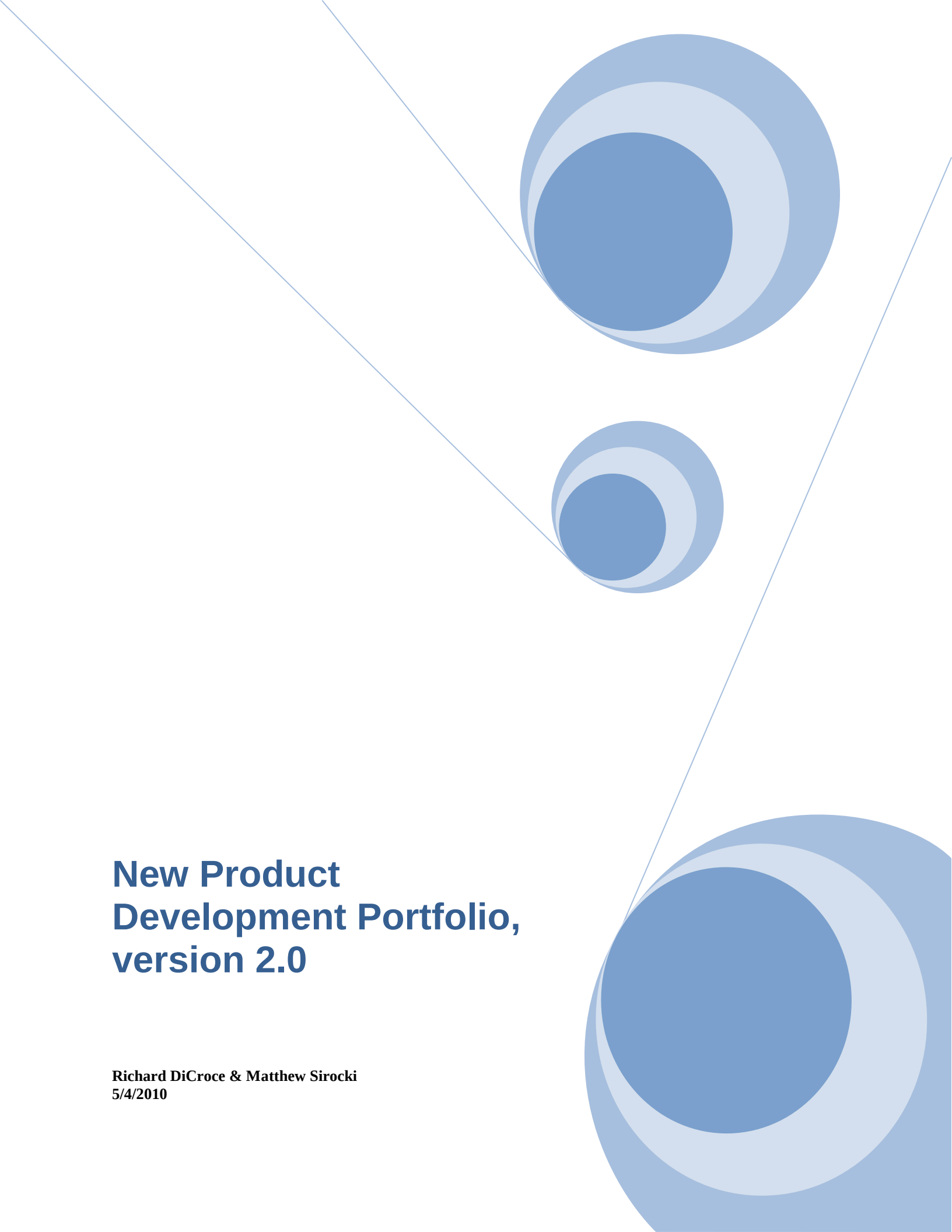


New Product Development Portfolio, version 2.0

Richard DiCroce & Matthew Sirocki
5/4/2010

New Product Development Portfolio, version 2.0

An Interactive Qualifying Project
submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
degrees of Bachelor of Science
and Bachelor of Engineering

by
Richard DiCroce, CS
Matthew Sirocki, ECE

Date:
May 4, 2010

Report submitted to:

Professor Erwin Danneels
Department of Management
Worcester Polytechnic Institute

TABLE OF CONTENTS

ABSTRACT

This project involved the development of an interactive web interface for a new product portfolio assessment tool. The tool's purpose is to assess the various new products that a company is developing in order to assist the company in making selection, prioritization, and resource allocation decisions. Users input numerical data about each project (e.g. unit price, unit cost, life cycle, etc) and score the project on 19 questions across four categories (competitive advantage, marketing resources, technical resources, and technical uncertainty) using a seven-point scale. The tool then makes a variety of calculations based on these inputs and allows the user to generate comparisons of the company's projects based on several criteria (e.g. net present value, return on investment, etc).

ABOUT THE PROJECT TEAM

Richard DiCroce is a junior at Worcester Polytechnic Institute pursuing a Bachelor's degree in Computer Science.

Matthew Sirocki is a junior at Worcester Polytechnic Institute pursuing a Bachelor's degree in Electrical and Computer Engineering.

INTRODUCTION

This project involved the development of an interactive web interface for a new product portfolio assessment tool. The tool's purpose is to assess the various new products that a company is developing in order to assist the company in making selection, prioritization, and resource allocation decisions. Users input numerical data about each project (e.g. unit price, unit cost, life cycle, etc) and score the project on 19 questions across four categories (competitive advantage, marketing resources, technical resources, and technical uncertainty) using a seven-point scale. The tool then makes a variety of calculations based on these inputs and allows the user to generate comparisons of the company's projects based on several criteria (e.g. net present value, return on investment, etc).

The tool is designed around the methodology articulated by Professor Erwin Danneels, the project's advisor, in a 2002 paper published in the Strategic Management Journal. The paper argues that "product innovation drives organizational renewal by exploiting and exploring firm competences" and subsequently defines some criteria for evaluating new product development projects based on estimations of the company's ability to build the product (technical skill) and the company's ability to sell the product (marketing skill).[1] The tool this project produced is geared towards making such evaluations simpler. Based on user input, the tool can generate charts and lists of product development projects adjusted for the amount of technical and marketing risk involved in developing each product.

This project developed the second version of this tool. Version 1.0 included only basic analysis functionality and had no collaborative features. The main goals for the second version were to expand the analysis functionality and provide some collaborative features. It was initially planned that version 2.0 would simply build on version 1.0, which was produced by an earlier IQP team. This proved to be impossible, for reasons that will be described in the next section. Consequently, version 2.0 is a complete rewrite. It duplicates the major functionality present in version 1.0, but does not reuse or extend any code from version 1.0.

[1] Danneels, E. (2002). The Dynamics of Product Innovation and Firm Competences. *Strategic Management Journal*, 1095-1121.

REVIEW OF PREVIOUS EFFORT

Version 2.0 was originally planned to extend the efforts of a previous IQP team. Version 1.0 was built using Adobe Flex for the web interface, MySQL as the back-end datastore, and PHP to communicate between the two. Version 1.0 was made available at www.npdportfolio.com and was used by many students in several of Professor Danneels' classes.

Professor Danneels judged the effort to build version 1.0 a success. Version 1.0 was capable of taking in the necessary user input and performing the appropriate calculations to generate the desired bubble charts. However, version 1.0 also had a variety of known problems and limitations:

- **Poor conceptual model.** It made an arbitrary distinction between company user accounts and individual user accounts. Professor Danneels indicated that this was done to prevent everyone from creating companies in the tool, but the distinction failed to accomplish this goal.
- **Poor user interface design.** Partly because of the poor conceptual model, it was not clear how to perform a variety of basic tasks and no help was provided. For example, just to begin using version 1.0, two separate user accounts had to be created. First, the user had to create a Company account. Then the user had to log in to the Company account and create a project. After that, the user had to log out and create a User account. Then the user had to log in to the User account and add him/herself to the company just created. At this point, the user could finally begin entering data.
- **Bugs.** Version 1.0 had a variety of known bugs. For example, sometimes it would report that user accounts had been created when, in fact, they had not. It also sometimes reported that all calculated outputs for a project were zero when they were not. Some users also reported problems with disappearing charts, requiring them to log out and log back in so they could continue working.

We initially believed that these were mostly surface problems that could be rectified by reworking the existing code. Instead, we discovered that they were systemic issues related to the way the entire version 1.0 project was handled. In the first two weeks of the version 2.0 project, we uncovered numerous additional, internal problems:

- **Incorrect or non-existent comments.** The version 1.0 Flex code contained few comments, making it difficult to determine exactly what any particular section of code was trying to accomplish. The PHP code was not much better; for example, it contained the comment "Query the database to see if the given username/password combination is valid" in almost every file, but the code never actually performed such a check.
- **Insufficient and incorrect documentation.** Outside of the code, the only documentation the previous team produced was their IQP report. This report did little to explain the internal operations of the code. Instead, most of the report was spent explaining what software was used in the project and justifying relatively trivial decisions (e.g. what web host to use, why the SHA-1 hash algorithm was not employed, etc).[2] Even worse, most of the few technical details the report provided were incorrect. For example, the report claims the project employed the MD5 hash algorithm to secure users' passwords; in reality, the AES encryption algorithm was used. The report also included a description of the schema used in the database, which we soon discovered was out of date and not the final schema used in version 1.0.
- **Pervasive security flaws.** As previously mentioned, the PHP scripts never verified user credentials before updating the database. In addition, most inputs to the PHP scripts were not escaped, so a wide variety of SQL injection attacks were possible. The client was allowed to handle securing the password, and did so using the AES encryption algorithm rather than a hashing algorithm, meaning that users' passwords could possibly have been decrypted into plaintext.

[2] Tidd, M., & Werner, U. (2009). *New Product Development Portfolio Website.*

- **Poor code organization.** The entire web interface was implemented in one giant Flex source file. This made navigating through the code extremely difficult.
- **Poor programming practices.** The Flex code included nearly every programming practice that should be avoided at all costs. For example, it made liberal use of global variables. Most variables were cryptically named, making it difficult to tell what they were being used for. Some functions involved in calculating outputs directly manipulated parts of the interface, while others were just stubs that called another function to perform the calculation. This made separating sections of code into their own source files effectively impossible.
- **Poor project organization.** The version 1.0 team does not seem to have employed any of the usual tools or techniques helpful in managing a software project. There is no indication that they used source control or any sort of local testing environment. From the contents of the web server, it appears that version 1.0 was developed directly on the server and that "good" code was occasionally backed up in a Zip archive and stored on the server.

Because of the substantial internal problems, we decided, about two weeks into the project, to scrap the existing code and commence a total rewrite.

APPLICATION ARCHITECTURE

The application's architecture has two major components. First, the back-end is powered by a MySQL database, with all interactions performed through a SOAP service written in PHP. Second, the front-end, written in Flex, provides a friendly user interface for manipulating data.

The architecture is designed such that it has high cohesion and low coupling. Each component is effectively a black box with internal operations that the other component does not know or care about. For example, the Flex front-end knows only that it is communicating with a SOAP service that implements a given set of functions; it does not know or care how the SOAP service is implemented or how the data is persisted. Likewise, the SOAP service knows only that it is communicating with a client that wants to use its services; it does not know or care whether the client is the Flex front-end or something else.

BACK-END DESIGN

The back-end has two major subcomponents. First, a MySQL database provides persistent data storage. Second, a SOAP service written in PHP provides a convenient and secure way of accessing the database.

THE MYSQL DATABASE

The full database schema is documented in Appendix B. It was developed using MySQL Workbench, a convenient tool for building and maintaining MySQL database schemas.

To summarize, there are five tables. The Users table contains one row for each user who has registered. The Companies table contains one row for each company that has been registered. The CompanyUsers table connects the two; the connection between companies and users is m:n, so there is one row for each <company,user> pair.

The Projects table contains one row for each project in the system. Projects may be considered part of a company, as the mapping from company to projects is 1:n. That is, a company may have many projects, but a project can only belong to exactly one company. Note that the database design allows for multiple companies to have a project with the same name. These are still separate projects; the constraint on the table is that each <company,project-name> pair must be unique.

The ProjectResponses table contains one row for each user response to a project. Responses may be considered part of a project, as the mapping from project to responses is 1:n. Note that the constraint on this table is that the <project,user> pair must be unique. In other words, each user is allowed to have at most one response to each project.

The general naming convention in use is camel case. Table names begin with an upper-case letter; field names begin with a lower-case letter. All fields containing monetary values are defined using the same data type. Likewise, all fields containing percentage values are defined using the same data type.

All tables are explicitly set to use the InnoDB storage engine. This is a critically important point! The default storage engine in current versions of MySQL is the MyISAM engine. *MyISAM does not enforce foreign key constraints!* The default storage engine can be configured, but we lack access to the necessary configuration options in the current shared hosting environment. Therefore, any tables added to the schema must explicitly be set to use InnoDB as well.

The default character set and collation for the entire schema has been set to UTF-8. In theory, this should allow for easy internationalization (if and when that becomes necessary) but characters outside of the usual ASCII set have not been tested.

THE SOAP SERVICE

The SOAP service provides a convenient and secure way for web applications like our Flex front-end to interact with the database. It is written in PHP using the NuSoap library. The NuSoap library is only sporadically active in terms of development, but many people have used it to build SOAP services and it was judged to be a better choice than the alternative options. One important strength of NuSoap is that it can generate the WSDL file for your service automatically; no other options, including the SOAP library that is now integrated into PHP, could do this at the time the project began.

SECURITY CONSIDERATIONS

Because we have no idea who might be calling the SOAP service, it is important to implement a pervasive system of security checks. The service architecture is designed to balance security with performance. The service exposes a login operation; this operation validates the user's credentials and sets a variable in the user's session to contain the username. Most operations require an authenticated user, and the username set in the session is used when this is the case. Note that sessions time-out (but are not necessarily garbage collected immediately) after 24 minutes by default. If a session times out and the client wants to continue performing authenticated operations, the client must re-login first.

A secondary layer of security involves ensuring that users have permission to perform the given operation. Since a user's permissions could change at any time, they are not stored in the session. Instead, helper functions are provided to query the database and check whether the user has the permission to do the requested operation relative to the company the operation is being attempted on. If an error occurs, these functions will throw an exception, which the calling function must catch and then proceed to abort the operation. It should be noted that there was initially an intention to provide site administration functionality, but the functionality was deemed unnecessary at this time. Nevertheless, the support necessary to provide the functionality was coded and remains in the back-end.

A third layer of security is aimed at preventing malicious attacks. The SOAP operations interact with the database using PHP's newer MySQLi extension. This allows us to take advantage of features added in newer versions of MySQL. Most of these features are not used. Some cannot be used; at the time of writing, the hosting provider does not allow triggers, stored functions, and some types of stored procedures. One feature is used pervasively: prepared statements. Prepared statements are one of the safest ways to make calls to a database; they completely prevent SQL injection attacks. All calls to the database from the SOAP service are made using prepared statements for this reason.

MISCELLANEOUS CONSIDERATIONS

There are some other issues that must be taken into account when coding for the SOAP service. For one thing, the SOAP standard considers "true" and "false" to be valid values for a data field having a Boolean type. PHP, on the other hand, will evaluate any strings other than "" (the empty string) and "0" as true when converting to a Boolean, meaning that "false" evaluates to true. A special stringToBool() utility function was therefore introduced to deal with this problem; it must be used whenever a Boolean value must be taken in from a client.

Another issue is that of concurrency. The SOAP service is not being heavily used, so this is currently not a problem and no attempt has been made to deal with it. This may become an issue in the future.

FRONT-END DESIGN

The front-end has three significant sub-components. The web service system interacts with the SOAP service in the back-end, the Data singletons hold retrieved data, and the Screen framework helps manage the user's interaction with the data.

THE WEB SERVICE SYSTEM

There isn't much to say about the web service system. Most of it is a black box of code auto-generated by Flash Builder for us. The metadata and super classes are not to be touched, but you can freely make changes to the data classes themselves.

Wherever possible, you should always check that the user has permission to perform an action before calling the web service. It's much more efficient to simply check if the user has permission, since the data singletons provide access to that data, than it is to make the call to the server and catch the error it returns. Having said that, since there are no provisions for concurrency, it is possible that the permission information might be outdated. It is also possible that the server will encounter an error trying to perform the operation. It is, therefore, important to always attach a fault handler to every web service call.

On another note, all of the code that calls the web service code currently does not provide for the possibility that the session may have timed out. It is considered unlikely that users will encounter this problem, but a check for this and code to perform an automatic re-login should be added to all service calls. Time constraints prevented the implementation of this code in the current version.

THE DATA SINGLETONS

Four singletons exist specifically to hold data returned from the web service that many parts of the front-end might need.

- **The CompanyData singleton** holds a list of all companies the user is part of, along with the user's current status in that company and their permissions.
- **The CompanyUserData singleton** holds a list of all users in the current company, their status, and their permissions, assuming that the user has permission to access this information. This singleton is also home to the filter list used in the analysis screens.
- **The ProjectData singleton** holds a list of all projects in the current company, as well as a reference to the currently selected project.
- **The CredentialsData singleton** holds the user object and the user's password. It was planned that this would be used to perform automatic re-login in the event of a session timeout, but this was never implemented (see previous section).

Note that extensive permissions checking is implemented in the front-end as with the back-end. This is less about security (since, if the user isn't allowed to perform the operation, the back-end should stop them) and more about better interface design; users should not be allowed access to functionality they aren't allowed to use.

THE SCREEN FRAMEWORK

In order to better coordinate the front-end's *many* screens, we crafted the Screen framework on top of Adobe's GUI framework. The Screen framework is well-documented in the code, so developers should look there for additional details. The Screen framework provides several useful features that make programming navigation

through the front-end simpler. The framework also provides a ScreenManager singleton that holds references to frequently used screens along with subclasses of the Canvas component that add features such as the loading box.

Flex 3 was in use when we began developing the front-end. Now that Flex 4 has reached release status, many of the graphical components in use are deprecated in favor of newly designed components that perform the same functions. We were able to convert many parts of the front-end to Flex 4, but time constraints prevented migration of the Screen framework. A future team should investigate this.

GENERAL COMMENTS ON FLEX AND ACTIONSCRIPT

Flex and ActionScript have many annoyances and idiosyncrasies that made development of the front-end difficult. These include, but are not limited to:

- **No generics.** You have to be aware of what type of object an ArrayCollection contains; Flash Builder cannot tell you. Additionally, you must cast objects you get from an ArrayCollection to the expected type. If the cast fails, no exceptions or errors are thrown; the destination variable is simply set to null.
- **ObjectProxy.** At least in beta versions of Flex 4, objects returned from the SOAP service that were nested several levels deep were not of the expected type and could not be cast to the expected type. Instead, they were of type ObjectProxy. This problem was avoided by eliminating the need for nesting, but it may still exist.
- **Limitations on constructors.** MXML files cannot have custom constructors. This forced us to make changes to our planned architecture by providing places in singletons where screens could pull relevant data from, rather than pushing the data to the screen when it is constructed.
- **No destructors.** ActionScript just doesn't have them.
- **No abstract anything.** ActionScript also doesn't have a concept of an abstract class or abstract methods, so if you want to create a class that can never be instantiated, you have to provide your own guarantees.

Appendix C contains entries from a team member's blog that are relevant to this project, including expanded discussion of the above issues and a variety of others. A great deal of architectural discussion is also contained in those blog entries. Future developers are highly encouraged to read this material before they begin programming.

RECOMMENDATIONS FOR FUTURE WORK

If time and resources permit, future developers may want to examine these issues, which we were unable to resolve due to time constraints. Some of these have been discussed in the above sections.

- **Handling concurrency.** As mentioned, no parts of the application are truly capable of dealing with concurrency issues. For example, if a user is granted a permission after they have logged in, they must log out and log back in for the change to take effect. This will be increasingly problematic as use of the application increases.
- **Handling session timeouts.** As mentioned, the front-end does not currently handle session timeouts. This presents a significant interface problem, requiring users to manually log in again.
- **Finish migration to Flex 4.** As mentioned, some parts of the front-end have been migrated to Flex 4, while others, especially the Screen framework, have not. The older graphical components in use are now deprecated and may be removed in a future version of Flex.
- **Update the NuSoap library.** Shortly before this report was written, a new version of the NuSoap library was released for the first time in approximately three years. An attempt to upgrade to the newer version revealed some breakage that needs to be investigated and resolved. We did not have time to examine this issue further.
- **More extensive testing.** We are not aware of any bugs in the current version, but that is no doubt because we have not had the time to perform the sort of extensive testing needed to uncover the bugs. Automated testing of the front-end would probably be difficult, but it should be investigated as the front-end was the source of most of the bugs we encountered. Unit tests of the back-end would also be beneficial. Such unit tests were planned for the current version, but time constraints prevented implementation.
- **Context-sensitive help.** Context-sensitive help was planned for inclusion in the front-end, but time constraints prevented implementation.

BIBLIOGRAPHY

Danneels, E. (2002). The Dynamics of Product Innovation and Firm Competences. *Strategic Management Journal* , 1095-1121.

Tidd, M., & Werner, U. (2009). *New Product Development Portfolio Website.*

APPENDIX A – APPLICATION SCREENSHOTS

Note that many screens in the application have a variety of states depending on whether an item has been selected, what the status of that item is, what permissions the user has, etc, so these screenshots do not encompass all possibilities.

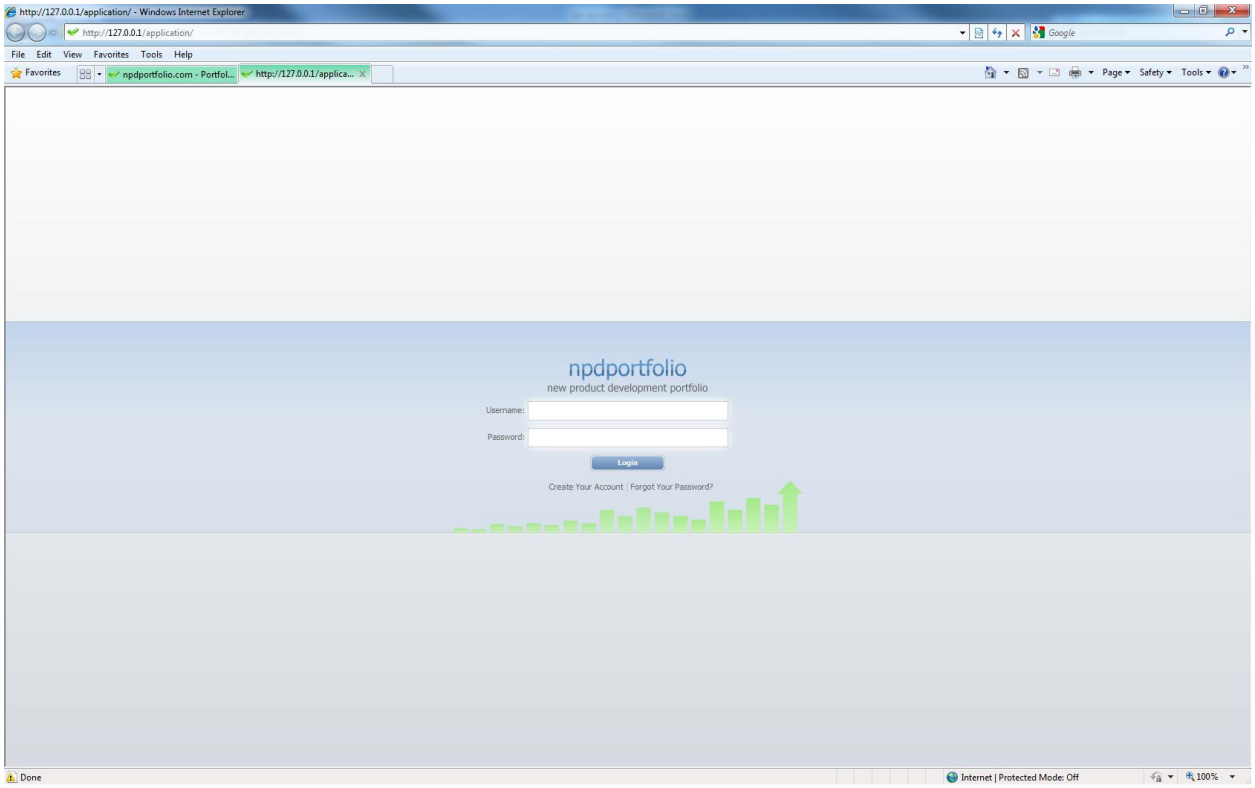


Figure 1 - Login screen

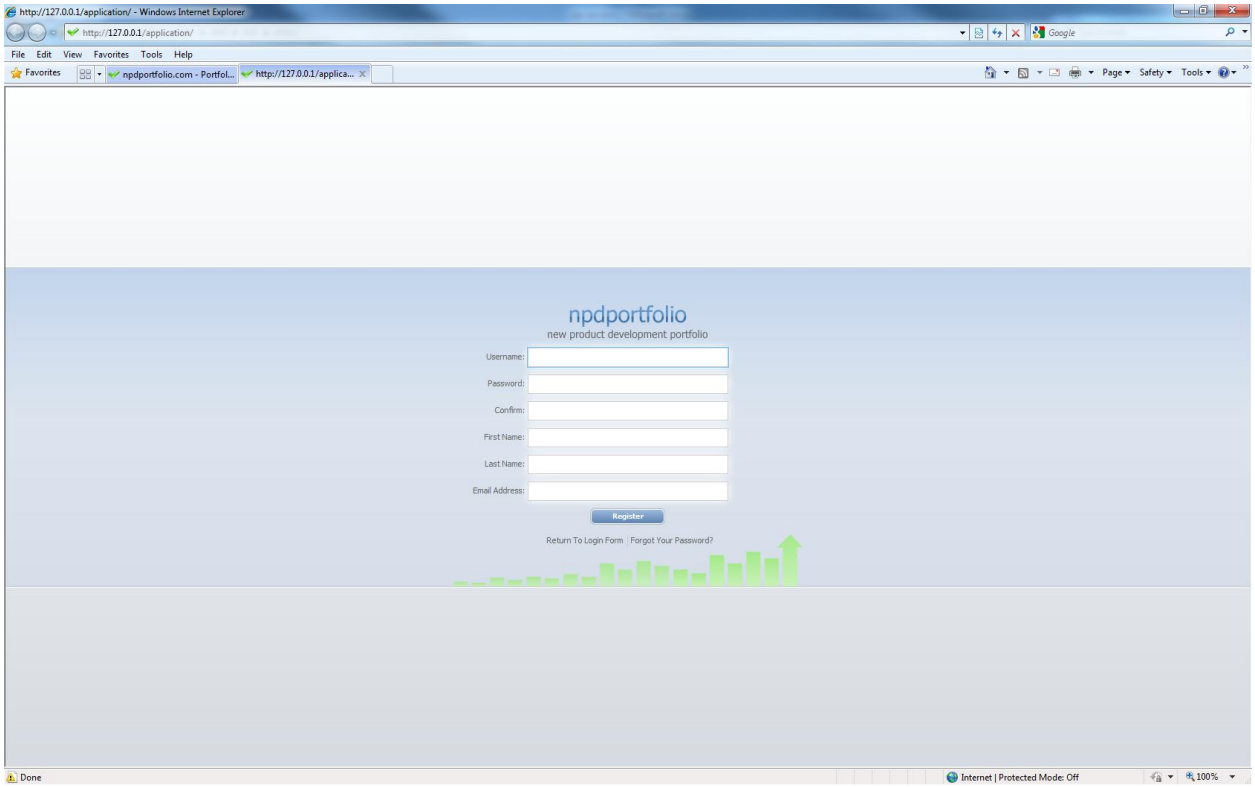


Figure 2 - Registration screen

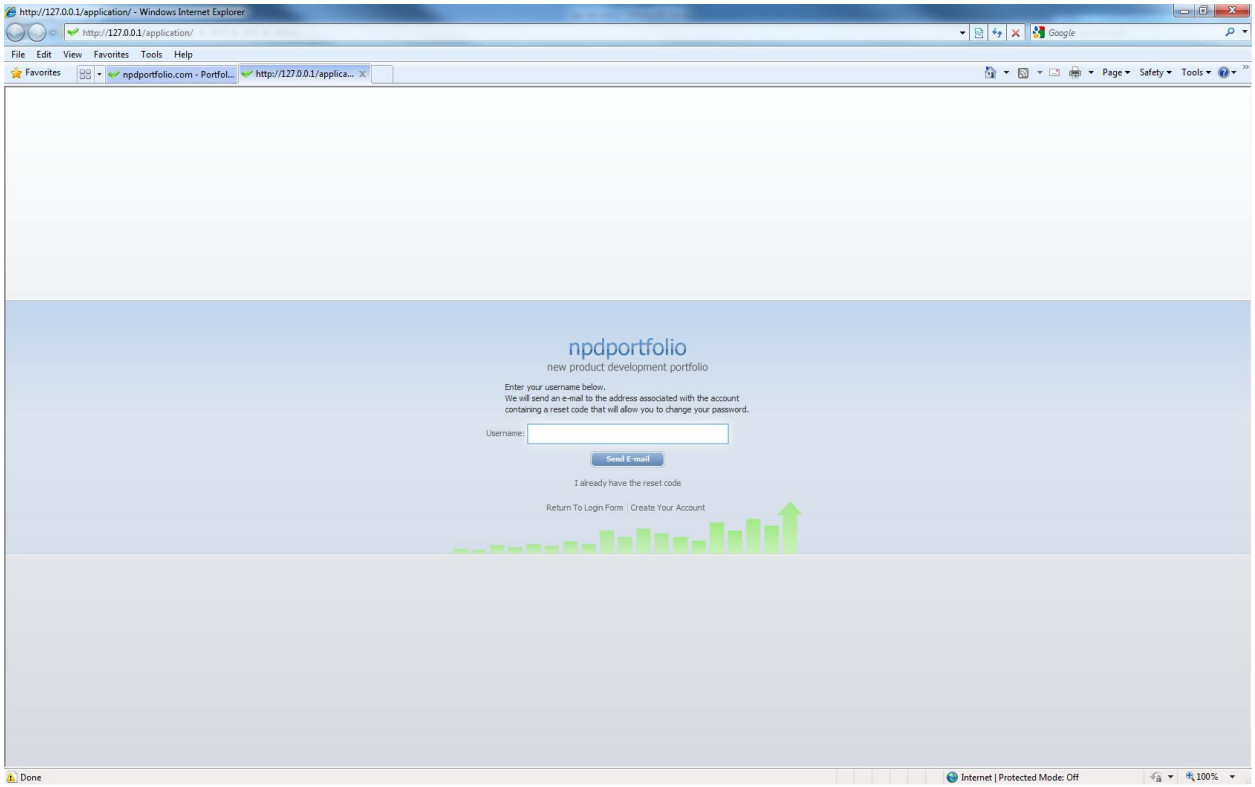


Figure 3 - Forgotten password form

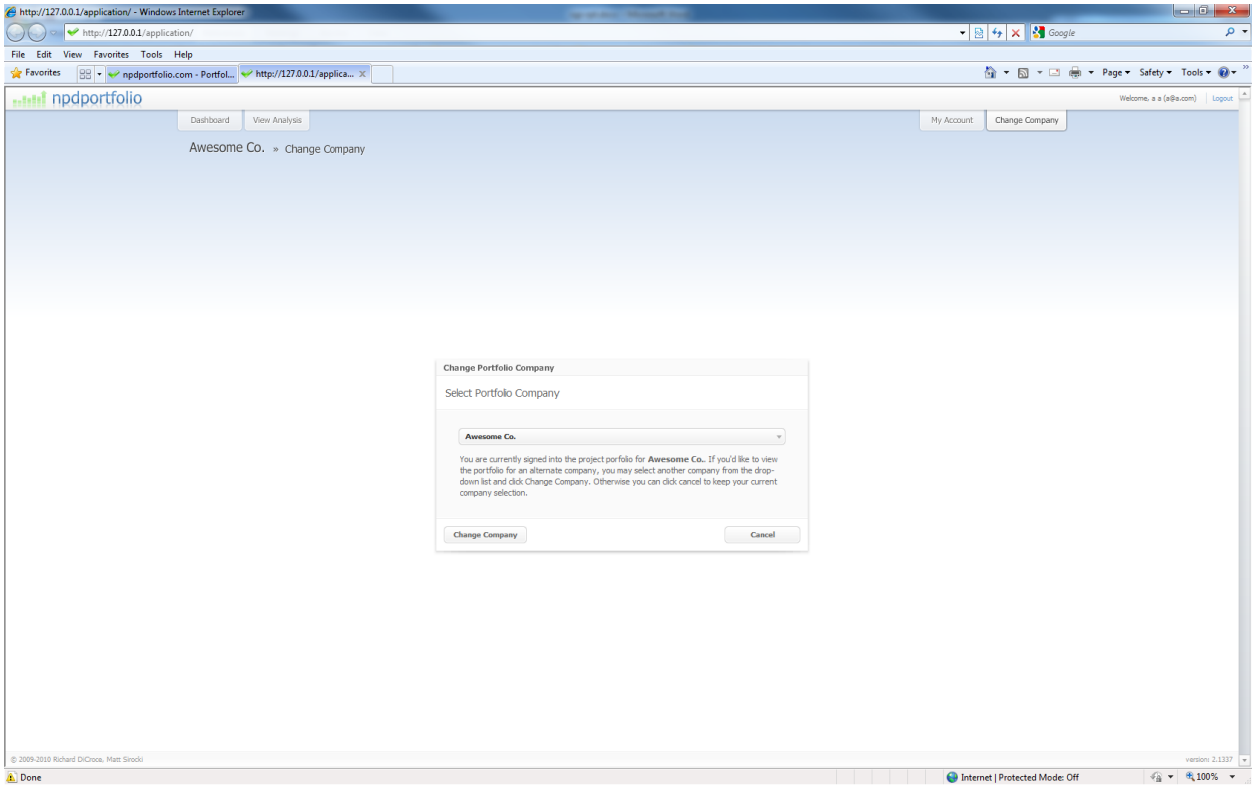


Figure 4 - Company selection screen (only shown if the user is part of more than one company)

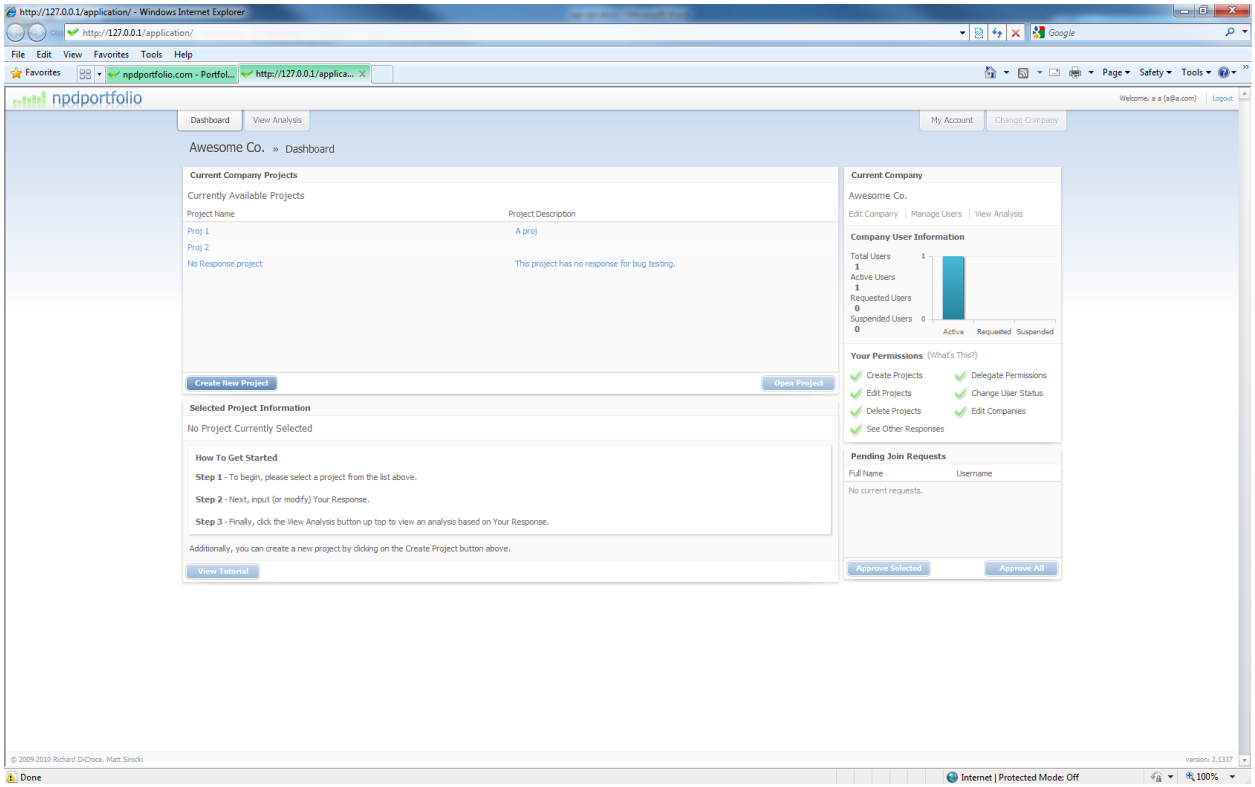


Figure 5 - Dashboard

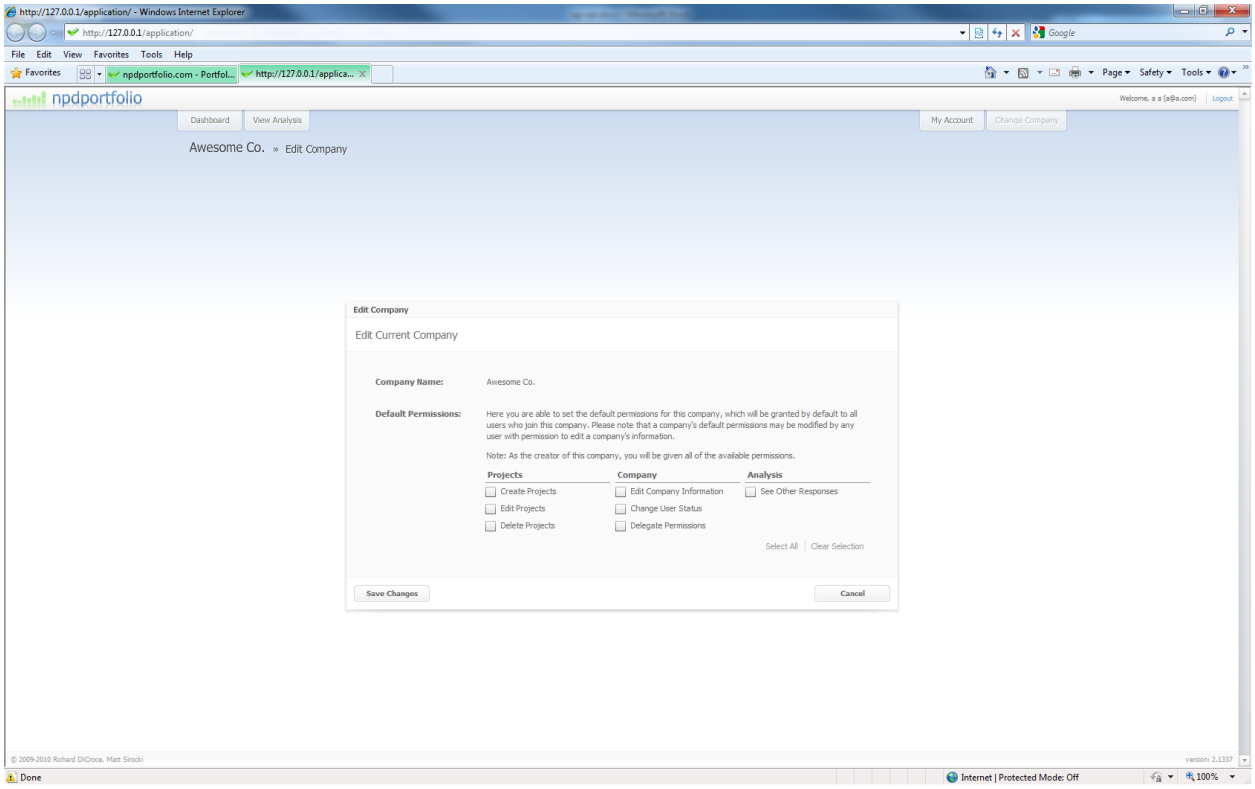


Figure 6 - Editing company information

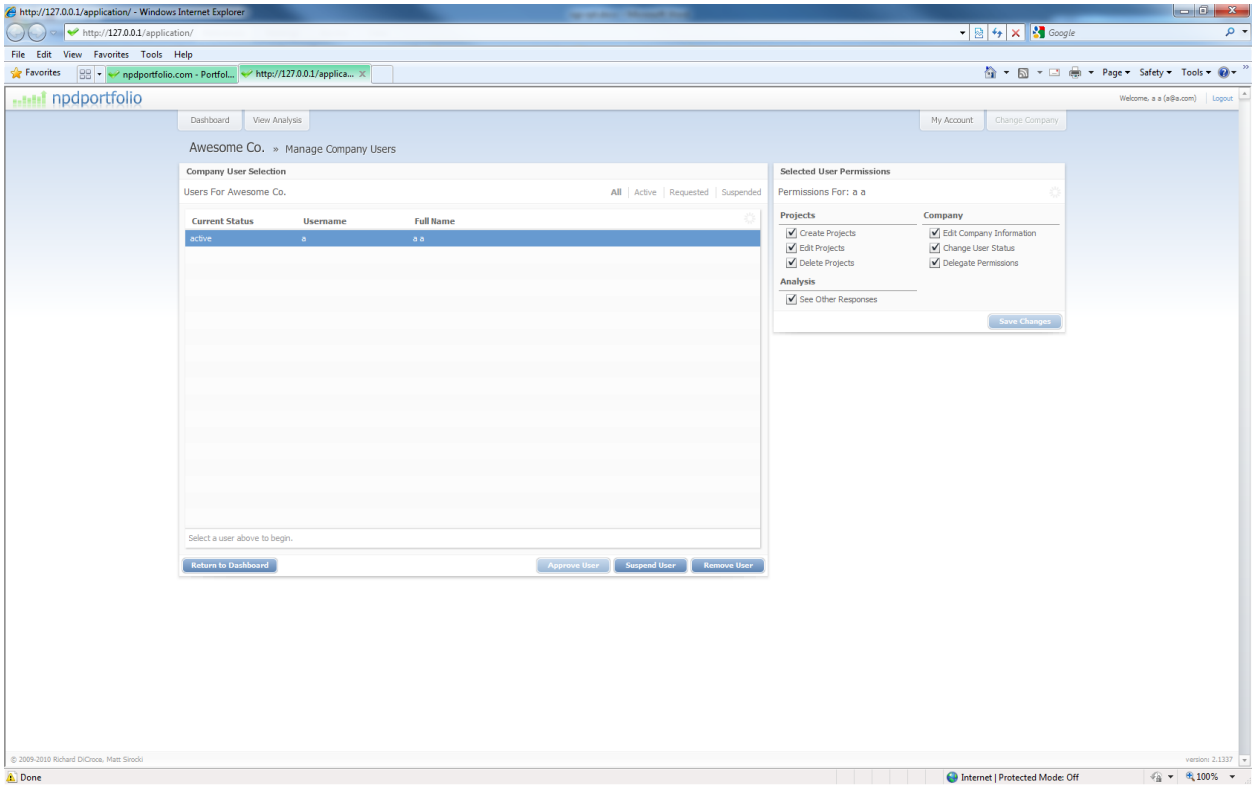


Figure 7 - Managing company users

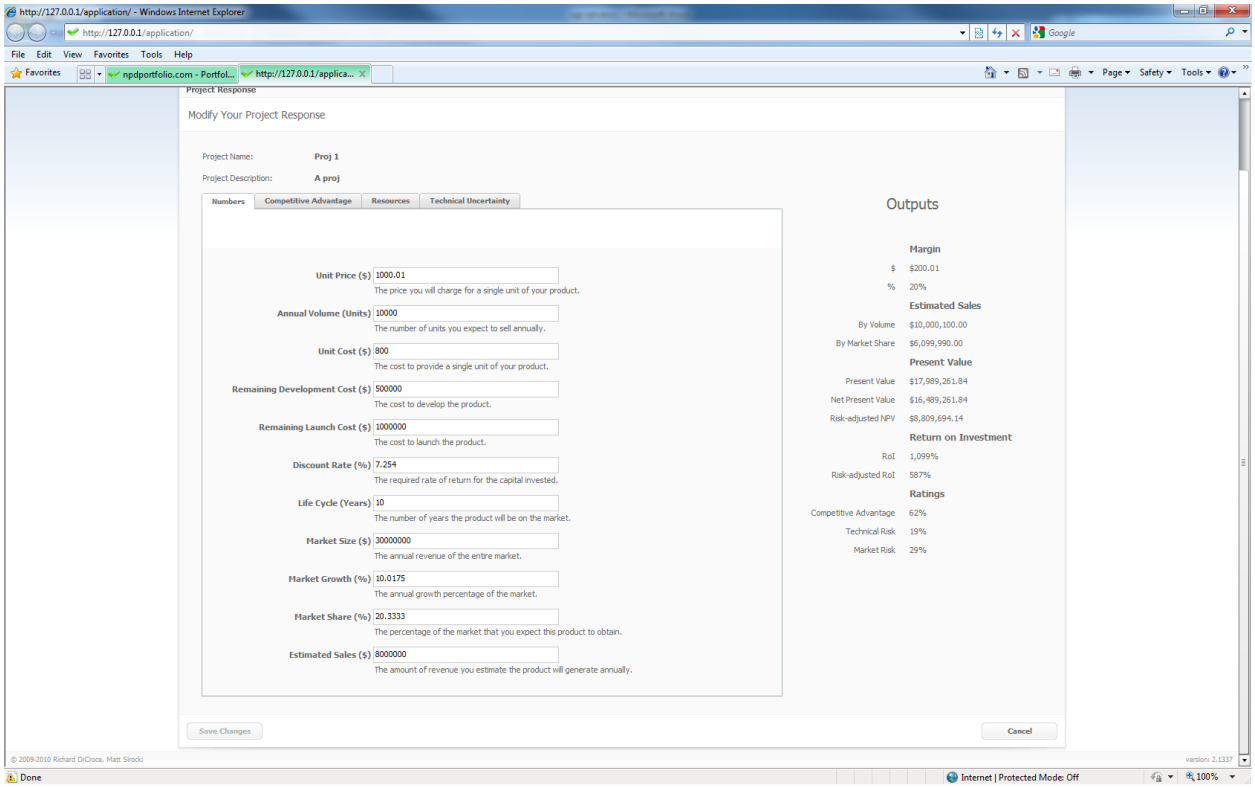


Figure 8 - Input project response (numbers)

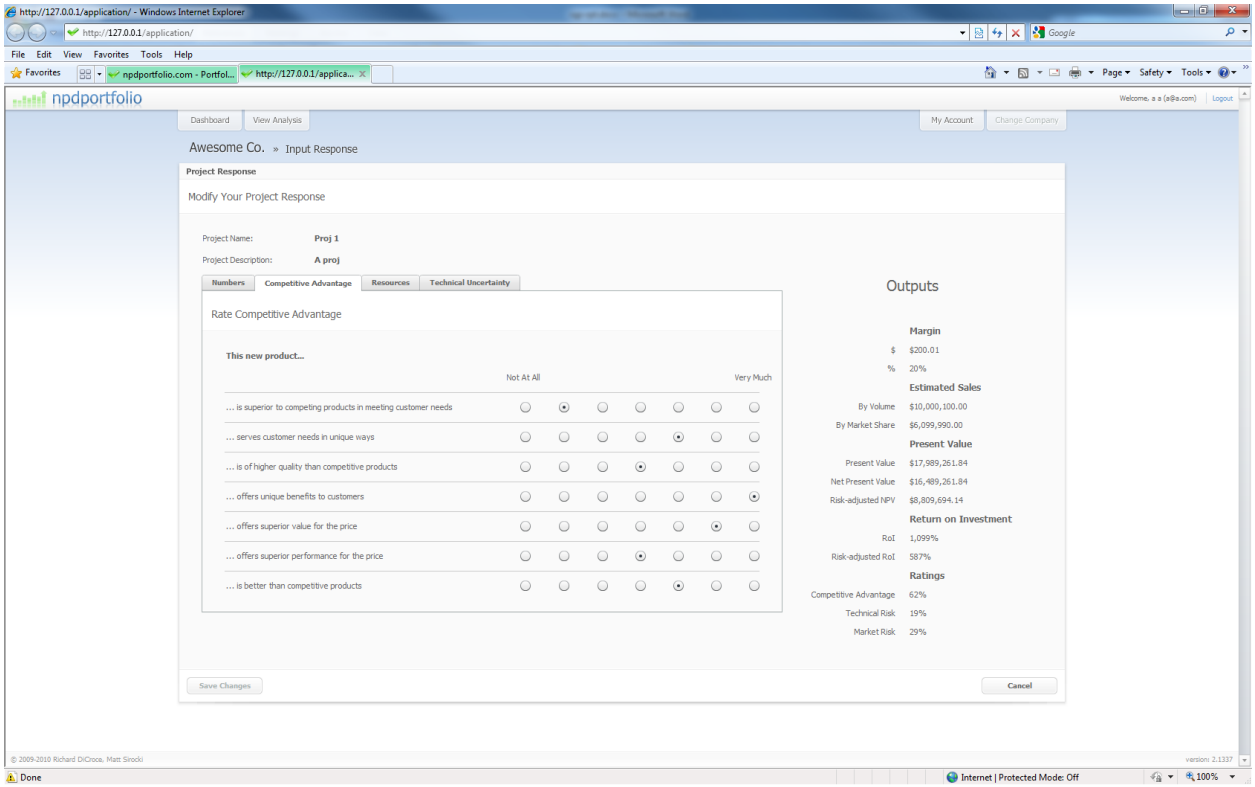


Figure 9 - Input project response (Competitive Advantage survey)

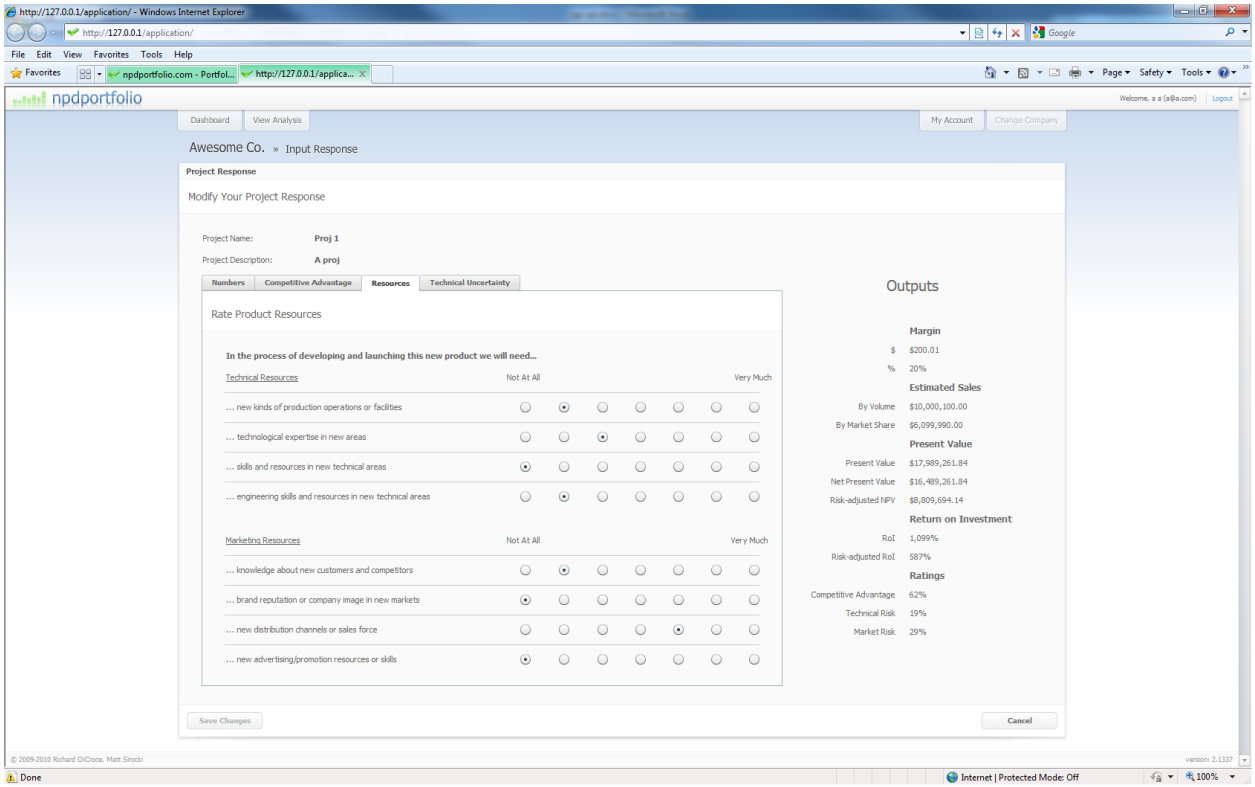


Figure 10 - Input project response (Resources survey)

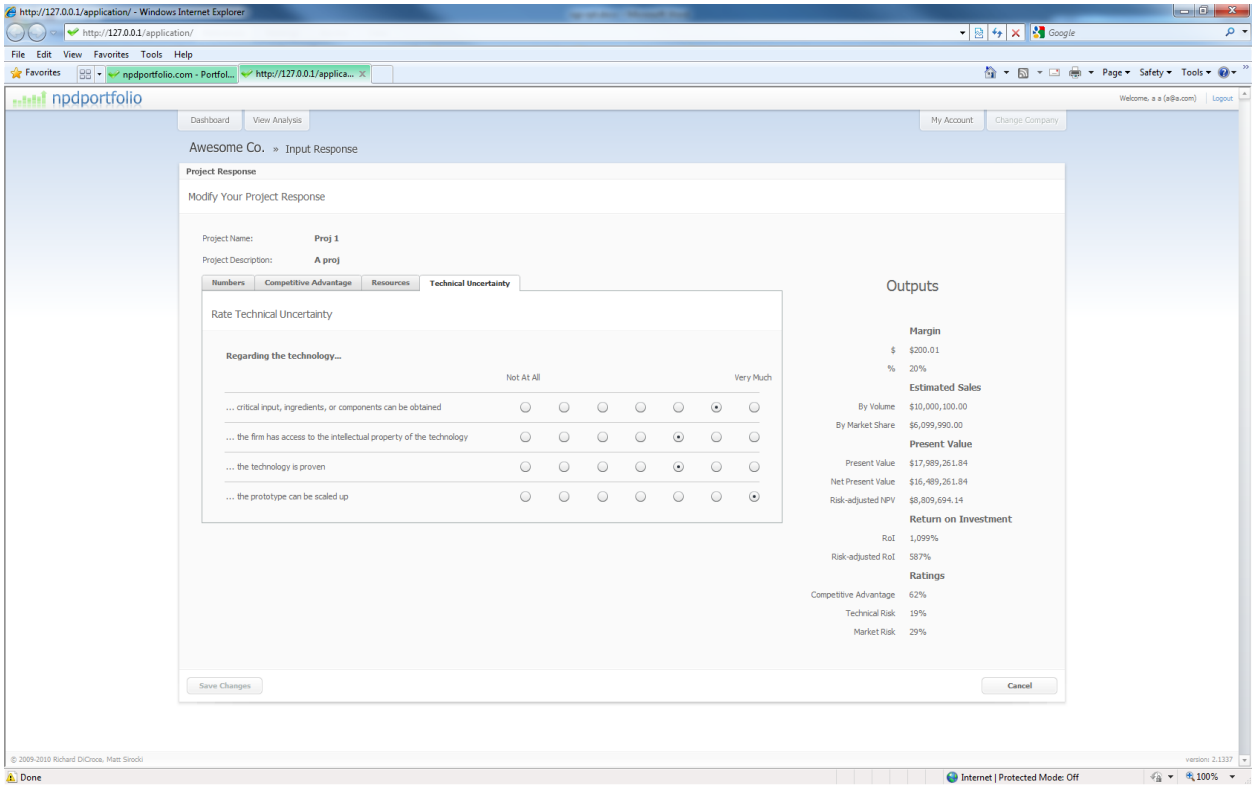


Figure 11 - Input project response (Technical Uncertainty survey)

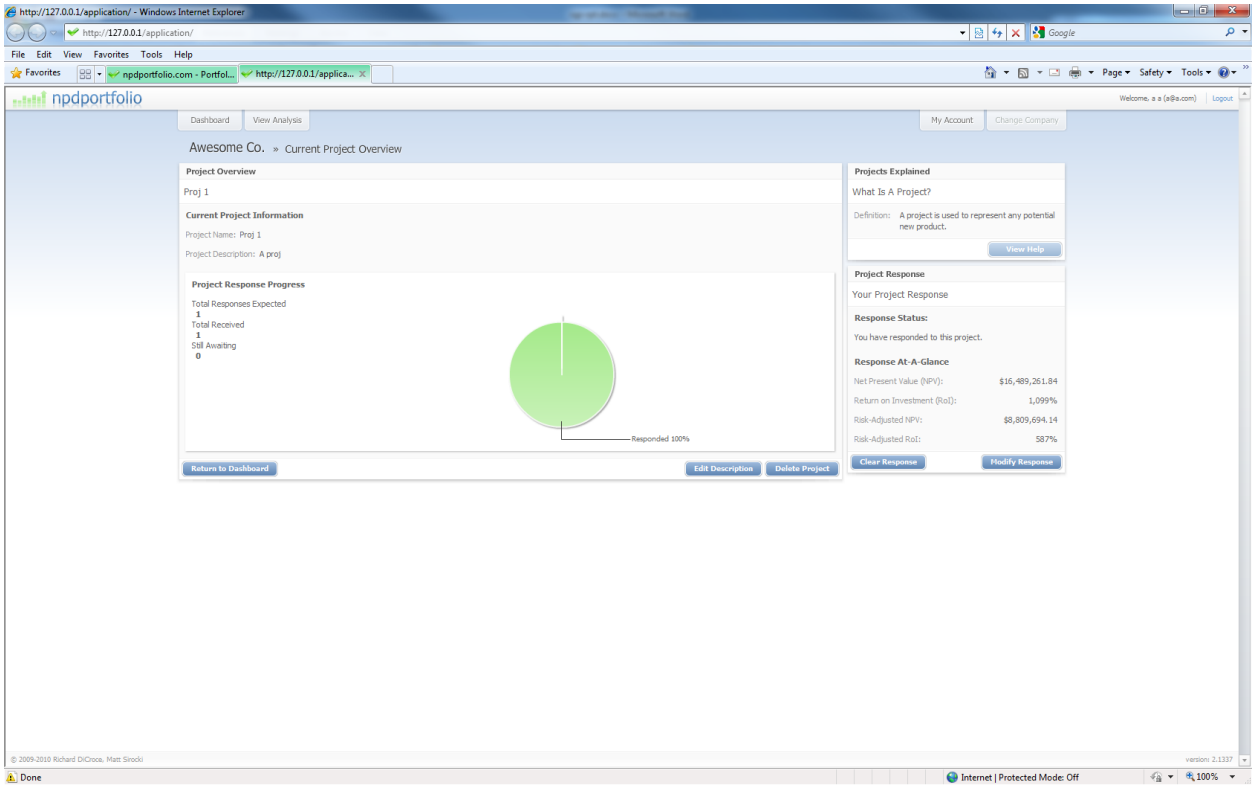


Figure 12 - Project overview screen

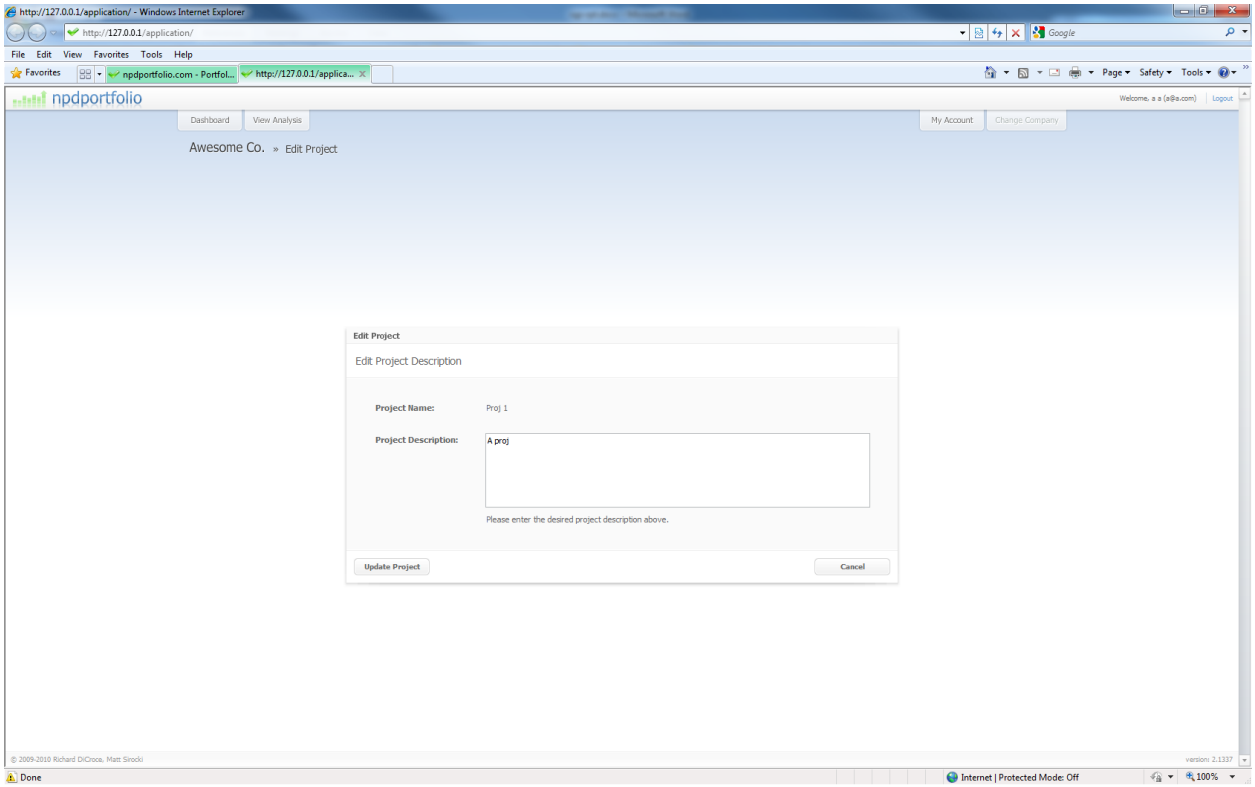


Figure 13 - Edit project

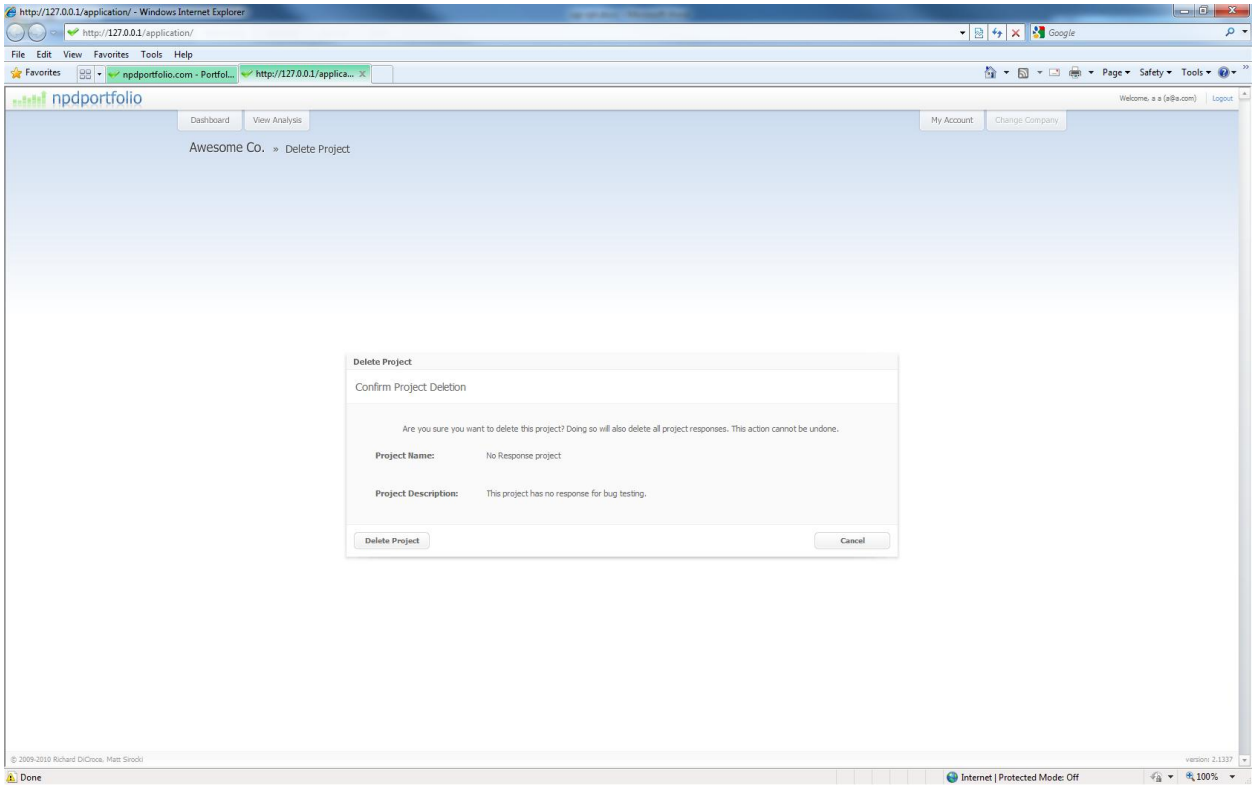


Figure 14 - Delete project

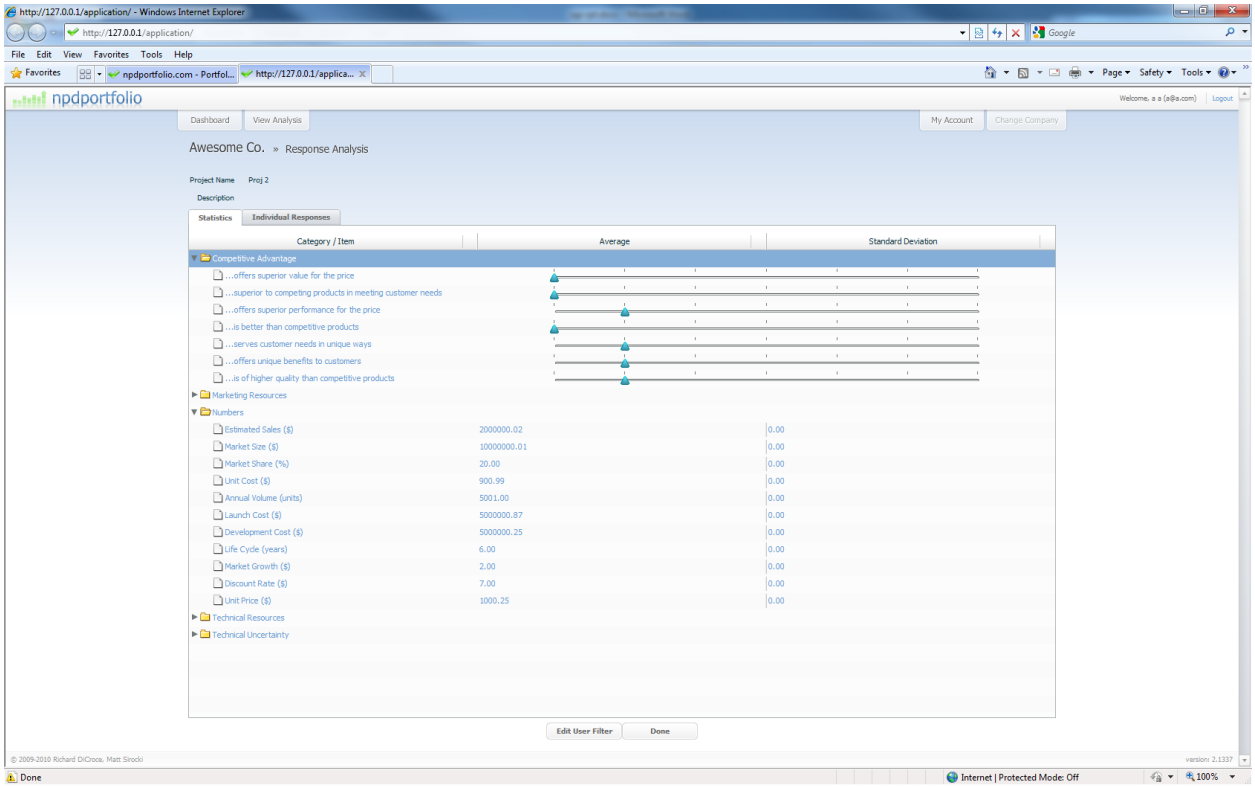


Figure 15 - Comparing multiple project responses

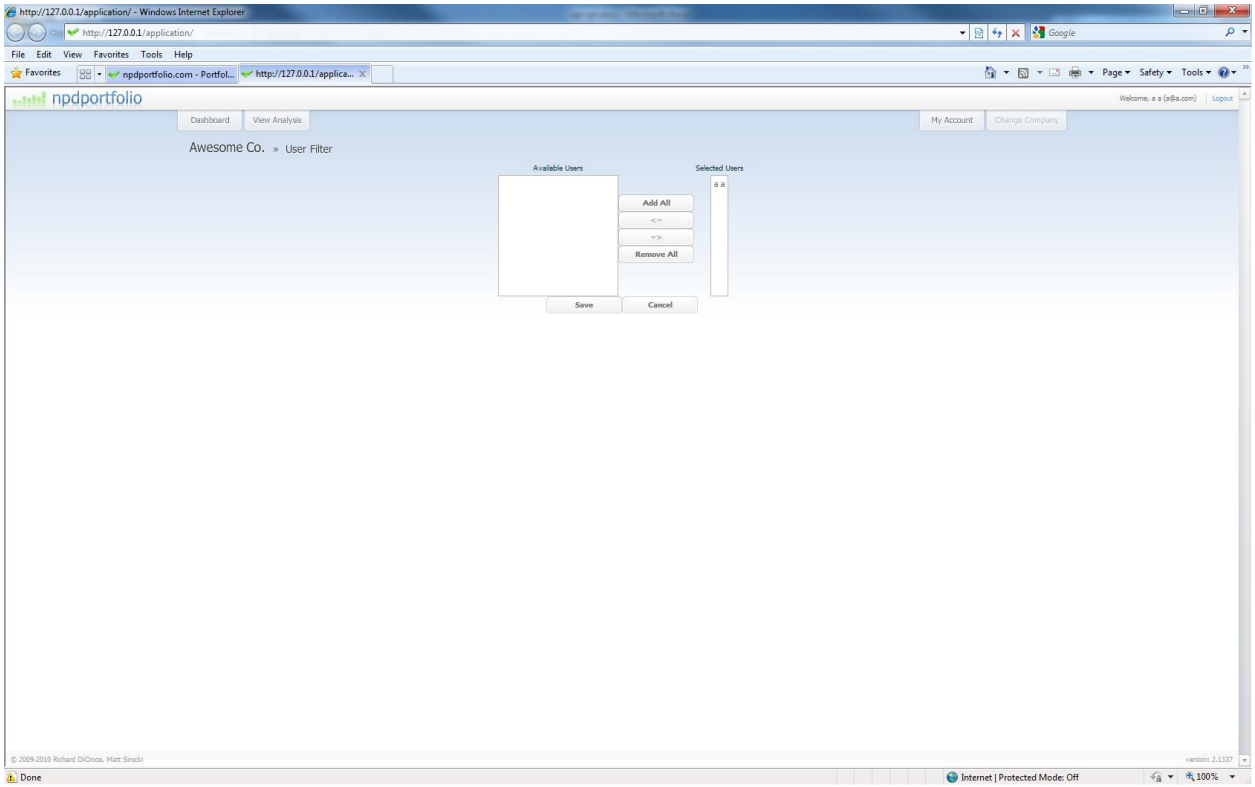


Figure 16 - Filter to select which users to compare project responses for

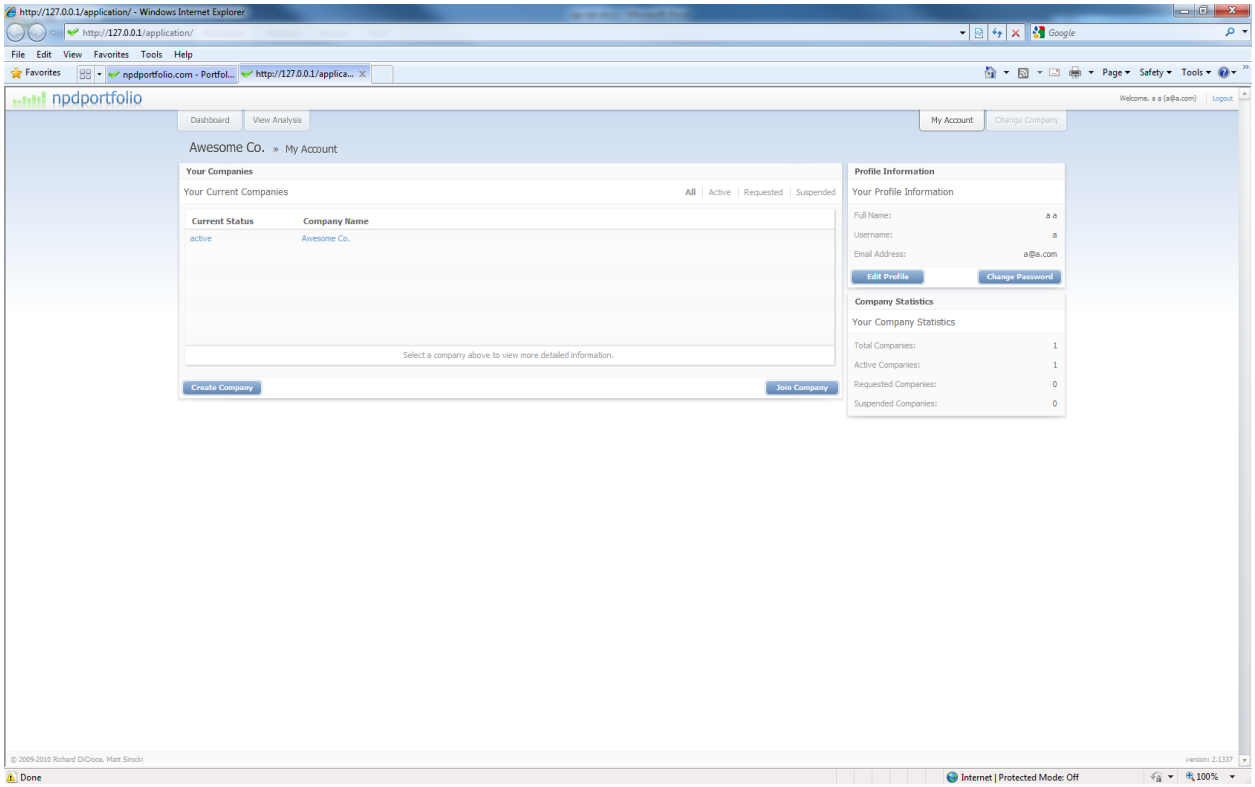


Figure 17 - My Account screen

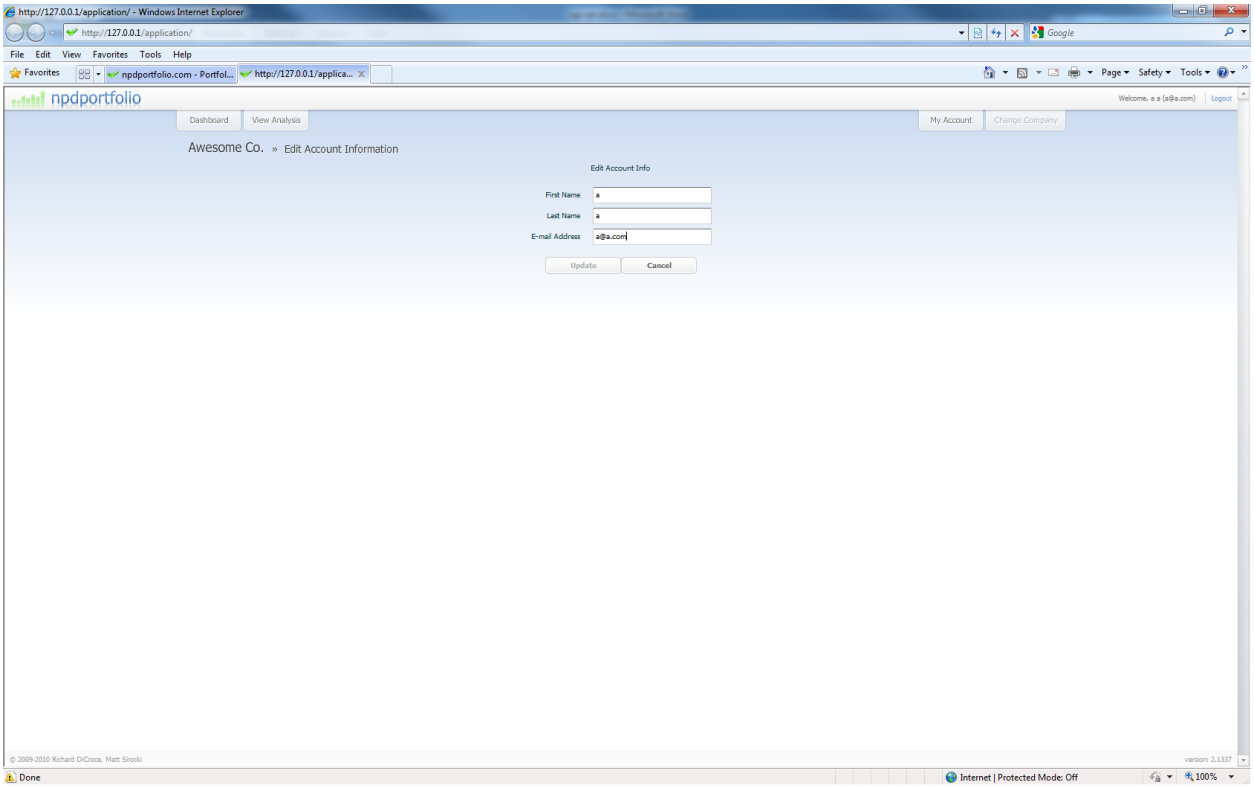


Figure 18 - Editing account information

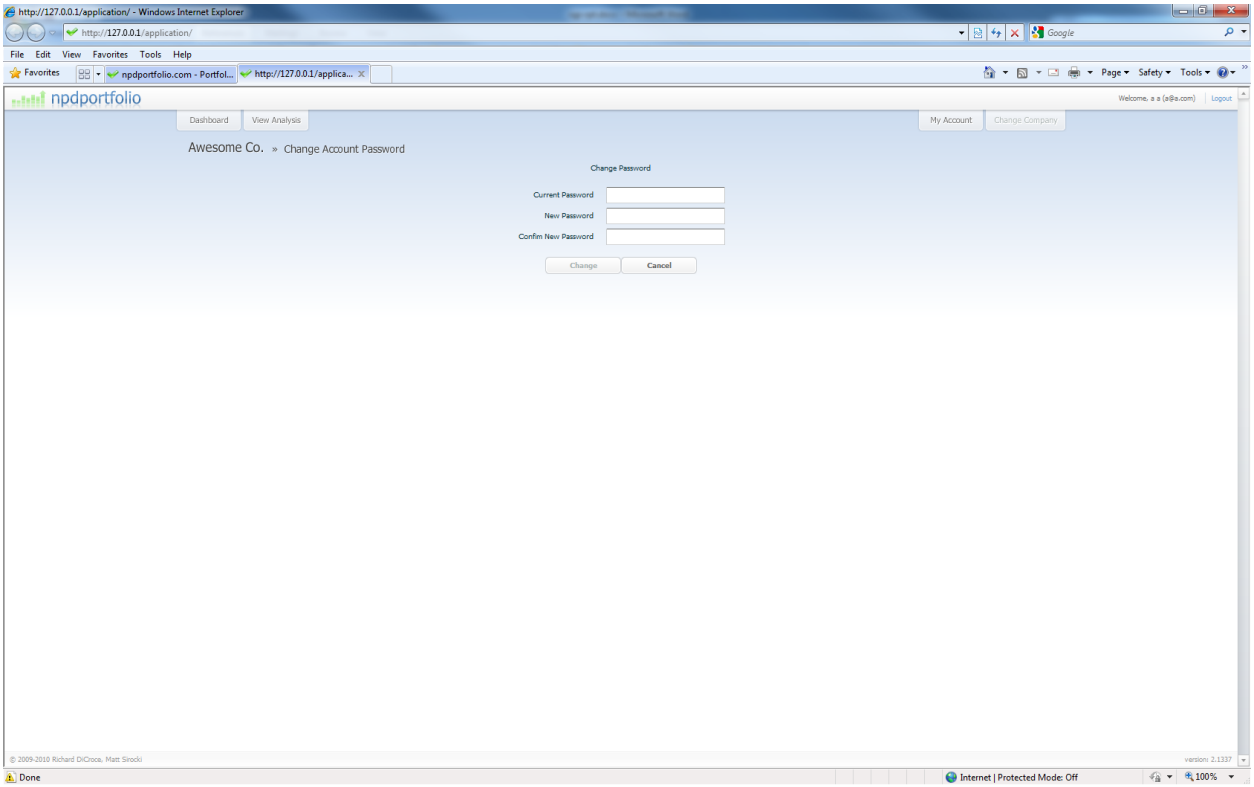


Figure 19 - Change password screen

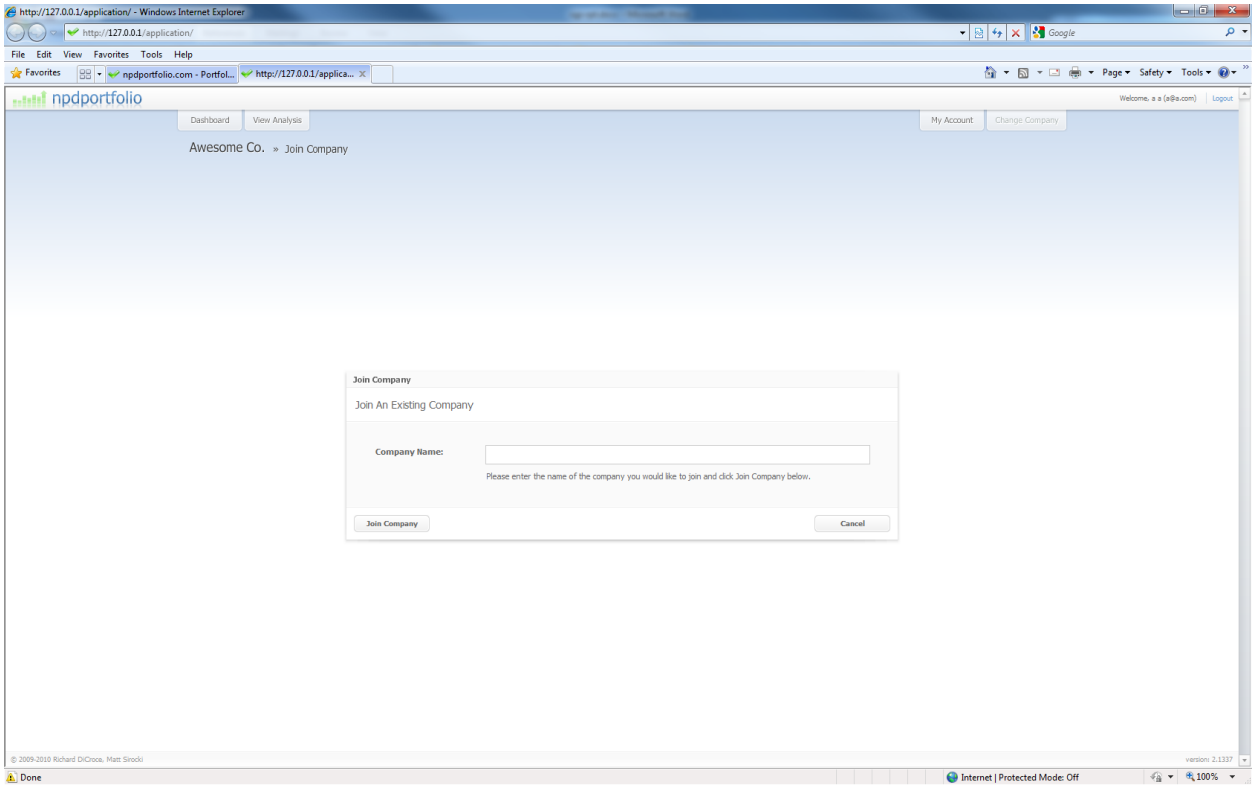


Figure 20 - Join an existing company

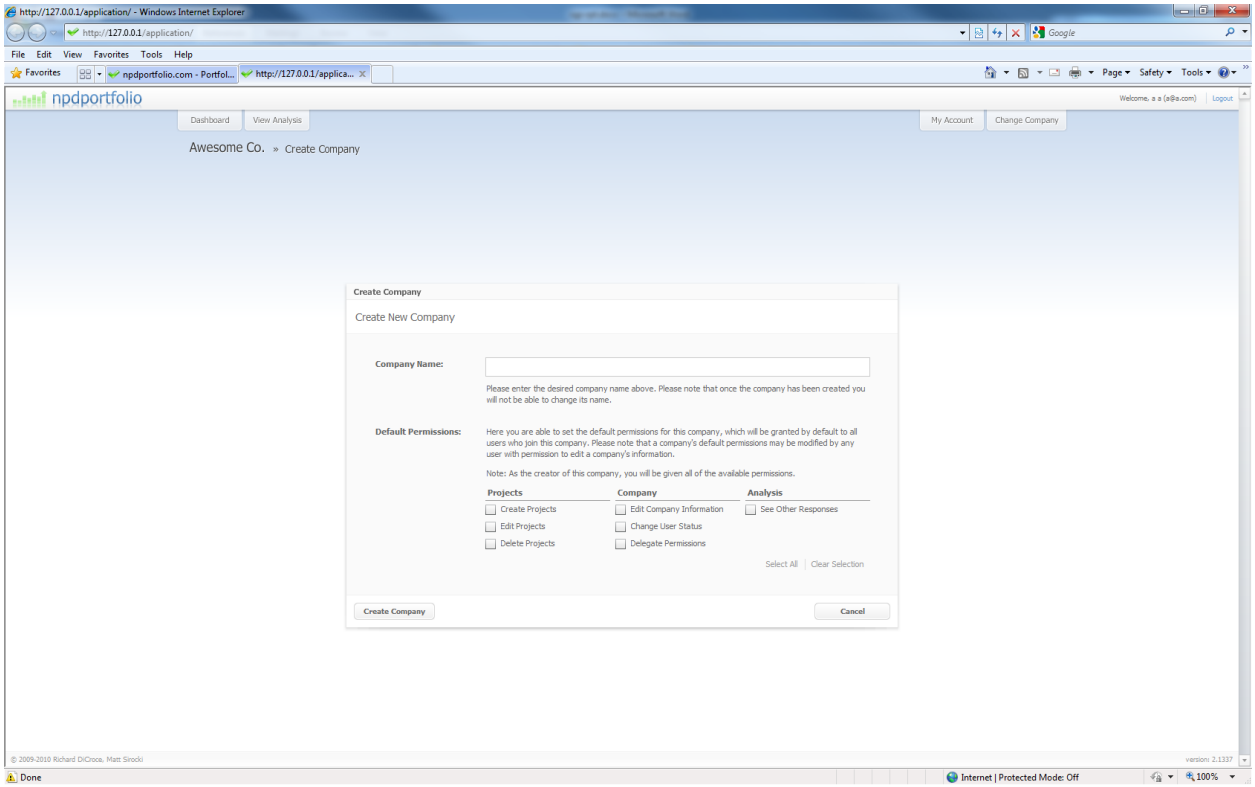


Figure 21 - Create a new company

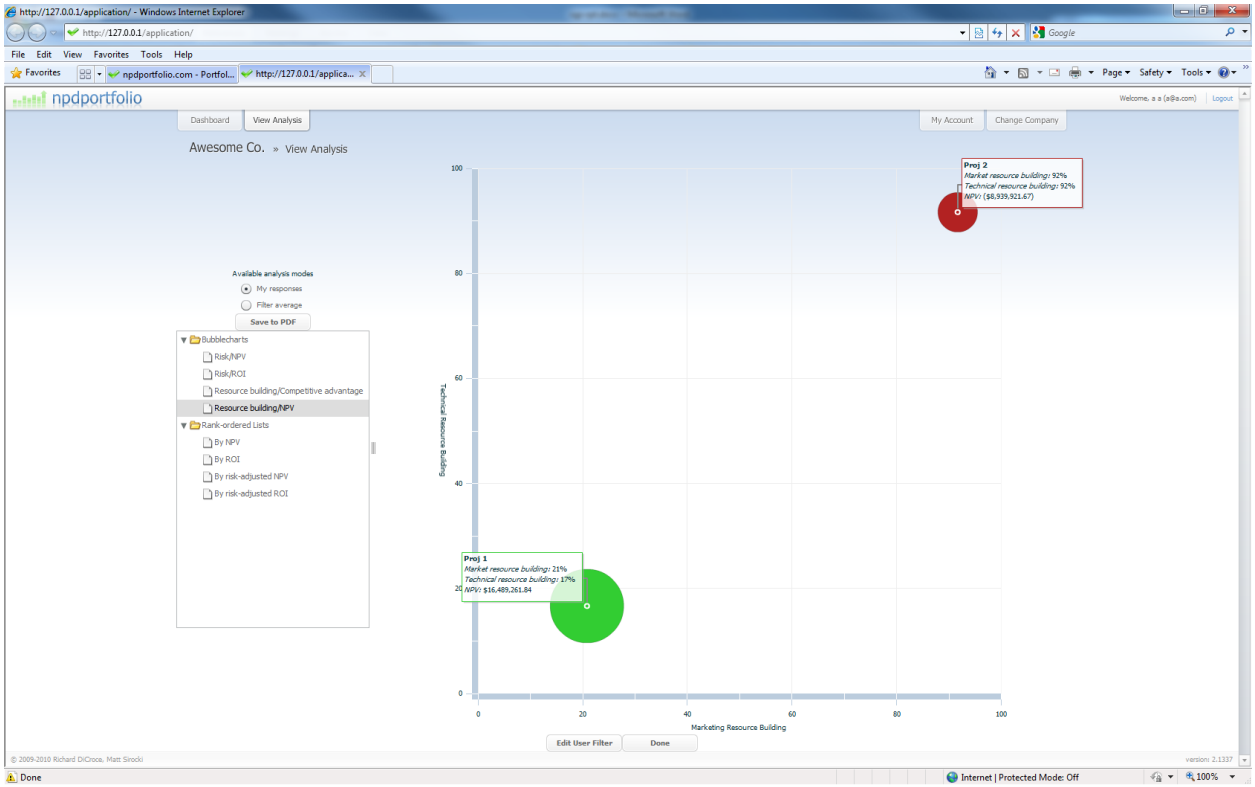
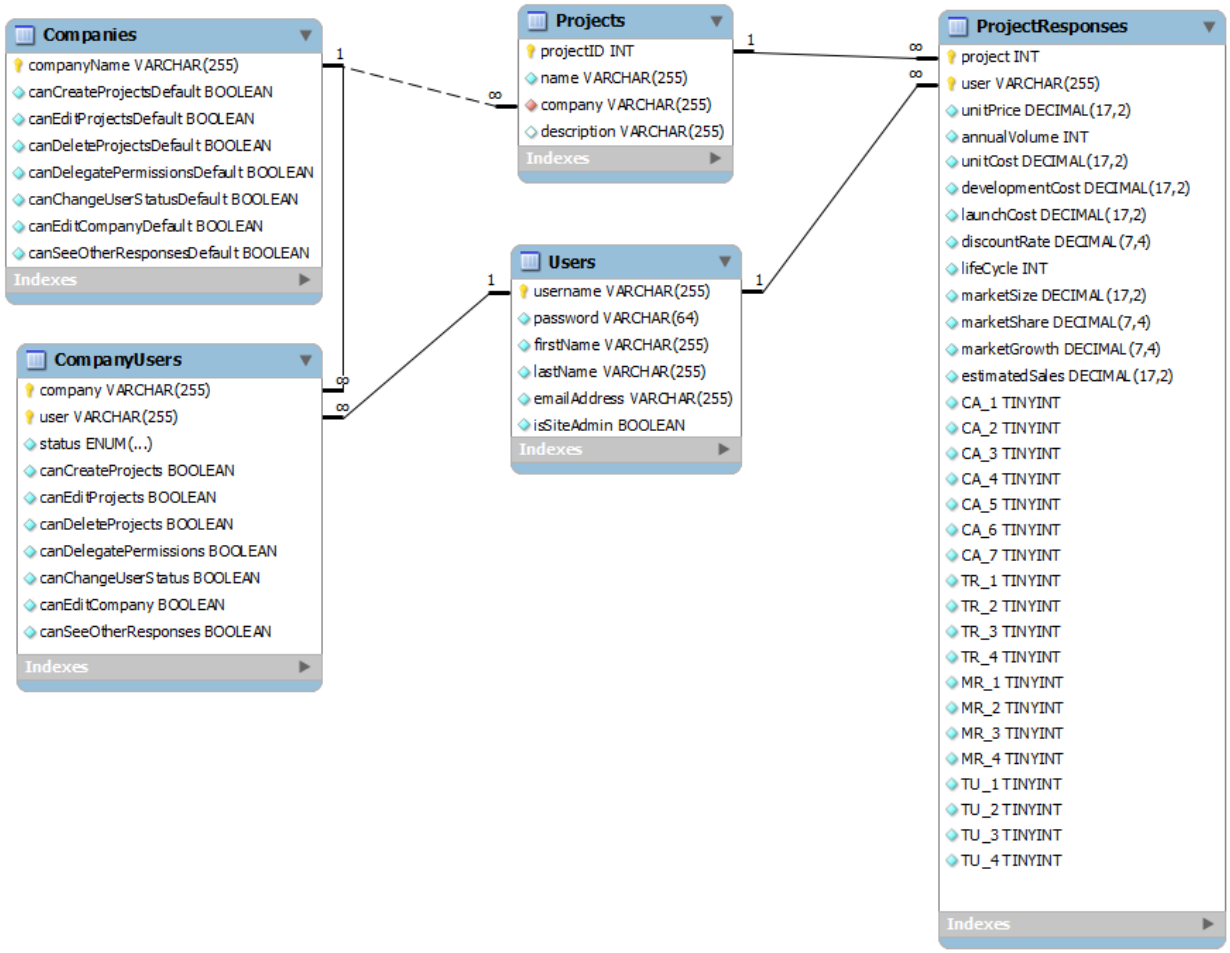


Figure 22 - Portfolio Analysis

APPENDIX B – DATABASE SCHEMA



APPENDIX C – DEVELOPER BLOG

Note that blog posts are in descending order, with the newest posts first. We recommend reading from the oldest posts forward.

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

IT'S OVER!

May 3, 2010 — Richard

Almost. I'm in the process of writing the IQP report now and I hope to have it finished in the next few hours. But I had to stop and tell everyone something. Something huge. Something so disturbing it might make your jaw fall off.

Adobe... actually **did something right**.

I'll give you a moment to compose yourself. Done? Okay, here's what happened.

As I was writing the report, I was perusing through the documentation we've accumulated on the SourceForge wiki. We use two libraries: NuSoap and FlexSpy. I figured I'd check to see if either of them had been updated. Well, lo and behold, NuSoap finally got an update after about three years. I tried the new version but it breaks stuff, so the next team will have to look into that problem. But it got me thinking about the hacks I added back in September so that it would return HTTP status 200 for SOAP faults even though you're supposed to return status 500. See, at that time, Flash Player had a bug where HTTP messages with status >200 got ignored.

Well, apparently, Adobe actually **fixed this bug** at some point! This is probably the reason for some strange behavior I'd been noticing, where things that should have caused errors didn't anymore. I hacked NuSoap to work around their bug, then they fixed their bug and the hackfix broke stuff.

Adobe actually doing something right. It's astonishing, I know. Don't worry though, I'm sure they're already back to being their usual incompetent selves...

Posted in IQP. Leave a Comment »

One More Reason To Hate Adobe

April 7, 2010 — Richard

So if anyone else out there is like me, you've been encountering a car wreck at the intersection of Outlook 2007 and Adobe Reader since October. Or maybe longer.

October was when I started migrating my computers to Windows 7, and since we're in the 21st century now, I figured it was time to bid 32-bit Windows adieu and move on to a 64-bit flavor. Which was when I discovered a really annoying bug. On 64-bit Windows 7 (and 64-bit Vista as well) the Adobe Reader previewer doesn't work in Outlook. So if a PDF is attached to a message, I can't look at it without firing up the Reader application. Not a *huge* problem, just an annoying one.

Not knowing about the fact that it was a problem in 64-bit Vista too (since that was an abomination that I absolutely despised), I figured this was an issue with Adobe Reader that would get fixed soon. HA! Earlier today, I got fed up with this and decided to see if anyone out on the Intertubes had a solution or a workaround. This thread on Adobe's forums was the first thing I stumbled on to. Apparently, this is a bug that *everyone knows about* but Adobe has **ignored** for ***three frakkin' YEARS!***

It gets worse: the fix is drop-dead simple. Change two registry keys and you're good to go! One of the posts in that thread linked to this page which provides a small program that applies the necessary changes. Naturally, as always when dealing with Adobe, there is a catch: Adobe's installer will change the registry keys back to the wrong values when you update or re-install Adobe Reader. So you'll need to hang on to that link and/or the program and re-apply the fix from time to time.

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

Words cannot express how angry this kind of crap makes me. I am seriously going to start a personal boycott of all things Adobe. Shouldn't be too hard since the only thing I use (other than Flash Builder, which is another abomination that will be eradicated from my computers once the IQP is finished) is Adobe Reader. I've heard good things about Foxit Reader, which I'm going to try out. I might make another post later about my experiences with that.

And now, in IQP news:

After I bitched in my last post about how Adobe said that it could take up to two weeks to approve my request for a free FB4 license, Professor Danneels e-mailed us to say that, "previously this approval happened very quickly" so we wouldn't be waiting too long. HA! Earlier this morning, I got an e-mail from Adobe in response to my e-mail asking them to expedite my request:

> Hi
>
> I'm afraid there is a back log due to the large volumes of registrations. Your registration will be viewed shortly.
>
> Alistair Hill

I'm not sure what's more pathetic: the fact that it took them *four business days* to answer my e-mail or the fact that the response is completely unhelpful. It doesn't even tell me exactly *when* they'll review my request, just that it'll happen "shortly" which could be Adobe-speak for a day, a *year*, or an **eternity** for all I know. Their response (or rather, the lack thereof) to the PDF previewer bug certainly is not encouraging. Once again, thanks for nothing, Adobe…

Posted in Epic fails, IQP. Leave a Comment »

The End Is Near?

April 2, 2010 — Richard

Everyone involved with the IQP has been asking that question lately. It sure seemed like the answer was yes… until today.

Before we get to that, let me recap my Tuesday for you. I wasted about four hours trying to work out how to save and print charts and graphs. The code required to accomplish this is actually remarkably simple. It's just that, once again, Adobe fails *horribly* in their internal implementation. When I tried to save an image of a bubble chart, the datatips got cut off. When I tried to print a bubble chart… well, I don't know *what* happened there, but suffice it to say the output was very messed up and completely useless. **FAIL!**

I spoke briefly with Professor Danneels yesterday about this and informed him that saving and printing would be impossible, but that I could probably pull off saving to PDF. He agreed that was better than nothing. I told Matt about this, and he said something along the lines of, "well, it would be pretty bad if **Adobe** Flex couldn't produce **Adobe** PDFs!" He had a good laugh when I told him that functionality actually *doesn't* natively exist – you have to use the AlivePDF library that somebody else made to do it. **DOUBLE FAIL!**

Which brings me to today. I was going about my business online when Matt messaged me and informed me that our Flash Builder trial license had apparently expired. I fired up my Eclipse installation and sure enough, Flash Builder stopped working. Now I don't have a problem with this; it's perfectly reasonable that trial licenses should expire since, after all, they are *trial* licenses. What I *do* have a problem with is the fact that I was just using the damn thing **two days ago** and it couldn't be bothered to tell me that my license **was about to expire**! **TRIPLE FAIL!**

Still, I figured this wasn't really a *huge* problem. After all, Matt and I are students, which means we can get free licenses for some Adobe stuff, including the final version of FB4, which I discovered was released about two weeks ago. So I hopped over to Adobe's website and filled out their form. They have issues where web design is concerned, apparently: having a scrolling frame inside a page that *already scrolls* is terrible design and is really annoying. **OVERFAIL!**

Anyway, they wanted a scan of my ID to prove I'm a student and said it has to have a "current date" on it. WPI does not put dates on their IDs, but in the interest of getting the license key as quickly as possible so we can keep working, I submitted it anyway. I figured that, if they don't accept the request, I can just have Professor Danneels write a letter on WPI letterhead stating that we are students and use that as proof instead (since that's the *only* other proof they'll take). After submitting the form, I was shuffled off to another page that helpfully informed me that it could take up to **two weeks** to process the request! Seriously, Adobe, *two fricken' weeks* to look at an ID and decide whether it's valid or not?!? What exactly are they doing, sending someone to WPI to hold the scanned copy of the ID up to my face to verify that I am who I say I am?!? **FAILTACULAR!**

Sigh So I've sent an e-mail to the address Adobe provided in case I didn't get an e-mail within two weeks asking them to please expedite the request if at all possible, seeing as there are only 3-4 weeks left in the project/term and we really can't afford to be handicapped like this right now. Further updates as I get them…

EDIT: So about five seconds after I posted this, it occurred to me that the trial became time-locked because we're in April now. So I set my computer's clock back to March and Flash Builder started humming right along! **FAILTROCITY!** I've passed this workaround off to Matt, so we are kinda sorta no longer hamstrung. Apparently, two Adobe fails *can* make a win!

Posted in Epic fails, IQP. Leave a Comment »

We're 95% done!

March 19, 2010 — Richard

"We're 80% done!" was a running gag when I took SoftEng in D-term last year. It refers to the fact that software developers tend to get stuck in this loop where the project seems to be perpetually 80% done. I think Matt and I are a victim of that loop too, except at 95% instead of 80%. We *are* almost done. Really. Most of the remaining changes are largely cosmetic, and the bugs we're aware of should be easy to fix.

These things **will** get done by the end of the term, if for no other reason than the fact that Professor Danneels is off to the University of Central Florida at the end of the term and thus would be unable to sign the CDR forms. There are other implications to his departure though. The plan, until now, was to leave a full-bodied project on WPI's SourceForge for the next team to pick up. Presumably, there *will* still be a next team, just not at WPI. Nevertheless, I can't see WPI's SF admins granting user accounts to UCF students.

This creates some problems, given that **everything we've done** is in the SF project. The SF project controls the SVN repository, which contains all of the code we've written. The SF wiki contains helpful notes and warnings about some of Flex's many gotchas. The SF tracker documents some of the bugs we've found and fixed (although we really haven't been making proper use of this feature). The SF Documents module holds some critical files that aren't in the repository, like the database schema and sample configuration files.

So now we have to find a way to "export" all this stuff out of SourceForge, which is ironic because the whole reason for putting everything on SF was to make it easier to hand it all off to the next team. I think what we're going to end up with is a gigantic ZIP file, comprised of the following:

- When we finalize the version 2.0 code, we can export the project from Eclipse into a ZIP file that anyone anywhere else can import. They'll lose the benefit of having the repository though, which would enable them to see how the code evolved and deprive them of the commit comments that explain why certain things were done. In theory, this could be avoided by exporting the repository wholesale, but the web interface doesn't give us that ability and there would probably be complications to doing this even if we could.
- The wiki can effectively be "exported" by simply printing all the pages to PDF and then merging them all together into one giant PDF file. Of course, this will make the embedded links to other resources in the project useless, so we'll

have to rewrite some of the documentation accordingly first. Alternatively, we might be able to copy-paste it into a word processor without too much trouble – this would at least enable us to link parts of the wiki together.

- We haven't been making proper use of the tracker and I'm planning to include a list of any known bugs in our IQP report (it should be noted that our code is completely bug-free… but it may have some undocumented features :-). The IQP report is something that should be included anyway, and I probably would have put it in the SF project for the next team to refer to.
- The files in the Documents module can all just be added to the ZIP as-is, since there's really nothing special about them. As with the code repository though, the revision history will be lost.

Hmm, those caveats are more numerous and potentially problematic than I expected. We should contact WPI's SourceForge admins (although we'd have to find out who they are first, since I haven't the slightest clue) to find out if they have a policy for situations like this (and, if so, what it is). If we're lucky, we might be able to strike a deal or something.

Of course, first, we have to finish that last 5%. This could prove challenging. All the classes I'm taking this term are heavy on projects and will be very time-intensive; Matt is in one of these courses with me, *and* he's taking an overload (4 courses instead of 3). Thus, progress is likely to come slowly in fits and starts. Also, I haven't even begun writing the IQP report yet, which is liable to take a while; an explanation of what we did and why is going to involve explaining all the previous team's screw-ups. Luckily I have my blog to help me with that. Although that could be a bad thing – the discussion of what the last team *didn't* do could end up being longer than the discussion of what we *did* do…

Posted in IQP. Leave a Comment »

Post-mortem: The Beta Bug

February 26, 2010 — Richard

Around 11AM today, I uploaded our first beta build. There was just one problem: it didn't work.

This was something I could not understand. Everything worked perfectly in our development environment, so what could the problem possibly be? All of the error messages and sanity checks in our code were for naught: no errors were being displayed. Even the Flash Debug Player was of no help: there were no NPEs, no errors or stack traces of any kind. So there was no indication of what was going wrong, and yet something was *definitely* going wrong: attempting to create a new user or login caused the loading spinner to spin indefinitely.

I didn't have time right then to figure out what the problem was, so I fired off some e-mails and briefly discussed the issue with Matt over IM. The only things we knew were that beta1 did not work, but alpha6 worked perfectly. Around 6PM I parked myself behind a monitor and started looking for what I had nicknamed "The Beta Bug". I started by performing a binary search: since alpha6 worked, some change between then and now obviously had to be the culprit. So I started testing revisions to try to determine where the problem was introduced. At some point in this process, I left the browser window open while preparing a new revision and discovered much to my amazement that it worked – it was just really, *really*, **really** slow. As in, slower than dial-up slow.

So then I wondered if this was affecting other revisions. I tried alpha6 again, and lo and behold, it was now experiencing a speed problem as well. So I tried beta1 again, but that still seemed to hang indefinitely. We never had any speed issues before, so I figured these things had to be related somehow. Since there were no error messages to go on, I decided to use Wireshark to look at the raw traffic that was being sent through the tubes. One capture run later and I'd found the problem: there was a PHP warning before the XML schema. In the past, this rightly caused Flex to complain. Loudly. Apparently, somewhere along the way, this became a situation in which Flex just fails silently. Great job, guys.

With the text of the problem in hand, I easily fixed it in just a few minutes. I uploaded the new PHP script and beta1 started working, albeit *very* slowly, like the other builds. I informed Matt of my success a few minutes later, and when he

tried it, he told me it was much faster. So I tried it again, and the situation seemed to be pretty much back to normal. Exactly what happened there is anyone's guess. Maybe the PHP errors were being logged somewhere and it was slowing down the works?

In any case, while we're pretty close, we're not done yet. The beta1 build has five known bugs, and Matt is still restyling stuff. My hope is that we'll have the bugs fixed by Monday or Tuesday, at which point we'll upload a second beta. That build will become the official version 2.0 if no other bugs are discovered. At least, that's the plan, but we all know how my plans usually go...

Posted in IQP. Leave a Comment »

BETA, BABY!

February 25, 2010 — Richard

PARTY TIME!!!1!1!oneone

Yeah, you heard me right! We have officially entered the thought-to-be-mythical land of **Beta**! This comes after an all-day marathon coding session in which I completed all three remaining items on the **Beta** List. Of course, in doing so I created or discovered a variety of bugs. Fortunately, Matt informs me that most of them are already fixed in his working copy. He's hoping to finish that tonight or early tomorrow so we can include those changes and bugfixes in our first official **beta** build. Yes, I am going to bold every instance of the word **beta** in this post because I am just that psyched right now!

Of course, there are still issues to be dealt with. Besides the aforementioned bugs, there are still some bugs in our tracker that need to be taken care of. The Filter bug in particular is a nasty one and it impacts critical functionality, so my top priority now is squashing that critter. There's no way a fix for that bug will make it into tomorrow's build though because I probably won't have any more time to work on this until Saturday. As it is, I already broke the 10PM Rule today (only by about 20 minutes though, so I guess it isn't *that* bad).

Also, I feel a need to correct a misstatement I made last week. I said that there was no way to compare two objects based on the values of certain properties without writing your own loops and sprinkling them all over your code. That's actually not true: there are some *findX* methods that do this. It still doesn't change the fact, though, that you have to build the necessary input every time, which is still annoying.

And to make up for that, I have a new epic fail to report. I changed a function so that, depending on a certain condition, one of two SOAP operations would be called. The two code paths are mutually exclusive. Despite this, MXMLC still throws an error about duplicate variable definitions. So I added a "2" to the variables in the *else* clause. Not hard to work around, I'll grant you, but still damn annoying.

Also I should report YAAEF that Matt and I have been dealing with, but I kept forgetting to post about. If you use only static methods of a class in your <Script> blocks, Adobe's import organizer helpfully decides that you aren't using that class and *removes the import!* I've gotten burned by this dozens of times now, when I edit an MXML file, change something that triggers the import organizer to run, then had the build fail because the organizer decided to make things too neat and removed an import that is actually necessary. D'oh!

Anyhow, that's all from me for this week, folks! Check back over the weekend and next week as Matt and I look to quash any bugs we missed, finish re-styling the GUI, and write the IQP report.

Posted in IQP. Leave a Comment »

The 10PM Rule revisited

February 19, 2010 — Richard

Three posts in three days?!? Can it really be?

Yes! It can! All thanks to the 10PM Rule. I was really pissed off last night, but 10PM rolled around so I stopped working. This was a *very* good thing, because

my fatigue and my anger were blinding me to the obvious solution to the problem at hand. This morning, I had an epiphany when I woke up and finished the analysis revamping in 20 minutes. Some user-friendliness improvements would be desirable, but it all works.

I've also decided that we **will** reach beta next week if it kills me. Of course, it probably won't, because the other three items on the Beta List will be considerably less trouble to implement (famous last words!). After that, it's all tweaks and bugfixes. Well, and meshing the application with the website, which we still haven't done, but given the way we plan to do it, it shouldn't take too long. Then, of course, we have to write the IQP report.

Okay, so I guess there's still a ton of stuff to do, but at least the end is finally in sight, which is a damn good thing considering that the term ends in two weeks. I plan to work on this over the weekend, and Matt's been hard at work as well. I'm hoping that by Monday we'll be able to produce our first beta build. More to come...

Posted in IQP. Leave a Comment »

To beta or not to beta

February 18, 2010 — Richard

So yesterday and today became all-day marathon coding sessions in a headlong rush attempt to get the "beta list" done by tomorrow. One could say that this effort is not going so well.

The problem of the day today? Trying to decide whether two objects are the same thing. It sounds so simple, doesn't it? And if we were using Java, it would be. Just override the equals() method and you're good to go.

ActionScript, naturally, has no such method. The documentation of exactly how comparisons work is vague at best, so I have no idea if an equality test is checking each of the object's properties or if it's just checking to see if its the same object (that is, the same memory location). I *think* it's the former, at least in general, but I have no idea how consistent this behavior is. Not that it matters, because when I want to compare two User objects, all I care about is if the username is the same. There's no way to make this comparison (as far as I can tell) without sprinkling $O(n^2)$ loops throughout the code. This is another one of those things that is unbelievably annoying.

Also annoying is this "duplicate definition" warning I just got. In Java, if I create a for loop and create a variable i in the loop definition, it gets GCed and goes away after the loop is over. Apparently, not so in ActionScript.

Anyhow, if you can't tell, I'm more than a little pissed off now, so after being at this all damn day with not much to show for it, I'm quitting for the night. No, beta is not happening this week; of the four items on the list, the filtering refactoring is about 50% done and everything else is 0% done. Over the weekend I might do some work on those other items and come back to the filtering on Monday.

Posted in IQP, rants. Leave a Comment »

More fun with SOAP

February 17, 2010 — Richard

This week's top priority is replacing the "master response" analysis system with the "filtering" system. A few minutes ago I made a commit to begin that process. Things are a little ugly right now. This change is non-trivial and required removing an entire database table, condensing two company permissions into one, and modifying or removing numerous SOAP operations and data types. Everything is patched up on the PHP side, but on the Flex side, I just commented out some buttons (and screens, by extension) in order to get the rest of the application to compile; they haven't actually been adjusted in accordance with the changes in the back-end yet.

In the process, I encountered YAAEF. Big surprise, right? Well this one is just utterly ridiculous. Since the master response stuff had to go, the SOAP operations attached to it had to go as well. It turns out that Flash Builder's auto-

generator fails epically at removing SOAP operations. As in, **it doesn't**.

Yes, you read that right. You remove a SOAP operation, and Flash Builder just closes its eyes, plugs its ears, and starts whistling, "I can't hear you!" It's not even enough to manually remove the multitudes of now useless generated files; the FML file still holds data on all the operations you removed, so you have to go through the FML file and manually strip out all the references to the removed operations and data types as well. Fortunately, the FML file is actually valid XML, so you can use an XML editor to make your life a little easier. And trust me, you'll want to, because without one, the file is more impenetrable than the WSDL it was generated from. I have no idea why Adobe decided they needed a new file extension; not using .xml for an XML file is an epic fail unto itself.

The real question is *why it doesn't remove the damn stuff for you!* It's not as though it's useful to keep the code hanging around; the only thing that does is make your life a living hell by making it so the code still compiles, but fails at run-time because the referenced operations **don't exist any more**. Seriously, I don't get this at all. I've run into a lot of epic fails where Flex is concerned over the last six months (holy crap, I didn't realize I'd been at this for that long until just now!) but somebody over at Adobe *really* dropped the ball on this one.

While I was in the middle of doing all that, Matt was having fun styling our application, specifically, trying to embed fonts. Professor Danneels also e-mailed us to ask whether we'll finally have our infamous beta release on Friday as scheduled. Right now, I have no clue. I sure hope so, but there's still a long way to go in a short period of time. Implementing the filtering is going to require creating a new screen, refactoring some existing screens, creating a new singleton, and implementing some new SOAP operations at the very least.

The good news is that the rest of my week is fairly open, so there's a decent shot this can be accomplished. All-day coding marathons, here I come!

Posted in Epic fails, IQP, rants. Leave a Comment »

The last two weeks, take two

February 10, 2010 — Richard

Okay, so it's been two weeks **again** since my last post. I know, I really shouldn't let it go that long, but it's easy to lose track of time. Anyhow, this should be a fairly quick post since it's closing in on 11PM and I want to close up shop for the evening.

Since my last post, nothing incredibly spectacular has happened with the IQP. Matt has been working on making the application prettier and more user-friendly while I've been plowing ahead with the unimplemented functionality. My focus lately has been on account management features since those comprised the vast majority of the unimplemented functionality. Well now they are all implemented: last week I implemented updating your user profile (i.e. name and e-mail address) and changing your password. I also prototyped an idea for the "forgot password" system last week; today I spent several hours implementing it.

I can also say that we now have a (relatively) well-defined set of goals that must be met before we reach beta:

1. Implement removing users from companies.
2. Implement editing project descriptions. (We will not allow changing the name; the company name/project name pair is how we identify projects when we're tossing data between the client and the server. Allowing the name to change would create a massive concurrency problem that we are not presently equipped to solve. Something for the next team to tackle.)
3. Implement changing company default permissions.
4. Revamp the analysis system as discussed during last Monday's meeting.

That last item is going to be the most complicated. We haven't even decided exactly how to do it yet; odds are I'll spend most of tomorrow tinkering with ideas. Once that's squared away, the other three items are relatively trivial to implement. I hate to make any statements containing dates, since we've been so far off in the past, but we're coming down to the wire here, so I'm going to do my best to have all four of those things accomplished by the end of next week.

On a semi-related note, I've decided that the post ranting and raving about how horrible Flex is will be postponed until the end of the term. That way I won't have to make an update post if I discover more epic fails in the last three weeks of the project.

Posted in IQP. Leave a Comment »

« Older posts

CaptainRichard's Blog

Pages
- Home
- About
- To-do List

Categories
- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

The last two weeks

January 28, 2010 — Richard

So it's been over two weeks since I posted anything. The reasons are two-fold. At first, there just wasn't anything worth posting about. Then I got so busy I didn't have time to post anything. Also I was in this weird funk for a while where I just couldn't seem to focus on anything.

Fortunately, I've adapted to the new term. My schedule this term is very irregular and, oddly enough, results in my having zero classes on Friday. Sounds wonderful, doesn't it? And it would be… if it weren't for the 8AM class I have on Mondays and Thursdays, one that's two hours long, no less. Do you think a legal ban on 8AM classes would pass Constitutional muster?

Anyhow, this term is going to be busy for me. Engineering Economics is not exactly a walk in the park (although it might help if class didn't meet at a time when everyone is still half asleep!) and the inquiry seminar is not going as well as I'd hoped. The fact that the seminar only meets once a week for two hours seems very unhelpful to me; it would probably be better if it met twice a week for one hour, because it's easy to forget about it. The IS is going to culminate in a historical paper; finding time to work on it should be interesting since Matt and I are going to have to write an IQP report this term too.

As for the IQP, we've made some significant strides since my last post. Matt's done a lot of work restructuring the internals of our GUI, including solving the garbage collection problem. At the same time, he reworked some of the rougher areas of the GUI that I had ported from the old system. All this means that we are now finally where I wanted to be from the beginning: a few commonly used screens (like the Dashboard) are saved and re-used rather than being discarded and then re-created, but most screens just get tossed when we're done with them.

While Matt was doing that, I was cleaning up some smaller issues. For instance, that infernal logout button FINALLY got what was coming to it. I also did some restructuring so that screens that need to know which project to display data for "pull" the project from the ProjectData singleton rather than trying to "push" the project to the screen from whoever was calling it. The push method was creating a lot of headaches and making the code far more complex than it needed to be. I don't know if I ever mentioned this before, but MXML components are not allowed to have custom constructors, so instead of just passing the project into the screen's constructor, we were being forced to construct the screen and then make a separate call to set the project. This was creating a bunch of problems, not the least of which was that when we set the project, the loading box usually didn't exist yet, which was constantly triggering NPEs. Finally, I also implemented the ability to delete projects. Our good database design pays off here, because a single statement to delete the project cascades through the rest of the database and also removes all of the project responses.

The list of unimplemented items is beginning to narrow now. I still don't think it's reasonable to say we're at beta yet, because we still have plenty of buttons that don't do anything. As someone pointed out to me, however, it's okay if we don't make it past beta; half of Google's products are still in "beta" after *years* of development! More seriously, though, I'm still confident we'll have produced something worth using, not to mention something that teams after us will actually be able to extend. It's likely to be missing some of those "wishlist" features but all of the features that are implemented will work the way they are supposed to.

The next thing I'm going to work on are some of the account management features. Right now, once a user creates an account, they're stuck with everything they entered. At the very least, users need to be able to update their

Search

Blogroll
- WordPress.com
- WordPress.org

Archives
- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc
- Register
- Log in
- WordPress.org
- WordPress.com

e-mail address and change their password. Also, we don't have a system for handling forgotten passwords. I think the fairly standard system of "enter your e-mail address and we'll send you a link that lets you change your password" should do nicely, but this will require some additions to the database schema and implementing it in a secure manner could be tricky. I can't really say how this will play out since I've never needed to implement a forgotten password facility before. And before someone suggests it, I am *not* going down the "secret question" route. I hate those infernal things; most of the time, I have even less an idea of how I answered the question than I do of what the password might be!

Finally, for anyone wondering where the *Flex Sucks: Greatest Hits!* compilation is, I will be posting it at some point soon, possibly even later today.

Posted in IQP. Leave a Comment »

An update from the trenches

January 11, 2010 — Richard

Work on the IQP over break has been sporadic. Today, I finally finished updating all of our existing functionality to work with the new internal architecture. Along the way, I polished a few things and fixed a few bugs. We are now back to where we were, except with a more friendly and intuitive user interface.

That's the good news. Now for the bad news: we aren't really any further along than we were before. There are still a lot of unimplemented things. About the only thing that this refactoring did implement was the loading spinners. That's certainly a plus, but it's not enough for us to be able to apply a "beta" label to the project with a straight face. At the very least, all the items on the to-do list in the wiki that begin with the word "implement" need to be finished before we can seriously say we're in beta. Right now there are seven such items, which we clearly have no hope of finishing by Thursday, the start of the term. People are actually going to be using this thing, starting this term, so they're going to have to live with alpha builds for a while, I guess. The silver lining of that cloud is that what's implemented is stable (at least as far as I have tested it), so users shouldn't encounter too many problems.

Of course, the re-GUIing isn't quite finished yet. There are still some vestiges of the "old" system and the lack of an account management page means that, under the new architecture, once a user is an active member of at least one company, they cannot create or join any other companies. This is a rather large problem and addressing it is my next task. And after that… well, let's just say that the logout button will finally get what's coming for it.

I should probably take a moment to discuss why I've ignored the logout button for so long. I was thinking that, from a security perspective, it would be a good idea to make sure that we don't keep potentially sensitive data hanging around in memory once we're done with it. It seems straightforward enough: once you're done with data, simply null it out and get rid of it. Of course, things aren't actually that simple. The fact that memory is managed by garbage collection means that you can't really "delete" anything from memory. All you can do is remove all references to it and hope the GC runs at some point and collects it. No, there is *not* an explicit way to trigger the GC. There is a system call that runs the GC but it only works on debug versions of Flash Player. There is also a widely-circulated hack method of triggering the GC but there's no guarantee that that method will continue to work in the future. I've decided that, as security risks go, data hanging around in memory is not such a big problem. After all, if a user logs out and is done, they'll probably close the window or tab that the application is open in anyway, which will unload the Player and remove the application and all its data from memory.

On the other hand, having extra junk hanging around is a large problem from a standpoint of efficiency. I'm going to have to mess with this more to be sure, but, as mentioned in a previous post, it seems likely that there are some hard references to the objects we create that will prevent our screens from *ever* getting collected.

Speaking of trenches, over break I have put in a little more time on Modern

Warfare 2. I've now reached level 33, and I've started tinkering with the AUG HBAR and the Danger Close perk, both of which I've just recently unlocked. In terms of ranks, I'm now nearly halfway to the top (level 70). In terms of XP, however, I'm only about 16% of the way there. The only things that I haven't unlocked and am really looking forward to eventually having are the ACR, which is probably the best assault rifle in the game, and the Model 1887s, which still seem to have ridiculous power with Akimbo even with the latest patch, which nerfed that a bit. I'm not going to have either of those any time soon though: getting the ACR will require me to have about double the XP I have now, and the 1887s… well, those aren't unlocked until level sixty-something-or-other. 'Nuff said.

In the meantime, though, I've been having plenty of fun with the SCAR-H and the P90. My favorite weapon to play with right now is the P90 with the ACOG scope. I wasn't sure I was going to like the ACOG scope at first, but after one game with it I was in love. Not only does it provide significant magnification, making it much easier to get long-range shots, it also has cross-hairs that don't go away even if you get EMP'd. This is a bigger problem than it sounds; other sights like the Red Dot and the Holographic are actually worse than useless under EMP, because adding a sight removes the iron sights, so you have no indication of exactly where the bullets are going to go. Sure, intellectually you know that all you have to do is get the enemy in the center of the sight, but it's not easy to tell whether you've done that or not, especially for long-range shots.

I've also started tinkering with Claymores. They're wonderful if placed appropriately, but equipping claymores means you can't equip Frag or Semtex grenades. This is something of a problem for me because I'm used to having grenades, so I have to make a conscious effort to remember that I'm using claymores instead. Grenades have generally been more help to me than claymores because I have a tendency to run-and-gun.

Speaking of which, a Semtex grenade got me a sweet triple kill in one game. I was playing Domination on Karachi and I was over on the balcony near C when I noticed the enemy starting to capture it. I rushed over and, before I really noticed how many enemy players were standing there, lobbed a Semtex right onto the middle of the marker. **BOOM.** One Semtex took out three guys and prevented them from capping the flag. Then, with perfect timing, a fourth guy shows up to help his dead teammates. I filled him with lead from my P90.

Yes, the P90 is definitely my favorite gun so far: solid range, huge fire rate, massive magazine, fast reload, and the recoil isn't too bad. The key is to fire in short bursts so that the gun settles and the recoil doesn't cause all your bullets to miss. Right now my main goal is to keep playing with the P90 so I can finish unlocking all the attachments. I have most of them already but I need another 30 kills or so to unlock FMJ which I have to use to get the Extended Mags. I also recently used the grenade launcher (or noob tube if you prefer) on the SCAR to unlock the shotgun. I haven't tried that out yet; maybe I will the next time I play.

On a final note, I unlocked the Stinger missile a while back and I've been using that as my secondary weapon a lot. Shooting down air support is *loads* of fun. Especially when you shoot down some guy's chopper gunner and he ends up dying three kills short of a nuke. Listening to the lobby conversation after *that* game was entertaining, especially since, in the next game, I ended up on the same team as the guy whose nuke attempt I thwarted.

Posted in IQP, Modern Warfare 2. Leave a Comment »

The 10PM Rule

January 5, 2010 — Richard

For the more technically-versed readers of this blog (which must approach zero since the *total* readers of this blog approaches zero), you probably noticed that I made a really stupid statement in my last post. Obviously, the "destructor method" would never have worked anyway because the object wouldn't get garbage collected until there were no references to it. But since the event listener would be a reference, garbage collection would never happen and the listener would never get removed.

This is why I have the “10PM Rule”.

Said rule basically says that I am not to write any code or even *think* about writing code after 10PM because when I do, I usually do stupid things. The Rule has its origins in my freshman year. Back in those days, the campus intranet had a vibrant DC++ community. (It doesn’t anymore, because some students decided to go overboard, prompting NetOps to shut it down.) There was just one problem: despite the many terabytes of stuff available on the network, there were still some things that people wanted, but the network didn’t have. As an answer to this problem, someone created the RequestBot, which simply sat on the DC++ chat room and kept track of what things people wanted.

RequestBot was nice, but it had plenty of shortcomings. A second RequestBot, nicknamed RubyBot (since it was programmed in Ruby; I think the original was done in Python), added some more features and took over, but there were still plenty of gripes. So one of my roommates and I decided to write our own grand RequestBot. This thing was going to have everything under the sun. It would have remote administration features, it would keep track of the status of a request, it would keep track of the popularity of a request, it would keep track of who was working on it, it would keep track of where the files were located once the request was filled, it would keep track of statistics to rank the top contributors, it would even send you a PM once the request was filled if you were following it. We whipped up a features document, decided to code this puppy in Java, and created a project on SourceForge to store the code (and it’s still there).

We used up a lot of spare time working on this thing. By the time NetOps killed DC++, it was actually in decent shape, though nowhere near finished. It was at that point, during the network’s final hours, that we discovered our efforts were being duplicated by at least two other groups. D’oh!

But getting back to the 10PM Rule, one night I was hacking away at this and needed to communicate between two components. Initially, I was planning for the communication to be simple, so I just used a string. Then I realized it needed to be more complicated. By that point it was around 11PM, and because my brain was barely firing on *any* thrusters at that point, I wrote a bunch of code that translated various objects to strings and packed them into a HashMap to be tossed around the application. Then I went to sleep.

Then I got up the next morning and wondered what the hell I’d been thinking. I ended up spending as much time undoing my fatigue-induced WTF as I had spent creating it in the first place. Spare time isn’t something I had a lot of even then, so this basically meant a wasted day.

And that was when I invented the 10PM Rule. I do break it on some occasions; today, for instance, I didn’t even wake up until almost one in the afternoon, so I decided the restriction could be eased somewhat. Nevertheless, that post has reminded me why I established the rule in the first place.

Anyhow, since that post a few days ago, I’ve discovered that the problem is actually much larger than it appears. I’ve “fixed” it by making components remove the event listener as soon as they don’t need it, and re-adding the listener if they need it again. For components where that isn’t feasible, I’ve also changed the event listener code so that listeners are added using weak references that won’t stop the object from getting garbage collected. Not that that makes much of a difference. I finally learned how to use the Flash Profiler perspective, by which I of course mean that I mashed random buttons until it told me what I wanted to know. And what it told me was this: our screens are liable to *never* be garbage collected. There are simply too many references to them, mostly from components we don’t control. Flex apparently employs reference-counting GC most of the time, until memory usage passes some magical point and it decides to kick in the mark-and-sweep method instead. But there’s no way to force a mark-and-sweep, and thus no way to test that our screens will ever get garbage collected.

But moving on from that, there is some good news. I’ve started the re-GUIing that Matt and I had discussed and so far I’m pleased with the results. Of course, there’s still a long way to go, but the components that I’ve refactored

into the new architecture have gotten much simpler as a result. I'm hopeful that we'll have a beta build using the new architecture ready for the beginning of C-term, but obviously I can make no guarantees. I'm also re-examining the idea of using FlexLib; it looks like it may be worth it to take advantage of this, despite the 300KB it will add to the application. We already have our own version of a prompting text field in a few areas (like the login) and I'd like to use the window shade thingy on the Dashboard.

Finally, I've noticed that the people who end up at this blog thanks to Google are usually searching for "flex sucks". Expect a meta-post of sorts on the subject, linking to all my rants about Flex, in the next few days.

Posted in Epic fails, IQP. Leave a Comment »

Happy New Year!

January 1, 2010 — Richard

…of Flex Hell.

That would be the subtitle if it were possible to have one. Yes, it's only **one hour** into the year 2010 and I *already* have new reasons to hate Flex.

Allow me to explain.

Matt and I talked a few days ago and agreed on what generally needs to be done with the UI. We concluded that, since cross-company functionality really doesn't make any sense (and we don't have plans to implement any such functionality even if it did) it makes sense to have the company be an application-wide setting. So, post-login, there are three possible situations we might be in:

1. The user is an active member of exactly one company. This is our sunshine case, where there's nothing extra to do beyond simply shuffling the user off to our new, mostly unimplemented per-company Dashboard.
2. The user is an active member of more than one company. In this case, we have to prompt the user and ask which company they want to be working in.
3. The user is not an active member of any companies. In this case, we have to prompt the user and ask whether they want to create a company or join an existing one.

Simple enough, right?

Well, obviously not if I'm posting about it here. I managed to get path 3a (no companies, user wants to create one) working. There was just one catch: it worked *too* well. What I want to have happen is that, any time a user creates a new company, the application automatically makes that company the "current" company and then sends the user off to the Dashboard. And that was exactly what happened, which was fantastic… except that I hadn't written the code to accomplish that yet.

A few minutes of investigation revealed that what actually happened was that the user was getting punted back to the post-login decision screen, then because they were only part of one company, that screen automatically navigated to the Dashboard. The reason lies in the event-driven architecture that I used to create the CompanyData singleton. Concerned that something like this would happen, I updated the code so that event listeners would be added as weak references. I figured that the decision screen would get garbage collected and then that reference would be removed at some point by the EventDispatcher code.

No such luck; the screen must still be being referenced somewhere, I guess. Of course, I didn't really want to use weak references to begin with. I wanted to create a destructor that would remove the event listener when the screen got garbage collected. Since that apparently isn't happening, this method wouldn't have worked any better, but the reason that I didn't use the destructor method is because you can't. Why?

Because ActionScript, a supposedly object-oriented programming language, does not support destructors.

Yes, you read that right. Yet another feature that practically *every other OO language under the sun* has, but ActionScript lacks. I think I read somewhere that this is because the ECMAScript standard, which ActionScript is derived from, doesn't support destructors. This means one of two things: either the standard sucks, or Adobe made a shitty decision when adopting a standard.

Actually, on second thought, that's not an "or" situation.

So I'm not quite sure what to do about this now. I'm quitting for the night (or rather, morning) and I'll pick up with this tomorrow. Odds are we're going to have to make some changes to our whole "Screen" paradigm, although I've noticed some other potential shortcomings anyway. I'll talk them over with Matt the next time we're both online.

Oh, and for those wondering about all the stuff that I said I'd be posting about over break in my last post two weeks ago:

- *IQP*: check.
- *Dishwasher'd parts*: probably going to be too busy to examine them.
- *That GalCiv2 game*: turns out I can't remember enough details for a truly interesting write-up, so not happening.
- *What I got for Christmas*: nothing worth talking about.
- *Other stuff*: there is other stuff happening, so maybe.

Posted in Home, IQP, rants. Leave a Comment »

Windows 7 Redux

December 16, 2009 — Richard

Now that the term is *almost* over and I'm already finished with Networks (since the final was on Tuesday), I've been left with a little bit of spare time. I suppose I could have used it to work on the IQP, but I decided that upgrading to Windows 7 was a more important goal. I can work on the IQP anytime, anywhere, as long as I have the IQPDE set up. An OS upgrade is a considerably more involved process and has to be done during a lull, in case anything goes wrong and you need time to iron it out. I wasn't expecting any problems, since I had tested the software I use regularly to make sure there wouldn't be any problems. (I posted about this about two months ago.)

So how did it go? Well, given that I'm dedicating a post to it, you can probably figure out the answer. Before I begin, I want to say that, amazingly, *none* (or almost none, but it's hard to know for sure) of the problems I encountered were problems with Windows 7; they were all problems with various applications. Frankly, I'm digging Windows 7. I didn't actually do any measurements, but everything *feels* faster, especially boot.

The first problem I encountered was my uncooperative motherboard. For some reason, it just did *not* want to boot from that darn install disc. Mucking with some BIOS settings and issuing some verbal threats got me past that.

The second problem I encountered was The Weather Channel's Desktop Weather application. I've been using one weather app or another for years. I started out with WeatherBug but that was chock-full of spyware, so I dumped it for TWC, which at least only displays ads in its screen. Well, I went to install it and was promptly informed that 64-bit Windows is not supported. My bullshit-o-meter immediately shot into the red zone. Why the hell would a *weather* application, of all things, not work on a 64-bit OS? It's not like it's doing anything particularly specialized or complicated, or at least, it shouldn't be, so it should work fine as a 32-bit app. In searching for answers, I uncovered a bunch of weather gadgets, which I slapped onto my desktop on my secondary monitor and will probably keep there, because they're pretty convenient. But I still wanted that application, partially just to stick it to the Man and partially because it provides some information that no desktop gadget can give you, like forecasts by the hour. So I copied the working application from my laptop (which has Windows 7 in 32-bit trim installed) and ran it. Guess what? It works *perfectly*. What would we do without completely pointless, totally artificial installer checks?

The third problem I encountered was trying to get the IQPDE up and running. The first six or so steps went off without a hitch, then I hit a brick wall trying to

get Eclipse set up. I began by installing the 64-bit JDK, but of course, 32-bit Eclipse can't use it. So then I had to go chasing around half the Web to find out where the 64-bit Windows Eclipse installs are hidden (64-bit versions for other OSes, naturally, are right on the usual download page). Then after I downloaded that, I installed Flash Builder into it. This failed epically, because, once again, Adobe == shit and despite the fact that Flash *Player* comes in 64-bit trim, Flash *Builder* does not. So I had to repeat everything I'd just done, except using 32-bit versions of Eclipse and the JDK.

Yes, I know, 32-bit would have worked fine to begin with, but I've been trying to use as much 64-bit software as possible. Why? Because.

Anyway, I did all that, installed FB again, and this time it all went off without a hitch. I breathed a sigh of relief, figuring I was out of the woods. Ha! I proceeded to install Subclipse, and *all* the plugins promptly disappeared from Eclipse. After *much* trial and error, I eventually decided that UAC was probably the problem (because I put my Eclipse installation in "Program Files (x86)"), so I disabled it. After that, it was smooth sailing. So UAC is *still* shit, but its not like it would've ever stopped any malicious behavior anyway, since I'm not stupid enough to open e-mail attachments from people I don't know. Or, you know, do anything else stupid and risky like that.

So I'm mostly back up and running now. I'm not looking forward to repeating this on my computer at home. On the plus side, the two months between my Windows 7 testing and now has been put to good use by *some* developers. IZArc had an issue with glitches in the context menu, which have been fixed. Opera's configuration files have changed so that you don't have to go mucking around with paths any more. And the official build of Notepad2 now includes the registry-based solution for replacing the default Notepad.

The upgrade pains have been worth it, though. Windows 7 is everything that Vista should have been. I'm especially happy that real multi-monitor support now comes baked into the OS. On XP, I had to use nVidia's tool to create a hotkey for moving windows from one display to another. In Windows 7, the Win-Shift-Left/Right hotkey is there to do this for you right from the beginning. Also, you can (finally!) re-arrange the order of programs in the taskbar. Even my Internet connection seems faster. Not sure if it's just me or if they actually put some work into streamlining the network stack.

On the downside, the fact that icons in the system tray are hidden by default bugs me, but as far as I know, there's no way to configure this behavior, so I guess I'll just deal with it. And you can't shut off your computer by typing Win-Up-Enter-Enter any more; it's Win-Right-Enter now. I think I talked about both of those things two months ago, but hey, they're still bugging me.

So this is the last post of the term. (We've entered *two*-third time! …No, I won't make that terrible joke again. Probably.) Over the break, expect to see posts about the IQP, the still-unexamined dishwasher'd parts, that game of GalCiv2 I mentioned a while back, whatever I get for Christmas (if it's worth talking about), and any other assorted random things I feel like talking about.

Posted in Epic fails, Home, IQP. Leave a Comment »

The 100th commit party!

December 4, 2009 — Richard

Yep, you read that right! At 1:56PM EST today, a commit was made, marking the 100th revision in the code repository! It was nothing particularly interesting; in fact, the change was a single letter: a correction of a typo in the build script.

…okay, so there's really *nothing* particularly significant about that event, besides the fact that it's a big round number. Hey, I'll take my celebrations wherever I can find them.

In other IQP news, some solid plans have emerged from my dissertation on problems with our UI a couple days/posts ago. Matt is mostly done with our CMS, so he's going to be joining me on the application, probably beginning next week. We've decided how we're going to re-architect it, although our ideas were hampered somewhat by the fact that ActionScript has no concept of an abstract class… or an abstract *anything* for that matter. Basically, it really

is the Big Empty Space all over again. We're going to subclass Canvas and add some methods for our own purposes, then make all of our screens subclasses of that (BigEmptySpaceCanvas, anyone?). Those methods are going to include things like providing a VBox (or some other UIComponent) to be placed in the taskbox on the left side of the window. This will allow us to utilize more of the blank space in the taskbox and remove some controls from the screens themselves, making them less cramped. As previously mentioned, we will also be getting rid of the whole "state = screen" paradigm; instead, a few screens will be saved and most will be re-created each time the user needs one. This should resolve some bugs in a couple places.

In other *non*-IQP news, the second Webware project has been shelved by class vote. Instead we'll have two more fairly extensive homework assignments, which is fine with me. We got the team grades back and apparently they were all abysmal… except *my* "team", naturally, for which Pollice had exactly two comments:

- Missing JavaDoc in some of your Java files
- You've restored my faith in students

I think I can live with 78/80. Also, the Webware final will be on the last day of the term, Thursday the 17th, as decided by coin toss after half of the class wanted it on Tuesday (the 15th) and the other half wanted it on Thursday. I suggested splitting the difference and doing it Wednesday, but for some reason that suggestion didn't go over so well. Then I suggested we should play Planning Poker, which got a good chuckle.

As for Networks, the final will be on Tuesday since the class doesn't meet Thursday. The deadline for project 2 has been pushed back by two days to Thursday the 10th (which was weird, because I had thought that it was due that Friday to begin with!). The last homework will be due on Monday (the day before the final).

The end result of all this is that, as I predicted earlier this week, I probably won't spend much time working on the IQP next week, since I'll need to be working on Networks 2 and probably the Webware homework (Pollice has described it but hasn't posted it or given a due date yet). Which sucks, because we're *almost* to beta status. I hope to put up the first beta build at the end of the term, but we've already seen what becomes of the best laid plans around here.

Posted in IQP. Leave a Comment »

Bridging the gap

December 2, 2009 — Richard

I've been talking about bridging/meshing/whatever-the-hell-other-terms-I've-used the app and the site Matt's been developing for… um… a while. (Can you tell I can't remember how long?)

Well, today, I finally actually implemented it. It ended up only taking me about 90 minutes to figure out, since there wasn't really *that* much involved. I would have been able to finish quicker, if not for the hack I was forced to implement. My plan had been that inside the <body> there would be two s; one containing the site content and one containing the application. The containing the app would have height and width set to 100% but display set to none. When a user entered stuff in the login form, some JavaScript would call the app to validate it and return whether or not the attempt was successful. If yes, the JavaScript would set display to none on the content and display to block on the app .

There was just one problem: this doesn't work. At least in Opera, it doesn't. Apparently, the browser won't actually load the app or some such unless it's actually *visible* on the screen. So I hacked my way around this problem by setting the height and width of the app to be 1 pixel and then changing them to be 100% when login succeeds. I *did* try setting them to be 0 pixels initially, and this actually worked in Opera, but not in IE. Of course, the "1 pixel hack" means that there's a pixel on the screen, and there shouldn't be. This could wreak *all kinds* of havoc with the site layout. Fortunately, that's not my problem.

And that was all I got done on the project today. The Networks homework took me longer than I expected, mostly because I actually read some parts of the book. More disturbingly, I actually found some of the parts that I read… ***gulp*** *interesting*. Don't worry, I'm sure WPI must have a Geeks Anonymous group…

As for tomorrow, well, who knows. I'd like to put in some time sketching out the GUI refactoring and I need to upload a new test build. I do still have to do the Webware homework, though, and I should probably do some more work on the Networks project. Until tomorrow…

Posted in IQP. Leave a Comment »

Analysis: check.

December 1, 2009 — Richard

It's hard to believe I've been at this for over three months now. Today marks a major milestone: the analysis functionality that is the whole project's *raison d'etre* (translates to "reason for being" for those unfamiliar with the saying and with French) has finally been implemented. I don't think I'd say it's perfect, but I *will* say it's a damn sight better than the last version.

Yes, it's been a long slog getting here, and finally having reached this waypoint feels damn good. Of course, it *is* just a *waypoint* – we haven't remotely reached our destination yet. Still, it's made me stop to consider the question, *where do we go from here?*

I've decided that the first thing on the list is the much-delayed app-site meshing, which I plan to tinker with tomorrow. After that, the GUI needs some attention, both internally and externally. There are a couple WTFs with the current implementation. For instance, we use states to change screens. I did it this way because it was the way the last team did it and I was too busy trying to figure out all of ActionScript's idiosyncrasies at the beginning of the project to question it. As it turns out, like everything else the last team did, this is a really bad idea and it's lead to having a lot of hacks. It will also make it hard to make the UI more user friendly; in particular, it will make adding loading spinners extremely painful. On the "outside", my grand plan isn't working quite as well as I'd hoped. For most screens, it's okay, but there are a couple (like the response analysis screen) that are a bit cramped. Also, while the "task pane" is a very intuitive way to expose the application's functionality to the user, we disable it when we go into a specific screen and it becomes wasted space.

So how am I going to fix this? Well, I don't exactly know yet, and I certainly don't have any implementation details worked out yet, but a vague idea is forming in my head. Work on this issue will probably consume whatever time I spend working on the IQP next week. Speaking of next week, it's liable to be more like the last couple of weeks than this week in terms of IQP work. I'll be spending lots of time working on the Networks project and the specifications for the Webware project should be out soon, so I'll probably be putting time in on that as well.

To wrap up this post, I just want to remind everyone (i.e. the two or so people besides me who actually read this blog) that we have **not** reached beta yet. I *will* be posting a new test build this week, but it will still be labeled as alpha, since there's still a *lot* of important functionality that's unimplemented (like that insufferable logout button). I don't anticipate releasing another test build after that until the end of the term; that build will probably be the first beta version.

Posted in IQP. Leave a Comment »

Meat and potatoes

November 30, 2009 — Richard

The meat and potatoes of the application are implemented. Finally. Partially.

After spending the whole afternoon and evening working on it, we now have the BOBs. We even supply appropriate data tips and label the axes. I had originally planned to allow users to select which projects they wanted on the charts but decided that was going to be too time-consuming to implement to be worth it. For starters, you obviously couldn't allow projects from multiple

companies on the same chart. That just wouldn't make sense, since the companies would have different competencies. Even if two companies were in the same industry making the same products, that doesn't mean they have the same engineering skill or the same marketing channels, so even that would be an apples to oranges comparison. Allowing to filter which projects should appear from a single company may be useful, but the point of the application is that it's a *portfolio* and it doesn't make sense to just ignore parts of your portfolio. Any project that's not ready to be included in the analysis shouldn't have a master response yet, and so can't appear, and any project that's either been approved or rejected should be deleted from the system (although this isn't implemented yet).

The rank-ordered lists aren't implemented yet; they're next on the list. Also on the list is possibly looking at a different analysis selection mechanism. The current system is okay, but it takes up a fair chunk of valuable screen real estate. Consequently, on a widescreen monitor (possibly on 4:3 monitors as well), it makes the graphs *very* rectangular instead of square, which makes them harder to read.

In any case, tomorrow I have to do a Wireshark lab for Networks, but after that I'll probably spend the rest of the day on the IQP. Over Thanksgiving, I actually *did* start working on the second Networks project. So far, I've invested at least six or eight hours and all I've accomplished is that the application can parse its config files and the movie files and it can display a non-working interface. It's going to be a *long* slog before that one's done, but fortunately, it's not due until a week from Friday, so I've still got plenty of time to work on it. Of more immediate concern are the Networks and Webware homeworks that are due *this* Friday. I'm guessing they'll each take 2-3 hours, so my plan is to do one on Wednesday and one on Thursday, with the rest of my time on those days going to the IQP, the Networks project, and (if I get so frustrated that I have the urge to kill something) playing Modern Warfare 2.

Speaking of MW2, I played a few hours on Sunday and managed to climb about 5 ranks. Of course, the difference between level 6 and level 11 is squat in terms of XP. Having said that, I think I'm getting better, since I seem to have gone from being terrible at it to just mediocre. My yardstick for this is the fact that whichever team I was on in Domination (think King of the Hill, except with three hills and you don't have to stand on it after you've captured it from the enemy) repeatedly got its ass kicked, except for the last two games, in which the team I was on kicked ass.

Now one could argue that this is simply an artifact of having better teammates, and it's entirely possible that that contributed. However, in the last game, I got a 4 kill streak and called in a Care Package for the first time (it was actually the second time I got a Care Package, but the first time I got it literally seconds before the match ended and didn't have time to call it in). It may also have something to do with the fact that the first gun that seems to be any good is the the SCAR-H, which isn't unlocked until level 10 or thereabouts. It seems clear to me that the M4A1 and the other early weapons just suck at killing players because I saw a couple videos on YouTube of a guy who is a very good player and prestiged after reaching level 70. In the first few gameplays afterward, in which he didn't have any of the good guns since prestige sends you back to level 1, the enemy wiped the floor with him.

In any case, I had some interesting things happen during the two hours I spent playing. For one, I discovered that the host selection algorithm must be easily fooled, because I had to quit a match about five seconds after starting when I noticed enormous lag. Checking the player list, *everybody* except the host had a one bar connection. Great work, Infinity Ward.

Amore interesting thing that happened was my clutch response to a no-win situation. I was nearly out of ammo for my primary weapon, so I was looking around for dead bodies so I could swipe a new gun. I happened to be walking around with my secondary weapon, the AT-4 rocket launcher (which I'm going to ditch, since it's utterly useless against air support, which was the reason I was using it), when I came around a corner and face-to-face with a player on the other team. He started shooting me and I was taking damage fast, so I had to make a snap decision. He was close enough that there was zero chance I'd be able to switch back to my primary and kill him before he killed me. Unfortunately, he wasn't close enough for me to knife him and I'd almost

certainly be dead before I got close enough if I tried to rush him. So I decided to do the only thing I could: I aimed the rocket launcher at him and pulled the trigger. Mind you, this was on Rundown in a small alley between two buildings. At that range, I knew the rocket would kill me as well, but I figured if I was going to die either way, I might as well take him with me.

The last interesting thing happened during a match on Scrapyard. I'm not sure if it was a glitch or if I just happened to get *really* lucky. I was pursuing an enemy player who clearly wasn't aware I was there until the last second, but by that point it was too late: I knifed him before he could even get a shot off. My finger wasn't even on the trigger of my gun, but somehow that knife kill earned me a headshot. Huh? I mean, I guess it's not completely absurd, since you *could* knife somebody in the head, but I don't think I'd call that a head**shot**.

Anyhow, to wrap up this post, I've started a new category just for MW2. Possibly I'll rename it later and use it to talk about other games I play. I'm planning, perhaps over winter break, to make a series of posts about a fairly epic game of Galactic Civilzations II I played a couple of months ago. Unfortunately, I didn't really take a lot of notes or any screenshots, and the save file has been lost. Despite the fact that my recollection will be somewhat hazy, I do want to write about it, because some interesting things happened in that game. I'm sure this must have you on the edge of your seat, but you'll just have to wait… Muwahahaha!

Posted in IQP, Modern Warfare 2. Leave a Comment »

I'm not dead yet!

November 24, 2009 — Richard

Yes, things around here have been pretty slow, lately. This might lead one to conclude that for the past three weeks or so, I've basically been ignoring my IQP.

One would be essentially correct in thinking that.

Two weeks ago, I *tried* to work on it, but Adobe's bugs sent me on a wild goose chase all day Wednesday and ended up leaving me exactly where I started. Last week I did do *some* work on it, finishing the SOAP refactoring and killing artf5630, but end users wouldn't care about the former and the latter would have been relatively easy to work around. And this week, it's Thanksgiving. 'Nuff said.

It doesn't look like this situation is going to change any time soon (as in, the *next* three weeks) either. Just as I'm recovering from the first Networks and Webware projects, the second ones are right around the corner. In light of this, I've decided that I'm going to have to do some work on some of these projects over Thanksgiving (although it's entirely possible that I'm full of hot air here) and will need to put in some work on the IQP over winter break (which I'd been hoping to avoid).

I'm sure you're dying to know what the second projects entail.

Well, Networks 2 is a peer-to-peer movie streaming program. Not *movie* movies, mind you; there are companies that have spent millions trying and failing to accomplish that. No, we're streaming ASCII movies. This is looking like it'll be Networked Pong all over again. I promise not to forget to zero my buffers this time.

Webware 2 isn't well defined yet. Some kind of a WPI information portal thingy built using AJAX and web services. There's some good news here, in that I'll be working with Shikhar. He was one of the people who actually did stuff (and more importantly, actually knew what to do in the first place!) on the SoftEng project. This means I'm guaranteed to have at least one other competent programmer on the team (unlike with Webware 1). Sign-ups only just started, so it's too early to know whether we'll have a third or not.

In any case, for the IQP, the BOBs need to be done by the end of next week at the latest. On that Friday, I'd like to have a new test build, which will hopefully be the last alpha build. First I need to decide how to implement it though; I'm still toying with ideas for how the UI should work. Also, I still haven't done the

internal UI refactoring that I talked about weeks ago. Having thought about that a little more, there are pros and cons to doing it that way. I'm going to examine hybridizing the approach. Simply put, there are *some* screens you don't want to dispose of and regenerate, like the companies and projects tables. Generating them takes more time and throwing them out means, from the user's point of view, that they lose their state (e.g. a user sorts a column, then updates their response to a project, then goes back to the now unsorted table), which is not desirable. Other screens, you don't simply *want* to get rid of, it's practically *required* that you get rid of them. If I create a new project, or join a company, I don't want to come back to that screen to discover that all the stuff I entered before is still sitting there, I want a fresh screen. This can be accomplished under the current system, but it's a little kludgy.

Also, I've thought up a bunch of features that fall into the "nice to have" category. In particular, being able to easily export and/or print data and charts from the application would be very useful. It would also be good to have some auto-refresh functionality. Whether or not it's feasible to do these things is questionable, especially given the time constraints, but I'll add them to the to-do list. I'll be updating the list shortly, since I didn't update it after finishing the SOAP refactoring anyway.

Posted in IQP. Leave a Comment »

« Older posts

Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

YAAEF

November 18, 2009 — Richard

I'm officially introducing this terminology: YAEEF, an acronym for ***Y**et **A**nother **A**dobe **E**pic **F**ail*.

After mopping up the regressions introduced by the SOAP refactoring, I decided to spend some time chasing down artf5630, the bug where having a < or a & in a string (like a company name) broke the application. Preliminary testing revealed that the problem seemed to be on the Flex end of things, as usual. Trying things like creating a company with those characters worked fine if I did it using the NuSoap client (we have a file test.php for checking things like this). So I started looking at debug traces and whatnot and reached the (erroneous) conclusion that the *real* problem was that NuSoap wasn't properly deserializing the input it was getting, since it appeared that the NuSoap client didn't bother serializing the input in the first place.

After an hour or so of fruitless effort trying to figure out where in the NuSoap code one would fix this, I decided to hack around the problem before the input even hit the NuSoap server. Two str_replace calls later and… the NuSoap client blew up in my face. A little more debugging and tracing turned up the fact that the NuSoap client actually *was* serializing the input, and the NuSoap server was deserializing it appropriately. So finally, I ran a debug trace and discovered that if I input, say **<123&456>**, Flex was serializing this to **<123&456>**.

I'm sure you can see the problem there: for some reason, the input was getting serialized *twice*, so it was impossible for NuSoap to deserialize it properly. Curiously, this occurred for < and &, but not >. Some searching on Google turned up this (if you don't have an account, use BugMeNot to get a login). Notice: the bug is "closed" with resolution "fixed", but according to the comments, the problem was discovered in Flex SDK 3.2, persisted in SDK 3.3 (despite the fact that it was supposedly fixed at that point) and apparently SDK 3.4 as well, since it continues to be a problem in SDK 4.0.

Fortunately, the tracker page contained a workaround that I was able to add to the Npdservice constructor that hackfixes the problem. Just another in a LONG line of hackfixes for Adobe's bugs. The amount of bad code offsets Adobe would need to buy in order to "undo" all of this crap would probably bankrupt the company. It's truly sad that this amount of epic failure exists in a professional product, especially one that you get charged an arm and a leg to use. I am seriously hoping to *never* see Flex again after this project is over, and wherever I end up in the future, if anyone is considering using it, I *will* argue against it until I'm blue in the face.

Posted in Epic fails, IQP. Leave a Comment »

Projects² (and MW2 reviewed)

November 18, 2009 — Richard

Like last week, I haven't updated in a while. And like last week, it's because I've been busy. I spent all day Monday and most of Tuesday laying the foundation for the Webware project. Tuesday night was spent doing the fourth Webware homework. Not counting today, we have three more days to finish the Webware project (it's due at 11:59:59 PM on Saturday). It will probably get done, but it'll probably be fairly unpolished. In a twist of irony, logout is implemented for the project I've spent two days on, but not the project I've spent two terms on.

Which brings me to what's going on with the IQP. I'm determined to get the SOAP refactoring done, maybe along with some other miscellaneous stuff since I probably don't have time to make any significant headway on project analysis

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

this week. I've spent the last two or three hours working on the refactoring. Much to my amazement, my first attempt to compile the Flex app after all those massive changes actually worked!

Of course, the app wasn't fully operational; I should have the last problem fixed shortly. My solution is a hack, but as far as I can tell, there are no non-hack solutions for the whole "data grid can't see nested data fields" problem. I've seen it done several ways: subclassing the data grid column class so it can see them (but this breaks sorting and some other stuff), using a proxy object to wrap the objects in the grid (similar to what I'm going to do, but would require more work), and overriding the label function (which also breaks sorting and requires overriding that function as well). My plan is much simpler: the objects in the datagrid are CompanyToCompanyPermissionsMapping objects, so I'll just add getters to that class that get the data out of the nested objects. Like I said, it's a hack and it should not be used *anywhere* in the code we write, just for the benefit of the datagrid.

During the refactoring, I noticed a bunch of other things that should probably be refactored as well. If time permits, I'll do them this week, otherwise they'll get added to the to-do list like everything else. Next week there should be no other projects to worry about, so I should be able to get project analysis implemented (or at least started).

On a related note, I've been trying to stay on top of so many things that I haven't had an opportunity to play Modern Warfare 2 since the weekend. I got home on Friday around 7PM and parked myself in the media room (which contains a big honking plasma screen and surround sound with a subwoofer that shakes the whole house if someone on screen so much as coughs) until midnight. Not because I got tired at midnight, but because I finished the campaign at midnight. Yep, that's right: brand new game, no idea how to navigate the levels, but I still finished the entire campaign in just under five hours. The campaign *was* exciting, except for the ending, which was stupid.

SPOILER ALERT!

The game really does a good job of engaging you emotionally. Fighting your way through the streets of Virginia and Washington, D.C., you really get that sense of desperation and hopelessness. It was especially noticeable when you're in DC and your chopper gets shot down. You're trapped in the wreckage and you lost your weapons. Your CO tosses you an M4A1; a bullet plows through his skull and kills him as he does so. You shoot at the seemingly endless Russian forces, but your gun clicks empty. Another soldier pulls out a fresh clip and tosses it to you, saying "that's the very last one!" You go back to shooting at the enemy but 30 rounds isn't much against dozens of enemy troops... and your gun clicks empty again. The other friendlies who survived the crash are equally screwed; one says "I'm out!" and another says "three bullets left!"

Fade to white. Level over.

Yes, the campaign really gets to you emotionally, so it's too bad the "surprise" ending is lame and utterly predictable. I knew what was going to happen from the moment the Secretary of Defense said to General Shepherd, in one of the inter-level cut-scenes, "You warned us. We should have listened." Surprise, surprise, he betrays and kills you (as Gary "Roach" Sanderson). Then, as Soap MacTavish (whom you spent most of CoD4 playing as), you team up with Captain Price and Nikolai to kill Shepherd and show the world the truth. You succeed in the former, but whether you succeed in the latter is up to your imagination, I guess, since there's no newscaster as there was at the end of CoD4.

The plot in general was not as good as in CoD4, in my opinion. In CoD4, everything was very sequential and logical in occurrence. In MW2, there are a LOT of non sequiturs. You're infiltrating a Russian base to retrieve an "ACS module". Okay, why? What does this thing do? That doesn't get explained until the cut-scene *after* the mission, when you find out that apparently the whole US military relies on it. This is how the Russians manage to invade the US, for those who were scratching their heads trying to figure out how *that* happened.

There are a lot of questions left unanswered altogether too. It turns out that the prisoner that Makarov hates so much but for some reason can't kill is Price. How

did he end up in a Russian gulag? Why couldn't Makarov kill him? The Russians invaded the US using the magical ACS module to keep us from seeing them, apparently. Still, how is it that absolutely nobody noticed this massive flying invasion force? Are there no surveillance satellites to see them? No civilian aircraft to ask air traffic control, "what the hell is that flying armada with Russian markings doing here?" And when a bunch of stations DO report picking up bogeys, nobody gets suspicious and scrambles a few fighters to investigate? Or are we supposed to believe that Shepherd somehow managed to block all of that?

What bothers me the most is Shepherd's reason for his betrayal. "Five years ago, I lost 30,000 men in the blink of an eye, and the whole world just fucking sat there and watched. But tomorrow… tomorrow there will be no shortage of volunteers, no shortage of patriots." So you thought our military was weak and decided to fix that by *orchestrating a devastating attack on* ***your own country?!?*** WTF?!? What about the civilian casualties this caused? Are they just acceptable collateral damage? What about the impact of the governmental disarray that results or the damage inflicted on the US economy? It's unfathomable to think that someone ascended to the highest ranks of the military, having sworn to protect our nation from all enemies, foreign and domestic, and then decides to do our enemies' job for them.

Maybe some of this will be clearer when I replay through the campaign a second time. Or maybe Infinity Ward needs to come up with something more plausible next time.

Posted in IQP, rants. Leave a Comment »

When life gives you lemons…

November 11, 2009 — Richard

~~Make lemonade.~~ Don't bother because those weren't lemons you got.

My day started off pretty well. I slept in until 11AM. I spent an hour on YouTube checking out what multiplayer looks like in Modern Warfare 2, which looks like an amazingly awesome game. I've been waiting for it since I first played CoD4 two years ago. Now that it's finally out, naturally, I am forced to wait. I ordered the game on Amazon and debated having it delivered to my apartment, but decided it represented too much of a temptation to not do work and had it shipped home instead. Since I obviously am not able to be at home on a Tuesday, I decided not to waste money on release-day delivery. I expected, due to the game's popularity, that it probably wouldn't get delivered until some time next week and I wouldn't be able to play it for another two weekends. Then Amazon got my hopes up by sending me an email saying they had shipped it. Of course, what that really means is that they paid for the postage and assigned me a tracking number, not that they've actually put it in a box and kicked it out the door. The USPS says they still don't have the package yet, although they've been told to expect it. This, of course, is absolutely meaningless because the USPS tracking system sucks. I've actually been holding packages in my hand before the USPS tracking system even acknowledged the existence of said package.

Moving on from that aside/rant, I then got an email from Professor Claypool saying the second Networks homework is delayed until Monday. This made me especially happy because I finished project 1 and lab 2 yesterday, and the Webware homework due Friday doesn't look particularly complicated. I figured I'd be able to spend all day on the IQP and make some serious headway.

Yesterday: *Basically, we're declining the help because it doesn't look like we need it. Although with my luck, something disastrous will happen and I'll be eating those words…*

Today: I hate it when I'm right.

No, I'm not considering reversing that decision, but something disastrous (kinda) did happen today. The top priority currently is a significant refactoring of the SOAP service, specifically, the data types it uses, in order to eliminate some confusion that had developed and to streamline things better (I talked about this in a post last week). Well it's usually not a good idea to make lots of massive changes at once. It's easier to do things piecemeal, changing and then testing a

few things at a time in isolation. In this case, it turns out we have no choice but to do it wholesale.

See, my plan was to tackle one SOAP operation at a time, which would mean that the base data types (things like Project and CompanyPermissions) would remain unchanged until the refactoring was mostly complete, at which point the data overlap would be eliminated. Well, the first operation I decided to tackle was getting the per-company permissions for the logged in user. I fixed everything up, compiled the app, ran it… and watched it blow up in my face.

It took me about half an hour to ascertain the first part of why: the object I was pulling out of the ArrayCollection was somehow null. It took another half hour to figure out that the reason it was null was because the type cast was failing. Remember: no generics in ActionScript, so when you pull an object out of an ArrayCollection, you have to cast it to the appropriate type. It took about five minutes on Google to figure out how to get the class of the object, which turned out to be ObjectProxy.

shakes fist ***OBJECTPROXY!!!!***

Yes, for some reason, instead of giving me an object of the appropriate class, Flex decided to give me an ObjectProxy that was sort of like it (but not really) instead. I went around in circles for about *four hours* trying to figure out why, since this behavior doesn't appear to be documented **anywhere**. It just wasn't logical: the SOAP service returned an array of CompanyToCompanyPermissionsMapping objects. Said objects contained a string and a CompanyPermissions object. Whoop-de-friggin-do! The getAllResponsesForProject operation creates a map in the *exact same way* and **it** doesn't have this problem! WTF?!?

I *knew* there had to be **some** difference between the two that caused them to be treated differently. Eventually I got so frustrated (and hungry, because it was approaching dinner time) that I decided to go get dinner and come back to this after. This method actually works remarkably well for me: I get frustrated over a problem, say "fuck it", go do something else, and then have a brainstorm as to what the problem is and how to fix it. Today was no exception. As cliche as this is, I was standing in the shower when the light bulb finally appeared over my head (and I swear, it was audible!).

Yes, it turns out that Flex decides to be a jerk and give you fake objects **only if the object contains another object nested at least *two levels* inside it**! The operation that worked didn't have this: it's a map from a User to a ProjectResponse, both of which are *contained in* some other objects but neither of which *contains* any other objects. The operation I was trying to refactor was returning a map from a string (a company name) to a CompanyPermissions object. Since I was trying to do the refactoring piecemeal, the CompanyPermissions object still contained a User object. By the end of the refactoring, this would be eliminated, but I needed to leave it in temporarily so that other parts of the application using the class for other as yet unrefactored operations wouldn't become uncompilable.

So, to sum it up:

Fine: UserToProjectResponseMapping, which contains a User and a ProjectResponse.

Not fine: CompanyToCompanyPermissionsMapping, which contains a string and a CompanyPermissions, *which contains a User.*

Hence, the only way to accomplish the refactoring is to do it wholesale and change everything at once, so we don't have any class containing two levels of nesting at any point. I've tested and confirmed that this is the cause. It would be really, REALLY nice if you had actually, ya know, **documented** this little "feature" of yours*, Adobe!* Or better yet*, never have implemented it at all!* Honestly, whose *brilliant* idea was this?!?

Anyhow, I can't pick up where I left off, having discovered this, because the process of making the discovery has left my working copy of the project so mucked up that I can't remember what I've messed with or where. So I'm going

to revert to the base revision copy and begin anew. I don't know if I'll manage to finish this refactoring by the end of the week. If I do, it'll likely be the *only* thing I accomplish this week for the project. I've been trying so very hard not to have any more "infrastructure" weeks and I especially wanted to start working on the analysis functionality. Clearly, *that* isn't going to happen until next week at the earliest. Just as I was a little swamped with Networks this week, the first Webware project is due at the end of next week. It's being done in small teams, and I don't even have a full team yet, never mind actually writing any code. *Sigh* Anyhow, we'll see how this pans out.

I'm contemplating creating a page on Wikipedia discussing the causes, symptoms, and (lack of) cures for actionscriptophobia, the completely justified fear of anything involving ActionScript, which I believe this latest incident has caused me to develop. Good idea? Let me know in the comments.

Posted in Epic fails, IQP, rants. Leave a Comment »

Quiet on the IQP front

November 10, 2009 — Richard

As the title implies, I've done nothing yet on the IQP. Mostly because of the Networks project. Said project has reminded me that:

1. Socket programming is a royal pain in the ass.
2. C and C++ need to be taken out back and shot. Repeatedly. Then torched, and preferably nuked, until not a scrap of code is left.

The latter point is born from my frustration trying to figure out why the hell I was getting all kinds of random junk mixed in with the messages I was sending back and forth. I spent hours trying to figure out why last night, and another hour or so today before I finally saw it. Here's the problem code:

```
// add to the buffer
char buf[256];
int bRead = recv(sock, buf, 255, 0);
if ((bRead == EAGAIN) || (bRead == EWOULDBLOCK)) {
// no messages available
return out;
} else if (bRead == 0) {
// socket closed by the other player
return "quit";
}
recvBuf.append(buf);
```

See the problem? Well, if you code C/C++ on a regular basis, you probably do. I don't, but I finally realized I was missing a line.

```
// add to the buffer
char buf[256];
memset(buf, 0, 256);
int bRead = recv(sock, buf, 255, 0);
if ((bRead == EAGAIN) || (bRead == EWOULDBLOCK)) {
// no messages available
return out;
} else if (bRead == 0) {
// socket closed by the other player
return "quit";
}
recvBuf.append(buf);
```

Note to self: when writing C/C++, always, *always*, **always** zero your arrays! This reminded me just how much I despise C/C++, because this would *never* happen in Java. Hopefully project 2 will be something that lends itself to a Java approach. Although, even with all the difficulties involved with doing project 1 in Java, it probably *still* would have been quicker than doing it in C++ given all the time I've wasted hunting down this godforsaken problem!

Anyway, I'm hoping to finish up Networked Pong by the end of today. Depending

on when I finish (and how frustrated I am by that point), I may do some work on the IQP later. Tomorrow (and Thursday) will probably be a mix of IQP, Networks, and Webware, since I have two other assignments due for Networks on Friday and will have one for Webware probably due Friday as well (still waiting for Pollice to post it).

On a related subject, Professor Danneels has offered to hire us some help for the IQP. Matt and I talked it over. A third person would have been great at the start of the project, but given where we're currently at, we've decided to decline the help. Yes, I know, it seems very hypocritical: back in A-term I complain that we need more help, then I post the above paragraph talking about how I don't have a lot of time to work on the IQP this week, and then I conclude by saying we don't want the help. There are actually two good reasons for declining the help:

1. There's actually not all that much more work to be done on the application. I hesitate to assign percentages (a lesson from SoftEng: don't fall into the "we're 80% done!" trap), but I'd definitely say it's over half done at this point.
2. Matt is almost finished with the website and our pseudo-CMS (that isn't so pseudo-y anymore), so he should be able to help me out with the application starting in the next week or two. In a twist of irony, I responded to a question Professor Danneels asked about skinning the app by punting the issue to Matt, who later told me it was his intention to skin it all along anyway.

Basically, we're declining the help because it doesn't look like we need it. Although with my luck, something disastrous will happen and I'll be eating those words...

Posted in IQP, rants. Leave a Comment »

Wrapping up the week

November 5, 2009 — Richard

After several hours of work today, I've resolved both issues I mentioned yesterday. I'm not completely happy with the solution, but it looks like that's as close as we can get to what I really want without making our own special subclass of AdvancedDataGrid, which I'm not particularly keen on doing. It's now closing in on 5PM, I need to finish the Networks homework (and ideally do the second Webware homework, since it's not too tough), and I have shows on later to watch (baseball is **FINALLY** over! *Fringe* fans rejoice, for there *will* be a new episode tonight!). Therefore, I've decided that this will be the last major work I do this week. I do need to upload a new test build though (which may involve debugging the test build script, since I haven't tried it since the Flex 4 migration).

So what's next week looking like? Well, it's tough to say. I definitely want to re-factor the SOAP service before implementing anything else; this is likely to take a solid day's work. After that, we (finally!) get to implementing the meat and potatoes of the application: the BOBs (Big Orange Bubbles – also known as bubblecharts, but that's not nearly as cool a name) and the rank-ordered lists. After that's done, there are a lot of smaller messes to clean up before I can honestly claim we've reached beta status, but if all goes well, I expect we'll get there in about two more weeks.

There is a catch here, though: the first Networks project is due next Friday, and it's fairly extensive. Networked Pong sounds simple, and it would be fairly simple if we were being given Pong code to extend. Instead, we get nothing and have to write the whole thing from the ground up. This project will consume a lot of my time next week, hence the hedging. Although we have the option to do the project in Java, it looks like that would be more of a hassle (amazingly) than suffering through doing it with C/C++, especially because we have to make sure the thing works on the CCC servers, some of which randomly don't have the JVM installed (but somehow have javac! WTF?!?). So I figure I'll just code the damn thing on the CCC servers using PuTTY and Xming and then I won't have any problems. This means that I don't need a development environment configured in any particular way (unlike with the IQP), so I can work on it from home, which

means I can work on it over the weekend, if I feel like it. We'll see what happens.

Posted in IQP. Leave a Comment »

Another one bites the dust

November 4, 2009 — Richard

Another item on the to-do list, that is. Sort of. Project response analysis is now implemented to the point of at least being usable. It needs more work:

- The survey questions averages/standard deviations are presented in numerical format like the "numbers" are. This is *usable* but not very *user-friendly*. The inputs are graphical in nature (a radio button group), so it would be nice to have some graphical way of displaying the calculations. I have an idea about this but I'm not sure how or if it'll pan out.
- All the calculations are reported to a gazillion decimal places. This is because the data itself is returned to the grid as a Number, which then converts it to a String. There are several possible ways to fix, but I haven't decided which is the cleanest yet.

I'll investigate these problems and hopefully fix them over the next day. Depending on how this goes, I may or may not get to the SOAP refactoring I want to do. If not, this will take priority next week. Right now, I need to turn my attention to the Networks homework.

Posted in IQP. Leave a Comment »

Wish you were here!

November 3, 2009 — Richard

...instead of me.

I know, I haven't updated in a few days. I've been busily trying to implement project response analysis. It's *looking* like it'll get done this week, but I make no guarantees. Currently, I have the UI all set up and the SOAP operations implemented (although not fully debugged yet). I've plugged in getting the "master" response (which will be used to make all the BOBs) but this is rather useless at the moment because updating the master response doesn't work.

Why doesn't it work? My plan was to store master responses in the ProjectResponses table along with all the user-supplied responses. The method for accomplishing this was simply to set the *user* field to null for master responses, implying (correctly) that the response is associated with the project, but not with a specific user. This does not work because we're using InnoDB as our storage engine. InnoDB, unlike MyISAM, actually checks foreign key constraints. The *user* field is a foreign key to the Users table, where it references the *username* field, which is a key for the Users table. A **non-null** key. Therefore, the foreign key in the ProjectResponses table can't be null.

How to solve? There are several options, creating problems ranging from "creates duplication" to "creates hacks". We could create a "masterResponse" user and set the user field to that for all the master responses; this is a massive hack. We could make the username field nullable, but this is also something of a hack and requires us to make an extra check in createUser. The final option I can think of, which is what I'm leaning towards, is to create a separate MasterProjectResponses table that is exactly the same as the ProjectResponses table except that it doesn't contain a user field. This creates a huge amount of duplication, but it's cleaner than the other options, since there are no hacks involved.

Once I'm finished sorting out the project response analysis functionality, I'm going to perform some refactoring on the SOAP service. Right now, we've got a bit of a mess where relationships between data types are concerned. For instance, CompanyPermissions contains a User element. Why? Because sometimes we need to explicitly specify which user is connected with a set of permissions; for instance, on the company user management screen, we get a

list of permissions and we need to know which user each one is attached to. But sometimes the user field is irrelevant (for instance, when we create a company, we send a CompanyPermissions object to define the default permissions for new users) or we already know who the user is (for instance, when we get the permissions for all companies the logged in user has a relationship with). This means that sometimes some fields are getting filled in and sometimes they aren't, depending on the context of the operation being performed. This is bad for two reasons:

1. It adds overhead, both in processing (where we have to process empty elements even if we know they're empty) and in transmission (where we need extra bandwidth and transmission time to accommodate empty elements).
2. It can create confusion on the Flex end. Since every CompanyPermissions object looks the same, how are we to know whether any given object we might be handed actually contains valid data in a field or not?

To resolve this, I'm planning refactor the SOAP service so that we end up with a poor man's implementation of a map. It won't be a "real" map, not on the Flex side, anyway. Having a pseudo-map should be good enough, though, and we can always write the necessary code to make it a true map later on, if need be. I've constructed a spreadsheet listing the operations that will need to be adjusted, so this shouldn't be too painful. (Famous last words, I know!)

On the "other classes" front, I've already finished the Webware homework due Thursday, but I haven't yet started the much longer Networks homework due Friday. I also haven't started the first Networks project, which looks like it's going to be painful, but fortunately, that's not due until next Friday. Still, it looks like my nemesis, the logout button, will live to do nothing another ~~day~~ week. *Shakes fist* *Some day, logout! SOME DAY!*

Posted in IQP. Leave a Comment »

Migrating to Flex 4, the finale!

October 30, 2009 — Richard

As the final (kinda) step in our migration to Flex 4, I have updated the wiki on SourceForge with all the things that changed. Of course, the migration isn't really finished because we're still using the MXML 2006 namespace, but as I said in a previous post, I'm not touching that until the final release of FB4. It's been added to the technical to-do list on the wiki.

deep breath ***IT'S OVER!!!!!***

ahem I've also talked to Matt, and he didn't accomplish anything significant this week either, which makes our weekly report pretty darn short. It also means that our release schedule is slipping, but I'm still confident we'll finish this on time. Perhaps not as polished as I would like. Only time will tell.

Posted in IQP. Leave a Comment »

Migrating to Flex 4, Part III

October 29, 2009 — Richard

I spent several hours today porting our code to the new web services system. All that nice stuff I said about in my last post? I take it all back. While it's good that the process for making a service call is now more intuitive, it came at a heavy price: almost zero type-safety. Since Flex doesn't have a concept of generics (except for Vector, which I'll get to in a moment), the old auto-generating method created oodles of subclasses for things like ResultEvents and collections of custom data types. The new system does none of this. I talked about why generic ResultEvents are a problem before, but I hadn't noticed the lack of custom collections. This turned out to be a huge problem.

Our old code for interfacing with web services took advantage of the fact that we received custom collections in our custom ResultEvents to hang on to a little type safety. These custom collections disappeared, leaving generic ArrayCollections in

their place. So I had a choice: roll my own custom ArrayCollections, or use Vectors. I opted for the latter. Vectors are currently the only thing in Flex that has a concept of generics. I think I talked about this before, but although I don't know how Adobe is implementing Vector, I suspect it's a massive hack because you can't create your own generic classes. Moving on from that aside, I rewrote a ton of code to use vectors instead of the custom collections. It turned out that this solution merely created *another* problem. Datagrids expect to be given something that implements ICollectionView, apparently. Vectors don't do this. So while the application compiled, all the tables were completely broken. I fixed this by introducing a VectorToArray utility function I found thanks to Google.

So basically, in our attempts to maintain as much type safety as possible, we now have a process that goes like this: the auto-generated code gives us an ArrayCollection, which we turn into a Vector and pass around our application, then turn *back* into an ArrayCollection for the Datagrids. As if that weren't bad enough, Adobe *still* hasn't fixed the bug that makes their auto-generator screw up when faced with a SOAP operation called "XResponse". So we're still stuck with "GetProjectAnswer" instead.

Even getting a key to extend the beta beyond 60 days was a hassle. I used the form on Adobe's site, which promptly e-mailed me a license key. The one problem: the license key didn't work. I e-mailed them last night and got a response this morning. Apparently, somebody over there was asleep at the wheel and forgot to update the license key form for beta 2, which won't accept keys from beta 1. Oi.

Anyway, we're now back to where we were when I started this little adventure. I've updated the test build script but I haven't actually tried it, mostly because there's obviously not going to be a test build this week since there's nothing new to show. Over the course of tonight and tomorrow, I'm going to merge the Flex4Migration branch into trunk and update all of the documentation on the wiki accordingly. I also need to start drafting our weekly report. That ought to be interesting. Maybe Matt did something this week that was actually *useful*.

Posted in IQP, rants. Leave a Comment »

Migrating to Flex 4, Part II

October 28, 2009 — Richard

The Flex4Migration branch is almost ready to replace trunk. After a few more hours of work today, I was able to resurrect the Ant-only build process. This required some significant changes to the build script and also an addition to build.properties. A few things I've discovered along the way:

- For some reason, you can't be looking at build.xml when you try to build and run the project. If you are, you get that "out of heap space" error I was yelling about yesterday. I guess this means I owe Adobe an apology, since their compiler apparently isn't *that* bloated.
- Adobe added some syntax highlighting niceties to the code editor in FB4. Sadly, they still fall woefully short of where they should be. When I edit a Java file in Eclipse, I get warnings and errors displayed as I type. In FB4, you don't get anything until you re-compile, and you have to use the Flex builder; the Ant builder won't do it, despite the fact that you're using the mxmlc task. This means we have to continue using the "look at ant.out to find out why it failed" method for fixing compile problems.
- Flex Profiler is no longer a hog that takes over the console. Instead, FB4 just empties the console window altogether and says "no consoles to display at this time" after the Ant build finishes which is just as (un)helpful.
- Content Assist is much improved. Like the syntax highlighting though, this still falls short. If you mouse over a field or method, you'll get a box (as in Java) containing the appropriate ASDoc, but **only** if the field/method/etc is part of the class of the object. If it's inherited from an ancestor class, you get nothing.
- The hanging bug reappeared when I returned to a full Ant build. When I began tracking it down, I discovered that setting the compiler for compatibility with version 3 fixed the problem. Curious, I checked Adobe's documentation to find out precisely what effect this setting has. The only thing on the list that

seemed to apply to us is that the default theme in Flex 4 is the "Spark" theme, as opposed to the "Halo" theme in Flex 3. So I changed the build file to eliminate the compatibility setting and tell the compiler to use the Halo theme. Voila! Problem solved. A new *theme* was what utterly broke the app in a real browser (though it continued to work in the standalone debug player, curiously). Great job, Adobe.

- I neglected to mention the "conditional compilation is broken" issue in my list yesterday. This was a problem with the error highlighting, but since that doesn't work anymore now that we're back to a pure Ant build, it's no longer an issue. Adobe's documentation for Flex 4 indicates that the system for conditional compilation hasn't changed, so apparently their syntax checker is just broken or something.
- The new web services system has a significant drawback in comparison to the old system. Since CallResponders are generic and no specific subclasses are generated, all you get when the call returns is a generic ResultEvent with a reference to a generic result object. Bye-bye Type-safetyville, hello Casting City.

What all this boils down to is that the list of remaining issues is now much shorter. All that's left are some more minor refactorings to organize things better, killing off the old web services code, and updating the test build process. On the administrative end, I need to get a license key for FB4 from Adobe because otherwise the trial expires in 60 days. Adobe says they'll give anyone who has an FB3 license a key for the FB4 beta, and I've found the form on their website, so this shouldn't be too much of a problem. Once all this is done, I'll replace trunk with the migrated branch and update the wiki accordingly. I've decided that namespace migration will wait until the final version of FB4 (and the Flex 4 SDK) is released; this will be added to the technical to-do list on the wiki when I replace trunk.

The bad news here is that I'll have spent most of my time this week doing the migration instead of adding new functionality. There's also going to be downtime at some point, probably next week, so I can upgrade my computer to Windows 7. This means that beta is probably going to get pushed back towards the end of B-term. Also, in case anyone was worried about it, the migration to Flex 4 will bump the minimum required version of Flash that end users will need to Flash 10. I was a little concerned about this, but according to some surveys (admittedly commissioned by Adobe, you can debate whether this biases them if you like), Flash 10 is now present on 90-95% of all computers worldwide, so it shouldn't be a problem.

Posted in IQP. Leave a Comment »

« Older posts Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

Migrating to Flex 4

October 27, 2009 — Richard

I've made enough progress with the migration from Flex 3 to Flex 4 to commit it to the repository. It's not ready for prime-time yet, so it's in a branch in case something bad happens and we're forced to stick with Flex 3. Somehow I resolved the one-click build being broken. I'm not exactly sure how; basically instead of trying to migrate the existing project, I created a whole new project and copied over the source files and set everything up again. The build process is not as good as it was before now, in my opinion, because now there's a kludge of Flex build and Ant build, where the Flex build handles compiling the app and copying the HTML template, but the Ant build retains its job of copying the PHP scripts. It may be possible to return to a pure Ant build, which I'd prefer, but right now, it works, so I've moved on to investigating other issues.

I've managed to figure out the new system for importing web services. It's radically different from the old way; all of the existing code for interacting with the web service will have to be updated to work with the new auto-generated code. The good news is that this is a fairly simple process, and the new system is more intuitive than the old one. In FB3, you add event listeners to the *service* rather than the request call, which seems a little awkward to me. In FB4, conversely, you create a CallResponder, add event listeners to that, then attach the AsyncToken the web service call returns to it. The best news of all is that the existing web service code can co-exist alongside the new code: the only thing I ported was createCompany, which still works just fine despite the fact that the rest of the app, including the login, is still on the old system. You know you've done a solid job of engineering your software when things like that happen.

Also the weird hanging bug vanished. I think it occurred because the core Flex libraries were being linked to (as Run-time Shared Libraries) instead of being compiled into the SWF as they are under the Flex 3 SDK.

Major outstanding issues, which will hopefully receive some attention tomorrow:

- I tried to do the namespace conversion but it looks like this is going to be a bit more involved than a simple copy-paste. In fact, I think I'm inclined to stick with the Flex 3 namespace until the final version of Flex 4 is released because the namespace/Spark/Halo issue has caused the Flex 4 SDK to radically change at least twice. Right now it looks like you need a mix across *three* namespaces to make things work: the MXML 2009 namespace only contains language-related stuff (as opposed to MXML 2006 which has everything and the kitchen sink), while the "existing" UI components are in the Halo namespace and the "new" UI components are in the Spark namespace. Naturally, not every Halo component has a corresponding Spark component. So I *really* don't want to port everything just to find out that Adobe is so fickle that they changed their minds **again** for the final version. On a side note, you really don't understand the definition of "beta", do you, Adobe? "Beta" means (or is supposed to mean, at least) that the product is mostly feature-complete and no more massive changes are being made.
- The development build process is kludgy right now; I'd like to go back to a pure Ant build, but for now we at least have something that works. The test build, I'm sure, is utterly broken right now.
- Most of the existing code is still using the old auto-generated service code, as mentioned previously. These need to be updated to use the new auto-generated service. As I said, nothing is currently broken in regards to the web service, but having two sets of service code is unnecessary bloat.
- The new Package Explorer makes it much easier to see how the code *should* be organized. I've already performed some minor refactorings to move things around, but there are a few more that should probably happen.

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

On a related note, I've got some crap due for Webware on Thursday. Heineman basically gave us an assignment that says "learn Perl and make massive changes to this thing I wrote 13 years ago and haven't touched since". I'll probably be messing with that most of the day tomorrow. Good thing Pollice is back on Friday and we (hopefully) won't have any more assignments with such insane deadlines.

Posted in IQP. Leave a Comment »

On Flash Builder 4

October 27, 2009 — Richard

Or more accurately, "On Flash Builder 4 Beta 2". If asked to describe what "migrating" a project from Flex 3 to Flex 4 is like in one word, my response would be "teeth". Then, when asked what the hell that means, I would reply, "well, the experience is like *pulling* teeth, but you asked for one word."

No, I haven't "upgraded" my development environment to FB4; I created a new Eclipse installation and a separate workspace to test so that my development environment wouldn't be affected. I noted that FB4 apparently is now compatible with Eclipse 3.5, so I used that in lieu of 3.4, which is what we're using currently because the plugin architecture changed somehow in 3.5 and Adobe didn't update FB3 accordingly (or FB4 beta 1 for that matter). I suppose this could be the source of the first major problems I encountered (will have to test FB4 beta 2 with Eclipse 3.4 to know for sure), which are:

- I kept getting a bizarre "Windows – No Disk" error message, which meant I had to click "continue" about fifty bajillion times because I have one of those infinity-in-1 card readers installed in my PC. Eventually tracked this down to what appears to be a new preference in Eclipse; changing the General -> Web Browser preference from "use internal browser" to "use external browser" fixed the problem.
- Executing the launch configuration (to bring the app up in your default browser) doesn't cause the Ant build to run first like it does in the current IQPDE (new acronym: IQP Development Environment – like it? although maybe NPDDE would be a better choice). I currently have no solution for this.

Things just went downhill from there. When I tried to run the build, the mxmlc task failed epically because the JVM ran out of heap space! Geez, Adobe, I know memory and disk space is plentiful these days, but you seriously managed to use up *sixty-four megabytes* of heap space?!? For those who haven't taken CS3013 (OS), the heap is where all the stuff you allocate dynamically at run-time goes. I have **never** encountered this problem before, not even with Eclipse, which is a pretty sizable Java app, or with the SoftEng project, which was pretty massive towards the end.

So after I finally managed to build the project using the Flex 4 SDK, which is also still in beta, I tested it out in my browser. Whoa! It looks *really* different with the new "Spark" UI components. Of course, there's also a random hang when you try to edit a project response. Naturally, I can't reproduce this in the standalone debug player, but it happens in both IE and Opera.

Having become extremely frustrated, I've quit working on this for now. Flex 4 and FB4 have some great new features, as mentioned in a post I made a while back, but if moving up is going to require this much effort, it may not be worth it. Not to mention that this still doesn't tackle a host of other things, like updating the namespaces on all our MXML files to use the the 2009 namespace (as opposed to the current 2006 namespace). This will no doubt break a variety of things. Oh, and Adobe randomly decided that "mx:" prefixed on everything is no longer a best practice. Well, they can bite me as far as that goes. Also the "data services import" functionality is different and the generator has been updated, so that will probably cause some issues as well.

Having said all of this, I'll probably continue tinkering with FB4. The new Package Explorer is a *vast* improvement over the Flex Navigator in FB3. Being able to see documentation without having to Google it is a major plus as well. I also hear

there's some new code completion stuff in there, like auto-generating event handlers and accessors. Skinning applications also appears to have gotten much easier in Flex 4, although having never tried to do so with either version, I can't really say for sure. Not to mention that Eclipse 3.5 feels a lot faster in general, and the Flex 4 compiler is definitely speedier than Flex 3's (a case of the classic memory vs. speed trade-off, perhaps?). Then there's also the fact that you'll be forced to upgrade eventually: Adobe's attitude, as with all other software developers, seems to be "support the last two major versions", so FB4 drops support for Flex 2. This means that when Flex 5 comes out, they'll probably drop support for Flex 3. Of course, that's no doubt in the far off future; they haven't even finished with Flex 4 yet. Still, always be prepared…

Posted in Epic fails, IQP, rants. Leave a Comment »

Back to work

October 26, 2009 — Richard

It's niiiiiine o'clock on a Saturday…

Oh, wait. It's Monday. Rats.

That means tomorrow is Tuesday, October 26, which marks the start of B-term. Despite being a Tuesday, we're on a Friday schedule tomorrow in order to make everything balance out. WPI loves doing that, making the first day of the term have a different schedule than it should. They did it in A-term too. Well their dastardly attempts to throw everyone off ***won't get me!*** …Mostly because in terms of classes, Tuesday == Friday for me this term.

As planned, I managed to not write any code for the project over the break. I didn't quite manage to stop thinking about it though. Mostly, I thought about the divider problem, probably because that was the last thing I worked on before the break. I realized that I had the wrong script for *SoftEng II: The Wrath of the Big Empty Space* (alternative name – *SoftEng II: Attack of the Big Empty Space*). You see, the whole reason the divider problem exists is because I didn't implement the Big Empty Space faithfully. ***Oh, forgive me, Big Empty Space, for I knew not what I was doing!***

In *SoftEng: The Big Empty Space Picture* (no, I will *not* call it *SoftEng I: The Phantom Big Empty Space* because there was nothing "phantom" about it), the BES was implemented in Java Swing. It worked by having a MainWindow that sub-classed JRibbon and contained a JSplitPane which contained a SidePanel (which was sub-classed from… actually I can't remember where we ended up, it changed a couple times) on the left side of the splitpane and the BES (more accurately, any panel sub-classed from BigEmptySpacePanel which was sub-classed from JPanel) on the right. To change what was in the BES, you simply called a method on MainWindow to replace the contents, which caused MainWindow to call a method on the current occupant telling it that it was about to get kicked out, then replaced the panel and called a method on the new occupant to tell it that it was now on center stage. There was some other stuff we added towards the end too, like "locking" the BES so the panel couldn't be replaced, but that was the basic flow.

I'm sorry to say I did not do this in Flex. Instead we effectively have three MainWindows, one for each of the three major areas (Projects, Companies, My Account). Well, actually, we only have *two* since that last one is completely unimplemented. But the point is that the divider problem exists because of this fact. The divider is in a different position in different areas because it's actually a whole different instance of HDividedBox (or more accurately, a "subclass" thereof). So the path to fixing the problem is obvious: have only one instance.

Actually achieving this is a little more problematic. Unlike in the SoftEng project, we need the side panel to change depending on what area we're in. We also need some way of communicating between panels in the same area (e.g. user selects a project to respond to in the ShowProjects table, so there has to be some way to tell or some way for the RespondToProject panel to find out which project was selected). I think this is going to be the first thing I do, before dealing with "master" results, since this is a fairly important and somewhat sizable refactoring. I haven't got it totally nailed down yet; I think it's going to

involve making some custom constructors for the MXML components involved, which is something I haven't done before, in order to pass references to other GUI elements around. Alternatively, there could be singletons to handle it, I suppose. Like I said, I haven't completely thought this out yet.

In other news, I'll be perpetually parked behind my computer this term. Last term, I took Foundations, which involved zero programming, and Business Ethics, which obviously didn't involve much computer work either. This term, I'm taking the first Networks course, which is sure to involve a fair amount of coding (probably in C/C++, so I'm dreading this already), along with Webware, which is sure to involve a bunch of coding was well (especially since Pollice is teaching it). It's a little ironic that I'm already one term into an IQP centered around a *web application* and am only now going to take Webware. Maybe, if I'm *extremely* lucky, Pollice will just let us pick our own projects, and I can get his permission to re-use this. Yeah, probably wishful thinking, but I can dream, right?

Posted in IQP. Leave a Comment »

Third-time

October 14, 2009 — Richard

BZZZZZZZZZZZZZZZZZZZZZT!

A-term has just expired, everybody. Well, it has for me, at least. My finals were done yesterday, and after working on the project the whole day today, I'm officially calling it quits until the start of B-term. There are some minor tasks to take care of, like rolling out a new test build, but I'm not doing any more coding. We have now entered third-time. Not half-time, because it's not half over. Terrible play-on-words, I know.

So where do we stand?

Well, the good news is that I've implemented surveys and all the project calculations are now working. They are also correct, as far as I can tell, but I'll let Professor Danneels be the judge of that. All of this has been committed and will be part of this week's test build. The bad news is that implementing logout didn't happen, and neither did the app-site bridge.

I *did* spend an hour or two tearing my hair out trying to figure out the divider situation. Until now, the dividers in the app didn't "stick"; that is, switching from one screen to another caused the divider to jump around because the divider "remembered" its position in each individual state of each of the three (though only two are implemented) primary screens. This was bugging me, so I decided to fix it. It turned out the solution was trivial and I was trying to over-complicate things: instead of having an HDividedBox that's half-empty in the "base" state and adding a canvas to it in each of the real states, all I needed to do was add an empty canvas to the base state and add the state canvases to that instead of the box. Feels like a hack to me, but it works and it's *much* better than the other hack I came up with. Of course, this still doesn't change the fact that the divider doesn't "stick" when moving from one screen to another (e.g. Projects to Companies), because those are different instances of HDividedBox. I'll rectify that at some point, but it's not nearly as annoying now, especially since I've gotten the divider to "initially" be at 200 pixels from the left side of the box instead of dead center which required me to keep moving the divider to the left *every time* I relaunched the app because the important content on the right side was getting squished.

On a totally unrelated note, bust out the champagne! After around six weeks of blogging, and deleting a dozen or so spam comments from spambots, I *finally* garnered a real comment from an actual person today! Glad to know my trials dealing with Flex Builder's bugginess are at least helping *somebody* else out, Steve!

Anyhow, that's all for this third. Stay tuned for the third-time commentary on other crap I have to deal with!

Posted in IQP. Leave a Comment »

Progress!

October 12, 2009 — Richard

Good news: the screw-up is officially fixed and, in terms of project functionality, we are now back to where we were before we knew the screw-up occurred. A little bit of new terminology I've coined: the term "project response" is now used when referring to the entire package of answers a user provides for a given project; that is to say, a "project response" includes (or rather, will include) both the numerical inputs and the answers to the marketing/tech questions. This name was chosen to distinguish from the "project survey" term which is just the latter. Of course, coining new terms doesn't come without problems. You've got to make sure nobody else is using it, copyright or trademark it, keep an eye out for assholes who try to steal it, and, if you're using Flex Builder 3, make sure that you don't have a SOAP operation called *yaddaResponse*.

Wait, what?

Yep, it's the attack of the bugs again. For some reason, having a SOAP operation called XResponse breaks the web service code auto-generation feature in Flex Builder 3. Instead of XResponseResultEvent containing an XResponseResponseType, it contains a XResponseRequestType. I do not know, nor do I suspect I want to know, what could possibly cause this to happen. All I know is that this bug has *ruined my life*, or at least my scheme to corner the market on terminology coinage. So instead of having a GetProjectResponse SOAP operation, we call it GetProjectAnswer instead. Yes, yes, "answer" is a synonym for "response", but it's not the same thing, damn it!

Shakes fist *ADOOOOOOOOOOBEEEEEEEEEEEEEEEEEEEE!*

Ahem In other news, I realized that some of our SOAP operations could potentially fall victim to race conditions. Specifically, any operation that first used a SELECT query to check a unique constraint (e.g. a new user tries to register, so we check if the username is available) and then used an INSERT if the check succeeded could be vulnerable. As an example, take two users trying to create an account with the same username. User A submits his request, followed by user B. The server begins processing user A's request, but the OS timeslices the process *after* the SELECT check but *before* the INSERT statement. The server then processes user B's request in its entirety; user B has now claimed the disputed username. The server then re-schedules user A's request, which fails with a generic error message because, as far as the script can tell, there's no reason the INSERT shouldn't have worked.

There are a few ways of dealing with this. The first is to ignore the problem; the odds of something like this happening are fairly slim, but I'd prefer not to leave it up to chance. The second is to use locking so that nobody can mess with the table between the time we execute the SELECT and the time we execute the INSERT; MySQL's documentation recommends against locking InnoDB tables, and locking would probably cause a fairly significant performance hit. Option three, which I opted for, is to simply perform a blind INSERT, without doing the check first, and then seeing if there's an error message about a duplicate key value. It's kind of a hack, but it's simple and gets the job done.

One thing I did notice is that even if the INSERT fails, InnoDB still performs the auto-increment (this applies to our Projects table, which has an ID as its primary key). I suppose theoretically you could use this to DDoS the service, by racking up the auto-increment until it passes the limits of the INT data type the column is declared as. According to my calculations though, even if the server could handle 100 requests per second (which, given that we're in a shared hosting environment and the database is located on a different system from the web server, seems unlikely), it would take a year or so to max out the column. If this did still become a problem, we could always fix it by just altering the table to use a BIGINT instead of an INT or possibly resetting the auto-increment counter somehow. Alternatively, we could adjust the schema and the PHP scripts to eliminate the use of an ID column; an ID column is being used mainly to save space (one INT column versus two VARCHAR columns).

I'm considering going the "dual VARCHARs" route anyway, because every time I want to retrieve a project response, I have to go to the Projects table to find out what the ID is using either a subquery or a JOIN. This is your classic *memory*

usage vs. *processing speed* dilemma: the ID column saves space (or it *should*, at least, though I haven't really checked any authoritative sources) but slows processing, whereas the VARCHARs would use more space but speed processing (since they eliminate the need for subqueries/JOINs). This is something I'm going to look into at some point, maybe later this week.

Speaking of this week, I abandoned my plan to have the Business final finished by tomorrow. Instead, I'll spread it out over tomorrow and Wednesday. I'm not sure how this will impact what I get done for the project for the rest of the week. I'll be happy if I can get the surveys implemented; figuring out the app<=>site bridge would be nice, and not having logoff is really starting to get annoying. Those items are on top of the list; analysis UI will definitely be delayed until B-term, as predicted.

Posted in IQP, rants. Leave a Comment »

Closing out the term

October 8, 2009 — Richard

As I predicted yesterday, I did nothing on the project today. Well, I did do *one* thing: I decided that the 1/10th of a step not implemented yesterday can wait. I've added it to the to-do list on the SourceForge wiki, which is getting **very** big. There are going to be a lot of things to mop up once the major functionality is implemented. This means that next week, I'm diving straight into cleaning up the project responses situation. I'm not exactly sure when that will happen, but it'll probably be later in the week. The Foundations final is Tuesday, and while the take-home Business Law final isn't due until Thursday, I hope to have that finished for Tuesday also. This would leave Wednesday and Thursday completely open for marathon coding sessions.

This of course will be followed by a week off, during which I intend to do absolutely nothing about this project. I'm starting to feel a little burn-out from the pace. I think this pace is necessary in order to finish, and Matt clearly agrees, since he told me earlier that he was up until 5AM today working on the website. The hard work is paying off; the parts of the app that are implemented work flawlessly (or so it seems from my relatively limited testing), and the website looks fantastic. Having said that, it's a well-documented fact that vacations improve efficiency, and I have some issues at home that need dealing with that I've been ignoring (dishwasher'd parts, anyone?).

And for those of you panicked that I won't update this blog over the break, thus depriving you of my rants and lousy sense of humor, I'll admit that it probably won't get updated as much, but I'm sure I'll find *something* to rant about. Even if it is just about how douchy the Republicans are being.

EDIT: No test build this week. Why? No point – with all project functionality disabled, there's nothing useful to tinker with.

Posted in IQP. Leave a Comment »

9/10ths of a step forward

October 7, 2009 — Richard

Short post because I'm tired. It's closing in on 11PM and I've spent the whole day implementing company user management. It is now mostly finished, except that it is not yet possible to kick a user out of a company. I haven't decided yet if I'm going to stuff this into the to-do list on the wiki for later implementation or just go ahead and finish it. I certainly have bigger fish to fry. In any case, it's unlikely I'll do any work on the project tomorrow, since I've utterly ignored the homework for my other classes. I don't expect that those assignments will take very long, but the Foundations homework in particular has a tendency to be tedious and mind-numbing. After I finish it, I probably won't feel like doing anything. And then there's the fact that I have three TV shows to watch tomorrow (FlashForward, Fringe, and The Mentalist, in that order).

Posted in IQP. Leave a Comment »

The cost of mistakes

October 6, 2009 — Richard

To think that during SoftEng, I thought the cost of hacking something together only to un-hack it the next week was expensive… now I'm looking at a far worse situation. At least in the hack/unhack situation, the implementation was mostly right, if incomplete. The cost is *far* higher when you mess up the implementation altogether.

Allow me to explain. Rewind to this weekend when I was spinning ideas for dealing with the "multiple surveys per project" conundrum. It turns out I was ~~slightly~~ ~~partially~~ ~~mostly~~ completely off base there. I was under the impression, until earlier today, that it was *just* the surveys that were per-user. It turns out that Professor Danneels actually wanted the *entire* list of inputs to be per-user.

As one might imagine, this has significant ramifications. It forces massive changes to the database schema, which then ripples through the rest of the application, effectively breaking almost everything I did last week. But it has to be done – this is what the customer wants, and it makes more sense anyway, thinking about it.

So now I have to decide how to go about cleaning up this mess that I made. I've updated the database schema already. For the SOAP service, I'll fix the data types, update the operations for creating and joining companies, and ignore the rest for now. For the Flex app, I'll do the same, and disable the project functionality for now. I'll focus on implementing company user management, as planned, then come back to the project issues once that's finished.

Speaking of plans, this means that the wonderful (if unrealistic, even at the outset) schedule I came up with last week is completely shot. It was unlikely before, but now it's clearly impossible that we'll have 100% of what was in version 1 by the end of the term. This doesn't mean that we're falling behind… stop looking at me (err, your screen) like that! I'm not kidding and I'm not bonkers… well, not completely anyway. The key thing to realize here is that we're implementing new functionality along-side re-implementing existing functionality, so having 100% of what was in version 1 is a somewhat useless metric.

So, new plan (will update the to-do list shortly):

1. Implement company user management.
2. *Fix* the project responses implementation. This step involves getting the "numbers" to work again.
3. *Finish* the project responses implementation. This step involves re-implementing the surveys from version 1 (and adding a new one for v2).
4. Implement "finalizing" a project, by which I mean inputting the "master" results for a project. These are the numbers that will be used for the analysis (the BOBs and the rank-ordered lists).
5. Implement the analysis UI.

Once the above list is completed, I think we can reasonably say the app has reached beta status; we should reach this point by mid B-term. Beyond that, it'll be mostly about adding features that aren't core to the application's purpose and polishing the app to a nice shine. The ideal longer-term schedule is to reach gamma (release candidate) status by early C-term, with RTM (or, from Wikipedia, RTW – release to web – which would seem more appropriate in our case) in mid C-term. The end of C-term will obviously be focused on writing the IQP report and documenting everything we did for the benefit of teams that come after us.

And since I *almost* forgot, and since I know you're on the edge of your seats wondering when it'll happen, yes, integrating the app with the CMS will be in there somewhere.

Posted in IQP. Leave a Comment »

I need a fly swatter

October 5, 2009 — Richard

Yep, you read that right. The bugs in the Flex app are starting to multiply out of control and they're popping up faster than I can kill them. I fixed several bugs

today, none of which I knew existed before today. Some of the bugs weren't possible to catch before since the functionality that was necessary to show they existed wasn't implemented yet. As more things get implemented, we'll be seeing this on an increasingly frequent basis.

I also (re-)discovered a rather serious bug that I thought I had quashed two weeks ago. Despite my "hack-tastic" solution, default company permissions are *still* not being set correctly. Therefore, I am making a detour from the schedule to investigate this as it is a rather serious bug, especially now that users can almost join companies.

Which brings me to what I spent most of my time on today. Users can now ask to be added to companies, at which point they will receive the default permissions for the company and be given "requested" status. Still to-do: implementing the company user management UI so that an authorized user can actually approve the request by changing them to "active" status. Also I spent about two hours figuring out how to integrate FlexSpy into our development builds (and remove it from test/production builds). I opted for a hotkey (Ctrl-Alt-Space) rather than a button somewhere on the screen (for simplicity and also so we don't run into a "I want to put something there so we have to move the FlexSpy button" issue later). Consequently, the integration is somewhat... quirky. Flex, as usual, sucks massively and lacks true support for hotkeys, so you have to focus on the application first. By focus, I don't mean "click somewhere in the app", because annoyingly that doesn't work. Instead, you have to focus on something that is *really* focusable, like a text input or a datagrid.

And to close out our grab bag of news, the rumors were right for once: FB4 beta 2 was released today. Hopefully I'll have time to tinker with it later this week.

Posted in IQP. Leave a Comment »

Jumping the gun

October 4, 2009 — Richard

It appears I was jumping the gun a bit earlier. With more research, I've discovered a few things I said in my last post aren't true. Mind you, this stuff wasn't immediately obvious; some of it took a bit of digging to uncover:

- **I said:** "As far as I can tell, they also haven't decided if they're going to give FB4 to students for free like they're doing with FB3 (more accurately, I think you can get FB3 for free as long as you have a valid .edu e-mail address)."
  The truth is: I managed to uncover a post on Adobe's forums indicating that FB4 *will* be available to students for free. The e-mail address thing was also wrong; you're required to provide an ID showing that you're actively enrolled or employed by an educational institution, or a letter on the institution's letterhead saying that you are. Also, the version that they're offering is FB3 Pro, not the less featured FB3 Standard. One could then infer that Adobe will similarly offer FB4 Premium (re-named to avoid confusion with Flash Professional) for free. This may end up being a moot point since...
- **I said:** "If you want the advanced datagrid and charting components (which we do, because half our app depends on them!), you have to get the Professional version which is an extra $450."
  The truth is: Well, it depends on who you ask. I found this page, which indicates that as of August, these components are now freely available. Adobe's FB4 release notes say that there is still a watermark, but that page hasn't been updated since June. I can confirm that following the steps on Adobe's site to download and "install" Flex SDK 3.4 and the data visualization components works, because I did this and performed a clean build of our app with no problems. Of course, this was done using an Eclipse installation with a registered version of FB3 Pro, so I can't say for sure whether the watermarks would be there or not if we only had the Standard version. Adobe's version comparison page indicates that the only other difference between Pro and Standard is that Pro also has some other fancy stuff like the Flex Profiler, so I find it somewhat difficult to believe that they would release these components for free.

On a side note, it irks me that FB3 ships with Flex SDK 3.2, since 3.4 fixes some big cross-site scripting hole. Honestly, Adobe, would it have killed you to update

the SDK you ship with your product?

Anyway, these developments tip the scale in favor of at least giving the FB4 beta a shot. There's a rumor on the Adobe forums that a second beta will be released tomorrow, so I'll wait and see if that happens. Either way, later this week I'm going to bust out my laptop (so there won't be any issues with having FB3 Pro installed) and install FB4. This will allow me to answer the "data visualization components" question above and allow me to test the procedure I documented in the wiki (to some degree, since it'll be with FB4 instead of FB3). Something I noted about the FB4 beta that bothers me is that it doesn't work with Eclipse 3.5 yet, just like the FB3 plug-in. Then again, the FB4 beta was released at the beginning of June and Eclipse 3.5 was released about 3 weeks later. I read somewhere on Adobe's forums that the changes in Eclipse's plug-in architecture were made late in the game and Adobe didn't have time to fix it before the release; hopefully, the second beta will resolve this. The final release is now scheduled for early 2010.

On a mostly unrelated note, I discovered that our development build script wasn't generating debug builds because that's off by default, so I've updated build.xml to turn debug on for development builds. This will remain off for test (maybe) and production builds because it adds bulk to the SWF. Making this change will make my life easier, because now I'll be able to see the exact file and line that a problem occurs on.

Posted in IQP. Leave a Comment »

« Older posts

Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

Weekend thoughts, 2e, Part II

October 4, 2009 — Richard

Remember how a while back, I said two-way binding couldn't be used because Flex 3 doesn't really have support for it? I noted at the time that Flex 4 *will* support it, but it's still in beta and thus, not ready for production-level projects.

I'm being forced to reconsider that judgment.

True two-way binding would be wonderful (if it doesn't cause any bizarre bugs, that is), but the main reason I am now sorely tempted to upgrade to the FB4 beta are the new features in the IDE, not the new features in the language. People who have used Eclipse to develop Java software can attest to how truly wonderful the experience is. Enter a dot after a variable, say, and Eclipse will automatically present you with a list of every method you can execute on that object, and selecting any of them will prompt Eclipse to display full JavaDoc for the method, explaining what it is, what it does, what it returns, etc. FB3 is sorely lacking in this department. It will give you a list of stuff in alphabetical order, but that's about it. No documentation explaining what anything does, so you either take a guess or go Google it to find out. Not to mention that in Flex, you get a whole lot more than just methods on a class, you also get properties, events, and a smorgasbord of other stuff; the worst part is that if I'm looking for, say, an event to attach a listener to, there's no way to filter the list to show just events, so instead I have to wade through the whole list, keeping an eye out for the "thunderbolt" symbol that differentiates events from the other things.

FB4 fixes all of this and then some. Watching some of the video demos of it on Adobe's website, I am impressed by some of the features added in the fourth major version of this product *that should have been in the **first** major version*. Seriously, does Adobe *enjoy* making developers' lives miserable? Or were they looking to line their pockets that much more by forcing people to pay for upgrades to get features that should already have been available? I don't know what the answer is but the fact remains that if I want access to these "duh"-level enhancements, I have to upgrade.

This presents something of a problem. Adobe hasn't decided how many limbs they're going to charge for FB4 yet. As far as I can tell, they also haven't decided if they're going to give FB4 to students for free like they're doing with FB3 (more accurately, I think you can get FB3 for free as long as you have a valid .edu e-mail address). Right now, we're using the licenses Professor Danneels paid for last year. Major version upgrades certainly aren't going to be free; Adobe may not even decide to offer them at a *discount* for all we know. This stuff ain't cheap either: minimum $250/license for FB3. *Minimum*. If you want the advanced datagrid and charting components (which we do, because half our app depends on them!), you have to get the Professional version which is an extra $450. That's for the *current* version; remember, we don't know what pricing for FB4 will look like, which makes it rather hard to ask whether an upgrade is in the budget.

Adobe's made sure that people can't just move to FB4 beta and then, if they decide the upgrade isn't worth the price, go back to FB3. Even if you don't use any features of the Flex 4 SDK, FB4 changes the files containing the project meta-data in a way that makes it impossible to open them with FB3, apparently. So a migration from FB3 to FB4 is a one-way ticket.

All of which leaves me stuck between a rock and a hard place. If I want access to features that will make me more productive (and are standard for other languages), I have to upgrade to a beta product, which may be unstable, without even knowing what the final version will cost. To say I'm conflicted about what to do here is a massive understatement. I'm going to weigh this some more and get some input from Professor Danneels about it. For now, I'll just have to keep trudging along with FB3.

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

Posted in IQP. Leave a Comment »

Weekend thoughts, 2nd edition

October 4, 2009 — Richard

First up is the project schedule for this week. As previously noted, I plan to work on joining companies and company user management this week. I would also like to prototype my idea for bridging the app with the website that Matt has been developing; this is something we added to our plans for this week in our report. If by some miracle all of that gets done, I'll move on to implementing surveys and analysis, which is tentatively planned for next week.

Speaking of surveys and analysis, it occurred to me that we have a bit of a conundrum to deal with. One of the new features for version 2 is that multiple users will be allowed to take surveys on the same project, and users with the appropriate permissions will be able to aggregate the responses, generating the mean, standard deviation, etc. Our database design takes this into account by separating the survey information from the project information. This means that there is no (and currently, cannot be) a "default survey" or some such, which presents a problem. The bubble charts (and some project calculations) require survey data (e.g. market/tech risk or competitive advantage) to work. So how do we solve this problem?

- We could create the notion of a "default survey" that any user will be allowed to edit.
  Pros: Could be accomplished easily by allowing the user to be null in the survey table.
  Cons: Would probably generate a lot of user confusion and UI/code duplication.
- We could allow the selection of a "default survey" from the surveys currently available (i.e. all surveys taken by any user for the given project).
  Pros: Shouldn't cause any confusion about surveys.
  Cons: Would effectively allow any user to see how any other user answered the survey. Professor Danneels has requested that survey aggregation be a restricted privilege, presumably to preserve the anonymity of surveys; this option would violate that.
- Similar to the above, we could simply say that the survey of the user who created the project is the one that gets used.
  Pros: Problem in cons above is mitigated.
  Cons: Problem is cons above is not eliminated, merely confined to users only being able to see the survey response for the person who created the project.
- We could auto-aggregate the available surveys.
  Pros: Would seem to be the most natural solution.
  Cons: Committees do not necessarily produce the best/correct answer to a given problem. Also does not solve the issue of being able to see other users' surveys; e.g. if only one user has taken the survey for the project so far, the "aggregation" is based completely on his/her answers.
- We could use the current user's survey as the default.
  Pros: Has none of the problems of the other options.
  Cons: Results are no longer reproducible because user A's charts will use user A's survey, while user B's charts will use user B's survey.
- We could disable any survey-related analysis until a survey aggregation is performed or an authorized user selects a survey from the ones available.
  Pros: Fixes all the above problems.
  Cons: Severely breaks the app's flow, since you have to wait until a user with the appropriate permissions takes action on your project before you can do any useful analysis. One of the main benefits of a tool like this is the instant analysis it provides.
- We could allow the selection of a subset of the above choices for dealing with survey data.
  Pros: Would allow users to have it their way.
  Cons: This is not Burger King and doing this is not like adding a tomato slice to a hamburger. Would immensely increase the complexity of the application and

the database, and would require significantly more time to implement as a result (effectively, this says "implement options A *and* B *and*..." plus the ability to pick between them).

None of these options are perfect. It's entirely possible there's an option I haven't considered here, so if anyone has any ideas, feel free to suggest them. If we have to pick from the above, my preference would be auto-aggregation. This feels like the most natural way of solving the problem, but it does have issues, as noted. This could perhaps be mitigated by adding the option (for appropriately permissioned users) of picking a given survey as the default, which would at least fix the "design by committee" problem. The other issue of being able to see how other users answered the survey only occurs if you know several things about the aggregation. Namely, you would need to know how many people have taken the survey and how a lot of them answered to be able to deduce how the remaining users answered. This is easy if you know that only one or two people have done the survey; if ten people have done the survey, determining how any one person answered effectively becomes impossible unless the other nine are willing to tell you their responses.

Now for an abrupt gear change to another topic. Over the weekend, I've stumbled onto two libraries that could be very useful on this project and I am definitely planning to integrate at least one of them.

- fxSpy – This "library" is more like an application that allows you to inspect a Flex application's GUI. You can see the hierarchy of how elements are organized, check out what their properties and styles are set to, and even change them on the fly! Very cool. Sadly, it doesn't provide any functionality that would allow you to, say, check out the contents of a bound data provider at run-time. Still, integrating this tool will eliminate the need to recompile/launch/login/navigate in order to test a purely cosmetic change. Obviously this will require some changes to the build script so it's not compiled into test/production builds.
- FlexLib – This is a library of useful GUI elements. The only ones I'd say we might be interested in right now are the prompting text input/area, so I'm not sure if this is worth integrating. Were this a desktop app, I'd certainly go for it, but this is a web app; size matters and I'm not sure that the benefit the library provides is worth the added file size (and download time, by extension). I will keep this library in mind; if we end up needing other elements it provides, I'll integrate it.

And that's all for this post. If I decide to work on the project later today, it'll probably be on the application-website linkage issue.

Posted in IQP. Leave a Comment »

The best laid plans…

October 1, 2009 — Richard

Well, you know what they say about them. In case you're still not clear on my meaning, no, I didn't get anything done on the project today. The Foundations homework in particular was a bitch to finish. On a more positive note though, I did create and upload a test build for Professor Danneels to play with, which I've included a link to in our weekly report. Said report is still being drafted but it's mostly ready.

Also, I decided that, in celebration of our big "useful functionality!" milestone and first official test build, now is a good time to start tagging code. Specifically, each week, whatever revision is used for the test build will be tagged with the date. So, if we wanted to, in the future, we'll easily be able to recreate the application as it existed at any number of points, since we also have available all revisions of the database schema in the documents section. I can't think of a reason why we'd need to, really, but you never know what might come in handy.

And that's all for this week, folks! Tune in next week for more exciting progress updates! Or this weekend, if I find the time to go look at the dishwasher'd hardware.

Posted in IQP. Leave a Comment »

Wednesday work, part II

September 30, 2009 — Richard

When things finally get rolling, man, do they **roll**. I just implemented saving project edits, only taking about an hour to do so. This is fantastic because *finally*, five weeks into the project, the app has acquired some useful functionality: you can enter project data, the app will crunch the numbers, and you can save the data. The UI I put together for doing so isn't perfect, but in my opinion, it's a damn sight better than the last version. It does things like, ya know, *validating* user input.

This is the first week I've actually felt satisfied with what I've accomplished. I did most of my Foundations homework earlier (it was, by **far**, the most tedious, boring, mind-numbing assignment I've ever had to do), so tomorrow I just need to finish that up and do the BL homework. Those shouldn't take long. After that, in lieu of starting to implement anything big, I'll probably do some polishing. We have a log out button, but it's not hooked up yet, so I'll probably tackle that. Also I'd like to finish streamlining the test build procedure. Once those items are finished, I'll tag the code and deploy to testing.

That should do it for this week. Next week I plan to implement joining companies and company user management in general, as reflected on the newly updated to-do list. After that, for the last week of the term, it's on to implementing surveys and project analysis. This will put us in an excellent position going into B-term, if I can pull it off. Unfortunately, at this point I'd give better than even odds to that schedule being too tightly packed. Editing projects happened quickly because most of the infrastructure was already in place: we already had a Project complex type in the SOAP server, we already had some other project SOAP operations implemented, and the editing UI was already finished; all I had to implement was making the Save button work. Implementing each of those major items is something I expect to take a while, more along the lines of implementing the entire project UI so far. We're talking new screens, new SOAP data types, new SOAP operations, the whole works.

Also it would be really nice to implement deleting projects at some point, since I accumulated 22 projects trying to kill that refresh bug yesterday. Yeah, I could delete them using phpMyAdmin, but… it's the *principle* of the thing.

Posted in IQP. Leave a Comment »

Wednesday work

September 30, 2009 — Richard

As I expected, I was able to finish the refactoring last night, so today I've been doing more work on project editing. We now validate all fields in the form; if any are invalid, the save button is disabled and the user can mouse over the field (outlined in red) to find out what the problem is. We also now calculate most of the output values, on the fly, as the user changes the inputs. We can't yet calculate any of the risk-adjusted values because we have no risk data right now, but the others are all implemented. Also all of the fields now have tooltips that explain to the user what the field is, and all fields note the expected units in parentheses.

This might be all I get done today, as I want to get some work done on my Foundations homework and preferably also finish my Business Ethics homework (well, I suppose Business *Law* would be more accurate now that we've moved into that phase of the course). In any case, the next bit of work to tackle is making the submit button work, so that the edits are saved to the database.

Posted in IQP. Leave a Comment »

RefactorFest '09, Take Two

September 29, 2009 — Richard

I started refactoring the app's architecture today. Might finish tomorrow, but I need to do homework tomorrow too, so no promises. It took forever because when I refactored the project screen, this truly strange bug appeared that resulted in the project list not getting refreshed after creating a new project. I

have absolutely no idea what causes the bug, despite much testing and analysis. Fortunately, I was able to hackfix the problem with a single line of code.

At the same time, I committed the project edit screen I was working on despite the fact that it's completely non-functional at the moment. This is still a high priority, after finishing the architectural refactoring. Said refactoring should be relatively quick and painless; I may even finish it tonight, because pretty much all that's left is just moving a few things around.

In other news, I have achieved my goal from yesterday of seeing double digits under weekly commits: between this time last Tuesday and now, I made 10 commits. Woohoo!

Posted in IQP. Leave a Comment »

Headaches, continued

September 28, 2009 — Richard

After I got project creation working earlier, I started working on project editing. Note that under our new system, surveys are (will be) separate, so this is just entering the raw numeric data. I began by making it possible to see what projects are available for editing, which was simple enough. Then I whipped up a reasonably good editing UI in a short amount of time, then I started trying to figure out how to plug things in. And that was where the headaches *really* started.

I wanted to do two-way binding between a variable and a field so that the field would automatically get its data and when the user changed the field, the internal variable would automatically be updated. Unfortunately, Flex 3 doesn't really provide support for this. You can fake it using a <mx:binding> to do whichever direction you don't have, but this caused the validator I attached to the field to do weird things, like complaining about invalid characters in an empty field. It took me forever to figure out that the binding was the problem. Flex 4 will have some level of support for two-way binding, hopefully without the strange side effects, but it's still in beta so I'm not inclined to use it to write production software.

Anyway, now that I have a better idea of what needs to happen here, I'm confident that a solid day's work will get this up and running. This will probably get done tomorrow, because I'm determined to see double digits under "commits this week" on SourceForge. Some work will probably get done on Wednesday and Thursday as well, but not as much, since I have homework for both of my other classes due Friday.

I'm also going to spend some time seriously considering (read: drawing messy diagrams on a whiteboard) the internal architecture of this application at some point this week, probably tomorrow. SoftEng teaches that you make architectural decisions at the point where the cons of postponing the decision outweigh the pros. I feel we're at that point. We've got the Global mess and projectMain is starting to become a bit unwieldy. Some architecting should hopefully resolve these issues, but I have to decide exactly how it should be done first.

Posted in IQP. Leave a Comment »

Flex headaches

September 28, 2009 — Richard

I had been planning to post late last night, but I was too tired. I was working on implementing project creation; I finished this just a few minutes ago. I hadn't expected it to take so long, but my initial sense overlooked the fact that I needed to implement some permissions checking along with actually creating the project. I also ran into a few roadblocks along the way.

The first roadblock was trying to get a combobox to display the right field. We have, effectively, an array of CompanyPermissions objects, but we only want the company name displayed in the combobox as the other information is irrelevant to the context. Turns out this is pretty easy to fix: just specify a labelField.

The second roadblock was caused by me not remembering code that I wrote last

week. I screwed up a few things, like checking $_SESSION['username'] instead of $_SESSION['user']. This turned out to be a good thing though, because it forced me to reconsider my error handling mechanisms when I needed to debug. I created some utility functions for permissions checking and initially had them return a boolean indicating whether the user can perform the action, or null if the check failed for some reason. Of course, returning null makes it pretty hard to pass anything back that might explain WHY the check failed. It occurred to me that exceptions would be a nice way of handling internal errors, so I did a Google search, and lo and behold, PHP has exceptions! So instead of returning null, we now throw an exception instead.

The third roadblock (well, it was actually the first chronologically, but it's much more exciting to save the best for last) nearly caused my jaw to fall off. I had been using a HashMap to store the company permissions in order to make accessing them by name easy. We aren't using that functionality yet, but I expected it would be handy later. Instead, it caused a problem because ActionScript doesn't have generics.

Yep, you read that right. No generics. Every other strongly-typed modern programming language I can think of has generics, although they may call them something else (generics is the Java term). This is very annoying, because it means that when you put something into ANY collection, when you go to take it out, you get it back as a generic Object instead of the class it actually is. You can cast, but this only ensures run-time type safety, not compile-time type safety. The only other option is to subclass the collection class and add methods that will return the more specific object type, but this still doesn't really solve the underlying problem. I've opted to go with this since Flex auto-generated it for us, but it still irks me.

What's worse is the fact that in Flex SDK 3.4, Adobe sort of but not really added generics. There's now a Vector class that takes an object type, but that's it. You're not yet allowed to create your own generic classes. I can't imagine why they would implement a half-baked solution like this; probably they're not actually implementing generics, just achieving the effect using some mad haxx.

In other news, I also worked on the test deployment procedure a bit. The database stuff is mostly nailed down, so now I just need to add the FTPing to the build script and we'll be where I initially wanted to be. I've noticed that there is an Ant task for talking to SQL servers and now I'm wondering if it's feasible to include that in the test build as well. This might be more work than it's worth though; I need to investigate further. In any case, this isn't on the top of my list, since we have a procedure that works, even if it's not as automated as I'd like. Implementing project editing is up next.

Posted in IQP, rants. Leave a Comment »

Weekend thoughts

September 26, 2009 — Richard

A few things I've been pondering or discovered over the last two days:

With the website template almost ready, it's time to start figuring out how to make a plain-jane HTML login form work with a Flex app. I initially thought of several different ways of doing this, and none of them were particularly appealing:

- Construct the page with the Flex app in it dynamically, passing in the username and password using FlashVars. This would work great, except for the fact that web browsers have a tendency to cache pages, which creates a security problem. Yes, there is the no-cache pragma, but this relies on the client browser to honor our request, which does not make for good security.
- Pass the username and password using a query string. As above, this would work great, except that browsers *also* have a tendency to cache links. There's no way I know of to ask them not to, and even if there were, the above argument still applies.
- Use one of the above methods, but only pass the username. This works, except it would make it impossible to auto-re-login in the event of a session timeout. That feature is one I feel is important, because it helps create a

seamless user experience. Also this would create a difference between logging in with the HTML form and logging in with the Flex app, since under the latter login, auto-re-login would work. Unless we crippled the Flex login to make it the same as the HTML login.

- Store the password in the PHP session along with the username. Then have the Flex app call a "getLoginCredentials" operation. Fantastic, until one considers the possibility of a session hijack, a man-in-the-middle attack, or any number of other attacks.

So we *really* don't want to do any of those. What options does that leave us with? I wasn't sure, until I realized that a series of things I'd come across in my search could be combined to forge a solution.

The first was this. Yes, apparently you can talk to JavaScript from Flex and vice-versa. Great, but not particularly useful until I came across this: a simpler way of embedding SWFs into a page. Adobe's "template" is incredibly obtuse, and according to the SWFObject docs, Adobe is actually going to be adopting SWFObject for its templates in Flex 4 because it's so much simpler. An idea began to form in my head and finally became a real possibility when I found this link in the SWFObject docs. Can you see it?

The solution is actually so simple the other ideas I had would probably be *more* complicated to implement. I had been ideating under the assumption that we'd send the user off to a different page when they logged in; my breakthrough occurred when I realized that doing so is completely unnecessary. Here's the basic flow of my idea:

1. The user will enter their credentials into the login form, and click "Login" (or press enter, or whatever).
2. Clicking login, rather than sending the user off to another page, will simply trigger a JavaScript function.
3. The function will load the Flex app. Alternatively, we could load the app with the page. The former has the advantage of not wasting bandwidth if the user is just poking around our site and has no plans to log in and use the app; the latter has the advantage that loading the app will take less time or even no time (if the app is completely loaded already) to load when the user clicks the login button. We could even shoot for a hybridization of both approaches, where the "load on click" method would be the default but the user can check a box or something that will set a cookie that would cause the app to load with the page.
4. The JS function calls a Flex function. The Flex function takes the username and password that the user entered into the form and calls the login SOAP operation.
5. The JS function makes the app visible, consuming the whole screen. This should be easy enough to do with JavaScript + CSS; simply set "display:none" on the that contains the page's content, then set "display:block" or some such on the that contains the Flex app. This div would have a class or ID that would have the width and height set to 100% so the app would take up the whole screen.
6. ???
7. Profit!

This idea will, of course, require the browser to have a rudimentary understanding of CSS, but I think Matt's template already more than covers any CSS that is necessary for this to work. It will also require the use of JavaScript, meaning the user will have to have it enabled for the site to work, but I think Matt's design already requires JavaScript to some degree anyway. Making JavaScript that works cross-browser can be a real pain in the ass, but a lot of the legwork is already done for us by SWFObject; all we need to make cross-browser is the code that calls the Flex function and makes the CSS changes.

Some people might consider this idea a hack. It isn't in my mind, because it has the potential to provide useful functionality, namely the fact that we could already have the app ready to go by the time the user logs in. It also makes for a smoother transition when logging in and out since you're not going to a different page; logging out will simply involve reversing the CSS changes and you'll be right back where you were when you logged in. This could even be used

to create a "back to site" button or some such, where the app would minimize and hover at the top of the screen as a bar or something and the user could go read documentation, follow tutorials, etc. Thinking about it, the feature possibilities this presents are endless.

Having said that, I'm definitely *not* looking for more features to add at this point. Such features as the one just described are nice frills but far from necessary, and we already have more than enough unimplemented things on our plate at the moment.

Now for an abrupt gear shift to SOAP. Remember how, way back when, I said that I doubted NuSOAP or Flex supported the document/literal wrapped style that a lot of people are using? Turns out I'm an unobservant dolt. I had thought that Flex was the one creating all those XRequestType and XResponseType classes. When I looked at the WSDL that NuSOAP generates, however, I realized those were all specified as request and response types to operations. NuSOAP automatically used the wrapped style when I specified document/literal. Sweet! This eliminates my worry about the possibilities for conflicting method signatures in the future as the SOAP service gets filled out with more operations, because it won't matter. The only thing we *can't* do is overload a method, i.e. have an operation someOperation(x) and another operation someOperation(y, z) – the names would create a conflict. But we shouldn't be doing any overloading anyway, since WSDL 2.0 makes it verboten. We only use WSDL 1.1, since that is all that NuSOAP presently supports, but it's always best to keep an eye on the future.

And that's all for this post, folks. Today or tomorrow (or Monday, depending on when I have time and when I feel like it) I plan to whip up a "proof of concept" page to make sure my HTML + CSS + JS + SWF plan will work.

Posted in IQP. Leave a Comment »

The Global problem

September 23, 2009 — Richard

I just made a commit that fixes the "companies list doesn't update" bug and I'm already thinking the fix is a bad idea. Well, maybe not a **bad** idea, per se, but a sub-optimal one.

Essentially, what I did was create our very own special Global class singleton. We had a different Global class before that I had imported, but I decided it wasn't cutting the mustard. Due to the fact that it obviously couldn't provide explicit fields or getters/setters, it was impossible to check type-safety at compile time. This is not good, but sometimes you have to deal with it. I decided it was unacceptable because it meant code completion didn't work in the editor, since the editor couldn't know what data type the field was, and figured it would be better to just roll my own.

It is *better*, but looking at it now, it's probably not the *best* way to do this. The Global class was introduced in the latest revision, and it's already a smorgasbord of getters, setters, refresh methods, and event dispatchers for a variety of things that have little to do with one another. This is how you end up with God classes and ISwissArmyKnifes. So I'm thinking this is going to get rewritten *again*, if not now, then shortly down the road. It (probably) makes more sense to separate things into their own singleton DataProviders or some such so that related functionality is grouped into individual classes rather than having one giant class that has everything plus the kitchen sink.

It's somewhat ironic that this developed this way. I did something similar on the SoftEng project and while it wasn't terrible, it wasn't great either. Guess I haven't learned my lesson *there* yet.

In other news, my work earlier today to automate (as much as possible) creating and deploying test builds hit a bit of a snag. I wanted to use MySQL Workbench's Forward Engineer function to create the tables in the database. GoDaddy requires you to explicitly enable the remote database access necessary to allow this to work. Unfortunately, the option wasn't present when I created the database. I e-mailed their tech support and was told this:

> Thanks for contacting Online Support. In order to have direct database access as an option, your account would need to be migrated. You would need to backup and remove your existing database(s) and provide us with permission to migrate your account if you wish to proceed. Please note: company policy prohibits employees from removing databases for customers. Once the migration is complete, you can then create the new database(s) and use the restore function to bring in the database structure and data.

Since, according to Professor Danneels, nobody is using the site right now, downtime isn't a problem. The functionality would be useful, so I'm going to go ahead with this. I also need to finish the build target so it will FTP the file structure to the server, which requires importing some libraries that the FTP Ant task needs. I plan to do both of these things tomorrow and hopefully have a deployment procedure nailed down by the end of the day. This will probably be the last work I do this week since, as previously mentioned, I have a final and a paper due Friday.

Posted in IQP. Leave a Comment »

« Older posts

Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

The "Test" build

September 23, 2009 — Richard

After the waffling I wrote about yesterday, this morning I decided to work on creating the "test" build and figuring out how to deploy it to the npdportfolio.com server. I have successfully created and set up a test build on the server (available at http://www.npdportfolio.com/npdtest), but this was a highly manual process involving a lot of trial and error. I've already automated creating the necessary file structure, now I just need to automate FTPing it to the server (this time out I did it manually using FileZilla). After that, I need to figure out how to get remote access to the MySQL server so I can use MySQL Workbench's tools to create the appropriate schema (this time I did it manually by logging in with phpMyAdmin and running an exported script). We'll also have to figure out how to make the website that Matt is developing fit in with this once it's done. For now we're using the skeletal "app is the whole site" method for our builds. Production builds aren't implemented yet but a production build is only trivially different from a test build so I don't expect we'll have any problems with that.

Also I discovered a rather large oversight on my part, namely that after creating a company, the app doesn't refresh the company permissions so the companies table is still empty. That's what happens when you're writing code at 11:30PM; added to the bug tracker on SourceForge.

Posted in IQP. Leave a Comment »

The state of the project

September 22, 2009 — Richard

Well, after 8 or 10 hours of work today, we can now create companies and find out what our permissions are in relation to companies we are associated with. This is a solid step forward, but a tiny one in the grand scheme of things. I'll hopefully be able to do some more work on the project tomorrow and Thursday, but I'm not sure what results will come out of that yet. My mother is coming to ~~visit me~~ decorate my apartment tomorrow, and I have a short paper and a mid-term on Friday that need attention at some point.

I haven't decided what to tackle next. I could tackle joining companies but this is going to require more work than it initially appears; the workflow is that you ask to be added to a company and then someone with the changeUserStatus privilege approves it. This means that implementing joining companies is actually a two step process, because first I have to implement asking to join a company and then I have to implement approving the request. These are two separate UIs, obviously, neither of which is written yet.

Alternatively, I could decide to tackle creating and editing projects next. I'm leaning towards this option because this functionality is more useful and would probably be less or the same amount of work as implementing joining companies. Of course, even once this is done it is still relatively useless because the analysis UI (the Big Orange Bubbles, or BOBs as Matt and I are referring to them) doesn't exist yet either.

Of course, I also need to get a "test" build target set up; this was on the list for last week and it didn't happen, which means it's on the list for this week too. I'd rather not let it go until next week, but again, I'm not sure how much more work I'll be able to put in this week. It's occurred to me that this project could probably benefit from a third team member, but it seems rather infeasible to bring someone else on board at this point, being half-way through the term. Getting work on the app done should be less of an issue once the website is (at least mostly) finished, since at that point Matt will have time to work on the app. In the meanwhile, I'm enjoying not having to worry about anyone else causing

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

merge conflicts when I make massive commits.

Posted in IQP. Leave a Comment »

The sound of silence

September 21, 2009 — Richard

Nothing happened over the weekend. Absolutely nothing. I didn't even check if any of my waterlogged gear still works. This was partly because I didn't have much time to do so, and partly because I figure that as long as I don't check, the parts will be both working and junk at the same time. It won't be until I finally do check that I'll *force* them to either be working or junk. Gotta love quantum mechanics. (The alternative of course is that there are a ton of different universes, each with a different permutation of working parts, but that's not nearly as cool in this case.)

Also, almost no work got done on the IQP. Well, aside from hauling a big-ass whiteboard into my apartment to design on and checking into the whole TLS situation. Turns out that GoDaddy offers SSL/TLS certificates for a single domain for $30 a year, a pittance compared to most certificate authorities. Of course, this certificate will only cover a single domain; any other domains or subdomains won't be covered. This is OK, but will require having three separate Ant builds: a "development" build to run locally on the development computer that doesn't use HTTPS, a "test" build that can be deployed to an npdportfolio.com subdomain that doesn't use HTTPS, and a "production" build that can be deployed to npdportfolio.com that *does* use HTTPS.

Alternatively, test builds could be deployed to and accessed from a subdirectory rather than a subdomain (subdomains actually just point to subdirectories on GoDaddy anyway, I believe). This would allow us to use HTTPS on test builds, but wouldn't allow us to eliminate test builds because, in order to not interfere with the production build, we still need to change the location of our scripts and SOAP service and also change the database credentials. We could also decide to go the subdomain route and get a wildcard certificate, but GoDaddy charges six times as much for those.

Also, I noticed that the hosting plan is up for renewal in about two weeks. It's a nitpick, but personally, I would've opted for the Economy Linux hosting plan over the Economy Windows plan. The cost is the same, but with the Linux plan, you can get more direct access to the account using SSH. Also GoDaddy's help documentation says that plan upgrades, e.g. from Economy to Deluxe, only take 30 minutes on Linux versus 24 hours on Windows. Of course, by switching, you lose support for ASP and ASP.Net, as well as MS Access and MS SQL. The Economy Linux plan also doesn't gain you any languages; if you want Perl/Python/Java/Ruby you have to upgrade to the Deluxe plan. Ultimately, the hosting platform doesn't matter much, becuase we're unlikely to use any of the benefits of either platform; it's just a matter of personal preference.

I probably won't get any further work done on the IQP until tomorrow, because I just finished the Foundations homework and I still need to read a bunch of stuff for Business Ethics.

Posted in Home, IQP. Leave a Comment »

Weekly Recap: The Slow Motion Picture

September 18, 2009 — Richard

I'm about to send off our weekly report to Professor Danneels. What exactly was done this week?

In my opinion, not bloody much.

We sort of have a website developed, and our app now lets you login. Big whoop. I'm not satisfied with how little progress we made this week. This may have something to do with my SoftEng experience, in which it seemed like I was doing a third of the work, Mike was doing a third, and four other people combined accounted for the last third while everyone else sat around picking their noses. This is probably fairly close to the truth, but the key is that it felt like there was constant progress because someone was always doing something.

This project has a team of two people, and Matt spent his time this week working on the project's website.

I don't know, maybe I have an unrealistic expectation of exactly how much work I'm capable of doing on this project each week. It could also be that I kept redoing things that were already done as I discovered that the way I initially did them presented a problem somehow (like the SOAP changes and the mysqli switch). I'm hoping there won't be any more of those next week; if there aren't, I should be able to get a lot more things that actually matter done. My main goals going into next week are to add some company and project functionality, and by the end of the week I want to have a deployment procedure documented. The goal there is that each week, we'll accompany the status report with a link to a subdomain on npdportfolio.com that Professor Danneels can actually go to and play with. I'd hoped to do that this week, but I decided to keep focusing on adding actual functionality instead because right now there's not much to show, so having a demo is rather pointless.

Here's to hoping that next week is more productive. Now I have to go find out if any of that soaked hardware is still any good.

Posted in IQP. Leave a Comment »

Two steps forward, one step back

September 17, 2009 — Richard

After finishing up the login business yesterday, I started tinkering with how various parts of the app should be organized. I've started with how to present company functionality to the user, since that's what we need to implement next before we can begin implementing project functionality. I was about to start writing some SOAP operations when something occurred to me; unfortunately, I can't remember what it was now. But I went off on Google and discovered that PHP's mysql extension is one of at least three ways we can interact with the database.

Until now, we had been using the mysql extension, which gets the job done, but it's not pretty. It was developed back in the days before PHP had any object-orientedness and so is a completely procedural system, which nowadays makes it rather awkward to use. I discovered that this extension is now in maintenance mode, meaning it will get bugfixes, but no new features. It already doesn't support any features added in MySQL 4.1 or later; this isn't a huge problem because we weren't really planning to use any whiz-bang features like stored procedures or triggers, but if for some reason we need to, we won't be able to with the mysql extension.

This lead me to do some investigating, which of course turned up the mysqli extension and PDO. Checking the npdportfolio server's PHP configuration revealed that while PDO is available, no drivers for it are installed, so using that is out of the question. The mysqli extension, on the other hand, is available to use. Mysqli fixes the problems the mysql extension has. It can be used in an OO style or a procedural style (for easy conversion from mysql), it's being actively developed, and it supports all those things we probably won't use. These benefits aren't huge, and the mysql extension isn't going away any time soon (according to PHP.net, it *will* be included in PHP 6, despite some other sites claiming the contrary), so I wasn't going to bother making the switch. Then I discovered one major benefit to going the mysqli route: prepared statements.

I first encountered prepared statements in B-term last year, taking Database Systems I, but had pretty much forgotten about them having not done any database work since. Prepared statements have several nice features. For one, if you're executing the same statement repeatedly with different parameters, using a prepared statement results in a significant performance boost because the server only has to parse and validate the statement once, as opposed to doing it on every execution. We don't do that anywhere now and probably won't. More importantly, however, prepared statements take parameters. Parameters that you don't need to escape.

This is a Big Deal. Yes, it's easy enough to escape your inputs; the problem is usually remembering to do it. One unescaped parameter could sink your whole

site, so not having to worry about doing it is very convenient. It also provides the sprintf-like functionality where you give an input and specify what data type it is (string, float, etc); this also helps to prevent attacks.

So I bit the bullet and rewrote the two existing SOAP operations to use mysqli instead of mysql. I attempted a few rudimentary SQL injection attacks afterwards and I'm pleased to report that they failed epically. Finally... an epic fail that's actually a *good* thing.

In other news, Professor Danneels e-mailed us some documents today regarding the project. Securing data and data transmission are on his list of issues. The former I think we're well on our way to having, but the latter is going to require some investigating. It's not something I'm overly concerned about at this point; all it really requires is adding TLS to the mix, which doesn't impact how we develop the SOAP service (TLS is after all an acronym for Transport Layer Security). Exactly HOW to add TLS to the mix is the question. This isn't something I've done before, hence the need to investigate. Will add this to the to-do list.

Posted in IQP. Leave a Comment »

We have login!

September 16, 2009 — Richard

Logging in to the application is now implemented. I imported a utility class to assist with this. Changes that have been made and problems they create have been documented in the wiki.

In dealing with some of these issues (session garbage collection, mostly), I ran into the question of whether REST would be more appropriate for our application instead of SOAP. After doing some research, I decided it would be best to continue with SOAP. It's not like we have a lot invested in our SOAP infrastructure, but a SOAP implementation appears to be the simpler solution to me.

- SOAP enforces certain things like data types, where REST does not.
- REST "operations" are basically just big long URLs. This has an advantage of requiring less overhead, but I can see this becoming a problem under certain conditions, such as when you're sending a lot of data (surveys, anyone?).
- Flex Builder 3 doesn't support importing REST-style web services, meaning that we'd either have to use Flash Builder 4 (yes, Adobe is renaming it), which is currently in beta and thus probably not suited for a production environment at this time, or we'd have to roll our own code to access the service, which would slow development considerably.
- Most options for creating a REST server in PHP seem to require installing PHP extensions and the like, which is not an option for us since GoDaddy doesn't provide us with shell access. I suppose we could use ASP but that would also slow development a bit.

So now that creating users and logging in is done, the next step is to start creating the "main" interface, the screen that the user will see after logging in. As previously discussed on this blog, we already have an idea of where we want to go with this. I'm going to take a break for a while, then come back and start working on that.

Posted in IQP. Leave a Comment »

WSDL Hell

September 15, 2009 — Richard

As I said earlier, it's not a huge bug, but artf5612, the bug about Flex importing the web service with arguments in the wrong order, was bothering me, so I did some research and testing. The situation is far more complicated than it appeared.

As it turns out, WSDL allows you to use two different binding styles and two different uses. The RPC (Remote Procedure Call) binding style is more intuitive, but the document binding style has the advantage that it passes XML validation. The literal use versus the encoded use seems to be mostly about whether type

info is included in the SOAP messages on top of the WSDL file (encoded) or not (literal).

NuSOAP uses RPC/encoded unless you specify otherwise. Apparently, while RPC/encoded is (technically) valid WSDL, it's not WS-I (Web Standards Interoperability Organization) compliant. Document/encoded isn't used, which leaves just RPC/literal and document/literal. Both of these are WS-I compliant, but RPC/literal may be deprecated in the future. Document/literal is, in any case, the preferred way of doing WSDL.

I have changed our SOAP service so the createUser operation is registered as document/literal, and I think I'll commit this, but I'm a bit wary of document/literal, frankly. It has a massive (in my opinion) disadvantage, in that the operation name is lost. This means that the only way to tell which function should be called using document/literal is to check the number and type of parameters it takes. And that means that if you have two functions that take the same number and type of parameters, it's impossible to tell which function to dispatch to.

Having said that, it "fixes" the bug, although it requires even MORE ActionScript to be generated than RPC/encoded did, and that was already 4 files plus 2 for each operation. Document/literal will require 4 + 4/operation apparently, because in addition to the XResultEvent and the X_request.as files, we now also get a XRequestType and XResponseType, which are classes that "wrap" the request and response. This results in a somewhat different and, in my opinion, less intuitive code style. But supposedly it is the better way, so I guess we'll deal with it.

In other news, Professor Danneels apparently *is* reading my blog, despite the mostly technical content. He just e-mailed us saying it seems like a good idea to ask users for their real name and an e-mail address, which is a good thing because I've already (re-)implemented some stuff under that assumption.

Posted in IQP. Leave a Comment »

Flex Builder epic fail (?)

September 15, 2009 — Richard

After doing the database schema yesterday, I tweaked it a bit today to fix a few things and then set about adjusting the user creation functionality I worked on last week to work with it. This involved changing the createUser PHP function, the SOAP server, and the Flex code. Then I hit a snag.

As part of that last step, I updated the SOAP service in Flex Builder so it could adjust its generated code. I noticed that something truly bizarre happened: Flex re-ordered the arguments to the createUser function in what appears to be a completely random order. It seemed odd, but I figured, whatever, it really doesn't matter what order the Flex code wants them in as long as they get sent correctly. Well it turns out that they *don't* get sent correctly. Using the order that the generated code specifies fails epically, while using the order that the SOAP server specifies them in works perfectly.

I have no idea why this occurs. Nobody else seems to either; in fact, *almost* nobody else even appears to have encountered this problem! The only references to the situation I could find on Google are here and here. It occurs to me that the problem could be in the way NuSOAP generates the WSDL file; one of the pages I linked to said they were using the NuSOAP library, the other didn't say what their web service was written in. In any case, this isn't a deal-breaker and probably isn't worth expending a lot of effort looking into, but it *is* **damn annoying**.

Something else that is also annoying is that for some reason the form alignment on the "Create User" page of the Flex app is messed up. Again, no idea why and not something worth spending a lot of time looking into (for now at least) since the form works fine. What is particularly curious is that if you switch back to the base state and then back to the newUser state, the misalignment disappears. Will start a bug tracker on SourceForge to note these things.

EDIT: Bug tracker is up and our inaugural bugs are added!

Posted in Epic fails, IQP. 2 Comments »

Rebuilding the database

September 14, 2009 — Richard

The goals for this week are to implement logins, some company functionality, and ideally some project functionality as well. Before we can do that, however, we need a solid database schema to build our functionality on. The schema from the previous version has a lot of problems:

- MyISAM, which ignores foreign key constraints, is used as the storage engine. Potentially, invalid values could have made their way into the database.
- All of the project data (minus the surveys) is stored as FLOATs; any developer worth their salt knows that you never, *ever* store currency data as FLOAT because you'll get bizarre rounding errors due to the fact that the data is stored in IEEE-754 format. For those who aren't familiar, IEEE-754 manages to hold an amazing range of numbers in a small amount of memory by using exponents to create the floating point. IEEE-754 technically allows the base to be 10 but this is very inconvenient implementation-wise, since computers work in bits; therefore, most implementations use the only other allowed option, base 2, which makes it impossible to exactly store some base 10 numbers. Therefore, an approximation is used instead and then rounding is applied.
- The field names leave something to be desired. They aren't very descriptive (what exactly is N_irr or N_lc?) but fortunately they **do** match the IDs that were used in the Flex app, which allowed me to figure out what fields in the database corresponded to which inputs on the screen.
- Lots of redundant data. A whole table of it, in fact. Every value in the Results table is calculated by the app based on the numbers from the Projects table.
- Superfluous references to various IDs; for instance, the Results table contains a *uID* column that references the Users table. It looks like this is responsible for the "calculated values on the project edit page are all zero" bug, because the getResults script looks up entries by user ID, not by project ID.

Clearly the schema needed a lot of work. I discovered that MySQL has a program called MySQL Workbench that you can use to create and edit your database schemas. It's capable of doing lots of neat stuff, like auto-generating the creation/update scripts for you and creating visual schemas in various common formats (UML, Crow's Feet, etc). I've used this superb tool to whip up a new schema that should address the above problems nicely. It could probably still use some tweaking and I'm debating a few things like whether we should ask users for their real name and e-mail address. Such information isn't strictly necessary but it could be used to do some nice things (personal greetings, anyone?) and some fairly important things (password reset emails).

Posted in IQP. Leave a Comment »

It never rains (but it pours)

September 13, 2009 — Richard

So I haven't posted anything in a few days, mostly because there hasn't been much to say. On Friday, Matt and I met for an hour or so in the ADP Lab to talk about what happened last week and what our objectives are for this week. We did a little database and UI design on the whiteboard. The main objectives this week are re-implementing companies and setting up the "main" UI. Also we plan to work on website design (having decided that having the app be the entire website isn't such a great idea) and project functionality, but these are secondary objectives.

You've read (or not) about *SoftEng: The Big Empty Space Picture*; welcome to *SoftEng II: The Wrath of the Big Empty Space*. Yep, that's right folks. Matt and I were tossing around ideas for the UI, and guess what we ended up with? Buttons up top to quickly navigate to different parts of the app and a tree/task panel on the left side for more specific actions in whatever part of the app you're currently using. John Vilk, who I worked with on the SoftEng project, happened to be in the ADP Lab at the time and said that wasn't good enough; "It needs a

ribbon!" Mind you, this is only a preliminary design, and it is subject to change, but it worked fairly well for the SoftEng project and I don't see any reason it won't work here as well.

And now for an abrupt gear change. Remember about two weeks ago, when I was complaining about my luck (or lack thereof) concerning hardware lately? Well I hadn't really done anything since then, partly because I was still undecided about exactly what I should do and partly because whatever I decided to do, I wouldn't have much time to work on it anyway. Well it looks like fate decided to solve my dilemma for me. This morning, our dishwasher decided to break. In the process, it spewed a ton of water all over the place, a lot of which leaked down into the basement.

Guess where my workbench happens to be located? Yep, that's right, **directly** underneath the kitchen.

So pretty much all of the hardware down there is now waterlogged. This includes both the stuff that's probably junk, like the unstable or broken motherboards, and stuff that's good, like a brand new DVD burner, some sticks of DDR memory, a terabyte hard disk, some video cards, and a whole Athlon XP-based system. I have no idea if any of this stuff is still any good, and I'll have to wait until next week to find out, because it's probably going to take a few days for this stuff to dry out. Thus, my plans for next weekend have been decided for me: see if any of my stuff is salvageable.

And now, I'm going to work on the Foundations homework that I've been procrastinating about since last weekend.

Posted in Home, IQP. Leave a Comment »

« Older posts Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

IT'S ALIIIIIIIIVE!

September 9, 2009 — Richard

Hmm, well I guess it would probably be more accurate to say it's un-dead, especially at this point. User creation now works without a hitch, but that's it.

My last post was only around half an hour ago, but I forgot to mention a bunch of other things that happened earlier:

- We got the day off to a solid start by fixing the "things in different places causes SVN war over build.xml" by extracting those paths out to an svn:ignore'd build.properties.
- While I decided to work on user creation/login, Matt said he'd start working on our new UI.
- I created db_init.php. As the name implies, this is the new SQL_UPDATE.php, except with a name that makes more sense.
- I whipped up our SOAP server and first SOAP action: createUser.
- I "imported" the web service into the Flex project. Finally, something Flex Builder is actually **good** at: point it at a WSDL file and it will auto-generate everything you need to use the web service for you!
- I discovered that MySQL's default storage engine (MyISAM) **ignores** foreign key constraints! Therefore our tables will use InnoDB instead.
- I updated and added some documentation to the project, including some notes on imported libraries and deploying to production.

Also I've now sorted out the problems I mentioned in the last post. A crossdomain.xml file fixed the localhost/127.0.0.1 problem and I used that distinction to test that a simple search/replace should make the project ready to deploy to production. My final commit for today cleaned up some debris that had accumulated from messing with the project structure last week.

Posted in IQP. Leave a Comment »

SOAP epic fail

September 9, 2009 — Richard

Well I've spent most of the day trying to get user creation and login fully implemented. I've worked up a database initialization script and the SOAP scripts accordingly, but I've hit a series of snags trying to do so.

The first was just me screwing up by naming the service and the PHP function different things. Then I discovered that NuSOAP returns HTTP status 500 on a SoapFault, which apparently doesn't make it to Flex, so I've hacked around this by having NuSOAP return status 200 instead. Then there was a problem with my INSERT statement, which I fixed.

Now I have two remaining problems. First, the localhost path is hard-coded into the AS files that Flex generated, so this would completely break in production. Second, accessing the Flex app from 127.0.0.1 doesn't work – you have to use localhost because that was what I used when I imported the web service and accessing something in a different domain isn't allowed for security reasons (that is, the domain the app is loaded from is "127.0.0.1" but the web service is on domain "localhost"). This would obviously also completely break a production environment since you would be accessing npdportfolio.com, not localhost.

Going to see if I can chase down these remaining problems, then calling it quits for the night.

Posted in Epic fails, IQP. Leave a Comment »

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

The rewrite commenceth

September 8, 2009 — Richard

Professor Danneels replied to our e-mail, saying we should do what we think is best and that if we want to start from scratch, that's OK as long as we retain all the original functionality.

Accordingly, I have commenced a rewrite. I'm about to make a commit that makes some important changes:

- I've created a *new_src* directory and updated the launch configuration to use the index.mxml in this directory accordingly.
- I've updated the build script so it is now possible to build the "old" and "new" versions of the app.
- The "old" source has been retained (for now) in the *src* directory. This directory will go away and *new_src* will become *src* once we have re-implemented the functionality and no longer need the old app as a reference.

This setup is, as far as I know, the most convenient way of doing the rewrite while still having easy access to the old source, which we will need for some things. I decided to begin the rewrite with the login, since that seemed a convenient starting point. I'm going to commit this "new" application and then start playing with how the main UI should look. Matt and I discussed this a little bit earlier but he's away from his computer at the moment. When he comes back later, I want to talk about getting the database schema and SOAP services going. At the very least I'd like to have user creation and login re-implemented by Friday. Ideally we would also have some company and/or project functionality working as well, but I think the more conservative login goal is more realistic.

This, admittedly, cuts against the SoftEng lesson that what you implement should be of the highest value to the customer, and login is definitely not at the top of the customer's priority list here. While I generally agree with that lesson, I'm still (somewhat) haunted by the fact that on the SoftEng project, once the login got to the top of the priority list, we had trouble actually implementing it because of other decisions we'd made. Suffice to say it was a little messy, or go see it for yourself; I posted the SoftEng blog about a week ago.

Hunger is now interrupting my thought processes. Time to go get some dinner. Will update the to-do list in an hour or two to reflect our new strategy and priorities moving forward.

Posted in IQP. Leave a Comment »

The waiting game

September 7, 2009 — Richard

Matt and I talked (or rather, typed) earlier, and he agreed with my conclusion that a user permissions system is better than the current method. We also discussed what approach we should take in terms of softening the whole "rewrite (again)" blow. Matt said and I agreed that we need to make clear that our goal isn't to change the purpose of the project or how it (generally) functions, and that the changes we're proposing aren't whimsical. To reinforce this, I suggested that we should accompany our weekly reports with something Professor Danneels can actually play with; namely, I want to create a v2.npdportfolio.com sub-domain and upload a recent, working build each week so he can assess our progress personally and not just take our word for it. Matt agreed this is a good idea, taking it a step further and suggesting we should hold on to the last *several* weeks of builds so he can even compare week-to-week progress, not just against production. I agreed that was a good idea, but there is a catch in that we can only have 10 MySQL databases, so we'll only be able to show at most the last nine builds, unless we decide that it's OK for multiple builds to use the same database (e.g. if there were no schema changes that week).

(Aside: It's really kind of a shame we're stuck with GoDaddy in terms of hosting; I've done some work with InMotion and their prices are comparable, but they don't impose limits on most resources, and if they do, they're so enormous you'll never hit them.)

So after that conversation, I e-mailed Professor Danneels as I said I would, and we are now awaiting a response. It's late and I don't expect to get one until tomorrow, so I decided to take another look at our build process. It kept bugging me that for some reason we couldn't use the Flex Ant tasks. It turned out that I was being retarded when I tried to do it initially and messed up, because I was able to get it to work fine now. This is important for several reasons:

- It should fix the "your workspace path can't have a space in it" problem because we're no longer using <exec> to do a shell command.
- It appears to fix the double-build bug (again).
- It massively improves build speed in the case that the Flex app doesn't need to be compiled (i.e. no changes to the Flex code) through the use of a cache file. Build time under such circumstances decreased from 6 seconds to 1 second on my system.

I have updated the "local test setup" page on the wiki in accordance with these changes. This was a particularly good thing because I forgot to do so after eliminating the BrowserLaunch hack, which meant we still had stuff in there about needing a JDK (which is no longer true) and what to do if <javac> fails (which we no longer use in the build).

And now, back to waiting…

Posted in IQP. Leave a Comment »

To rewrite or not to rewrite

September 6, 2009 — Richard

Matt and I have been joking about the idea of scrapping the existing code base and performing a total rewrite for the last week. I've reached the point, however, where I now think this is not just a viable idea, but a good one. I skimmed through the Flex book I got off of Adobe's website and it's clear that even the existing UI isn't coded properly. We have bizarre things in the Flex code, like HBoxes and VBoxes being used to create what are effectively forms that don't quite align correctly, instead of just using a Form that would automatically do the layout.

Ignoring that, however, let's briefly summarize what's on the to-do list right now:

- Refactoring the database, which will involve some pretty heavy changes.
- SOAP-ifying client-server communication, which is effectively a total rewrite.
- Making massive changes to the UI so it doesn't feel so clunky.

Just doing those things is going to require a near total rewrite *anyway*. Little of the existing code base will be left by the time we're finished. So why bother dealing with it at all? It would be faster and cleaner to just start from scratch. Some of the code is probably salvageable, and we can reuse that in the rewrite, but it would be better, in my opinion, to move that good code to a rewrite than to try and rewrite the existing code around it.

The thought that got me seriously considering the potential for a rewrite is that the user/company distinction is very klutzy. It effectively reinvents a user permissions system in a crummier way. Think about it: "company" users can make projects and groups, whereas "user" users can edit project information. My question is whether this is something that Professor Danneels requested or if the last team just decided to do it that way. I think it's a bad system; it's unintuitive and breaks the app's flow. Say I'm in charge of implementing use of this tool at some company. I'm also going to be one of the people using the tool at that company. I first have to create a Company account and create the projects. Then I have to *log out* to create a User account, log back in as a user, and join the company before I can start editing project data. Now this is a big company, and a lot of people want to use this wonderful tool. I can't afford to waste my time servicing everyone else's project creation requests, so I just give out the password to a few people. Then one of those people gets fired for some reason. Now I need to change the password (which is currently impossible, by the way! something else for the to-do list) and let everyone know what it's been changed to, otherwise the ex-employee could log in and delete all of our projects!

Seeing the problems here?

A *much* simpler and more effective way of doing this (from the user's point of view; from a programmer's PoV, it will be a bit more complicated) is to just implement a permissions system. Everyone who wants to use the tool creates a user account. Users can create companies, at which point they gain "company administrator" permissions for that company. Other users can create their own accounts and ask to be added to the company, which company administrators can approve or deny. Company administrators can grant permissions like project creation to users who are part of their company. In this fashion, all of the problems in the above example are solved. App flow isn't broken anymore: I create my account, log in, create a company, and I can immediately begin creating and editing projects all using the same login. I need to grant some co-workers the ability to create projects? No problem! Just pop over to that section of the UI and do so. Then later when one of them gets canned, I just go back to that screen and kick them out of the company.

This idea is, at its core, focused on security; I've been so busy being concerned with the security of the project's infrastructure that I hadn't yet considered the security of the application itself. It has a lot of problems in that area at the moment, like just adding a user to a company by the user entering the company's ID (and thus gaining instant access to all project data with no approval process), being unable to change your password, and the fact that if you want multiple people in the same company to be able to create projects, they have to share the same login. It's possible that some of this UI was requested by Professor Danneels; I'm not sure, but after I talk to Matt about this and get his thoughts, I intend to e-mail and ask. If we decide to implement a permissions system, scrapping the current Flex code is really going to be the only way to do it, because we'll effectively be building a whole new application anyway.

Of course, one might ask: if you're going to do a rewrite, why use Flex to do it? You've sure been complaining about it a lot, Rich!

To which I would answer: true, but mostly I've been complaining about how Flex has been abused in creating the current app rather than complaining a lot about the language itself. Sure, it has some drawbacks, but what language doesn't? It actually does have things to recommend it; for example, it has a gazillion different UI elements available to use, which generally understand how to use the data they're bound to automatically. Something I said early in the project was that Flex was a bizarre choice because it lacked the ability to connect to a remote database. What I hadn't quite grapsed at that point is that we need to not treat this like the typical desktop application; it really is a web application that should talk to a web service. And web services are something Flex can handle easily. Combining these facts, I think Flex actually is the right way to go here; I just think we need a fresh start to take full advantage of it.

On an unrelated note, another potential issue that occurred to me is concurrency. It's possible for multiple users to edit the same project at the same time, for instance, and whoever's edit gets saved last wins without anyone knowing what happened. We should probably figure out a way to deal with this (some sort of locking, maybe?), but it's far from being a top priority. Will add it to the to-do list.

Posted in IQP. Leave a Comment »

Make war, not SOAP

September 5, 2009 — Richard

So since it's the weekend and I can't do any development because I don't feel like setting up the test environment on my home computer, I thought I'd take some time and investigate a problem with the current code base that hasn't received much attention yet. Namely, the whole client-server communication thing.

Yeah, I know, I complained about it a lot about a week ago, and pointed out a lot of stuff that is wrong with the current system. I made a few suggestions for improvement, like using PHP sessions to keep track of the logged in

user/company and having the server do hashing for validation rather than the client. These were the relatively obvious things that immediately came to mind.

Having had some time to think about it since then, the reality of the situation is that we have a bunch of ad hoc PHP scripts that do stuff. This presents several problems:

1. Ad hoc-ness is generally bad for application stability, becuase it means nothing is standardized. Something that is true in one place may not be in another, and duplication tends to be a problem. A composite data type may have x fields in one place but because of inconsistent changes it has y fields in another.
2. It's hard to detect errors. The current XML "schemas", if they can be called that, make no provision for the possibility that something might go wrong on the server. When something does, like for instance when a PHP script fails because it's making incorrect assumptions about table schemas, the failure is completely silent. The other half of this problem, of course, is that the Flex app makes no provisions for errors in communication either, but even if it did it would currently be impossible to give the user a meaningful explanation of what went wrong.

The first way of fixing this that occurred to me would be to reduce the ad-hoc-ness by documenting the schemas in use, standardizing data types, and introducing an <error> element into the schema that any call could potentially return. Then I realized that there was a standard already in widespread use that does all this: SOAP.

SOAP used to stand for Simple Object Access Protocol, but according to Wikipedia, this acronym was deprecated with version 1.2 of the standard. Upon realizing that SOAP-ifying the project would be the best way to fix the aforementioned problems, I dug into the problem and did my best to work out a plan for implementation. It's doable, but there are some snags.

PHP is not a very SOAP-friendly language. It now has a built-in SOAP library but said library has several drawbacks. For one, it can't auto-generate a WSDL document; WSDL is a nasty, confusing piece of work and I would not want to write it by hand. WSDL is important because this is how applications accessing the server will find out what functions are available, what parameters they take, and what they'll return. This implies the second drawback, which is that PHP is not a strongly-typed language, but WSDL expects data types to be specified. On top of all that, PHP's SOAP library lacks support for WSDL 2.0, which isn't really a problem but certainly isn't a benefit either.

I considered going with a different language, thinking that perhaps Perl or Python might have better support for SOAP and that there's no reason we're *required* to use PHP. These languages are also weakly typed, but I hoped they might have better libraries. This thought process was short-circuited, however, by the realization that the npdportfolio.com server is run on a Windows hosting account. GoDaddy provides only PHP, ASP, and ASP.Net on their Windows accounts. I guess we could examine the ASP options, but that's a whole new language to learn on top of Flex/ActionScript and ASP.Net is designed to make drawing webpages easier, so if we went to that we might as well just start over and re-do the whole damn project in ASP.Net.

So if we're stuck with PHP, what are our options? Well, we've already seen that the SOAP library that it comes with isn't great, but the major deficiency, the inability to generate a WSDL file, could be addressed with a number of third-party solutions developed to deal with this problem. We could also just go with the nusoap library, which is very similar to the included library but actually gives a damn about type-safety and thus is capable of generating a WSDL file automatically; drawbacks here are that this library seems to be unofficially dead development-wise, it's slower than the default implementation (the default implementation is written in C, whereas nusoap is written in PHP), and it too lacks support for WSDL 2.0. Having said that, it looks like a lot of people are still using nusoap and the various official forums for the library are very active. I'm therefore leaning towards using nusoap in lieu of the default implementation.

SOAP-ifying the project will have numerous benefits, some of which I've already

mentioned, like built-in facilities for error handling. It will have one other major benefit, however: it will provide an API that anyone can use to interact with the site. SOAP and WSDL are after all designed to facilitate the development and use of *web services*.

And now, on an unrelated note, some other thoughts and things I've done since yesterday:

- Expressing horror at the idea of VARCHAR fields as primary keys was a knee-jerk reaction, because it's generally a bad idea, but the way they're actually being used at the moment makes sense. Using an artificial IDENTITY field instead would actually be *bad* database design in several places, like the user and company tables, where we would only want to allow one record with a given username in the table. In other words, the username itself would already be the primary key, so adding an IDENTITY to serve as the primary key would result in denormalization.
- After discovering that we really have no choice but to use PHP, I went on to the production server and ran phpinfo() on it to find out exactly what we can work with. Promisingly, PHP 5.2.5 is currently installed, which is fairly recent, and all of the default packages seem to be present. I saved this to a static HTML file and added it to the Documents section of the project on SourceForge.
- The code as it currently exists is in such bad shape that I think we're going to have to make an important call this week. My attitude when dealing with team projects is that if you make changes, you shouldn't commit them until you've tested them extensively and verified that your changes don't break anything. This avoids situations where team dynamics are adversely affected because developer A gets pissed at developer B because B broke the build and nobody could do any work until it got fixed. It also helps to ensure that you aren't taking steps backward, because anything that gets broken also gets immediately fixed, so at the very least, you're treading water and not sinking.

  With this project, however, I think that for the next couple of weeks, sticking to the idea of not breaking things is going to be more detrimental to progress than helpful. For instance, fixing the issue of silent failures will require changes to both the backend (PHP) and frontend (Flex); if I fix the frontend first, I'll effectively be writing code that will just get tossed shortly afterwards when we rewrite the backend and I'm forced to change the frontend again. If I fix the backend first, then the whole project is broken until I can fix the frontend accordingly. If I try to do both at once, that's going to take forever, Matt will probably commit something in the meantime, and merge conflicts will almost certainly result. Not to mention that reworking the database schema has to be in there somewhere as well. So I think the policy for the next several weeks is going to have to be relaxed from "don't break functionality" to a more lenient "don't break the build" as we work to fix things, starting from the backend and moving forward (DB -> PHP -> Flex).

And finally, the title of the post is a reference to the Zero Punctuation review of *Assassin's Creed*.

Posted in IQP. Leave a Comment »

Weekend plans

September 4, 2009 — Richard

This weekend, being a long weekend courtesy of Labor Day (and WPI's decision to take the day off), I've decided that my goal will be to read the book I just found on Adobe's website. It's not an excerpt, it's a whole 130-page long book, published by O'Reilly, who usually have good books, so I'm hopeful that this will be informative.

I've also just e-mailed our first weekly report to Professor Danneels basically explaining that users won't care about anything we did this week, and will probably not care about anything we do for the next several weeks either. This is because right now we're working just to get the project organized enough to be able to do things users *will* care about, and so many parts of it have such bad structure that untangling all of it is going to take some time. Doing so provides benefits for the developers, but not for the users. All of the work I did this week

resulted in an application that still looks and acts *exactly* the same way it does on the production site. Internally some things have improved but this is invisible to the user.

Something we learned in SoftEng is that when giving a progress report you should show how things have been improved from the user's standpoint, because generally that is all the customer cares about. I was, therefore, somewhat reluctant to send a weekly report basically saying the above, because it cuts against that lesson, so I included the fact that I think users will eventually see *some* benefits from the restructuring. Some of the bugs in the application are almost certainly the result of its poor structure, so fixing the structure will fix the bugs. Unfortunately, I have no concrete examples I can point to as yet.

On a final, unrelated note, I discovered today that one of the people who worked on this last year is in my business ethics class, so I guess if I want to I could now bring my phone with me to class and take a picture of his face to pin on the wall behind my computer and throw darts at it.

Posted in IQP. Leave a Comment »

RefactorFest '09

September 3, 2009 — Richard

RefactorFest '09 began today as I started separating index.mxml into more manageable chunks. Here's the commit I just made:

> *This commit begins the process of cleaning up the garbage heap that is index.mxml:*
> *- The "base state" is now pretty much a blank slate. The login canvas is now displayed in the "Login" state, which the app now initializes to and returns to on logout.*
> *- The login canvas has been separated into its own login.mxml file. Anything that was in index_scripts.as that was specific to the login state has been moved to login.mxml in a <mx:Script> block, which will probably be moved to its own ActionScript file later.*
> *- Anything that needed to be shared was moved to common.as, but this is of little help. I discovered that variables defined there will not be application-wide, even if the file is included in the component mxmls. Therefore the only thing this is useful for are all the imports, which should really be defined on a per-component basis. This file is a massive hack and will go away at some point.*
> *- Because of the aforementioned variable problem, I have changed (read: corrected) the way that logins are processed. Now clicking the login button, if the login is successful, results in an event dispatch to the main part of the app, which now has special functions for handling logins (i.e. note the user name, login type, and change state).*
> *- Some dead code and other assorted junk has been removed.*
> *- Because Flex Profiler is such a pest and immediately takes over the console even if the build fails, preventing us from finding out WHY it failed, I have created an ant.out file that the output gets redirected to. This is an annoying workaround, but at the moment I don't have a better solution.*

So after two days of work, I finally managed to take ONE state out of index.mxml. That leaves three to go, along with cleaning things up afterward. A lot of hacks are going to get inserted in this process because of the way that data is currently handled, which is to say that it's not really. Basically everything was just a global variable. This is bad programming form, especially because it makes it harder to see what things any given part of the app is interacting with. With things being separated out, this now effectively becomes impossible to maintain, so we'll be moving to an event-driven mechanism, at least for now.

This work is long, arduous, and tedious, but it is making an improvement and having other beneficial consequences. Index.mxml was 1553 lines in length a week ago; now it's been cut down to 1040 lines. A bunch of dead code and other assorted programming debris has been removed because mxmlc complained about it. I've also gained some experience dealing with Flex.

There are going to be a lot more hacks inserted before this is over though. Truly fixing things is going to require revamping the PHP scripts, which I would prefer not to do until we've refactored the database. Those are going to be our main goals going into next week.

Posted in IQP. Leave a Comment »

Flex sucks

September 2, 2009 — Richard

After around eight hours of working on the project today (or rather, *trying* to work on it) I've concluded that I officially hate Flex and that it is the most obtuse "language" I have ever had the misfortune of encountering. Allow me to explain.

As I said in an earlier post, my first goal is to break the giant index.mxml file up into smaller chunks. Ideally I was looking for a structure that would go something like this, at least for now:

1. The big block of script goes in its own file.
2. The HTTPService stuff gets its own file.
3. The base state of the app becomes a blank slate and the app goes to a "Login" state instead of the base state when it initializes…
4. …thus allowing me to pull all of the states/canvases out into their own separate files.

I figured out how to do #1 about an hour into today's work, but I was kind of working on all the above goals simultaneously, so I can't legitimately say that finishing that goal alone took an hour. Figuring out #3 took a bit more doing but I managed to work it out.

Then I started on #4, beginning with the Login state since it was convenient. This was where the Flex compiler started going bonkers. First it complained about missing imports in the new login.mxml file. I don't know why, because the things it complained about are part of the standard mx namespace, but whatever, I made a <mx:Script> block and started hacking away. Then it complained about calls to undefined functions, so I cut-pasted those from the index_scripts.as file I had extracted the ActionScript bits into. Then it complained about undefined properties and variables and I started to see why the last guys just packed everything into one giant source file: because separating it is a bitch.

I had thought I would be able to treat MXML code similar to how I would treat Java or C code. I could extract things to their own separate files and just reference one file from another when appropriate. The problem is that MXML "code" is not really code and Flex is not really a programming language (it's not Turing complete); Flex is just a prettier layer on top of standard ActionScript. That's what the Flex compiler does, in fact: it translates MXML to ActionScript and *then* compiles it.

This unfortunately makes separating things unnecessarily difficult and annoying. For instance, if I want to have an HTTPService call that is accessible from more than just the file it's in, I'll have to first rewrite it in ActionScript and then reference the ActionScript file where I need it. I have no words to describe how unusual, unintuitive, and outright annoying this is.

On the plus side, it is helping me eliminate dead and redundant code. I've already cleared out several unused variables and functions after mxmlc complained about them.

Right now, I have something that at least builds but unfortunately doesn't actually work. Clicking the login button is having no effect right now and I'm not sure why yet. Will probably figure this out sometime tomorrow, as I'm planning to quit soon for the night.

Posted in IQP, rants. Leave a Comment »

The OTHER other shoe drops

September 2, 2009 — Richard

I have no classes on Wednesdays this term, meaning I have all day to play with

the project.

Lucky me.

I've been trying to understand how exactly the Flex part of the application works. Looking over some Flex tutorials has been helpful, but unfortunately most of what I read there doesn't really apply to the code we've got. For instance, I've looked at information on HTTPService stuff. Pretty much all of them include a resultHandler and, in many cases, a faultHandler. This is all very nice and neat. Our existing Flex code uses neither.

I'm thinking this may be the source of some of those bizarre bugs. The calls are made asynchronously, so if the server takes a while to do whatever it's supposed to, the data won't be available. All of the fields and datagrids and whatnot are mostly bound to the lastResult method/field/whatever. It would seem more natural and logical to me that you would call the service and it would put the data in a variable when it completes and you would then bind to that variable, and I'd like to change to that method. Of course, doing so will be easier said than done, I'm sure. Flex Builder is pretty much no help at all here. When dealing with Java, for instance, I can click on a variable and do Ctrl-Shift-G and get a list of every use of that variable in the entire project. This doesn't work with Flex Builder (well, it works, but not as well) becuase there's a lot of embedded scripting and text, which Eclipse doesn't know to look at.

I've also been looking at the issue of having all the Flex code in one giant source file. This is annoying on many levels; it makes it hard to navigate, but more importantly, it makes it hard to collaborate. SVN can reconcile edits in different locations in the same file up to a point. Then you start getting merge conflicts, which I really hate dealing with. Last night, Matt said he wanted to tackle fixing the database structure and I told him to have fun. He'll need to mess with the PHP scripts to do that, so I've decided that my first goal is going to be separating the Flex mess into multiple source files. Right now, I'm going to break for lunch, after which I'll bust out my hatchet.

Posted in IQP. Leave a Comment »

« Older posts

Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

Why source control is important

September 1, 2009 — Richard

So Matt and I were trying to get moving on this project. We hit a teeny tiny snag about two hours ago, when we couldn't get the pretty orange bubbles to appear on our charts no matter what we did. At first we suspected that we had old Flex code but then I discovered that the SQL_UPDATE.php script was old and was generating a database schema that was inconsistent with what the PHP scripts were expecting. This was only true for two of the six tables though, so it wasn't immediately evident. After discovering the problem, I went to the production site and had phpMyAdmin generate the correct schema, which I then tested and committed.

This is (part of) why source control is important – so that it's always VERY clear whether you have the latest code or not! And so other people don't have to go chasing around to eight thousand different sources of code to get ALL of the latest source code.

We also just discovered that Matt has the debug version of Flash Player and I don't (currently, at least), so he is getting annoyed with a ton of error messages. Apparently the last guys didn't have the debug version installed either. One more thing for the to-do list...

Posted in IQP, rants. Leave a Comment »

Coding Horror

August 31, 2009 — Richard

Well after all that wonderful success, you knew the other shoe had to drop. Matt followed the setup procedure I laid out and, after a few hiccups which have been documented in the wiki, he got up and running.

Then we started tinkering with the app and poking a bit more at the source code. Matt noted that the database design is abysmal and will require some significant reworking. Seriously, VARCHARs for primary keys?!? No uniqueness constraints for the tables connecting users to companies to groups. Then we noted a bunch of HCI nightmares, which are obvious even to the two of us, neither of whom have taken the course yet. Things like missing field descriptions, either duplicate graphs or (more likely) mislabled axes, massively incorrect spelling and grammar, the list goes on.

I'm much less hesitant to break things now, because they can't get much worse. Well, I suppose the app could be written in MUMPS. That would be a lot worse. Since I'm the security nut, I think I'll probably work on the PHP situation first. I got the distinct impression that Matt would like to take a hatchet to the database, and as far as I'm concerned, he can have at it. I get the feeling that these two things alone will solve a great many problems, after which we'll need to tackle the HCI problems in the Flex app. Doing so is probably going to require revamping the XML schemas being used to communicate between Flex and PHP, so I'll have a look at that while I'm shoring up security. Documentation should end up getting worked on through all of that.

And then, we will have an app that is actually, ya know, *stable*. And *working*.

Posted in IQP, rants. Leave a Comment »

More success!

August 31, 2009 — Richard

Excellent news! We now have a fully integrated and automated development and testing environment for the project! After much wrangling with Eclipse, I

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

finally got it set up to do what I wanted: build the Flex app and then sync the PHP scripts and HTML bits into the htdocs directory. I have updated the page I created on the SourceForge wiki about setting up the environment accordingly. Those instructions should probably be tested at some point but I don't see any reason why they shouldn't work.

Now *real* development can begin. I will be updating the to-do list shortly to reflect these recent developments. On a side note, I have noticed a "bug" of sorts with the build script. For some reason, it runs twice the first time you run the project after opening Eclipse. Not exactly sure why but it's a minor problem. Will add it to the to-do list.

Oh, and the infamous SoftEng blog is now available. You can find it here.

Posted in IQP. Leave a Comment »

Success!

August 31, 2009 — Richard

Quick post before I go to sleep:

We now have an Ant build file that can compile the Flex app and sync the PHP scripts all in one go. I didn't realize, however, that the html-template thing Flex creates has to be filled in, so I'll fix that tomorrow, using the method described here. After that all I should need to do is create the Java app that will launch the browser, clean up the build and launch configurations, and I should be ready to commit this. Then we can get started on real work.

Posted in IQP. Leave a Comment »

Ant Redux, Part II

August 30, 2009 — Richard

So my research on Friday night told me how I can (hopefully) compile and copy everything in one fell swoop, but the question of how to open the page in a browser was left unanswered. After a little more research today, I've discovered several possible ways of doing this.

1. Use the <exec> task to run, say, iexplore.exe with the appropriate argument to open the page. This has the benefit of being simple, but not very portable. Also people may want the page to appear in their default browser (like me), which is not IE, in which case we'd need a different path for every browser on every platform. Not very attractive.
2. Use the <script> task to run a Python script to open the browser. Said script is literally two lines long, which is very simple, and would open the default browser. The downside is that this would require importing the Jython library, which doesn't appear to be very current with the C version of Python. I don't know when the webbrowser module was added, so I have no idea if this would even work. Of course, I suppose I could install the C version of Python and use <exec> to run the script from there but that's making things much more complicated than necessary and it eliminates the cross-platform benefits.
3. Use the <javac> and <java> tasks to compile and run a Java program to open the browser. Support for doing this was added in the Desktop class, which is new to Java 6, so a relatively new JVM would be required for this option. That's not a huge problem though, and I'm more confident that this would work than #2, so I'm planning to explore this route first.

Another thing that occurred to me is that it would be really awesome if Ant could automatically start Xampp before running the browser. There is no cross-platform way of doing this; any "cross-platform" method would involve detecting which platform is present and running commands appropriately. This isn't really a big deal though, so I'm not sure if I'll toss this in or not. Right now I'm leaning against including it, but we'll see.

Of course, with my luck lately, this whole enterprise will be derailed by something I didn't anticipate…

Posted in IQP. Leave a Comment »

Ant Redux

August 28, 2009 — Richard

I know, I said I wasn't going to work on this over the weekend, and I'm not. Really. Not gonna write a single line of code.

...Yeah, I think I don't believe me either. Anyhow, the problem of getting the build process completely integrated was bugging me, so I spent the last hour or so navigating through the series of tubes to figure out how to fix it. It was looking like I might have to go off the deep end and write an Eclipse plugin or something, but then this page hit me over the head with the blatantly obvious: Flex provides a command-line compiler, so use it dummy! That's all the Flex <mxmlc> Ant task is doing! This doesn't eliminate the need to reference the Flex SDK or the potential issues with external resources in different locations, but this can be used to confine them to the Ant file; just specify the appropriate directories as properties and then people can change them as necessary for their setup.

The main point is that I now know (I think) how to make this work. The Ant build can handle everything: building the Flex app, copying it and the PHP to the htdocs directory, then launching the web browser. Of course, in order to make THAT work, I need Eclipse to use the Ant build by default, as this page kindly explains how to accomplish. The only question I still have is what the situation with launch configurations are in such a setup. Right now I think (can't check since I don't have the files here) that the Flex builder and launch configuration are being used. I know I can change the builder but I'm not sure how or whether that meshes in with the launch configuration. Something to experiment with Sunday night or Monday afternoon.

The main reason this is bothering me is because we need this to happen. Getting everything integrated into a one-click build-copy-run will speed up development immensely; otherwise we're stuck with either having to do multiple "builds" (at least two, one real build for the Flex app and one fake "build" to copy the PHP scripts) or having to handle copying the PHP scripts manually. Neither of those is particularly efficient; it makes me wonder how the last team dealt with the problem. It seems like they probably worked everything separate and manual, although it's hard to say for sure. They don't mention having a project on SourceForge or any sort of source control (e.g. SVN) in their IQP report, nor did they describe their testing environment (probably just FTP'd it to the web server; again, not efficient, hence the early emphasis on getting a working testing environment set up locally), but given the omissions and incorrect statements in that report that I ranted about a few posts back, that doesn't mean any of my hunches are correct.

For anyone reading this who really wants to see the SoftEng blog, you'll have to wait a few more days. I will post it, but WordPress limits what sorts of files I can upload and I had originally saved it as an XPS (Microsoft's version of PDF), which they won't allow. I tried to install a PDF printer but it looks like I accidentally removed support for that on this computer when I nlited the installation disc. Whoops!

Posted in IQP. Leave a Comment »

Weekend

August 28, 2009 — Richard

For various personal reasons, I drive home to Berkley every weekend. On past programming projects (read: SoftEng), this was not a problem because I could continue working at home. With the amount of effort required to get a test setup working for the IQP, however, this is not something I'm planning to do. I expect that it will turn out that this is for the best, though. Something you learn in SoftEng is that you should work at a sustainable pace. When I worked on the SoftEng project, I didn't do that, partially because I found the project interesting but more because I was afraid that if I didn't stay on top of the project, it would disintegrate; for those who haven't heard about it, I was basically one of two guys in charge of a 16-person project (the other guy being Mike Diamant, who is now an alum), and in the final analysis, the two of us alone accounted for about 50% of all commits (some 937 of them!). For people who really want to know

how that project went down, I'm planning to attach the massive blog page I had on the project's SourceForge wiki to another post I'll make at some point over the weekend.

Okay, this post is rambling. Short version: I'm taking weekends off. Should help productivity during the rest of the week though.

On a final, random note, it occurred to me that it would be nice to make some unit tests for the PHP scripts. Going to go make a "todo list" page now.

Posted in IQP. Leave a Comment »

Ant epic fail

August 27, 2009 — Richard

Okay, NOW I'm done for the evening. For the past two hours or so I've been struggling to construct an Ant build file that would allow one-click build and run of the whole project. First I tried ant4eclipse, but it doesn't know how to handle a Flex run configuration. Then I figured I'd try to use the Flex SDK's Ant tasks and then do a shell command to execute the browser, but for some reason Ant isn't finding the tasks file. Will pick up with this another day.

Posted in IQP. Leave a Comment »

Development setup

August 27, 2009 — Richard

Well, after several hours of work, I now have an Eclipse installation fully kitted out with SVN adapter (Subclipse), PHP dev tools (Eclipse PDT), and the Flex plug-in. This required more work than it sounds because the Flex plug-in doesn't like Eclipse Galileo (3.5), so I had to downgrade to Ganymede (3.4). This in turn created a problem with installing PDT, but I managed to sort it all out. I also have XamppLite set up on my system and I can now get the latest Flex code to compile and appear in my browser with one click. I've also:

- Created a project and source repository on SourceForge. The "version 1" (or rather, version 3) code has been imported, with the Flex and PHP code in the same Eclipse project.
- Taken the liberty of getting rid of the AS3Crypto source import and replacing it with the pre-compiled version.
- Created a page in the project's wiki on SourceForge explaining the steps I took to get my setup working.

Still on the to-do list before serious development can begin:

- Figure out how to make Eclipse/SVN ignore the project properties file. This eliminates problems with people having different external resources (like the AMP stack) in different places; if we don't do this and install in different directories, every time we commit to SVN, the properties file will be updated and break the other person's setup.
- Figure out how to "link" the PHP sources in the Eclipse project to the PHP scripts in the htdocs directory. It's possible to create links in Eclipse, but then it won't commit the linked files to SVN. Probably this is going to involve some sort of script or Ant task that will cause the htdocs/php directory to be deleted and copy the PHP directory over it every time we run the app.

So it's not perfect yet, but we've made a huge step in the right direction. I may do some more work on this tonight or, having spent at least 5 hours today on it already, call it quits for the evening.

Posted in IQP. Leave a Comment »

FIRST!, Part II

August 27, 2009 — Richard

So we had our meeting at noon today. The good news:

- Nobody is using the site right now, so we can break things to our heart's content.

- Professor Danneels isn't mandating any weekly meetings, just a weekly status report to let him know what's going on.

The bad news:

- I've downloaded and installed Flex Builder 3, which effectively goes like this:
  Eclipse **-** all the good stuff **+** stupid garbage **=** Flex Builder 3
  This does not make me particularly thrilled. Especially since Adobe charges $250/license for this junk!
- This is actually version FOUR of this project, the first two iterations apparently having been scrapped because they were just that bad.
- My previous rant can now be extended further, as I've just discovered that the IQP report from the last guys contains some pure fiction. IQP report says:

  The website employs MD5 password encryption, a widely used password encryption method.

  Ignore the fact MD5 is a *hashing* algorithm, not an *encryption* algorithm, and take a look at what the actual code says:

  // 1: get an algorithm..
  var name:String = "aes"+"-"+"ecb";

  Oh yeah, AES-ECB is *exactly* the same as MD5... NOT! The key, for those interested in such errata, is "duck".

So, moving off of the rant, my first tasks, having now installed ~~Eclipse~~ Flex Builder and imported the Flex end of the project is to get a project set up on SourceForge, after which I'm going to start tackling the first item on our agenda, which is to revamp the PHP/SQL end of the project. I have a feeling that doing so may reveal the cause of and/or fix a variety of bugs that Professor Danneels and others have noted in the current version.

Posted in IQP, rants. Leave a Comment »

« Older posts Newer posts »

CaptainRichard's Blog

Pages

- Home
- About
- To-do List

Categories

- Epic fails
- Home
- IQP
- Modern Warfare 2
- rants

FIRST!

August 26, 2009 — Richard

FIRST!

...oh, right, nobody else can be first because I'm the only one who can edit this. To anyone who is bored enough to read my programming notes and other rants, welcome! This blog is going to be serving primarily as my programming journal for the NPD IQP project during A, B, and C terms. It will probably also include other projects I feel like keeping track of and non-programming rants if I get pissed off about something.

So let's get this off to a good start with a rant about the code I'm inheriting from the previous team that wrote the initial NPD code. It's a semi-messy Flex/PHP/MySQL mashup.

Flex basically just provides the GUI and some calculations. I haven't dug too far into this code yet, but my initial observations are:

1. Comments? What comments? We don' need no steenkin' comments! Well, the HTTPService stuff is commented but that's about it. And it looks like they used the copy/paste method of programming too:

   <!–
   @name: getProjects
   @function: to get a list of a user's project names
   @return xml list of project ids
   – >

   And then a few lines later...

   <!–
   @name: deleteProject
   @function: to get a list of a user's project names
   @return xml list of project ids
   – >

   So even the comments can't be relied upon to be correct. Wonderful.

2. My cursory inspection revealed a "deleteUser" HTTPService that points to a PHP script that doesn't exist on the web server. This service is referenced exactly once in the entire code base. I guess nobody ever tested to see if that button worked...

The PHP is sandwiched in there because Flex doesn't have any native method for talking to a remote database. The choice of Flex for the front-end, consequently, is mystifying to me, but it's what we've got so I guess we're stuck with it ("we" being me and Matt, the whole team for "version 2" of this project). PHP I'm a bit more familiar with than Flex, and I've poked around in the scripts a bit more. Observations:

1. Security? What security? You mean the honor system isn't good enough? Yep, apparently the previous coders just assumed that nobody would be able to figure out where their PHP scripts for interacting with the database are or at least wouldn't be able to figure out how to use them. Right now, if someone really wanted to wipe out the entire database, it'd be piss easy. This comment seems to be in every file:

   / /Query the database to see if the given username/password combination is valid.

Search

Blogroll

- WordPress.com
- WordPress.org

Archives

- May 2010
- April 2010
- March 2010
- February 2010
- January 2010
- December 2009
- November 2009
- October 2009
- September 2009
- August 2009

Misc

- Register
- Log in
- WordPress.org
- WordPress.com

The code that follows does nothing of the sort. Of course, such queries would be unnecessary anyway if one were to use Sessions to keep track of the logged in user. So I guess they saved me the trouble of having to strip that out.

2. As a tangent to the above security rant, if you're constructing an SQL string, you REALLY ought to escape ALL the inputs. The scripts that I've examined only escape inputs that the user can directly manipulate in the application, completely ignoring the possibility that someone might just post to the PHP scripts directly, in which case ANY input could potentially be a problem. This should be easily fixed by converting to parameterized stored procedures instead, which would seem to be more optimal anyway because then you're not reconstructing the same strings every time you run a script.
3. Oh yeah, and there's an SQL_UPDATE.php that anyone could execute from the comfort of their browser that re-creates the whole database schema, dropping the existing tables as a first step in doing so.

SQL databases don't like me and I don't like them, but for this project (and undoubtedly other ones in the future) we'll have to put our differences aside. The database looks well-structured, but the IQP report from the previous team leaves something to be desired in terms of explaining what all the fields are. What exactly is `mresc'? Marketing RESourCes? Looks like we're gonna have fun finding out.

Oh, and it looks like there's also a bunch of random junk on the server, but I'll have to do some testing to know for sure. We're having our first meeting at noon tomorrow, after which I plan to set up a testing environment on the server and get started. I've already read Professor Danneels' paper that explains most of what's going on concept-wise, and I'll probably read the other two he sent us at some point in the next few days as well.

Whew! Felt good to get that rant out. Until next time...

Posted in IQP, rants. Tags: IQP, rants. Leave a Comment »

Newer posts »