 Objective

To work at a company that requires skilled engineers to help realize its vision; hopefully giving me an opportunity to solve interesting problems and learn something new along the way.

 Education

**Neumont University -** [**https://neumont.edu**](https://neumont.edu)Bachelor of Science in Computer Science, 2009

 Skills

**Web**ASP.Net (MVC5, WebAPI, Razor) - HTML & CSS & JS & AJAX - Knockout - Angular (v1) - ODATA - RESTful APIs - XML / JSON / Content negotiation - Telerik/Kendo - JQuery - limited NodeJS experience (and npm & bower) - WebForms

**Middle Tier**.NET Framework 4.5.1 - ADO.Net - Entity Framework - NHibernate - LINQ - IOC containers (Autofac, Unity, MEF) - MSFT Enterprise Library - Logging frameworks (Cloud-based, Windows EventLog, custom) - FluentValidation - AutoMapper - TPL Dataflow / Task-based programming (async/await)

**Databases**MS SQL - DbUp - limited experience with PostgreSQL, mySQL, MariaDb, DynamoDb, Redis, MongoDB

**Infrastructure**Bamboo - Octopus Deploy - Cruise Control - NAnt - Continuous Integration (CI) - Continuous Deployment (CD) - limited experience with other OSes (OpenSUSE, CentOS, etc.) - Powershell - Ansible

**Theory**Metaprogramming (C# mostly) - Design patterns - Agile methodologies - Localization / Internationalization

**Tools / Other**Subversion - Git - JIRA - Confluence - Crucible - TFS - Sharepoint - ReSharper - Visual Studio - limited Java / Python / Ruby skills - Section 508 & ARIA accessibility - ISO / CMMI experience - Slack - NUnit / xUnit / MS Unit / Moq - Nuget - Redgate tools - Facebook / Twitter / Foursquare APIs - LeanSentry - Rollbar - Raygun

References

Available upon request.

 Work Experience

**August 2014 - Present**

Fishbowl, Inc - <http://www.fishbowl.com/>

**Software Engineer**

My work at Fishbowl was often variegated, changing depending on the needs of clients and the availability of resources that could develop the software necessary to meet those needs. Given my versatility as a full stack developer, my specific responsibilities were generally in flux from one project to the next. I’ll attempt to illustrate some significant accomplishments among those projects.

Continuous Integration / Continuous Deployment overhaul

For many years Fishbowl had been using CruiseControl and NAnt in order to deploy its products. There were a few problems with how it was set up: It was slow, it had lots of opaque behaviors, it did not allow concurrent builds or deploys, so on and so forth. After making a small prototype, I gradually overhauled most of our products into the new setup. Deployments went from an error-prone multi-hour affair to a painless process that could be done in minutes. We also have some Java projects, so I am comfy with Java & Maven in Bamboo (though Octopus is specifically for Windows). In tandem with this, I also turned many of our dependencies that were once filepath-based into internally-hosted nuget packages (with respective docs in our Sharepoint).

**Relevant tech details:** Bamboo, Octopus Deploy, Powershell, shell, Maven, Unit test runners / parsers, Nuget

Single Sign On (SSO) consolidation

Fishbowl has had a sign on mechanism since its inception. However, it eventually acquired another company (particularly for its mailing engine), and thus ended up with two disparate login systems. This portion of technical debt has been around for several years but eventually needed to be tackled in order to introduce some usability enhancements (a fancy sidebar for navigation across products).

***Relevant tech details:*** Active Directory (AD), Security Assertion Markup Language (SAML), Data migration & facade work, unified API for navigation menu.

API-Based UI re-design

When one of our products sorely required a UI re-design, we had to migrate it from ASP.Net WebForms to MVC5 (as the new UI framework was all in Razor). We thought an API-first approach would help us test the code better, and also give us an easy path for exposing the relevant data to other products that may later require it. In the process the code was made more modular, more testable, and more extensible than it was before. I also built a few interesting IQueryable helpers to enable full text search.

**Relevant tech details:** MVC5, WebApi 2, Autofac, xUnit, Kendo MVC, Moq

Segmentation UI widget & API

For several of our products, we needed an HTML widget that could access our new “Segmentation” API which can segment (i.e. facet) user data in several ways. The widget itself was interesting because it was pure HTML/JS/CSS (with a small dependency on JQuery and a few other libraries), which we could then encapsulate in an easy MVC helper, which was then encapsulated in a nuget package. Since the API could potentially induce expensive operations, the full stack (JS to API to DB) was all async with cancellation support. While the JS provides “real-time” data, the underlying model was mostly intended for “easy” segments that could be calculated ahead of time.

**Relevant tech details:** Angular (v1), Jasmine, Protractor, Gulp, Bower, NodeJS / npm, MVC, WebApi

Social Media Publishing

Many clients wanted to be able to publish the content of their mailings to their social media accounts when the mailings go out. We created a UI with which they can create social posts from their mailings and modify the content as desired. The feature itself sounds trivial, but since social posts could be edited independently of mailings and we actually have two mailing systems (one built on top of the other with added workflow and business rules), it became slightly more complex as there were many workflows and mixed states in which the mailings and posts could exist.

Assorted other work

There’s some other work that doesn’t quite stand on its own but is worth mentioning: Understanding our mission-critical software (mailing system), database performance tuning, mixed-stack work: Java & Python, Code reviews, architectural planning, improving our documentation (by an order of magnitude).

**July 2009 - August 2014**

REI Systems - <https://www.reisystems.com>

**Sr. Software Engineer**

Much of my time at REI was spent on projects within the Health Resources and Services Administration (HRSA) program. Chief among the HRSA products is the Enterprise Handbooks (EHB) which is the nickname given to the web site in which much of their work is done. Most of my work involved transforming complex business rules into well-designed systems.

Audit Tracking Component

For this project, there was a business need to see snapshots of data that had to be visualized in a way that organized it into sections and fields. This was a large crosscutting concern needed to be easily applicable with any data model and solution.

**Relevant tech details:** Implemented auditing as a crosscutting concern via attribute decoration, interceptors (using Unity), and Reflection to accommodate varying data models. Since we were limited to 3.5, use of dynamic wasn’t an option.

HIV/AIDS Bureau (HAB) Formula, Initial Release & Center for Disease Control (CDC) Integration

For HAB, they calculate funding for grantees based on several data points. The purpose of this module was to read information from all those data points, display relevant information to the user, and appropriately make calculations automatically. The result saves several weeks of work that was previously done manually by the client. During this release I was responsible for the successful integration with external systems while overseeing another developer to help make those changes.

After the initial release of HAB Formula, the client wanted the ability to drop an excel file suiting a specific format onto a file server which can then be read into our system. Since this behavior could be done while formulas are in progress, it followed a two-step staging process and then would explicitly be loaded into formulas if desired.

**Relevant tech details:** Used dependency injection to allow multiple implementations for file retrieval, validation, and parsing. Internally used Aspose for reading excel sheets into data sets for parsing. Knockout for some UI binding & validation. WebForms.

HRSA EHB Funding Memo Re-Design

In addition to upgrading to our new internal technology platform, this re-design introduced several advanced new configurations for the manner in which a “Funding Memo” is created. The “Funding Memo” is essentially a description of the funding that is to be award to several potential grantees. With all of the new configurations introduced, the UI had to be flexible to accommodate any mix of configurable parameters. As one can imagine, it was also critical that all of the funding specifications calculated the desired total. I was also responsible for onboarding a few new developers throughout the project.

**Relevant tech details:** WebForms.

HRSA EHB Post Award Re-Design

This project was a substantial enhancement to the project I first worked on in the company. We upgraded the solution to the new internal technology platform and implemented several workflow-related enhancements. On this team I was responsible to effectively coordinate two more senior developers to leverage the existing code base while introducing enhancements.

**Relevant tech details:** Migrated Classic ASP (in Visual Basic) to WebForms.

HRSA EHB Awards Re-Design

The awards process in EHB (Enterprise Handbooks) is the phase in which money and terms and conditions are created to be given to the appropriate grantee. My role in this project was to create the entire “funding information” module, which allowed the responsible HRSA employee to distribute money. It was a large team of about 10 developers and occasionally required a team effort in order to integrate modules.

**Relevant tech details:** WebForms.

HRSA EHB Post Award 2.0

As an entry-level software engineer, my responsibilities were to create the internal review module for “EHB Post Award” module. Grantees receiving funds from HRSA would submit applications to meet the terms and conditions imposed upon them, which would then be reviewed by HRSA employees. This behavior also had to be implemented for another module (prior approvals) in which a similar review process was necessary, but based on a different subject matter. For this reason, the code base and data model had to be flexible enough to accommodate both modules.

**Relevant tech details:** WebForms.