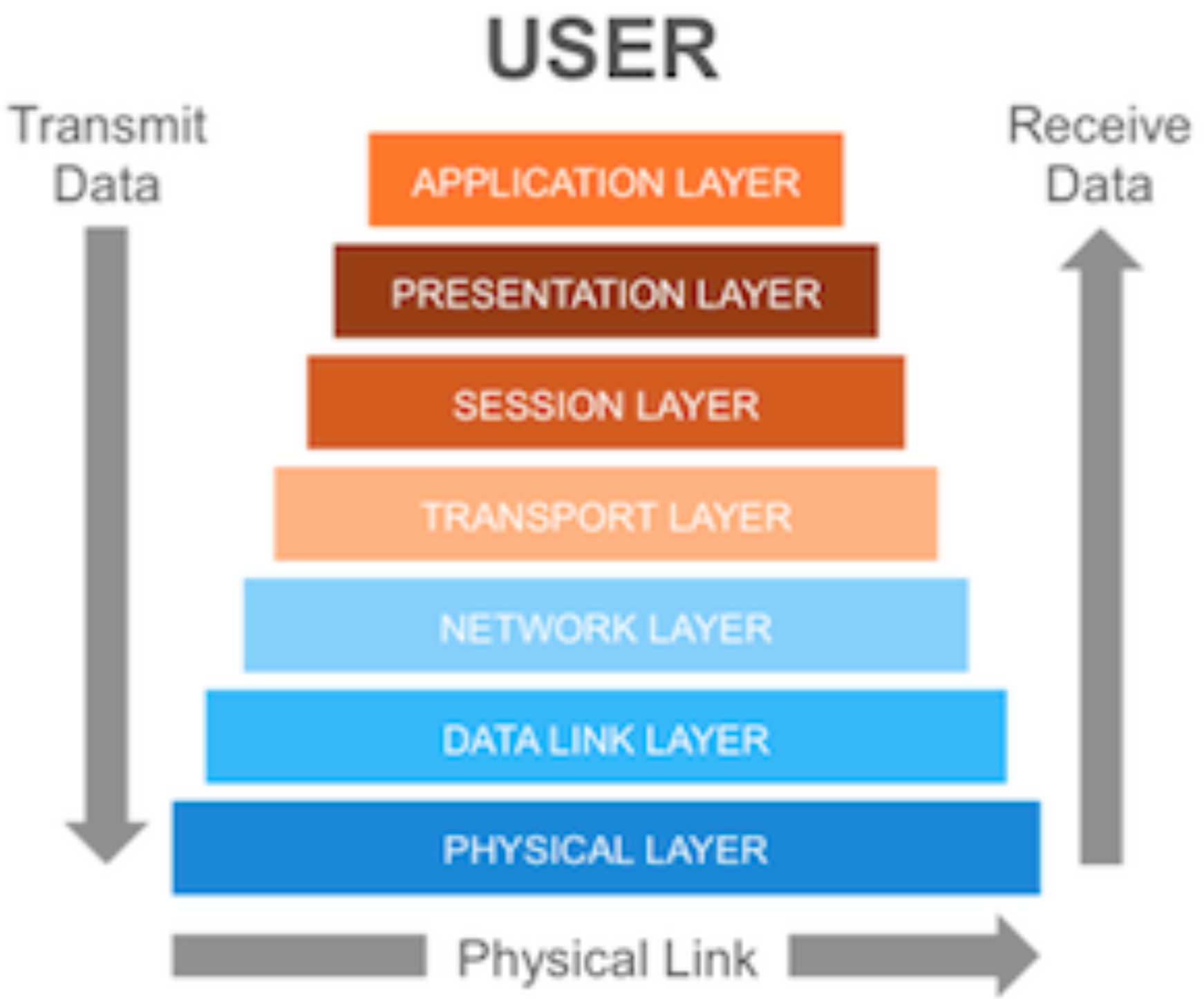
```php
<?php

use Wheeler\Fortune\Fortune;
use Mockery as m;

class FortuneTest extends PHPUnit_Framework_TestCase
{
    public function tearDown()
    {
        m::close();
    }

    public function testCanCreateFortune()
    {
        $this->assertNotEmpty(Fortune::make());
    }
}
```

Functional Test

A quality assurance process that bases its test cases on the specifications of the component under test

Sometimes called Integration Testing.

Think of it as testing a slice of your application stack.



I HATE THIS IMAGE. ITS A PLACEHOLDER TO GET THE POINT ACROSS.

Test Driven Development

* Write a test that fails
* Write code to make it pass
* Repeat

Does TDD make sense for everyone? Not at all.

But what it IS really good at is getting you “in the mode”.

Behavior Driven Development

* Write a test that fails
* Write code to make it pass
* Repeat

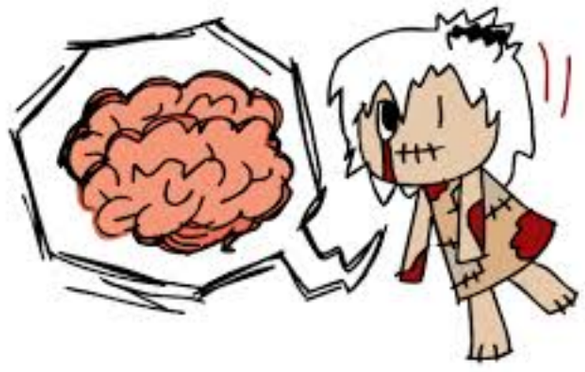
I didn't mess up my slides... it's pretty much the same exact thing, just a different spin.

BDD is ... behavior driven. It's analogous to unit vs functional testing. BDD is testing the public-facing side of the application.

Every app has behavior... hopefully.

(that means it can't resist the testing)

Selenium



Zombie.js

Phantom.js

```
var Browser = require("zombie");
var assert = require("assert");

// Load the page from localhost
browser = new Browser()
browser.visit("http://localhost:3000/", function () {

  // Fill email, password and submit form
  browser.
    fill("email", "zombie@underworld.dead").
    fill("password", "eat-the-living").
    pressButton("Sign Me Up!", function() {

      // Form submitted, new page loaded.
      assert.ok(browser.success);
      assert.equal(browser.text("title"), "Welcome To Brains Depot");

    })

});
```

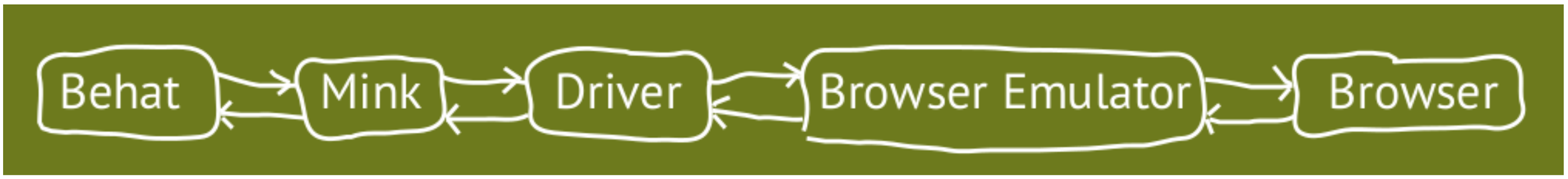

This gets tedious but the idea at first glance is to test what your users are seeing.

There are tools to make this less of a chore.

Behat Homepage: http://behat.org/

Mink Homepage: http://mink.behat.org/

Go check these out and find something similar for your language of choice. Behat is very similar to Ruby's cucumber. So, if you're writing ruby applications, "cucumber" is your buddy!



This is the most professional diagram I could find on the topic.

But you see that Behat is a testing framework that uses Mink to drive a headless browser emulator to test-drive your application according to rules you set.

elstamey 4 days ago Got the zombie driver installed and configured. Upd

1 contributor

file | 11 lines (9 sloc) | 0.298 kb

```
1  Feature: Search
2    In order to find buildings on campus
3    As an end-user
4    I need to be able to search a buildings directory
5
6    Scenario: Find a building
7      Given I am on "/maps/client/"
8      When I wait for page to load
9      And I fill in "search" with "Page Hall"
10     Then I should see "Page Hall"
```

In Summary...

They’re important and form the pass/fail basis for whether or not an application is allowed to be deployed.

Talk a little about continuous integration and deployment... but not a lot of details.

Mitigating Brain Drain

Make your developers happy...

This is by far the easiest way to keep domain expertise from leaving an organization.

Documentation

* External documentation works, but must be kept up to date
* **Meaningful** comments in code

External documentation will work just fine as long as it is made a “part of the process” like everything else. The first stop after a new feature is discussed and planned out should be an update to the documentation.

Meaningful comments are important.

```
public class SuperMan implements SuperHero {

    ...

    @Override
    public void fly() {
        // If the costume is not equipped.
        if (!costume.isEquipped())
            return;
    }

    ...

}
```


Developers know WHAT the language is doing. Meaningful comments should explain complex logic and more of the WHY a particular block of code is implemented the way it is.

```
    /**
     * This method takes a formatted string of year(s) and stores it as a multi-dimensional
     * array of operator+value expressions.
     *
     * Comparison operators can be encoded in URL.  Any PHP library wrapping this service should encode
     * the URLs.  For human usage, it may make more sense to allow unencoded operators.
     *
     * Operator Encoding
     * -------- --------
     * =        %3D
     * >        %3E
     * >=       %3E%3D
     * <        %3C
     * <=       %3C%3D
     *
     * Query parameter format:  ?year=2008,2009,1982-1989
     *
     * @param $year
     */
    public function setYear($year)
    {
        /** Split query parameter by comma for different years. */
        $years_raw = explode(',', $year);

        foreach($years_raw as $expression){
            /**
             * Every expression is either a single year or a range of years separated by a dash (-)
             */
            preg_match('/^([0]{1}|[0-9]{4})(?:-)?([0-9]{4})?$/', $expression, $matches);

            /**
             * From the docs on 'matches':
             *
             * If matches is provided, then it is filled with the results of search. $matches[0]
             * will contain the text that matched the full pattern, $matches[1] will have the text
             * that matched the first captured parenthesized subpattern, and so on.
             *
             * In this specific case:
             *
             * $matches[1] is the year (or beginning year, in a range)
             * $matches[2] is the ending year, if in a range; null if not range.
             */
```

Wednesday, November 20, 13

These comments are actually a little out of date, but you get the point. This particular snippet of code comes from a service where we expose a “year filter” in a publications web service. The original format was something like, “year=2013,2012,<=2009”, so I included the encoded values of those symbols that should be used. This isn’t the case anymore and I have already updated source!

The current implementation uses the “Query parameter format” you see in the docblock.

Also, the preg_match API in PHP (in my opinion) isn’t 100% clear for an unfamiliar or junior developer. Because of that, I include documentation from PHP.net on what actually gets stored in $matches. You can also do things like:

```
/**
@see php.net/documentation-url
*/
```

SOLID Design Principles

While documentation is a great way to encapsulate domain knowledge into software, there are other things we can do to make traversing a codebase “safer” and more intuitive.

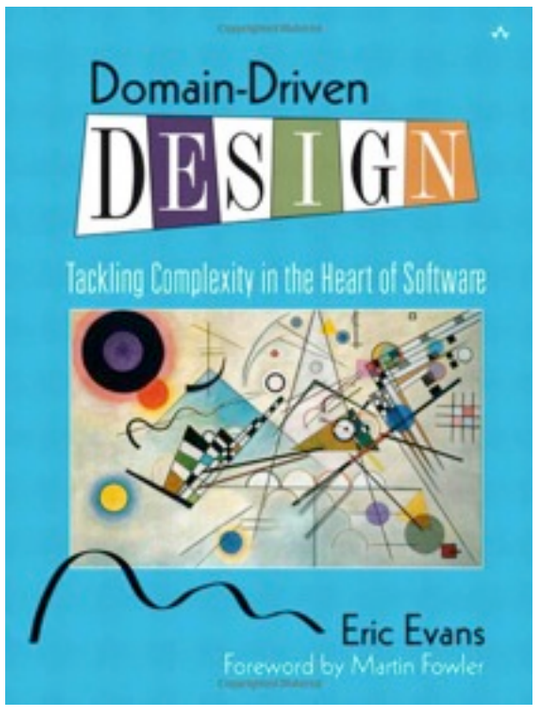
SOLID Design Principles – when applied together – make it more likely that a programmer will produce an easily maintained and extendable application.

* Single Responsibility Principle
* Open / Closed Principle
* Liskov Substitution Principle
* Interface Segregation Principle
* Dependency Inversion Principle

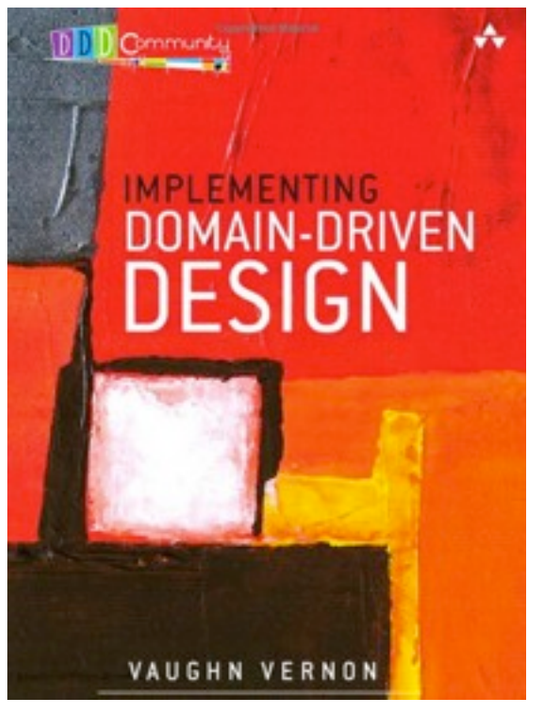
http://en.wikipedia.org/wiki/SOLID_(object-oriented_design)

Do some googling on this or send me an email! I didn't want to dive in deep on this since we had a diverse audience, but I love to talk about and teach development best practices where I can.

Domain-Driven Design



+



Wednesday, November 20, 13

Invest in a domain-driven design where it makes sense.

DDD is a topic worthy of days in seminar. It basically comes down to this: A iterative process for examining and extracting greater insight in a domain area; creating a ubiquitous language between domain experts and software developers that is carried to the code.

Evans' book is a VERY valuable to any aspiring software engineer. My opinion on this book is that given enough time, any motivated developer is GOING to end up doing a version of what he says, just by common sense. However, this book helps you skip steps that might take 4-5 years of experience. It's really great!

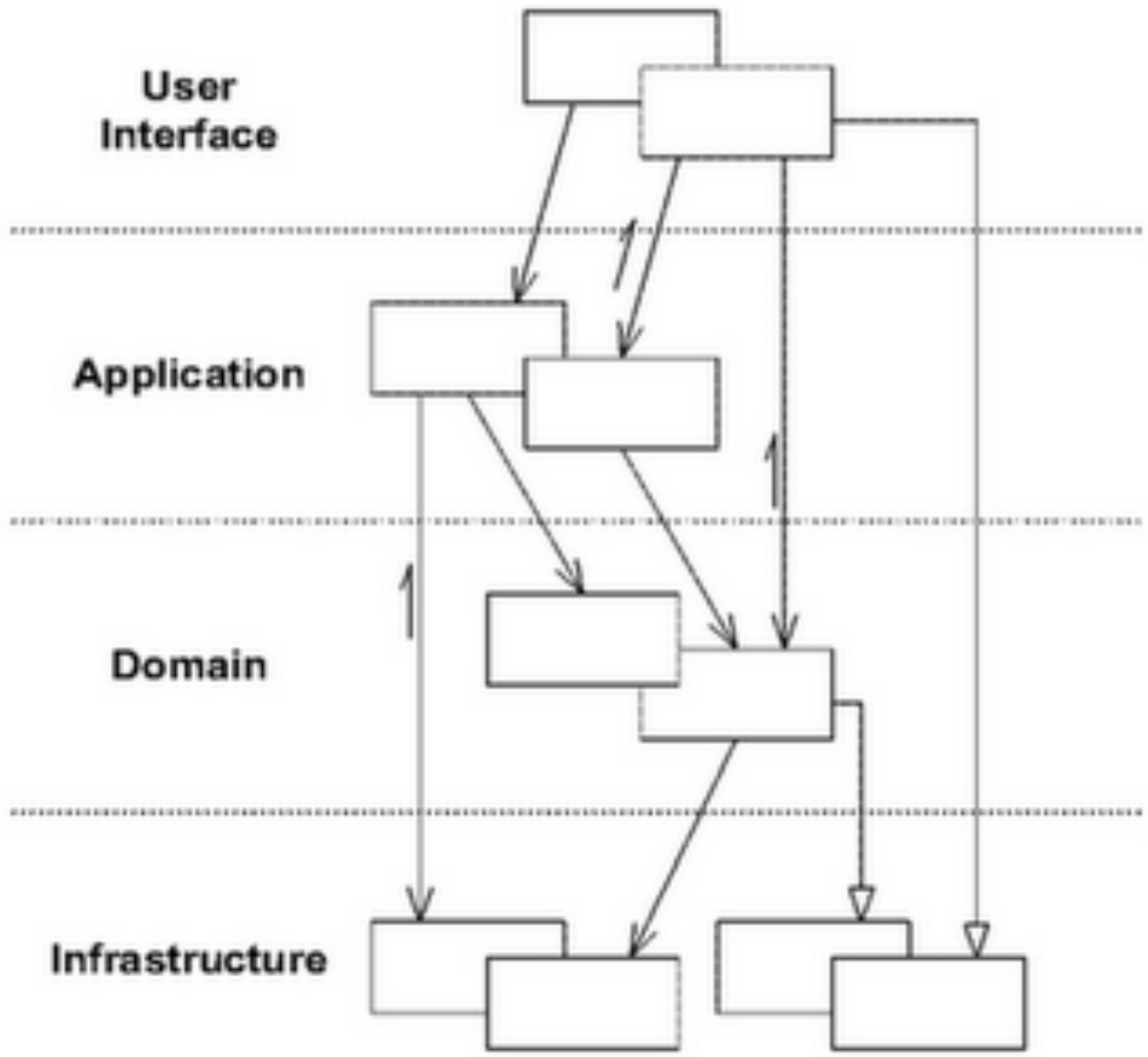
Vernon's book is a follow-up on Evans' work that speaks closer to the "how exactly do I apply DDD to existing apps and future work". Vernon's analogy was that Evans' work is the 10,000ft (more theoretical) explanation of the paradigm and that the goal for Vernon's book was to take the airplane down, land it, walk around, get the lay of the land, and then take off again. It also has Evans' "sign-off" as an excellent resource in applying the concepts.

Two books you can get for under $100 on Amazon that I guarantee are worth the investment. Or check them out from your University library! Haha.

... find a set of best practices that fit and apply them as they make sense for your team.

Finally... What's this SOA Stuff?

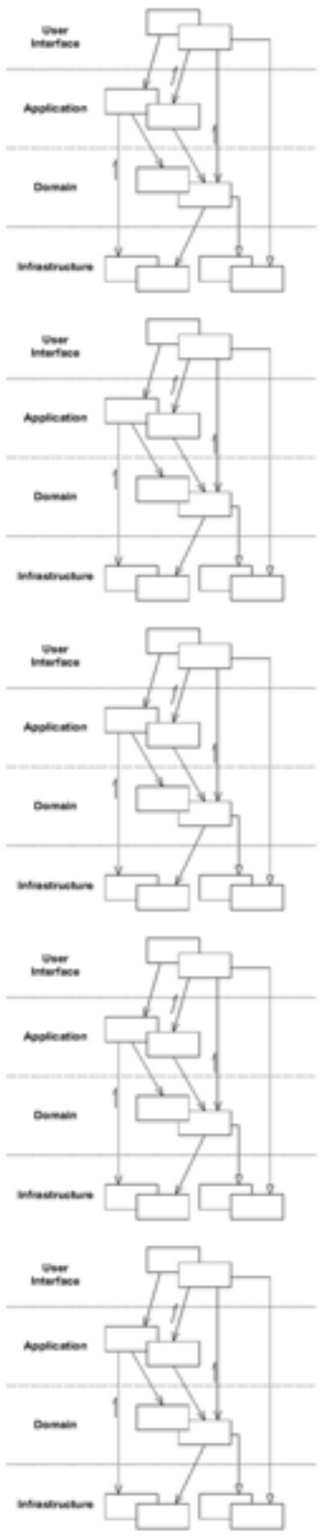
Let’s start with something familiar.



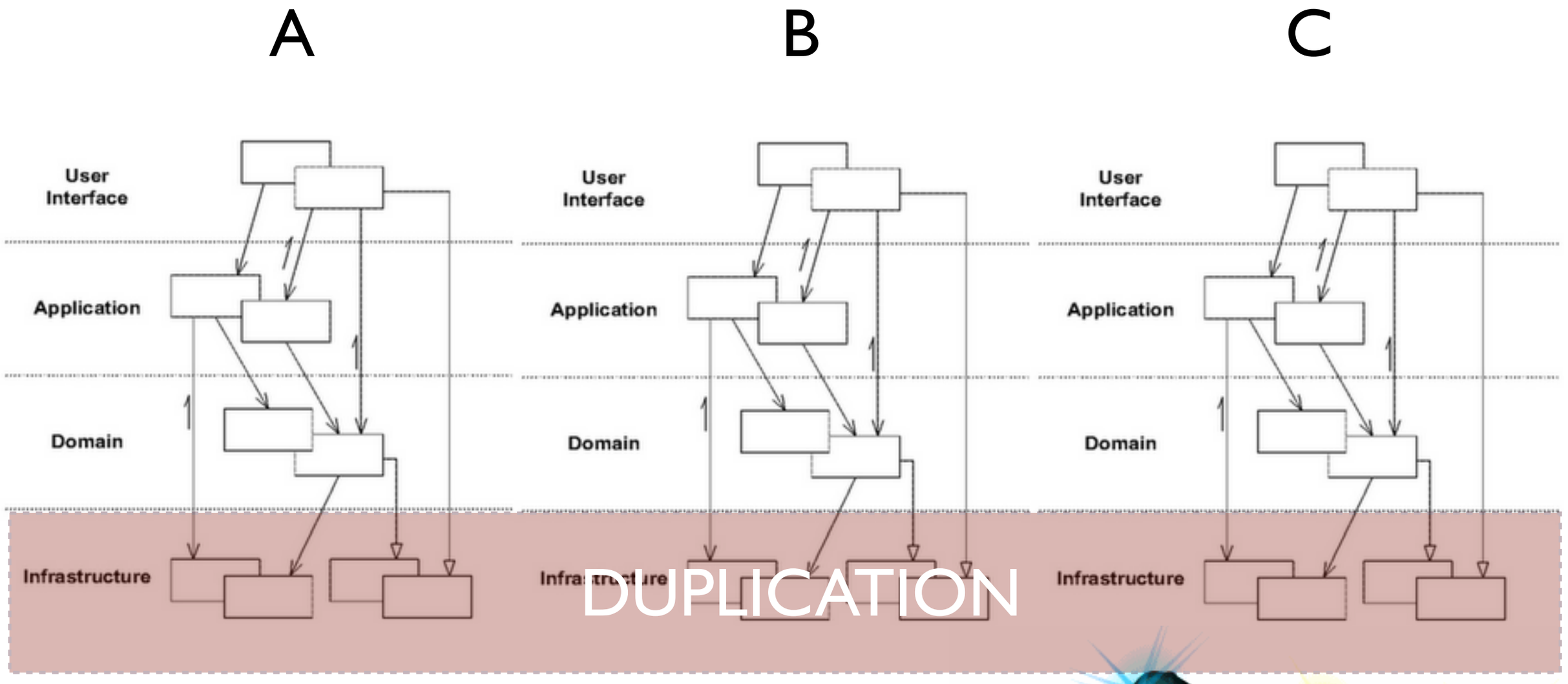
There is a LOT of virtue in this pattern.

Communication always flows downward through the stack, never up. The infrastructure knows nothing of the domain that's on top of it. It simply provides infrastructure services to be consumed.

Layered architectures start to fall apart in a few different ways.



As attempts are made to integrate layered applications with other services in interesting ways, your technology “stack” becomes a teetering tower.



Also, if you look at many layered apps side-by-side, they exhibit a high amount of duplication in the infrastructure layer.

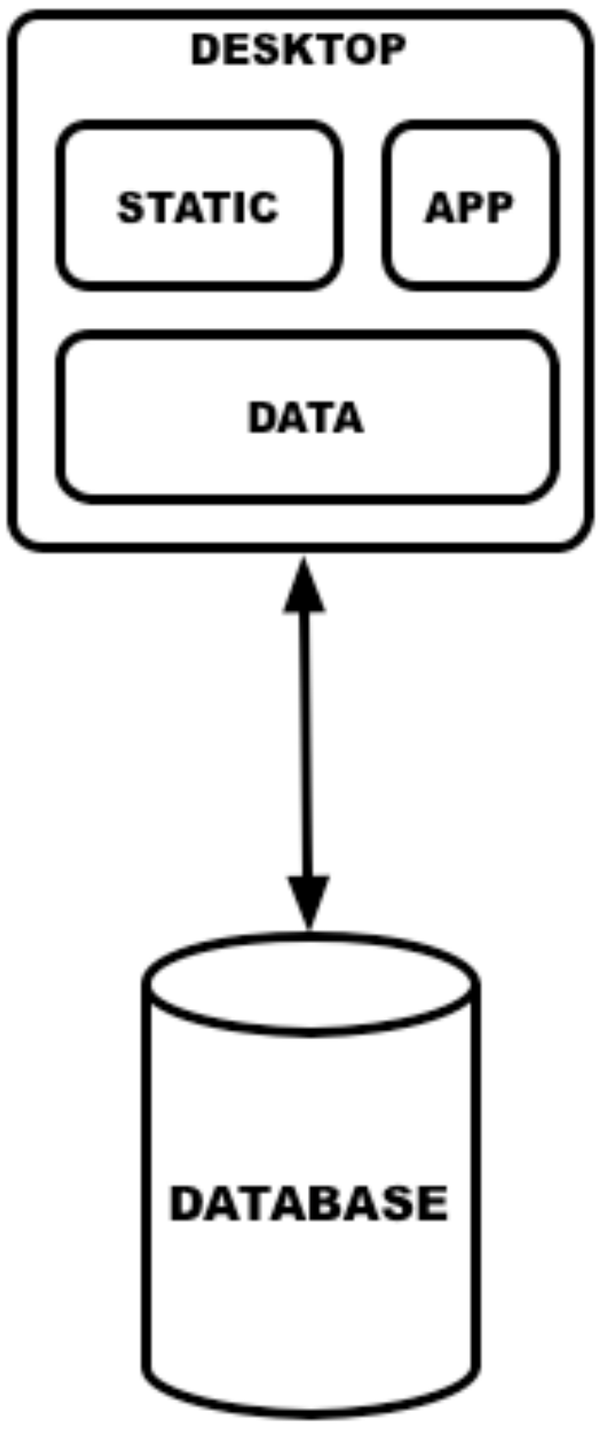
Steve’s tired of writing file uploaders... poor Steve.

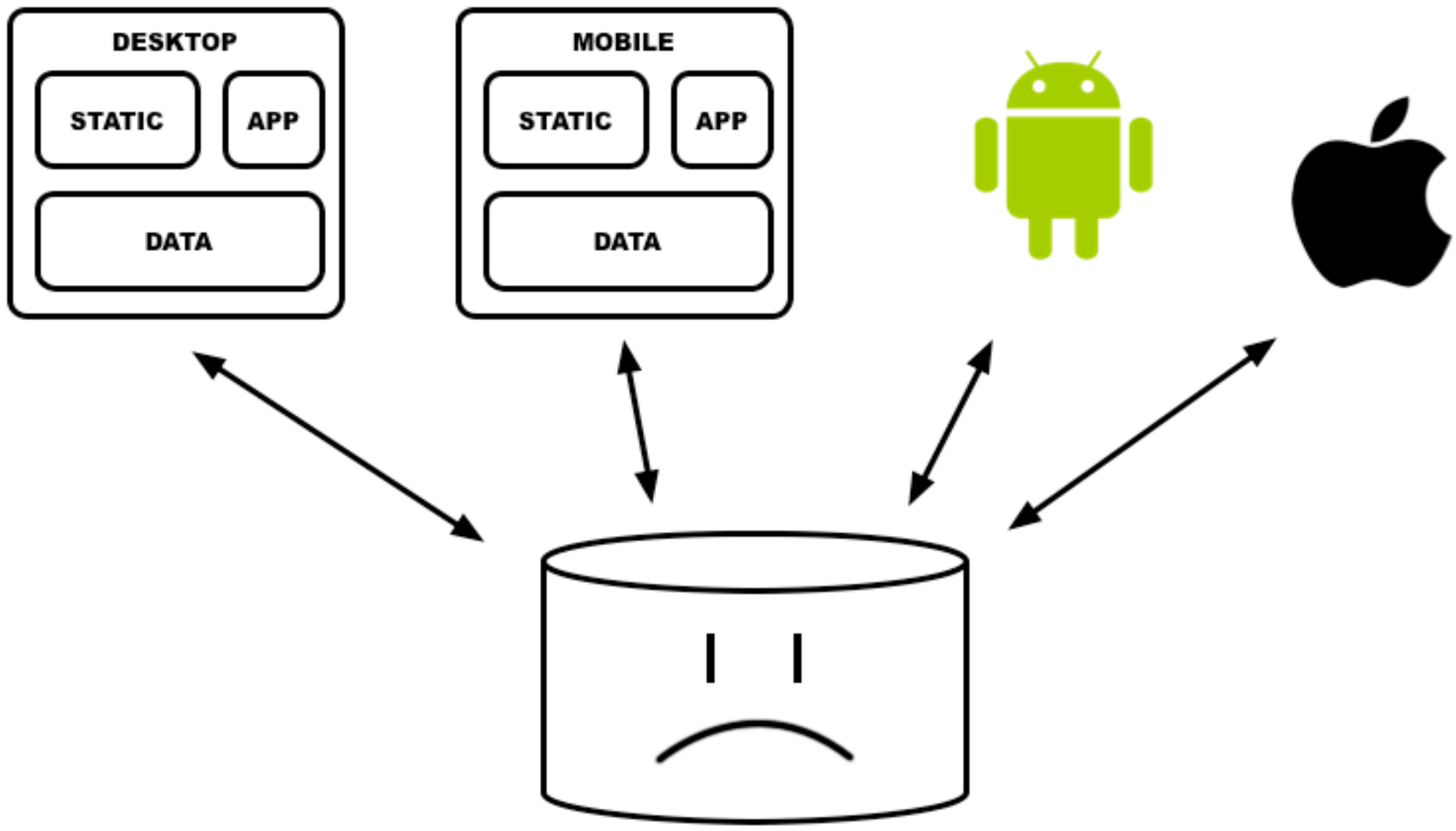
Service Oriented Architecture

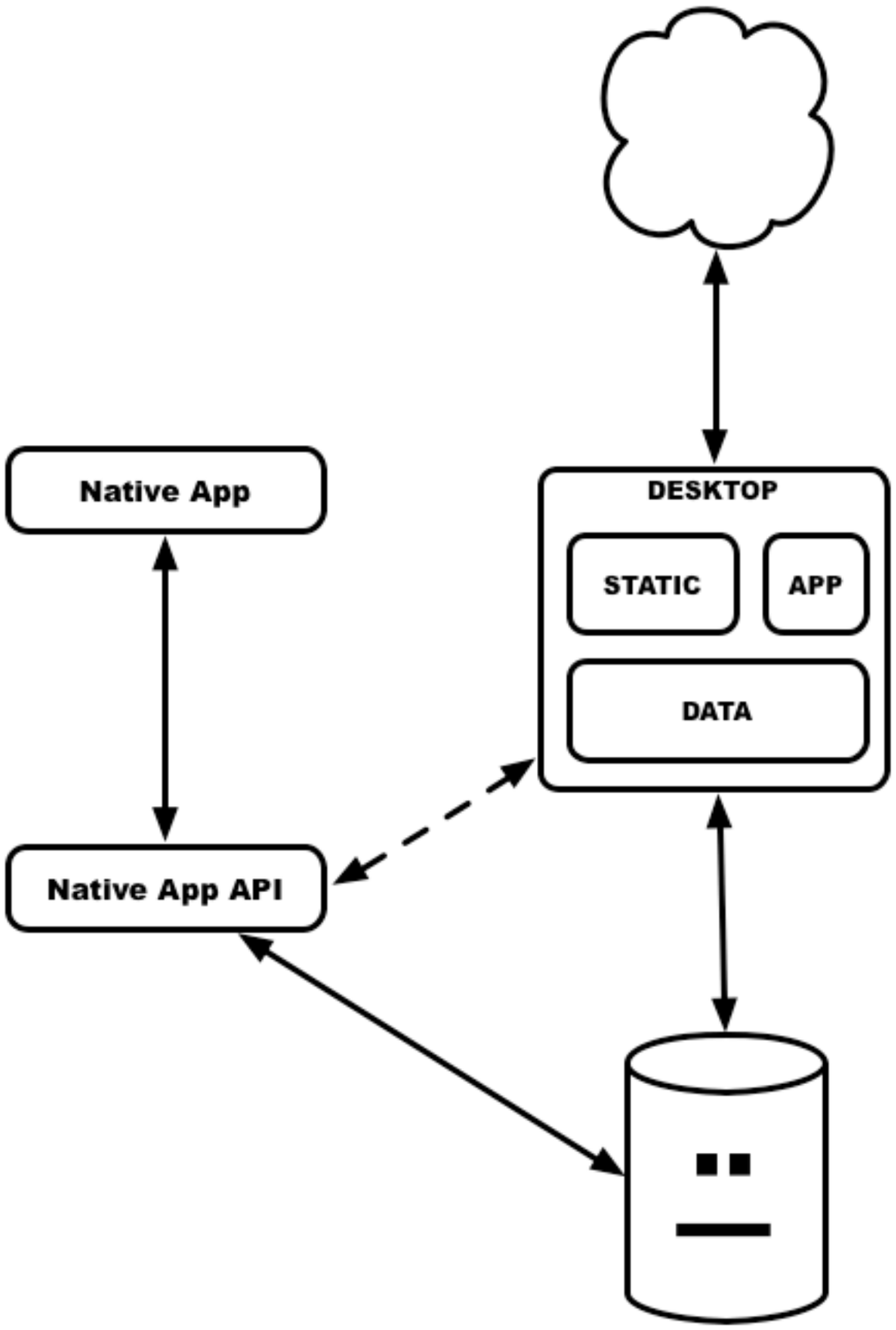
* An architectural pattern for developing discrete pieces of software to provide very specific functionality to other applications over a uniform communication protocol

So we're talking about APIs

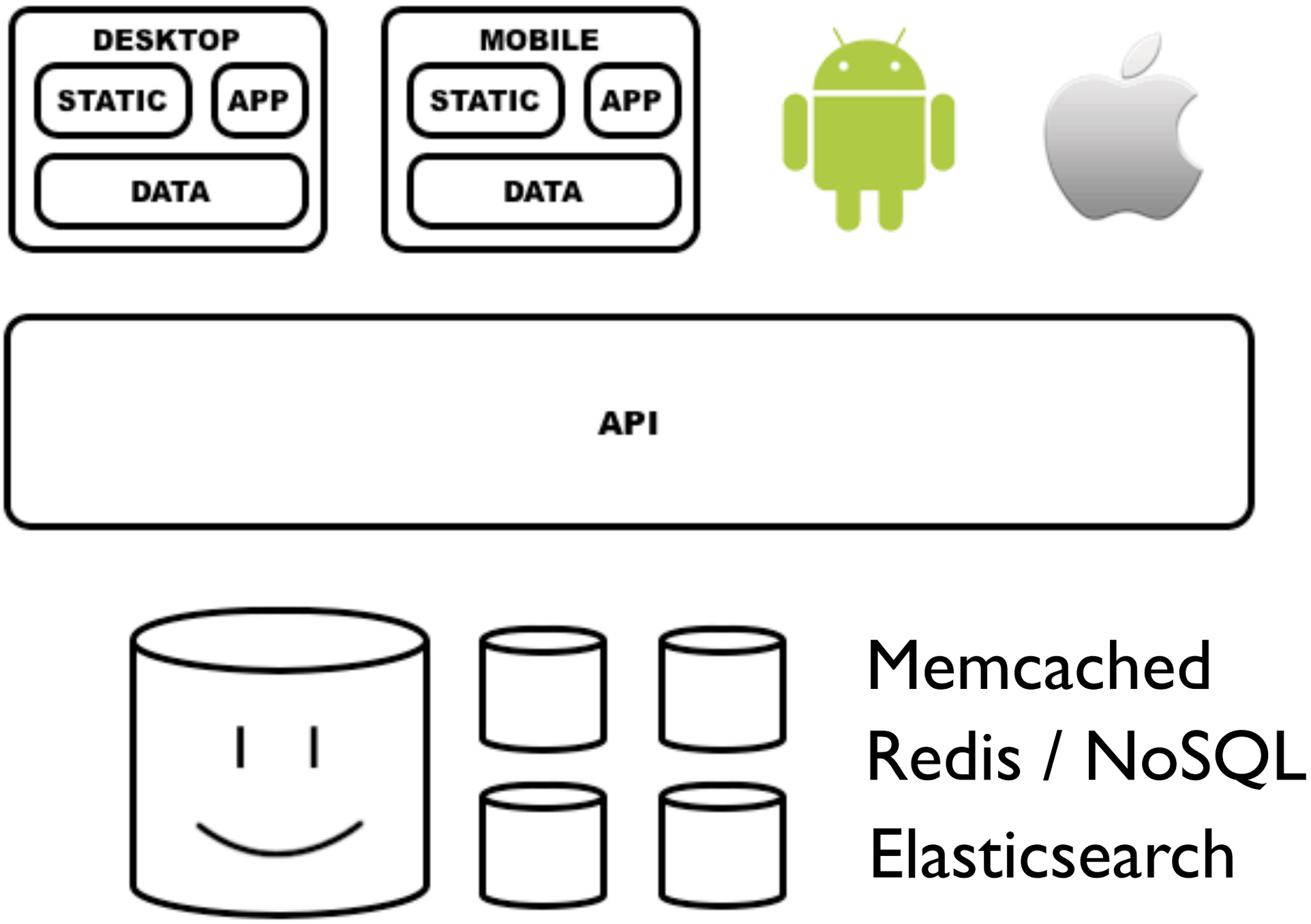
"Uniform communication protocol" is an interesting part of this definition. Because we're working with APIs and I need to create a uniform interface to my service, I am afforded the ability to communicate with multiple languages, platforms, and environments. As long as they can talk over HTTP, my API can provide service to whoever and I can also form a bridge between otherwise incompatible platforms. LOTS of strength there.


DESKTOP
STATIC
APP
DATA
DATABASE


DESKTOP
STATIC
APP
DATA
MOBILE
STATIC
APP
DATA



Data model is feelin’ kinda odd at the moment.



Shows ideal long-term configuration

Questions?