Active Analytics: Adapting Web Pages Automatically Based on Analytics Data

By

William R. Carle II

A thesis submitted to the

School of Computing

in partial fulfillment of the requirements for the degree of

Master of Science in Computing and Information Sciences

UNIVERSITY OF NORTH FLORIDA

SCHOOL OF COMPUTING

The thesis “Active Analytics: Adapting Web Pages Automatically Based on Analytics Data” submitted by William R. Carle in partial fulfillment of the requirements for the degree of Master of Science in Computing and Information Sciences has been

Approved by the thesis committee: **Date**

Dr. Karthikeyan Umapathy
Thesis Advisor and Committee Chairperson

Member 2

Member 3

Accepted for the School of Computing:

Dr. Asai Asaithambi
Director of the School

Accepted for the College of Computing, Engineering, and Construction:

Dr. Mark Tumeo
Dean of the College

Accepted for the University:

Dr. John Kantner
Dean of the Graduate School

TABLE OF CONTENTS

FIGURES

TABLES

ABSTRACT

It can be difficult for web designers to adapt the site's design to fit with changing usage trends. Web analytics tools give designers a window into website usage patterns, but they must be analyzed and applied to a website's design manually. Analytics data should be taken into account when designing the user interface (UI) of a website and should be constantly revisited to evolve the design of the site as needs change. A framework for marrying live analytics data with user interface design could allow for interfaces that adapt dynamically to usage patterns, with little or no action from the designers. The goal of this research is to create a framework that utilizes web analytics data to automatically update and enhance web user interfaces.

The first problem addressed in this research is how to extract and summarize analytics data from popular web analytics tools such as Google Analytics. In this research, we present a solution for extracting this data via web services and transforming it into reporting data that will inform user interface improvements. Once the data has been extracted and summarized, we expose the summarized reports via web services for use by our client side UI framework. This client side framework will dynamically update the content and navigation on the page to reflect the data mined from the web usage reports. The resulting system will react to changing usage patterns of a website and update the user interface accordingly, reducing the time it takes users to find the content they are looking for.

To evaluate the effectiveness of the proposed framework we will perform end user testing comparing average time to complete a task. We will use the University of North Florida's website for evaluation. We will create a new version of the site that utilizes our framework, and compare the average time it took users to find specific content using the current website versus the updated version.

Chapter 1.
Introduction

Modern Web analytics tools are an incredibly valuable asset for any organization with a strong Web presence. These tools track user actions on a site and offer insight into what users want from the website and what they have trouble finding. Popular tools like Google Analytics (Google Analytics, 2014) are widely used by sites across the internet, but the value they provide fluctuates greatly depending on how well the tracking data is analyzed and acted upon. Analytics tools are a valuable source of usability and behavioral data that is too often overlooked or not used to its full potential (Phippen, Sheppard, & Furnell, 2004). For analytics data to be properly utilized an organization would need to keep a constant eye on these statistics and update the site design to reflect the changing needs of users (Prom, 2011). Such constant vigilance and development is often not feasible for many organizations. Ideally an automated system could keep track of trends discovered through analytics and adapt a site in real time, with little to no interaction from a developer.

1.1 Problem Statement

For content driven websites, relevant navigation and placement of information should be the top priority to help drive as many people as possible to informational pages (Phippen et al., 2004). Too often however, a site is designed to the specifications of content owners rather than to the needs of actual visitors to the site. With many stakeholders involved in site design and variety of contents, it is sometimes challenging for a designer to argue for a site update that removes rarely used information and pushes useful information to the forefront. This can often lead to busy and difficult to use sites that don't take into account what visitors actually need from a site. Sometimes you need hard data to

convince someone that the link to their very specific niche of a webpage isn't as important as some other navigation options. Web analytics tools gather data on usage patterns of a website. Data gathered by web analytics can help designers address the usability problems of a site and keep track of changing usage patterns, keeping a site usable as needs change. The problem with web analytics tools is that they require active users to track usage pattern data and act on it in a timely manner to keep the site user interface (UI) useful.

There has been extensive research in field of web usability but further work needs to be done to marry good web interface design with analytic data that tells designers what visitors to a site really want to see. Good practice in web usability should be paired with analytic data to ensure that a site is not only easy to use but also surfaces the information that visitors are actually interested in. This is not a one-time process, rather a repeated process as a visitor's needs will change over time. The majority of visitors may need to find information on certain topics during certain times of year, and a static site design cannot react to the changing needs of users. Unless site owners are constantly watching these trends and adapting site design to fit these needs, a site will quickly become less usable (Prom, 2011)

As a case study, we will look at the official website of the University of North Florida. There are nearly 70 different links on the homepage alone and the relevance of these links changes over time. We will evaluate the current design of the homepage and various other high traffic pages on the site including the library homepage, and use these pages to test the effectiveness of our system. The university content owners have neither the manpower nor the web design expertise needed to keep up with these changing trends year round. The university needs to reevaluate the design of its site with analytics data in mind and needs a way to adapt the site over time as visitors' needs change. For example: most

student users would not use a course registration link during middle semester but would likely use that link during the registration windows at the end of a semester, semester breaks, or the beginning of a semester. Trends like this need to be acted upon in a timely manner and ideally without involving actions from a web designer. By automatically detecting these trends, and taking action such as moving the registration link to the top of a list of menu items, or subtly drawing attention to it through styling, users will be able to find their intended destination quicker.

1.2 Contributions

The goal of this research is to develop a generic automated system that will monitor the vast amount of data gathered from web analytics and adapt web pages in real time to reflect usage trends. This thesis seeks to create a system for automatically processing tracking data from services like Google Analytics and transforming that data into adaptations of a site's user interface. By modifying the user interface dynamically according to usage patterns we will improve the usability of the site and surface information that is important and relevant to the current visitors.

There is plenty of research into how to apply analytics data to improve a site's usability, but most of the proposed solutions require human analysis and manual action (Prom, 2011). While there will always be a need for designers to work on improving the usability of a site, we believe some of this burden can be offloaded to an automated system. For example, if a designer sees that that a certain page on the website is experiencing consistently high traffic they would need to adapt the navigation on the site to make links to that page more prevalent. We believe we can automate these and other similar tasks with

our framework. We plan to implement a functioning, real-world system to adapt interfaces to the changing needs of users.

1.3 Plan

As a case study for this research, we will use the University of North Florida main website (UNF, 2014). The university has a single unified content management system that covers most of the web presence for the entire university. The sheer size in terms of content and navigation items on this site make it a perfect candidate for the automation of user interface improvements. Since 2009 UNF has been gathering tracking data using Google Analytics totaling to over 9.6 million unique visitors and over 94 million page views. We plan to use this wealth of data to develop an automated way of analyzing visitor trends and applying the lessons learned from the available research on web usability to develop a smart web site that can adapt to changing user needs over time.

The first challenge in realizing this vision is gathering and acting upon a wealth of analytics data. We plan to tap into the analytics data gathered over the past 5 years on the university website and transform that data into a format that can be queried and reported on in real time. Using the Google Analytics API (Google Analytics, 2014), we will query past analytics data as well as recent trends in site usage and store that data in a simple local reporting data store.

Once we have set up an interface to extract the usage data, we will write a web service interface that can be called from a web client to expose the common usage patterns of the page the user is on. The

challenge in this module will be reporting on and summarizing the usage data quickly so the client code can make adaptations to the user interface in time to serve the user's needs.

The final piece of this solution will be a client framework that will be able to query the reporting service and take action on the data provided. The challenge here will be to make a user interface framework that is generic enough to apply to a wide range of site designs and navigation structures. The idea of this piece being that a web developer can utilize it to provide suggestions as to what a user may require on the current page. Based on the usage patterns of this page it will execute a set of rules to adapt the interface by increasing the visibility of frequently accessed content and navigation items.

Once our system is in place, we will work on implementing it on the university main site. We will create a mirrored version of the site that will use the system to make automated improvements to the user interface. With the mirrored version of the site is in place we will test the effectiveness of these changes by asking users to find certain popular content by navigating the site. We will compare the average time it takes users to find the requested content to determine the efficacy of the automated improvements. In addition to this quantitative analysis we will survey the users to obtain qualitative data on the automated user interface changes.

1.4 Organization

This thesis will be divided into five sections. In the second chapter, we will give an overview of web analytics and web usability concepts, and review the current state of the industry. We will also perform

a literature review which will analyze the current state of the art research in web analytics and web usability. In this section, we will find and summarize sources that relate to the goal we are attempting to accomplish, we will focus on papers that offer insight on how to analyze web analytics data and how to create usable web interfaces. In the third chapter, we will discuss the design science methodology (Hevner, March, Park, & Ram, 2004), and how we plan to apply its guidelines to conduct our research. In the fourth chapter, we will discuss our implementation of the automated analytics system. We will present the architecture of our solution and discuss how we implemented the different pieces of the system. In the fifth chapter, we will apply our fully realized system to a mirrored version of the UNF website. We will outline the process of implementing our system in a real-world scenario, and discuss the pitfalls and lessons we encounter along the way. In the sixth chapter, we will evaluate the effectiveness of our system by subjecting the dynamic mirrored version of the UNF website to various user tests. We will directly compare the current version of the site with the dynamic version to get a sense of effectiveness of our system. In the seventh chapter, we will discuss potential future improvements and other possible directions to take with our prototype and research. Finally we will compile our results and form a conclusion on the state of our research and its potential utility for organizations like UNF.

Chapter 2.
Background and Literature Review

2.1 Background

In this section, we will discuss various concepts relevant to our proposed dynamic analytics system. The two main concerns of our system are web analytics and web usability, we will discuss these two topics at length to provide an overview of the state of the industry. We will also provide a brief overview of other relevant areas used in this research which includes web services and data warehousing. Understanding of both of these concepts will be necessary to properly design our system.

2.1.1 Theory Background Topics

2.1.1.2 Web Analytics

Web analytics is the measurement, collection, analysis, and reporting of Internet data for the purposes of understanding and optimizing a web page (Prom, 2011). The origins of web analytics can be traced back to the practice of web usage mining. Web usage mining involves analyzing web server logs that record every request made to a web server. The idea is that by mining server activity logs, reports could be generated about usage patterns on a site (Kumari, Praneeth, & Raju, 2014). There are various problems with this method of obtaining analytics data. Most of these problems revolve around the fact that web server logs keep track of every single request made to a server (Mican & Sitar-Taut, 2009). Because every request is logged even requests that don't represent normal user actions, raw web log data can be inaccurate and must be properly filtered. For example, every request for page content is recorded separately including images, stylesheets, and script files. Recording of each individual request can result in a lot of noise in server logs which can complicate reporting. Another problem with these

logs is that all clients are logged equally including bots and search engine crawlers. Data from bots and crawlers are not relevant when trying to determine the behavior of humans on a website, and should be excluded (Mican & Sitar-Taut, 2009). In the end, data obtained from web usage mining is definitely useful, but a better solution was needed. This better solution had to be designed from the start with the intention of logging user activity specifically for reporting, and this is where web analytics comes in.

Analytics tools have been constantly evolving since the early days of web usage mining. Modern analytics tools offer a robust set of reporting tools that can help users determine usage patterns on a website as well as provide other important data about the site. Some of these additional reports include information on how the user found the site, the geographic location of users, the devices used on the site, and much more (Google Analytics, 2014). The current market landscape for web analytics tools is skewed in the direction of Google Analytics, one report form 2011 put Google's market share at 81% of all websites that use analytics tools (W3Techs, 2011). We will focus mainly on Google Analytics because of their dominance in the market, their wealth of features, and their lack of a service fee.

Google Analytics offers a wealth of reporting tools that surface information about almost every aspect of a site's user base. For our research, we will focus on a subset of these tools that surface mainly user behavioral data (Beasley, 2013). User behavioral data reports include user flow paths that show how users navigate a site, content drilldown reports that show the most popular pages on the site as a whole as well as on a given page, and traffic source reports that show how users reached the site (Google Analytics, 2014). Figure 1 shows Google Analytics a user behavioral flow report for UNF website.

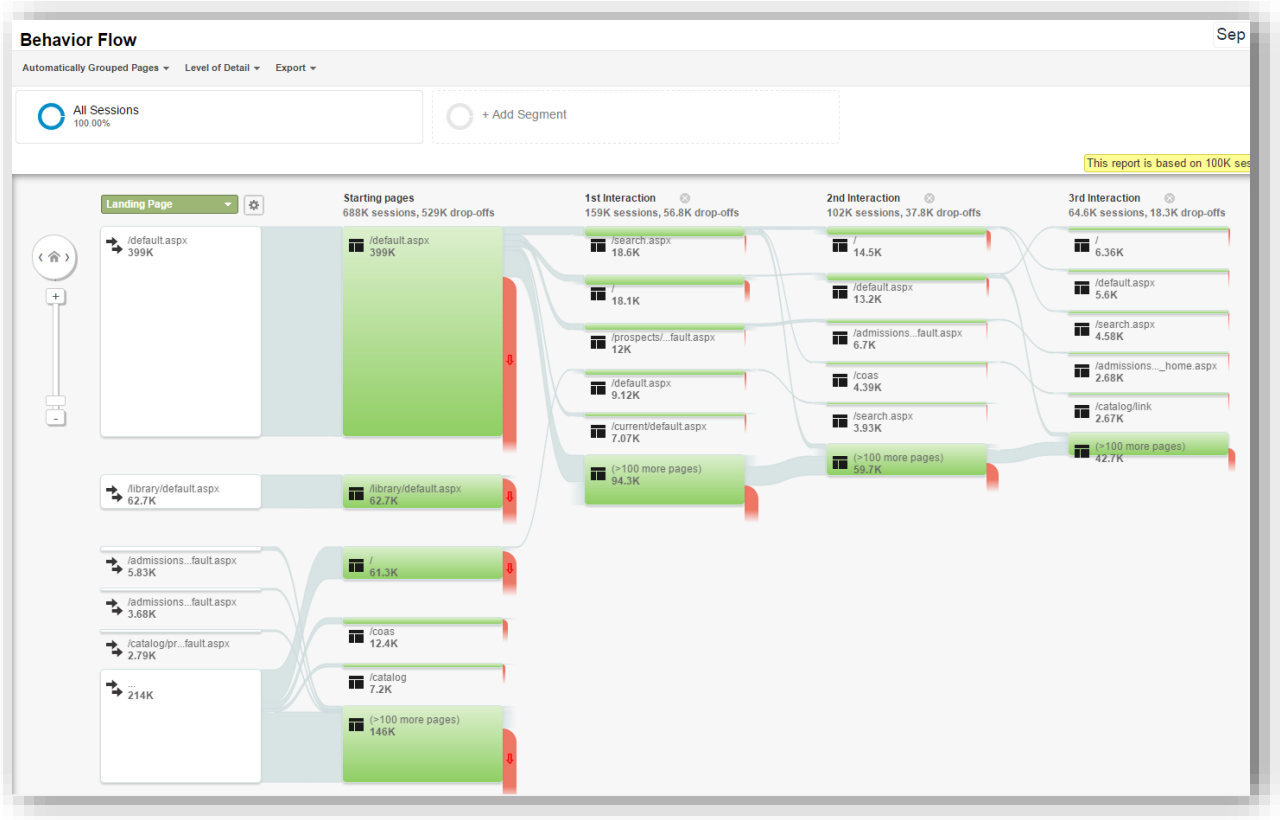

Figure 1. Google Analytics Behavior Flow Report

To extract this reporting data programmatically we will utilize Google Analytics' extensive reporting API web services. These APIs are exposed as REST web services and offer a programmatic endpoint for most of the data exposed in the Google Analytics UI. The API requires OAuth 2.0 authentication to query the data and has quota limits for each user, therefore we will likely need an intermediate layer that will query the API and cache the results for use on a high traffic website (Google Analytics, 2014).

Reports like the one above offer a window into user behavior on a site. From the report above we can see the top initial landing pages where users entered the website. We can also see the most common paths taken once on that page. We can see that most users started on the UNF homepage, and from there performed a search landing them on the "search.aspx" page. Going down the chain of user

interactions we see that some of the most popular destinations for users starting on the homepage are: Admissions, COAS (College of Arts and Sciences), Catalog, Library, etc. These links should be featured more prominently on the homepage, especially pages like COAS or Admissions which despite being available straight on the homepage, often took users multiple interactions to find. This report represents a one month snapshot of time and may only represent user needs for this specific period of time. Because of this, reports like these need to be re-evaluated multiple times a year to adapt to changing user needs. For our purposes, the data presented in this visual graph are also available in raw format from the Google Analytics API, we will discuss our process for extracting this data later in section 4.2.1.

2.1.1.2 Web Personalization

Analyzing usage data and adapting a user interface is a concept that has been around for a long time. Many different sites utilize user browsing patterns to determine additional products or information a site visitor may be interested in. A good deal of research has been done exploring the idea of web personalization. Usage patterns of individual users are analyzed and categorized into profiles that seek to predict their future behavior (Mobasher, Cooley, & Srivastava, 2000). The idea of this process being that users with similar needs and tastes would browse in similar patterns and additional products and information could be recommended to them. Most modern shopping sites utilize this type of analysis, for example Amazon.com (Amazon, 2014) has a recommendation feature that gives users suggestions based on the activity of users with similar shopping patterns. Where we will differentiate ourselves from these well-developed practices is that we seek to facilitate overall site improvement rather than personalization for individual users. Rather than personalize a site based on similar users' behavior we

aim to improve the overall usability of a site based on global usage trends, using these trends to dictate the layout of a site.

2.1.1.3 Web Usability

Web design is a complex task with many different facets to consider especially as it relates to web usability. It can be difficult to develop a site that takes into account all the possible areas of usability. In an effort to document the different types of usability concerns and provide a sort of checklist for web developers, various web usability standards have been developed. Many organizations have created their own sets of usability standards including International Organization for Standardization (ISO) 9241, ISO 25010, and ISO 1340 (Bevan, 2005). These standards cover various aspects of web applications and their development including: design process and evaluation, optimizing the user experience, accessibility, page layout, navigation, and many others. The different sections contain specific suggestions that a developer should apply to their site. For example in the "designing page layout" section of some usability guidelines it may suggest establishing a level of importance for the content, or placing important elements in the top center of the page. For navigation they may suggest providing feedback on the user's current location or keeping main navigation links always visible (Herring & Prichard, 2012). These are just a few examples of the many usability standards offered by various organizations. For our purposes we must ensure that, as we adjust page navigation and structure based on analytic data, we adhere to these usability standards and adapt the UI to more effectively implement the suggestions they provide. Adherence to web usability guidelines has been proven to positively effect a user's perception of a site, and it is in the best interest of web developers to be familiar with, and apply these guidelines to their work (Bevan, 2005).

One of the better sets of usability guidelines found in our research was the one created by the U.S. Department of Health and Human Services (U.S. Dept. of Health and Human Services, 2006). These guidelines put in simple terms the consideration that web designers need to take into account when designing a usable website. There are over 200 guidelines in the HHS document, each with a detailed description, example, and importance rating. For the sake of brevity we will not include the full listing of guidelines here, but instead will include some of the guidelines most relevant to our research in Table 1 below.

Table 1. Select HHS Usability Guidelines

#	Guideline	HHS Comments
5:2	Show All Major Options on the Homepage	Users should not be required to click down to the second or third level to discover the full breadth of options on a Web site. Be selective about what is placed on the homepage, and make sure the options and links presented there are the most important ones on the site.
5:7	Limit Homepage Length	Any element on the homepage that must immediately attract the attention of users should be placed 'above the fold'. Information that cannot be seen in the first screenful may be missed altogether - this can negatively impact the effectiveness of the Web site. If users conclude that what they see on the visible portion of the page is not of interest, they may not bother scrolling to see the rest of the page.
6:2	Place Important Items Consistently	Put important, clickable items in the same locations, and closer to the top of the page, where their location can be better estimated.
6:5	Establish Level of Importance	The page layout should help users find and use the most important information. Important information should appear higher on the page so users can locate it quickly. The least used information should appear toward the bottom of the page. Information should be presented in the order that is most useful to users.
7:2	Differentiate and Group Navigation Elements	Clearly differentiate navigation elements from one another, but group and place them in a consistent and easy to find place on each page.
7:11	Use 'Glosses' to Assist Navigation	'Glosses' are short phrases of information that pop up when a user places his or her mouse pointer over a link. A 'gloss' provides a preview of the type of information that will be found behind a link. Users prefer the preview information to be located close to the link, but not placed such that it gets in

		the way of reading the link. A gloss can be created by defining the Title attribute for a link. However, designers should not rely on the 'gloss' to compensate for poorly labeled links.
9:5	Highlight Critical Data	Visually distinguish (i.e., highlight) important page items that require user attention, particularly when those items are displayed infrequently.
10:2	Link to Related Content	Users expect designers to know their Web sites well enough to provide a full list of options to related content.
10:5	Repeat Important Links	Establishing more than one way to access the same information can help some users find what they need. When certain information is critical to the success of the Web site, provide more than one link to the information. Different users may try different ways to find information, depending on their own interpretations of a problem and the layout of a page. Some users find important links easily when they have a certain label, while others may recognize the link best with an alternative name.
11:4	Ensure Visual Consistency	Visual consistency is the consistent use of design elements such as typography, layout, colors, icons, navigation, images, and backgrounds. While users can overcome certain inconsistencies (e.g., entry fields, pushbuttons), consistent interfaces can reduce errors and task completion times. It can also reduce learning curves, and increase user satisfaction.
11:11	Highlighting Information	One study found that participants were able to complete tasks faster when the interface contained either color-coding or a form of ranking, but not both. The presence of both seemed to present too much information, and reduced the performance advantage by about half.
12:2	Place Important Items at Top of the List	Experienced users usually look first at the top item in a menu or list, and almost always look at one of the top three items before looking at those farther down the list. Research indicates that users tend to stop scanning a list as soon as they see something relevant, thus illustrating the reason to place important items at the beginning of lists.
12:4	Display Related Items in Lists	A well-organized list format tends to facilitate rapid and accurate scanning. One study indicated that users scan vertical lists more rapidly than horizontal lists. Scanning a horizontal list takes users twenty percent longer than scanning a vertical list.

2.1.2 Implementation Background Topics

2.1.2.1 Web Services

The goal of a web service is to expose a programmatic interface for transmitting data or performing actions over the Internet. Web services are called by the code of other systems to integrate data and functionality across a network. We plan on utilizing web services for two of the main components of our solution. For our solution we will exclusively be using REST web services. REST stands for Representational state transfer, and is characterized by stateless service endpoints that explicitly use the HTTP methods such as GET and POST. REST web services are services to manipulate XML (or other data formats) representations of web resources using a uniform set of stateless operations (Booth et al., 2004). REST web services are designed to be simple and adhere closely to the basic HTTP protocol. As a result of this all persistence and state management must be handled by the application.

Authentication and authorization for REST services are usually handled through the use of authentication tokens. Most REST web services offer some form of authentication using temporary authentication tokens or permanent application key tokens. Temporary tokens are often used for client side applications, and involve some authentication process with the service provider, usually OAUTH, which will provide a token that will last a limited amount of time before that authentication process must be repeated. Permanent tokens are pre-shared tokens that are often associated with a specific developer account, and are designed for server side applications that will connect directly to the REST services using this secret token (Booth et al., 2004).

Google Analytics uses REST web services to expose the reporting data, in order to extract this data we will need to authenticate to their services and extract this data. Google analytics uses permanent pre-shared application tokens, which require minimal setup (Google Analytics, 2014).

In addition to extracting analytics data via web services we will need to create REST endpoints to expose summarized and pre-computed data to our client-side framework. These services will be open, and will not use authentication tokens because these endpoints need to be exposed directly to anonymous clients. We will provide more details on the design of these web services in section 4.2.4.

2.1.2.2 Data Warehousing

We will provide brief discussion on data warehousing as it relates to our proposed system. We will not be able to rely solely on the Google Analytics API for all our reporting. We need the ability to query summarized reporting data on every page load. To do this we cannot simply call the analytics API, as this would greatly increase the time it takes our page to fully load. We will also need to pre-compute and store summarized usage statistics to further increase speed. For this task we will use some well-established data warehousing techniques (Fasel & Zumstein, 2009).

A data warehouse is a subject-oriented, integrated, time-varying, non-volatile collection of data in support of a decision making process (Inmon, Strauss, & Neushloss, 2010). In other words it is a way to store data about certain subjects as they change over time. This is a good fit for the kind of data we are attempting to gather and analyze. In our case the subjects are the web pages being visited by users. We need to analyze how traffic to and from these pages changed over time. Because we are using a warehousing database in a real time manner we will need to develop a warehouse that can respond quickly while still providing the subject-oriented time-variant strengths of a traditional warehouse.

The first process that needs to take place when developing a data warehouse is the design of the schema. A simple data warehouse schema, known as a start schema, includes two types of data: facts and dimensions. Facts are the central object of a star schema and contain the summarized data from snapshots of time. The dimension tables radiating off of the fact tables provide the detailed information about the objects represented in that snapshot of time in the fact table. This schema allows for historical record of statistics over time by querying for summarized data (facts) based on different attributes of business data (dimensions) such as dates, product names, etc. (Inmon et al., 2010). Because we are planning on using NoSQL database technology that isn't as heavily designed around relationships, we will be flattening this idea of a star schema, while also retaining some of its core features. We will discuss our specific implantation in more detail in section 4.2.2.

Another important aspect of data warehousing is the Extract Transform Load (ETL) process. This involves pulling data from a transactional data source, transforming it into a format more suited for reporting purposes, and loading it into the warehouse. The ETL process maps the schema of the transactional database to the schema of the warehouse dimension tables (Inmon et al., 2010). It also performs data summarization tasks to store statistics about dimensions in the fact tables. We will be performing a continuous ETL process based on pages a user is requesting. We will be mapping the data pulled from the Google Analytics APIs to the documents in our data store when pages are requested for the first time by a client. As part of this process we will be making multiple calls to the Google Analytics API and combining the data from multiple queries into single facts about page navigation trends. We will give a detailed description of this process in sections 4.2.1 and 4.2.2.

2.2 Related Work

From our research into web analytics and its application to web user interface design we found that it was a well explored topic with research dating back to the early days of the web. We found that the techniques of web usage mining and its applications to site personalization have been around for a long time and are relatively well explored. The more recent trend of using web based analytics tools such as Google Analytics to improve web usability is also a well-represented topic. We did, however, find a gap in the published literature relating to improving site usability based on analytics data in an automated fashion. We chose to focus our research on taking the knowledge from published sources about improving usability based on analytics data and finding a way to apply those methods in an automated way. In this literature review, we present some of the most useful sources we found relating to this topic and will discuss how we plan to use the existing research to develop our solution.

2.2.1 Web Usage Mining

The idea of gathering website usage data for use in improving site design began with the concept of web usage mining. Web usage mining involves analyzing web server logs and drawing conclusions about usage patterns from these logs. Traditional data mining techniques such as loading the data into analysis cubes in star and snowflake schemas and reporting on that data are used to track individual users and find overall trends of usage on the site. In Büchner's paper on web usage mining for marketing purposes he outlines a process for creating a generic reporting cube for analytical data (Büchner & Mulvenna, 1998). This paper offers some insights on how to organize and report on web usage data. With so much data constantly flowing in from high traffic websites these reporting

techniques could prove useful for our research. This paper focuses on using web usage mining and reporting for ecommerce purposes to help drive product strategy for companies, which is not the primary focus of our research and it may not entirely apply (Büchner & Mulvenna, 1998). The paper used web log data from an online retailer to perform its analysis. Because their primary focus was retail applications, the research doesn't entirely apply to our goal of improving usability and finding informational data rather than products. The paper also devotes a good deal of time discussing the extraction of web log data which is irrelevant for our purposes ads we are using web analytics data that is gathered for us. What we can take from this paper is some insight into how to architect a data store based around web usage data (Büchner & Mulvenna, 1998).

Another common application of web usage mining is user personalization. From the web usage data mined from server logs it is possible to extract profiles of user activity and match other anonymous users to those profiles. In the paper by Mobasher et al. the idea of mining user profiles is presented (Mobasher et al., 2000). Based on user navigation patterns, they form profiles of user activity and attempt to match live user activity to these profiles. If a user's activity fits one of their mined profiles they then automatically offer the user suggestions of other pages or products they may be interested in. This approach to automatically guiding a user based on analytics data is somewhat similar to our proposed process. The way they generate user profiles and determine other pages a user might be interested in could be very useful in the implementation of our solution. Where we believe they fall short is in the area of updating the user interface. This paper does not go into concrete ways of improving user experience, it is more focused on matching users to profiles and suggesting links. The paper is also based on data mined from web logs which can be unreliable and misleading as compared to modern web analytics tools due to the nature of data collected in web logs (Mican & Sitar-Taut, 2009). With our research, we plan to expand on the ideas in this paper and focus less on matching users

to profiles and instead making general user interface improvements based on overall site trends (Mobasher et al., 2000).

There are some inherent problems with any web usage monitoring system that must be overcome if any useful data is to be mined. The paper by Mican et al. covers some of the difficulties that must be considered when mining usage data (Mican & Sitar-Taut, 2009). Mining data from web server usage logs was the standard way of finding out what your users were looking at on your site until web analytics came along. The problems identified by this paper about this kind of data mining include things like search engine bots, content requests that are part of a different overall page request, differentiating between content pages and navigational pages, and various other problems. Although these problems were addressed in relation to web usage mining rather than web analytics, we believe they provide good insight into some of the problems we may face wen mining web analytics data. These problems must be taken into account when analyzing analytical data, especially when that data will be used for automatic changes to a user interface. For our research, we plan to use some of the insights presented in this paper to evaluate whether the analytics data we are mining are legitimate user behaviors. This paper does not draw any significant conclusions about content pages versus navigational pages which will also need to be a consideration in our final design so we will need to do our own research in that area (Mican & Sitar-Taut, 2009).

There is extensive research into web usage mining as it applies to selling products. The data mined from user activity can be applied to other ends rather than just trying to recommend more products and services. The paper by Kumari et al. explores the potential of using web usage mining and user profile analysis to improve the structure and content of a website and track how user interests change over

time (Kumari et al., 2014). This constant analysis of changing trends over time is a key tenant of our research which is why this paper is useful for our purposes. This paper also takes this analysis a step further and does not only analyze usage patterns but also analyzes the content that the users are viewing. By analyzing the content of a page that a user ends up on, they draw conclusions based on analysis of that content to find other pieces of content that may be related semantically to the content the user found. This paper focuses mainly on web usage mining and is also concerned with generating user profiles, which is not the direction we want to take with our research. Although this paper does not apply specifically to the ideas we are pursuing it does present some very interesting points especially related to content driven websites rather than product driven websites (Kumari et al., 2014).

Analyzing data from web metrics is a complex task, the data is overwhelming and the potential pitfalls are abundant. The paper by Weischedel et al. performs an extensive case study on the use of web metrics (Weischedel & Huizingh, 2006). The papers seeks to find the limitations of analyzing web metrics and finding the alternative data sources that help supplement this data. The paper draws some interesting conclusions on web metrics analysis including the idea of gathering queries made from particular pages and using that data to determine what information should be included on that page. It also champions the usefulness of customer opinion data to supplement hard log data to gather some qualitative information that may help improve the design of a site. This paper focuses mainly on clickstream data obtained from server logs which can be unreliable and lead to incorrect conclusions. Because we plan to use web analytics as opposed to log based web usage mining many of the conclusions reached in this paper do not apply. Despite the limitations of this paper it does offer some interesting conclusions about applying knowledge gained from usage data into concrete site improvements (Weischedel & Huizingh, 2006).

2.2.2 Web Usability

Gathering and analyzing the usage data is only half of the problem we plan to address in this thesis. These metrics on user behavior are useless without the concrete design improvements that follow them. There are several sets of usability guidelines that attempt to address the design considerations of a site. The paper by Lai et al. analyzes one of the industry standard sets of guidelines, the Microsoft Usability Guidelines (MUG) by applying the Repertory Grid Technique which is a qualitative evaluation methodology used heavily in market research (Lai, Xu, & Tan, 2009). This paper offers some valuable insight into what users are looking for in a web page in their own words and categorizes them into actionable areas based on the MUG. One of the important points presented in this research was the emphasis on relevance on a site, the idea that content on any given page is relevant to the core users of that page. This is one of the core ideas of our research, by mining data from analytics as to what other pages are most useful to other users of this page is backed up by these updated usability guidelines. This paper offers some suggestions on how to improve usability of a site such as increasing icon size for important elements, but it doesn't go very far in suggesting user interface improvements, we will have to draw these conclusions from other areas of our research (Lai et al., 2009).

For some concrete ideas on improvement of a website's usability we will look to other sources, specifically a paper by Webster et al. entitled "Enhancing the Design of Web Navigation Systems" (Webster & Ahuja, 2006). This paper tackles the topic of navigation usability. It looks at the global navigation of a site and emphasizes concepts like a sense of where you are, and where your next click will lead you, and what content you will find at that link. This concept will be important for our work,

some indication of what other users found after following a certain path could lead to subsequent users finding relevant information quicker. This paper examines the idea of global navigation, a common navigation element across the whole site that shows your current location in the site, to reduce the perceived disorientation of users on a site. The paper compares three different versions of a site, by asking participants to find specific information on the site. The findings of the paper suggest that simple local navigation systems often behaved better than global navigation, perhaps because users were presented with fewer choices and less of an information overload. For our system, we will take into account some of the lessons learned in this research to help us adapt navigation systems using analytics data (Webster & Ahuja, 2006).

Chapter 3.
Research Methodology

3.1 Design Science Research Methodology

For this thesis we will be utilizing the Design Science Research Methodology. Design science research involves the creation of new knowledge through the design of novel or innovative artifacts and the analysis of the use and/or performance of those artifacts along with refection and abstraction (Vaishnavi & Kuechler, 2013). The real point of the design science research methodology is the idea that design is research and the act of designing an artifact is a valid method of conducting research. What the design science methodology stresses over the typical design process is the idea of knowledge contribution. A design project should have a strong focus on contributing knowledge to the field and sharing the results.

3.2 Design Science Research Guidelines

Design science research sets forth various guidelines that provide a framework for executing the design process. These are not strictly enforced guidelines for the design process, rather they are simply aspects to consider during the design process (Peffers, Tuunanen, Rothenberger, & Chatterjee, 2007). Below is a listing of those guidelines and how we plan to meet them for our research.

3.2.1 Design as an Artifact

The purpose of this guideline is to ensure that the research works towards producing a viable artifact in the form of a construct, a model, a method or an instantiation. For our research, we plan on producing a working example of our dynamic analytics framework. This will be our physical working artifact that we will be able to test and evaluate. We will outline this artifact in detail in chapter 4.

3.2.2 Problem Relevance

The point of this this guideline is to define a specific research problem that is relevant to the business problems of the real world and justify the value of a solution to that problem. The problem this thesis is seeking to address is the degrading usability of websites over time, as user needs change, and the large amount of manual work that must be done to maintain a site's usefulness. There is a need for an automated way to utilize the web analytics data that is already being gathered on many sites to keep a site up to date and reflective of current user needs.

3.2.3 Design Evaluation

The goal of the evaluation guideline is to examine the effectiveness of the finished artifact on the problem. We will use the live UNF website, and the alternate version of the site discussed in the previous activity, to perform A/B testing with site users. The users will be asked to accomplish a task on one version of the site. We will compare quantitative measurements such as time to accomplish the

task, as well as qualitative measurements in the form of user surveys for the two versions of the site to determine if our artifact is effective in solving the problem.

3.2.4 Research Contributions

The research contributions guideline states that the research should provide contributions in the areas of the design artifact, design foundations, or methodologies. The objective of this research is to create a generic and reusable platform for querying web analytics data, analyzing usage patterns, and using that data to adapt the user interface of a site. This artifact will contribute to the growing field of adaptive analytics and will serve as an example of how to dynamically use analytic data to adapt sites. This system should be generic enough to apply to any site while also allowing levels of developer customization to fit an organization's individual needs. The resulting site will adapt dynamically to traffic patterns and lead users to their destination more quickly. We believe that this research will contribute significant insights into the field of dynamic analytics especially in areas outside of E-Commerce.

3.2.5 Research Rigor

The purpose of this guideline is to ensure that the decisions made when implementing an artifact are well informed and represent the best possible solution to the problem. Decisions made in the development of the artifact should be justified and backed up by research. For our research we plan to back up every decision with specific research and exhaustive analysis. We plan to justify each of our

decisions according to the best information available to us. Essentially our research rigor will be derived from the effective use of the existing knowledge base.

3.2.6 Design as a Search Process

This guideline states that the design process for an artifact should be a search process to find the best possible solution to the problem. For our research, we have done extensive searching into the field of analytics and plan to use various existing tools to help us architect our solution. We are not starting development of our system from scratch, we are taking the state of the art technologies available today and expanding on them with our own ideas. We will continue to evaluate alternative options as we develop our solution keeping in mind that we are always searching for a better solution to the problem.

3.2.7 Communication

The final guideline of the design science research methodology is communication. The problem and its importance as well as the resulting artifact and its effectiveness should be conveyed to relevant audiences. In adherence to this guideline this thesis will be defended in a public forum and the resulting research will be published along with the source code of the resulting artifact.

Chapter 4.

Dynamic Analytics Framework

In this chapter, we will discuss our development plan for the solution and justify the decisions we make during each step of the design process. We will first discuss the architecture of the system which includes components for extraction of data from Google Analytics and the client side framework that will adapt the website. We will present different options available to us in terms of frameworks and technologies, and justify our final choice based on the features of that technology and our ability to learn and implement that technology in a timely manner. Next, we will evaluate the feasibility of our timeline for development. We will take into account our knowledge of the technologies in use and the estimated time it will take us to perform the development tasks. The end result will be a detailed development plan including dates for completing milestones. Finally, we will discuss the economic feasibility of our complete system. We must address the costs required to develop and host each of the components involved in the system. Where possible, we will use open source technologies to avoid large costs, but there will likely be some cost associated with server hosting. The end result of this chapter will be a detailed plan for development of our system that will prepare us for the development portion of our research.

4.1 Website Improvement Process

The current process for updating the design of a website based on analytics data is a primarily manual process involving multiple stakeholders. The basic process involves a web team that can consist of many different specialized individuals including developers, designers, marketing personnel, content creators, etc., generating reports from analytics tools and using those reports to make decisions about website

design (Weischedel & Huizingh, 2006). The developers and designers then go to work updating the underlying code, the design, and the content of the website. Those changes are published to the live site and the cycle continues again as analytics data are gathered on the new design. This process needs to be repeated often to maintain the usability of a site as user needs change. Figure 2 provides diagrammatic overview of the website improvement process.

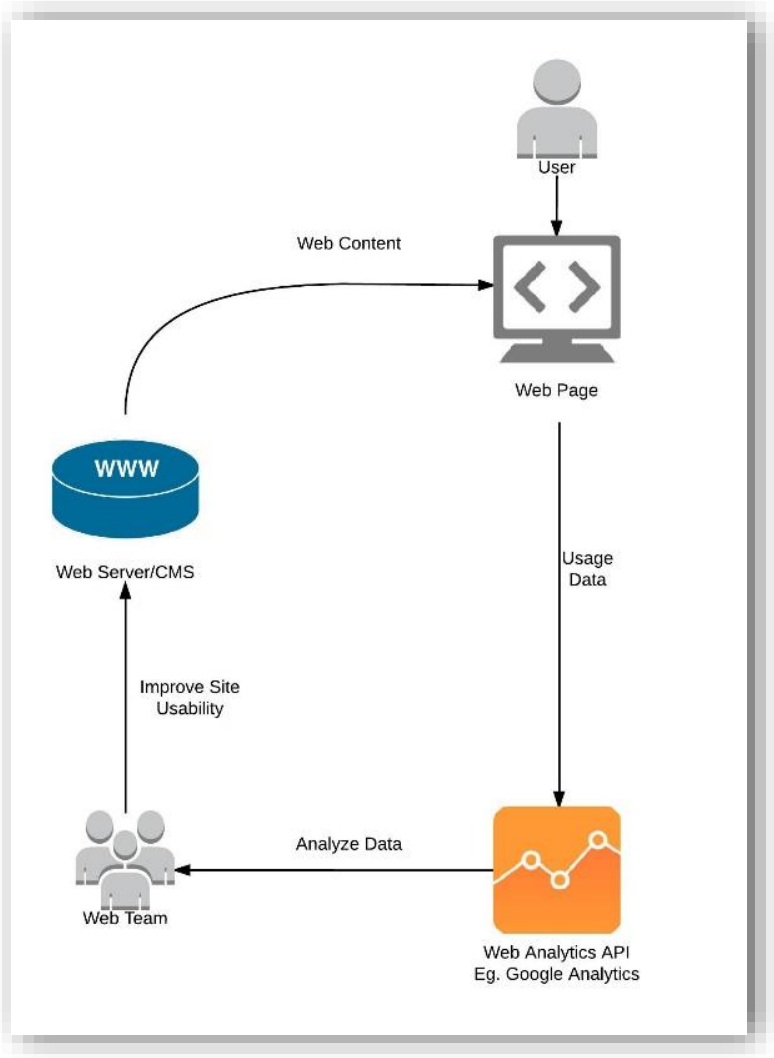

Figure 2. The Website Improvement Process

4.2 Dynamic Analytics Framework Architecture

While the process outlined above cannot be completely be replaced by automated processes, we believe that our system will be able to take over some of the smaller, more data driven decisions. This would free up the web team and allow them to focus on the more sweeping and important interface changes. Our system will read and analyze analytics data and apply the lessons learned from this data to site improvements, much in the same way that a web team does. To do this our system will need to consist of four major components. Firstly it will need a mechanism for extracting data from Google Analytics reporting APIs. We will then need to store that data in a way that it can be quickly extracted and used for interface updates. We will then need a service layer that can expose that reporting data to the client framework which will be responsible for updating the interface. Each of these layers will be discussed in more detail later in this chapter. Figure 3provides a quick overview of how each of these pieces fit together.

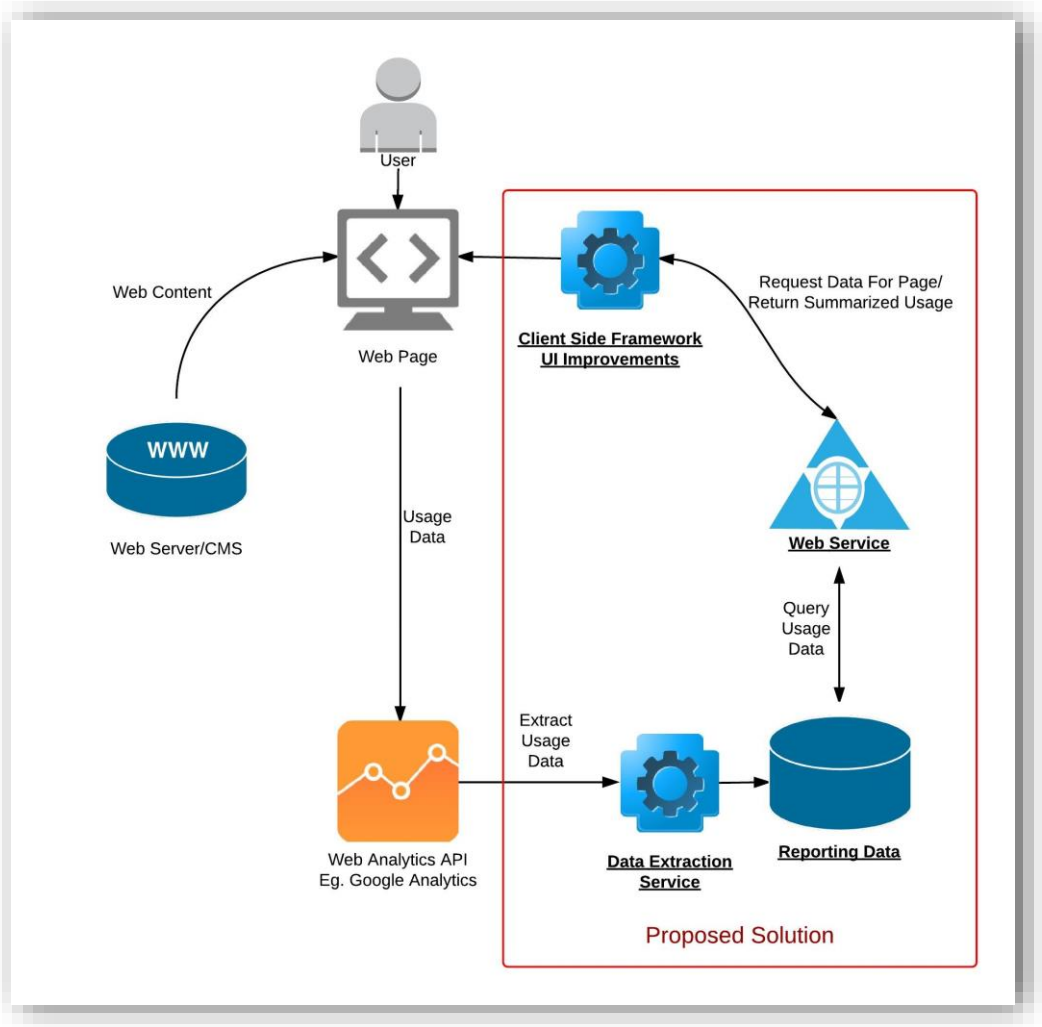

Figure 3. Dynamic Analytics Framework Architecture

4.2.1 Extracting Analytics Data

The first task our system must accomplish is extracting live analytics data from our web analytics provider. We have chosen to use Google Analytics for our system because of its overwhelming market share and comprehensive feature set. Google currently enjoys an 81% market share in the field of web analytics as of 2011 (W3Techs, 2011). Google also offers a feature rich reporting API that will allow us to extract the analytics data and use it for our own purposes. These APIs are REST web service based and

we will be able to easily tap into them with a simple server-side REST client. The Google Analytics reporting API does have two drawbacks that prevent us from using it in real time to query reporting data. Firstly, because it is a free service Google imposes rate limits on its reporting API to prevent overuse. Secondly, because the API is mainly intended for reporting it does not provide the kind of speed necessary for us to use it to update an interface in real time (Google Analytics, 2014). Because of these limitations we must extract the reporting data, transform it into a format more fit to our purposes and store it ourselves.

To facilitate this, our system will receive incoming requests and first query our database to see if we have already cached the reporting data for that request. If data for that page is not found, the Data Extraction Service (See figure 3) will be executed to extract the data from the reporting API. This process will be a separate module of the Web Services application we will discuss later in section 4.2.3. This module will be run on a separate thread, and will be responsible for asynchronously updating the data store with the data gathered from the analytics API. When data on a page is not available in our data store, or the data gathered previously is expired, the service application will create a new threaded task to update that data then return to the client. We will store this reporting data with time stamps so we can enforce an absolute expiration time for these reports. This will allow us to keep a history of activity over time while also obtaining data on new trends. If the data for a given page request is not found or if it is past its expiration, the middleware will fetch fresh data from the Google Analytics API, store it in our data store, then return it to the browser. Figure 4 provides flow chart representation of this process.

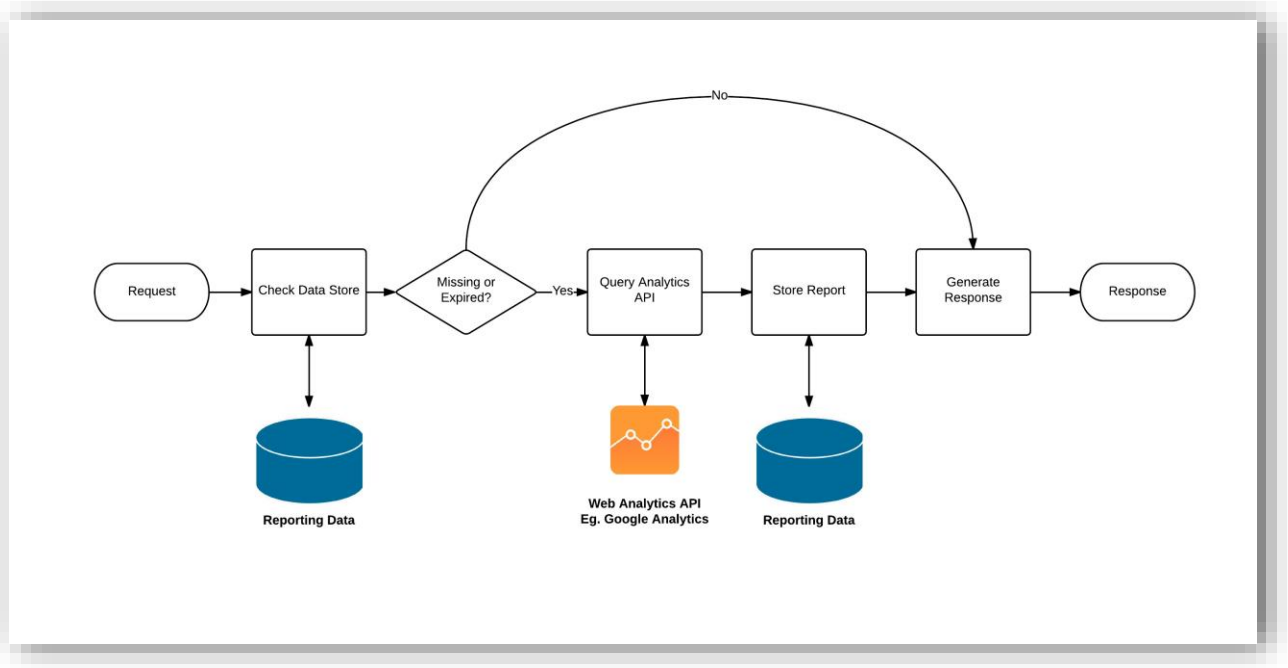

Figure 4. Querying and Storing Analytics Data

The data we are extracting from the Google Analytics API will need to be transformed and mapped to our database structure. The Google Analytics API provides all of its data via a single REST endpoint. To that endpoint we will pass a set of dimensions and metrics which will determine the data we get back. Dimensions represent attributes of single items such as pages (title, path, etc.) whereas metrics represent computed statistics about those pages (views, time on page, etc.) Table 2 shows how we will query the reporting API data and how those dimensions and metrics will be mapped to our database PageSnapshot schema. The query to extract GlobalTrend data will be very similar, we will simply remove the filter parameter. More information on the database schema is outlined in section 4.2.2.

Table 2. Analytics API Queries

Building PageSnapshot

Property	Type	Mapped To
PageURL	String	Provided
DateRetrieved	Date	Current Date
PrevPages	Page[]	Navigation Query
NextPages	Page[]	Navigation Query
CommonDestinations	Page[]	Navigation Query
Searches	Search[]	Search Query

Navigation Query **Returns: Page[]**

Dimensions	Metrics	Filter	Sort
pagePath **OR** **exitPagePath**	pageviews	prevPage = PageURL OR nextPage = PageURL OR pagePath = PageURL	pageviews Desc
pageTitle	avgTimeOnPage		
	exitRate		

Result Column	Mapping
pagePath	Page.PageURL
pageTitle	Page.PageTitle
Pageviews	Page.Hits
avgTimeOnPage	Page.AvgTimeOnPage
exitRate	Page.ExitRate

Search Query **Returns: Search[]**

Dimensions	Metrics	Filter	Sort
searchKeyword	searchResultViews	prevPage = PageURL	searchResultViews Desc
exitPagePath			

Result Column	Mapping
searchKeyword	Search.Keyword
exitPagePath	Search.Destination
searchResultViews	Search.Hits

4.2.2 Analytics Data Store

The next component of our system is the analytics data store which will be used to store the reporting data we queried from the Google APIs. Our data store will share many characteristics with data warehouses. Data warehouses are subject oriented, time variant, and nonvolatile stores of summarized reporting data (Inmon et al., 2010). Our data store will incorporate all of these properties. The data structure of our database will be based on subjects such as the summarized analytics data of a given webpage and the pages users navigated to next. These will be stored as a single document in our database (see below for a detailed data design.) We will also store our data in a time variant and nonvolatile way. We are interested in analytics data in snapshots of time. Because of this, we will store the analytics data pulled from Google Analytics with time stamps to indicate when it was pulled from the API, this will allow us to look back on changes in traffic patterns over time. Although we are following many of the concepts of traditional data warehousing we are not constraining ourselves to typical data warehouse design.

We are designing our database with facts and dimensions, just like a traditional data warehouse start schema. Because NoSQL relies less on relationships between documents we will be flattening out the facts and dimensions of our star schema into a single document. For example, our PageSnapshot object (See figure 5 below) will represent a fact, that fact will contain summarized data about a specific web page at a specific time. The time the data was retrieved, the data about the page itself, and its related pages are all dimensions that can be used to query information about that fact. As you can see below in figure 5, the facts and their dimensions are stored in the same document, which is more consistent with NoSQL document based data design.

For our data store we will be using a NoSQL database (Pokorny, 2013). NoSQL data stores provide a few key advantages we are interested in. NoSQL offers schemaless design allowing us to easily expand our data models to add new functionality. As we discover new important metrics about user patterns and expand our framework, we will need to expand the data model and add additional summarized statistics. NoSQL gives us the ability to do this on the fly without completely redesigning our database schema. This will give us a good deal of flexibility during the design process. Most NoSQL is also very horizontally scalable, meaning we can easily scale our single database to account for increased traffic. This means that even with our relatively limited resources and funds we will be able to create a scalable database that could be applied to a very popular website like the UNF website. By simply requesting additional instances of our datastore we can rapidly increase the performance of our framework. Finally, it offers extremely quick reads and writes across multiple instances with an "Eventual Consistency," meaning we can very quickly perform writes to the data store and eventually get consistency with other users on different instances. Because we are not writing a purely transactional system we are not necessarily concerned with the immediate consistency between queries offered by traditional SQL databases, and as a result we will be able to take advantage of the performance gains afforded by having multiple independent instances of our datastore (Pokorny, 2013). We will discuss our specific choice of NoSQL technology in section 4.3.

The data design of our data store (see figure 5) will use the concept of documents (Pokorny, 2013). We will define document types for the different features of our framework. Below are the data definitions of our documents in UML format, essentially they are documents containing key value pairs with sub documents containing their own key value pairs. These nested documents are stored together rather

than in traditional in related tables.

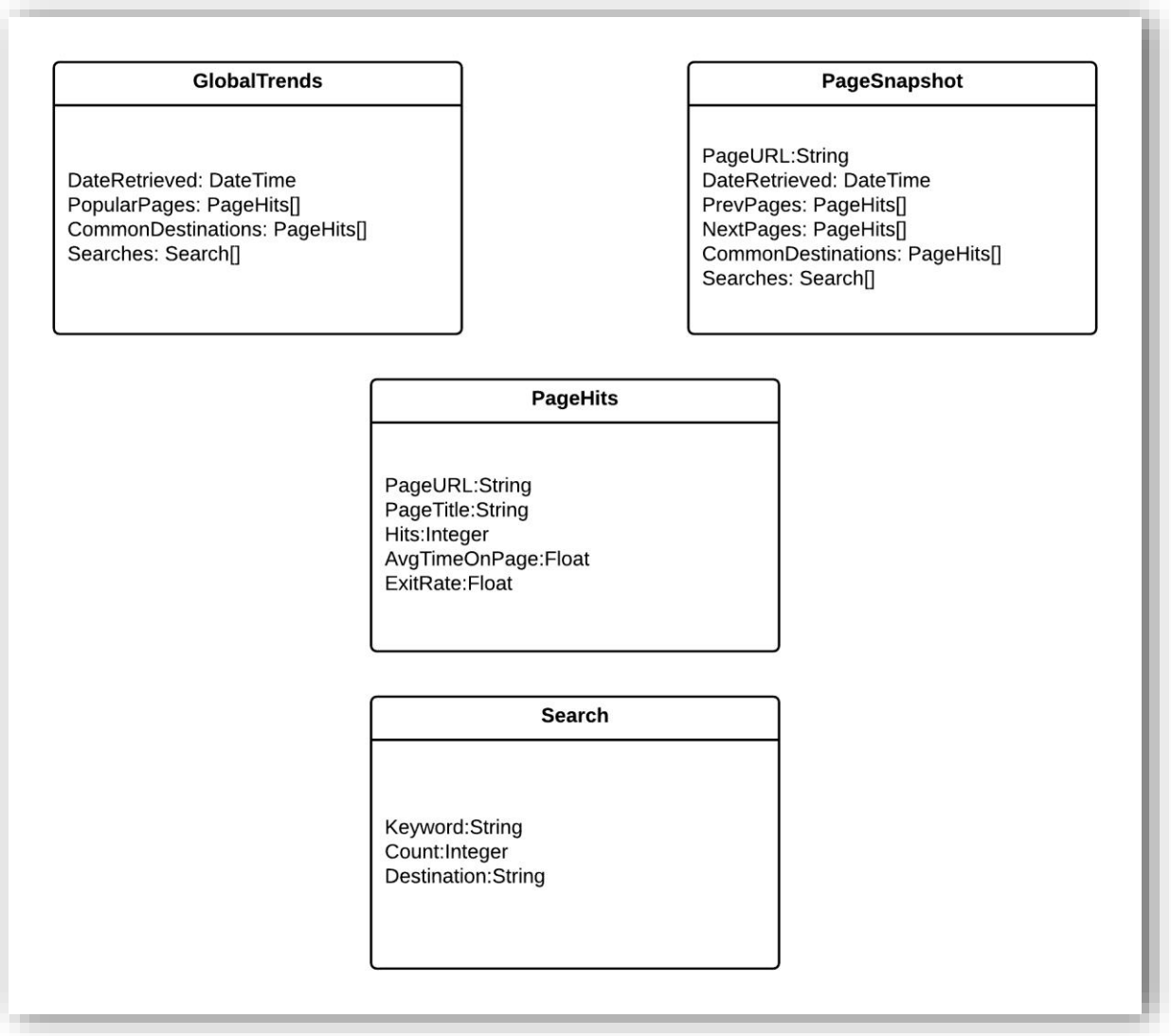

Figure 5. Data Store Schema

We have two main document types: GlobalTrends and PageSnapshots. The GlobalTrends documents contain information about the most popular content on the site overall. This information will be queried from various endpoints of the Google API and consolidated in a single document. These documents will have time stamps of when they are retrieved allowing us to set an expiration time for this data as well as track changing usage trends over time. The PageSnapshot documents will contain information about

specific pages a user is visiting. They will contain information about the pages that are often navigated to next and the common end destinations when navigating through this page. These documents will also contain information about searches performed from this page and where those users eventually ended up. All this data will be collated from various queries to the Google API and stored in in this format to maximize retrieval speed. The sub documents of PageHits and Search will contain the raw data about page hits and search queries and will be contained within their parent documents.

We will place indexes on the DateRetrieved and PageURL properties to improve performance of select queries on these properties.

4.2.3 Web Service Layer

Once the analytics data has been gathered and stored in our database, we will need to expose that data to client browsers. We will need a web service layer that will serve as the endpoint for queries on page analytics data. We will expose this data via REST web services that return the summarized data from our data store in JSON format. We will have two endpoints, one that will return the current global usage trends on the website, and one that will return usage trends for specific pages. Figure 6 provides UML class representation of the REST web service.

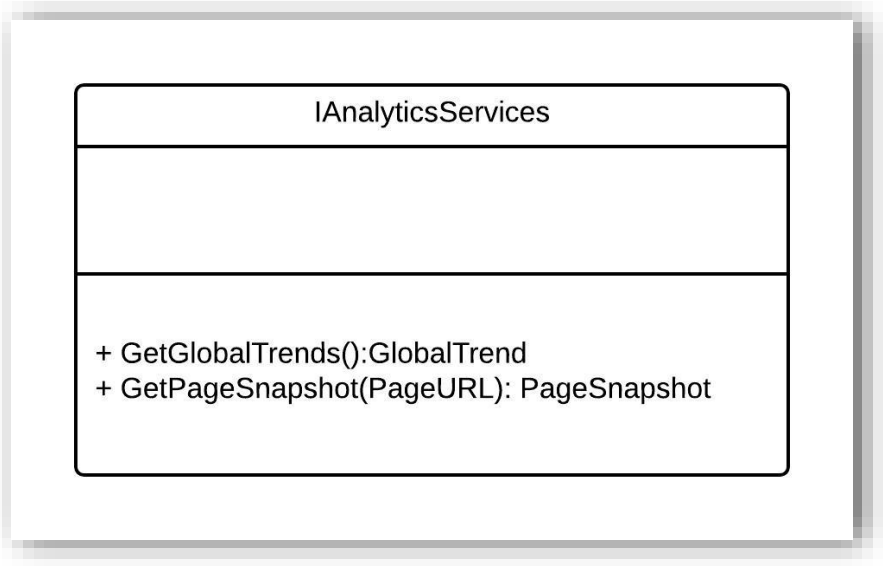

Figure 6. Web Service Interfaces

The web service application will be responsible for determining where the data is pulled from. The logic for determining whether the cache and the database are up to date will exist in the web service layer. In addition, the web service layer will be responsible for creating another threaded task to update the data store when it is discovered to be out of date. Figure 7 provides a UML sequence diagram of the data flow logic that will determine where the analytics data will be pulled from when the web services are called.

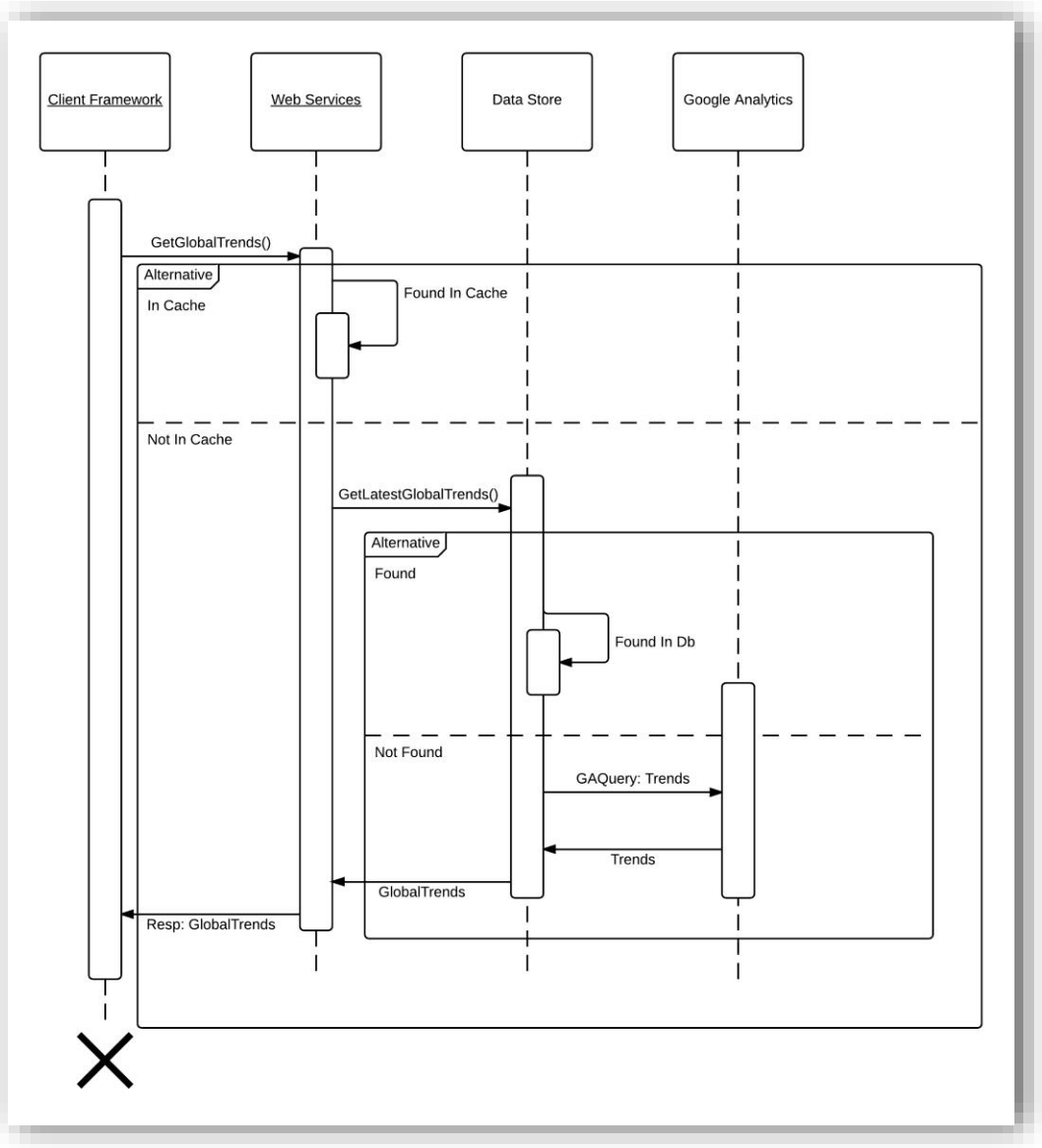

Figure 7. Web Service Sequence Diagram

The most important factor for these web services will be speed. These services need to return the requested usage data as quickly as possible. To accomplish this we will heavily utilize distributed caching technologies. We will talk more about the specific technologies in section 4.3 below.

4.2.4 Client Side Framework

The final component of our system is the client side framework that will do the work of adapting the user interface of the web page. The primary function of this component will be the adaptation of the user interface based on the data retrieved from the web services. The goal of this component will be to change the interface in subtle ways that surface more popular navigation options, while not changing the interface in a way that disorients returning users. To do this we will need to take into consideration all the lessons we learned about web usability which we outlined in our background and literature review chapter.

This component will also need to be highly customizable for web developers. We want to surface the analytics statistics we are gathering in such a way that developers can define behaviors based on the data returned. Developers will be able to subscribe to certain events in the client side framework that will rank navigation options on a page and allow developers to assign different styles to navigation options of different popularity. The client side framework will also have functions that return popular and trending topics on the site as a whole (global trends) which will allow developers to create sections of a page that always display the most important links on a site, and dynamically update as those popular links change.

This client side framework will be written using a language called CoffeeScript. CoffeeScript is a language that compiles into standard JavaScript and offers various syntactical and structural benefits over vanilla JavaScript (CoffeeScript, 2014). This language will help us create a well architected JavaScript framework that will be easily maintained and expanded upon. Object oriented design can be

very difficult and verbose when using standard JavaScript. With CoffeeScript we will be able to more easily design a client side framework that is maintainable and extendable

We will also utilize the virtually industry standard jQuery framework to assist with Document Object Model (DOM) manipulation tasks. The DOM is an interface for dynamically accessing and modifying the content and structure of HTML documents via JavaScript (W3C DOM Interest Group, 2005). To update the user interface of a website programmatically we will need to manipulate the DOM by adding styles and HTML elements. jQuery is used industry wide for client side user interface design and will greatly speed up our development efforts for the client side framework over vanilla JavaScript (jQuery, 2014).

4.3 Technology

While researching different technologies to build our system, we found many different options that each offered their own unique advantages. In the end, we settled on a technology stack that would allow us to easily integrate all the modules of our application while also providing high performance scalability. We have analyzed two different possible technology stack options outlined below. We will need to find technologies to fulfil the following requirements.

Table 3. Components

Server	Host web application
Web Application Language	Application logic Handle REST API requests
Cache Technology	Store frequently accessed reporting data
Database	Long term data storage Historical reporting data

4.3.1 Google App Engine Technology Stack

Our first technology stack choice is the Google App Engine platform (Google App Engine, 2014). This platform offers a high performance in-memory caching strategy backed up by a cutting edge NoSQL database infrastructure based on Google's own BigTable technology (Google App Engine, 2014). This will allow for automatic caching of frequently accessed data without additional programming effort integrating cache and database technologies. It also offers full featured web application hosting for our REST web services and data extraction service layer using the Python language. We are already familiar with this technology stack, so the learning curve should be small. App Engine also promises to be highly scalable if we need to subject the system to heavy load (Google App Engine, 2014). The reason we considered the App Engine technology stack is it offers all the components we require in a single integrated stack. With minimal integration work we will be able to satisfy all of our technology requirements. The pricing model for App Engine is also reasonable, and we will discuss the details later in section 4.5. Table 2 provides summary of Google App Engine technology stack.

Table 4. Google App Engine Technology Stack

Server	Google Cloud Platform (Google App Engine, 2014)
Web Application	App Engine Python Runtime Environment
Cache	Google NDB Datastore
Database	Google NDB Datastore

4.3.2 Microsoft Technology Stack

As an alternative option, we have also chosen another technology stack that could satisfy the same technological requirements as the App Engine stack. We are also very familiar with the Microsoft .NET

technology stack and we can use a collection of other tools to produce the same environment that is packaged together with app engine. The integration effort for this technology stack would be significantly higher than the app engine stack. The Microsoft .NET MVC framework will allow for development of REST web services and the creation of services for extracting data from the analytics API. The Redis cache server will allow for in-memory storage of frequently accessed reporting data, and the mongoDB database will allow for more permanent storage of historical reporting data. The difficulty of this technology stack will be the integration effort between the components. There are frameworks available for integrating these different technologies, but the integration and installation efforts would be significantly higher than the Google App Engine stack which comes pre-installed and integrated out of the box. The pricing for this stack will likely be higher than the App Engine stack and it will take more effort to integrate each piece. Table 5 provides summary of Microsoft technology stack.

Table 5. Microsoft Technology Stack

Server	Microsoft Windows Server 2013 running on Amazon EC2 Web Services (Amazon Web Services, 2014)
Web Application	Microsoft .Net MVC4 Web API (Microsoft ASP.NET, 2014)
Cache	Redis Cache Server (Redis, 2014)
Database	mongoDb (mongoDB, 2014)

4.4 Timeline

The ultimate goal for completion of this thesis is the end of the spring 2015 term. This section outlines the timeline for development of our system, evaluation of the proposed solution, and writing of final thesis document, and thesis defense presentation. This prospectus will serve as the first half of the final

paper and we will update it as we work through the development process. We will organize our development process into sprints, loosely following the agile methodology (Martin, 2003). At the beginning of each 2 week sprint, we will create a backlog of features to be completed in that sprint. At the end of that sprint we will generate a working prototype for testing and evaluation. Below are the milestone dates for the development process.

Table 6. Timeline

December 1st	➢ Present prospectus ➢ **Start Sprint #1** o Setup development environment o Create overall project structure
December 6th	➢ Present at SOC symposium to get feedback on the research idea, proposed solution, and the approach
December 15th	➢ **Start Sprint #2** o Create Database o Contact IRB in regards to evaluating system using human subjects, begin registration/approval process if necessary
December 29th	➢ **Start Sprint #3** o Data import from Google Analytics
January 12th	➢ **Start Sprint #4** o Create web services layer o Create client side framework
January 26th	➢ **Start Sprint #5** o Testing the system functionality o Implement system on copy of UNF website o Begin organizing user testing
January 30th	➢ Deadline for graduation application
February 9th	➢ **Development Complete** ➢ **Begin controlled experiments to evaluate the system**
February 13th	➢ Deadline for committee membership
February 23rd	➢ Finish first draft
March 9th	➢ **Finish final draft**
March 23rd	➢ **Defend**
April 3rd	➢ Deadline for thesis defense
April 17th	➢ Deadline for thesis submission

4.5 Budget

Below we have outlined two separate budgets for each of the technology stack identified for the development of our system. All the development tools and machines we plan to use are either already owned or free and open source.

4.5.1 Budget: Google App Engine Technology Stack

Google App Engine charges by the hour per running application instance, for outgoing network traffic, file system and database storage, and read/write operations on the database. Table 5 provides summary of a very liberal estimation of our potential usage. These prices are very low and even if we vastly underestimated our usage, the pricing will still be very reasonable

Table 7. Budget: Google App Engine Technology Stack

Google Analytics 50,000 requests/day	Free
Google App Engine 5 Instances 150 instance hours 500 MB outgoing network traffic 500 MB file system storage	$7.79/mo
Google NDB Datastore 5GB stored data 100,000 read & 100,000 write operations	$1.02/mo
Total	$8.81/mo

4.5.2 Budget: Microsoft Technology Stack

For the Microsoft technology stack we plan to utilize mostly open source and free technologies. For application hosting we will use Amazon's EC2 dedicated hosting platform which charges for running instances only. We will be able to create a server instance an only pay for it while the server is running, keeping costs low. Table 6 outlines the costs for this strategy estimating 150 running instance hours.

Table 8. Budget: Microsoft Technology Stack

Google Analytics 50,000 requests/day	Free
Amazon Web Services EC2 Windows Server 2013 Large Instance	$0.329/hour (running instances only)
.NET MVC 4	Free
Redis Cache Server	Free (Open Source)
mongoDB	Free (Open Source)
Total (150 hours)	$49.35

4.5.3 Technology Stack Choice

After analyzing the two technology stack options outlined above, we have decided to utilize the Google App Engine stack. The App Engine stack offers tighter integration between different components of our framework, all the components mentioned above are integrated out of the box and built into the App Engine API. In contrast, the Microsoft technology stack would require installation and integration of the different open source components needed to develop our full solution. In addition to the extra integration efforts needed for the Microsoft stack, the costs of running the servers is also a factor in our

decision. Because the Microsoft stack requires a dedicated virtual server as opposed to a shared application hosting environment, the cost to run our solution would be significantly higher. The development efforts in terms of application logic for either stack would be comparable, as we have experience developing with each these technology stacks. Although both options would fit our needs, we believe that the App Engine stack would ease development and result in a better architected solution.

Chapter 5.
Real-World Application

As part of the development process of our system, we will apply the framework we have developed to the University of North Florida website (UNF, 2014). We have access to the source code of the UNF website and full access to the UNF Google Analytics account and data. We have also received approval from ITS to use this data for our research. We will produce an alternate version of the UNF website which utilizes our framework, and applies analytics based user interface changes to the site for the end user testing we will describe in chapter 6.

We plan to implement our framework on multiple pages of the UNF website. We plan to install our framework on the UNF Homepage, the Library Homepage, and Current Students page. We have chosen these three pages because they are some of the highest traffic pages on the site, and have many different navigation options that can overwhelm users. Our goal is to make these pages easier for users to navigate by surfacing the links most commonly used on each of these pages. The UNF homepage alone has over 70 links to other pages, we believe we can draw attention to the most important links on this page based on current user trends. Our ultimate goal is to offer users the navigation options they are looking for without having to use the search feature on the page. This chapter will be expanded upon as we apply our framework to the university website with details of our implementation.

Figure 8. UNF Website Homepage

Chapter 6.
Evaluation

As we mentioned in the previous chapter, we will be building an alternate version of the UNF website and testing the two versions of the site with different users. We will be measuring the time it takes users to complete actions, and gathering qualitative responses about their experience navigating the site. In this chapter we will outline our plan for evaluating the two designs.

6.1 Testing Process

Our testing process will use a simple A/B testing approach to evaluate the new design of the site alongside the original design. A/B testing is a popular method of evaluating alternate designs of a user interface. Normally A/B testing assigns random users to different versions of the same interface. Statistics about decision time, conversion rate, and user satisfaction, are compared between the two designs and a decision is made based on the success of one interface over another (Kohavi, Longbotham, Sommerfield, & Henne, 2009).

We will be following the same overall idea, but will be using a less programmatic, and more manual approach. Because we cannot place our alternate design on the live UNF website we will not be able to gather large amounts of statistics with automated A/B testing. We will instead be presenting the updated interface to users in person, and asking them to complete some simple tasks, such as: “navigate to the course registration page.” To some users, we will simply present the current live UNF website and ask them to perform the same tasks we asked the other users to complete as a control scenario. We will randomly assign the new version of the UNF homepage or the existing homepage to users. We will also ask them to complete a short survey about their experience with the new interface when the testing is complete. We will ask users to perform multiple different tasks on different interfaces to make up for our relatively small sample size.

Because users will become familiar with the version of the site they first use, we will not be able to ask the same user to perform the task on both versions of the site. We will however, show both versions of the site to the user at the end of the process so they can offer qualitative feedback about the two versions of the site side by side. We will also be able to mine some data from the Google Analytics reports on the existing live site to compare with our in-person testing.

6.2 Our Study Participants

We will be asking for volunteers for our testing process and not offering monetary compensation. We plan on asking mainly UNF students and staff to participate. Because users are volunteering their time, we plan to keep the testing process very short and only take up 5-10 minutes of the user's time. Our goal is to create a semi-automated testing process that will guide the users through a set of small tasks, and record the time along the way. At the end of the process, the users will be asked to provide feedback on the interface in the form of a 1 to 5 star rating system and a free text comment box. We will post an item about the study participation in the Osprey Student Update and send requests to a few faculty members within and outside the School of Computing to request the participation of their students in the study.

6.3 IRB Approval

Because our research will use human testing we have contacted the UNF Institutional Review Board (IRB) to ensure we are following their guidelines. The initial response we received seemed to indicate that we will not need full IRB approval to perform our research. We will however, submit a formal request to them and include their official response in our final draft.

Chapter 7.
Future Improvements

This chapter will be completed after we have finished development of the system.

Chapter 8.
Conclusion

This chapter will be completed after we have finished development and user testing of the system.

REFERENCES

Amazon. (2014). Amazon. Retrieved from http://www.amazon.com/

Amazon Web Services. (2014). Amazon EC2. Retrieved from http://aws.amazon.com/ec2/

Beasley, M. (2013). *Practical web analytics for user experience : How analytics can help you understand your users*. Amsterdam: Morgan Kaufmann, an imprint of Elsevier.

Bevan, N. (2005). Guidelines and standards for web usability. *Proceedings of HCI International 2005, Lawrence Erlbaum,* Las Vegas, Nevada.

Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C. & Orchard, D. (2004). Web services architecture - W3C working group note. Retrieved from http://www.w3.org/TR/ws-arch/

Büchner, A. G., & Mulvenna, M. D. (1998). Discovering internet marketing intelligence through online analytical web usage mining. *SIGMOD Rec., 27*(4), 54-61.

CoffeeScript. (2014). CoffeeScript. Retrieved from http://coffeescript.org/

Fasel, D., & Zumstein, D. (2009). A fuzzy data warehouse approach for web analytics. In M. Lytras, E. Damiani, J. Carroll, R. Tennyson, D. Avison, A. Naeve, . . . G. Vossen (Eds.), (pp. 276-285) Springer Berlin Heidelberg. doi:10.1007/978-3-642-04754-1_29

Google Analytics. (2014). Reporting developer guides. Retrieved from https://developers.google.com/analytics/devguides/reporting/

Google App Engine. (2014). Google app engine: Platform as A service. Retrieved from https://cloud.google.com/appengine/docs

Herring, M., & Prichard, J. (2012). The effect of web usability on user's web experience. *Proceedings for the Northeast Region Decision Sciences Institute (NEDSI),* , 207-215.

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Q., 28*(1), 75-105.

Inmon, W. H., Strauss, D., & Neushloss, G. (2010). *DW 2.0: The architecture for the next generation of data warehousing: The architecture for the next generation of data warehousing* Elsevier Science.

jQuery. (2014). jQuery. Retrieved from http://jquery.com/

Kohavi, R., Longbotham, R., Sommerfield, D., & Henne, R. (2009). Controlled experiments on the web: Survey and practical guide. *Data Mining and Knowledge Discovery, 18*(1), 140-181. doi:10.1007/s10618-008-0114-1

Kumari, G. V., Praneeth, P. 2., & Raju, V. P. (2014). An application of web usage mining framework for mining dynamic web sites. *International Journal of Advanced Research in Computer Science, 5*(2), 91-93.

Lai, L. T., Xu, Y., & Tan, F. B. (2009). Attributes of web site usability: A study of web users with the repertory grid technique. *International Journal of Electronic Commerce, 13*(4), 97-126. doi:10.2753/JEC1086-4415130405

Martin, R. C. (2003). *Agile software development: Principles, patterns, and practices* Prentice Hall PTR.

Mican, D., & Sitar-Taut, D. (2009). Preprocessing and Content/Navigational pages identification as premises for an extended web usage mining model development. *Informatica Economica, 13*(4), 168-179.

Microsoft ASP.NET. (2014). Learn about ASP.NET web API. Retrieved from http://www.asp.net/web-api

Mobasher, B., Cooley, R., & Srivastava, J. (2000). Automatic personalization based on web usage mining. *Commun.ACM, 43*(8), 142-151.

mongoDB. (2014). mongoDB. Retrieved from http://www.mongodb.org/

Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems, 24*(3), 45-77.

Phippen, A., Sheppard, L., & Furnell, S. (2004). A practical evaluation of web analytics. *Internet Research, 14*(4), 284-293. doi:10.1108/10662240410555306

Pokorny, J. (2013). NoSQL databases: A step to database scalability in web environment. *International Journal of Web Information Systems, 9*(1), 69-82.

Prom, C. (2011). Using web analytics to improve online access to archival resources. *American Archivist, 74*(1), 158-184.

Redis. (2014). Redis. Retrieved from http://redis.io/

U.S. Dept. of Health and Human Services. (2006). The research-based web design & usability guidelines. Retrieved from http://guidelines.usability.gov/

UNF. (2014). University of north florida. Retrieved from http://www.unf.edu

Vaishnavi, V., & Kuechler, B. (2013). Design science research in information systems. Retrieved from http://desrist.org/desrist/

W3C DOM Interest Group. (2005). Document object model (DOM). Retrieved from http://www.w3.org/DOM/

W3Techs. (2011). Usage statistics and market share of google analytics for websites. Retrieved from http://w3techs.com/technologies/details/ta-googleanalytics/all/all

Webster, J., & Ahuja, J. S. (2006). Enhancing the design of web navigation systems: The influence of user disorientation on engagement and performance. *MIS Quarterly, 30*(3), 661-678.

Weischedel, B., & Huizingh, E. K. R. E. (2006). *Website optimization with web metrics: A case study*. Fredericton, New Brunswick, Canada: ACM.