Självständigt arbete på grundnivå

Independent degree project - first cycle

Datateknik
Computer Engineering

Customized Analytics Software
Investigating efficient development of an application

Tomas Altskog
MID SWEDEN UNIVERSITY
Department of Information and Communication Systems (IKS)

Examiner: Ulf Jennehag, Ulf.Jennehag@miun.se
Supervisor: Jimmy Åhlander, Jimmy.Ahlander@miun.se
Author: Tomas Altskog, toal1201@student.miun.se
Degree programme: Master of Science in Engineering – Computer Engineering, 300 credits
Main field of study: Computer Science
Semester, year: Spring, 2016
Abstract

Google Analytics is the most widely used web traffic analytics program in the world with a wide array of functionality which serve several different purposes for its users. However the cost of training employees in the usage of Google Analytics can be expensive and time consuming due to the generality of the software. The purpose of this thesis is to explore an alternative solution to having employees learn the default Google Analytics interface and thus possibly reducing training expenses. A prototype written in the Java programming language is developed which implements the MVC and facade software patterns for the purpose of making the development process more efficient. It contains a feature for retrieving custom reports from Google Analytics using Google’s Core Reporting API in addition to two web pages are integrated into the prototype using the Google Embed API. In the result the prototype is used along with the software estimation method COCOMO to make an estimation of the amount of effort required to develop a similar program. This is done by counting the prototype’s source lines of code manually, following the guidelines given by the COCOMO manual, and then implementing the result in the COCOMO estimation formula. The count of lines of code for the entire prototype is 567 and the count which considers reused code is 466. The value retrieved from the formula is 1.61±0.14 person months for the estimation of the entire program and 1.31±0.16 for a program with reused code. The conclusion of the thesis is that the result from the estimation has several weaknesses and further research is necessary in order to improve the accuracy of the result.

Keywords: Google Analytics, MVC, Facade, Java, COCOMO, SPL
4 Implementation

4.1 Tools
4.1.1 Java
4.1.2 Google Analytics

4.2 Prototype
4.2.1 GUI
4.2.2 MVC
4.2.3 Facade pattern
4.2.4 Project Template

4.3 Software Estimation
4.3.1 Size

5 Results

5.1 Size
5.2 Estimation

6 Discussion

6.1 Prototype
6.2 Conclusions
6.3 Further research
6.4 Ethical Aspects

References

Appendix A: COCOMO II 2000
Calibrated Post-Architecture Model Values
# Terminology

## Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>FXML</td>
<td>JavaFX Extensible Markup Language</td>
</tr>
<tr>
<td>MVC</td>
<td>Model-view-controller</td>
</tr>
<tr>
<td>COCOMO</td>
<td>Constructive Cost Model</td>
</tr>
<tr>
<td>SPL</td>
<td>Software Product Line</td>
</tr>
<tr>
<td>SLOC</td>
<td>Source Lines of Code</td>
</tr>
<tr>
<td>KSLOC</td>
<td>One thousand Source Lines of Code</td>
</tr>
<tr>
<td>FP</td>
<td>Function Point</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
</tbody>
</table>
1 Introduction

This introductory chapter describes the motivation for the necessity of the research done in this thesis and how the problems presented are intended to be solved.

1.1 Background and problem motivation

A lot of companies utilize Google Analytics to analyze their web traffic. In order to effectively utilize this data the companies' employees usually participate in a course with the aims to teach them the various capabilities of the software. These courses can be expensive and depending on the work requirements of the employees it can be difficult to teach them all they need to know within the scope of one course. Additionally a lot of the employees who are required to take these courses might not be familiar with IT in general and might find advanced software such as Google Analytics to be confusing.

Google Analytics is a general purpose analytics program and thus it contains a lot of different features with varying degrees of relevance depending on the user's needs and proficiency with the program. Therefore users who don't have advanced knowledge and experience with the software or IT in general might not be able to make use of it efficiently for the specific tasks they require. Furthermore the courses employees might be undergoing can be expensive and time consuming, a problem which would be exacerbated if the specific job has a high employee turnover.

1.2 Overall aim

The purpose of this thesis is to investigate an alternative method for allowing a company and its employees to effectively utilize the varying functions of Google Analytics. The proposed solution is to create a bespoke software whose design is specified by the user and displays information from Google Analytics relevant to the specific task of the employee. Because of this customization it might be possible to create a system which does not require extensive education and resources to be employed. However the design and development of this customized software could also be expensive. The goal of this thesis is to investigate how the development process of this software can be streamlined and thus more cost-effective.

1.3 Scope

The scope of this thesis will focus on creating a prototype software program using various methodologies related to software product line development. This prototype should allow for an estimation to be made of the amount of work and resources required to deliver a customized software product. This estimation could then be used to make a comparison with the cost of the traditional method of educating employees. However a detailed comparison between the two meth-
ods is beyond the scope of this thesis and will be left open for other researchers in the future.

1.4 **Concrete and verifiable goals**

These are the specific goals that the prototype should fulfill.

- Prototype written in Java implementing the MVC pattern.
- NetBeans Project template with folder structure similar to the prototype.
- Facade pattern implemented as Java class to simplify the use of Google APIs.
- The prototype should have the functionality to retrieve a custom report from Analytics with chosen metrics and dimensions.

The prototype will then be used to create an estimate of the amount of work hours required to reproduce the software program. This estimation will be done by counting the number of lines of code from the prototype and then applying that to the estimation model COCOMO II.

1.5 **Outline**

This chapter provides an introduction to the thesis by explaining the background and motivation for the work being done along with the purpose and goals of the research. Chapter 2 contains the theoretical information relevant to the thesis. Chapter 3 describes the methods chosen for completing the work of this thesis, such as programming languages, development environments and estimation models that were used. Chapter 4 highlights the important technical design decisions that were made in the thesis, specifically how the prototype was designed and implemented with the Google APIs. Chapter 5 presents the results of the software size and estimation using the estimation model COCOMO. This result is then analyzed in chapter 6 and from which conclusions are drawn, furthermore a recommendation for future work is given and a discussion on the ethical aspects of the work is given. After chapter 6 the works sources and appendices can be found.
2 Theory

2.1 Google Analytics

Google Analytics was launched in November 2005 by Google Inc after they had acquired the company Urchin whose product was the precursor to Analytics [1]. Analytics is a tool for monitoring web traffic with a wide range of functions that allow the user to access specific information about how their website is being used. It is the most widely used analytics program in the world with a market share of 82.8% as of January 1, 2016 [2]. Analytics is free for users with less than 10 million data hits per month, these data hits include the combined volume from pageviews, screenviews, events, and transactions [3][4]. If this limit is surpassed the user has to either throttle the amount of data they register or upgrade to the Google Analytics 360 package. [5]

Each web page with Analytics monitoring it has the Google Analytics Tracking Code (GATC) implemented in the form of either the analytics.js JavaScript file or the legacy ga.js file [6]. This script contains all the functions that Analytics needs to gather the information that the user wants to track and send to the Google servers using HTTP requests [7]. The script also sets cookies on the client side for distinguishing between unique users and throttling request rates [8]. The data flow of this is illustrated in figure 1.

![Google Analytics Data Flow](image-url)
2.1.1 Custom Reports

A large part of Analytics functionality is centered around its custom reporting tools that allow the user to decide which information they are tracking on their web pages a report should contain. These custom reports are built using the dimensions and metrics stored by the Analytics database. Metrics are the measurements of user activity such as Sessions and Pageviews while dimensions break down these metrics across common criteria. As an example for the Sessions metric the user can sort the data using a dimension metric such as City which would result in a table where the user can read how many sessions any particular city has accrued. The user can define multiple metrics and dimensions for their custom reports as shown in the example in table 1. [10]

<table>
<thead>
<tr>
<th>DIMENSION</th>
<th>DIMENSION</th>
<th>METRIC</th>
<th>METRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>Browser</td>
<td>Sessions</td>
<td>Pages/Session</td>
</tr>
<tr>
<td>Stockholm</td>
<td>Microsoft Edge</td>
<td>12000</td>
<td>4.2</td>
</tr>
<tr>
<td>Sundsvall</td>
<td>Chrome</td>
<td>3500</td>
<td>2.7</td>
</tr>
<tr>
<td>Sundsvall</td>
<td>Firefox</td>
<td>500</td>
<td>2.0</td>
</tr>
<tr>
<td>Sundsvall</td>
<td>Microsoft Edge</td>
<td>250</td>
<td>3.1</td>
</tr>
<tr>
<td>Göteborg</td>
<td>Safari</td>
<td>7000</td>
<td>5.3</td>
</tr>
</tbody>
</table>

Not all metrics and dimensions make sense to pair together since their individual scopes may differ. The various scopes any metric or dimension may belong to are users, sessions, or actions. For example the scope of the Sessions metric is sessions, which is not the same as the scope of the Page dimension which is actions, and pairing the two is therefore not possible. [11]

The user can also utilize custom dimensions and metrics to report data that is relevant to their systems and organization. This data can come from specific content of the web page that is not available and/or applicable on all web pages that utilize Analytics and thus not measured by default by the Analytics tool. For example a news reporting website might want to track the authors of articles as a dimension in order to measure their individual impact on specific metrics. The data for custom dimensions and metrics can also come from external sources such as in-store purchases or visits and will then be available for analysis alongside online measurements. [12]

Aside from table-type reports Analytics also can create Motion Charts that allow the user to plot specific dimension values against specified metrics and then graphically see how these relationships have varied over time. [13]
2.1.2 Events and Goals

Events is the way Analytics tracks individual user interactions with a web page that don’t generate an entire pageview. Interaction with flash elements, ad clicks, gadgets, videos and more will trigger an Event that will be stored by Analytics. Each Event consist of four components: Category, Action, Label, and Value. The Category component categorizes different elements on the website under a shared name such as “Article”. By combining the Category with an Action component the user can track specific actions that are linked to that Category such as “Shared to Facebook”. The Label and Value components are used to store optional information such as an article’s name and reading time. The full example Event can be seen in table 2. [14]

<table>
<thead>
<tr>
<th>Category</th>
<th>“Article”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action</td>
<td>“Shared to Facebook”</td>
</tr>
<tr>
<td>Label</td>
<td>“Article name”</td>
</tr>
<tr>
<td>Value</td>
<td>Reading time</td>
</tr>
</tbody>
</table>

Goals are a way to measure certain activities on a website that contribute to the overall goal of the system, for example one of the goals of an online retailer is to get visitors to purchase their products. Typically goals are linked to visitor conversions which is when a visitor does something valuable to the organization. Through Analytics the user can define goals for their website which allow them to track how their clients are reaching that specific goal. There are five types of goals: Destination, Duration, Pages/Screens per session, Event, and Smart. A Destination goal is when a specific location on the website loads. A Duration goal is a desired duration for one session. Pages/Screens per session goals are the desired amount of pageviews per user session. Event goals are when specified Events are achieved and Smart goals that through machine learning find the most likely conversion path and create goals accordingly. [15]

Another relevant feature is the Intelligence Events which automatically alert the Analytics user to high statistical variations in their data or the user can specify custom conditions for when they should be alerted. [16]

2.1.3 Individual user permissions

Analytics allows an admin to assign and remove permissions to specific users within the system. The permissions exist on three different levels: Account, Property, or View. For each of these levels a single user can be given the following permissions: Manage Users, Edit, Collaborate, Read and Analyze. [17]

2.1.4 Real-time analytics

Real-time data tracking allows the Analytics user to monitor their visitor data live. By monitoring specific metrics and dimensions live the user can see the effect of marketing campaigns and other events that might affect their website
Customized Analytics Software – Investigating efficient development of an application
Tomas Altskog

visitor patterns. By utilizing the Real Time Reporting API the user can also create live updated content on their website such as the amount of users currently browsing a specific page or how many products are available in inventory. [18]

2.1.5 Flow Visualization

Flow Visualization is a tool that allows the Analytics user to graphically view which paths their visitors follow throughout the website. The Users Flow report visualizes how users from different sources, or other dimensions, navigate through the website’s pages and the relative user traffic volumes for each page. An example of a User Flow report is shown in figure 2. The Behavior Flow report shows how the users engage in page content and the Goal Flow report shows which path the users took to a specific goal. [19]

![User Flow report example](image)

Figure 2: User Flow report example. [20]

2.1.6 Education and Training

There are a variety of ways to learn how to use Analytics for employees and private individuals alike. Many free resources are available online that allow users to manage the education and learning process to their own discretion [21]. For companies the process to integrating Analytics might vary. A company with an IT department might have the resources to educate in-house talent or recruit individuals with the required skills. Other companies might hire consulting firms that specialize in digital analytics to integrate Analytics into the organizational tool set. Courses for managers and IT professionals alike vary in price range from roughly 600 to 1300 SEK an hour and depending on the extent of education required could amount to a lot of resources spent. [22]
2.1.7 **Developer content**

For developers who are developing third-party programs and want to utilize Analytics data Google has created the Core Reporting API. This API supports the programming languages Java, Python, PHP, JavaScript, and Objective-C as well as the .NET framework. The API provides the various functions required to query the Google servers for the Analytics data which then can be manipulated within the third-party applications source code. Figure 3 shows how the Core Reporting API (Export API) relates to the other standard interfaces and Google’s database. [10]

Additionally the Analytics Embed API can be used to develop web content using the JavaScript programming language. By adding the following code to a web page the API’s various functions become available on the web page. [24]

```javascript
(function(w,d,s,g,js,fs){
g=w.gapi||(w.gapi={});g.analytics={q:[],ready:function(f){this.q.push(f);}};
j=d.createElement(s);fs=d.getElementsByTagName(s)[0];
js.src='https://apis.google.com/js/platform.js';
fs.parentNode.insertBefore(js,fs);js.onload=function(){g.load('analytics');};
}(window,document,'script'));
```
Google APIs use the OAuth 2.0 protocol to authorize and authenticate the applications using them. By creating credentials through the Google Developer Console a developer is given access to the APIs that are registered for the given credential. [25]

2.1.8 Privacy
User privacy is a large concern when implementing Analytics. Both for the Analytics admin as well as the website’s users whose browsing experience is being tracked by the software. The main concern for the admin is that Google could potentially utilize their organization’s user data for Google’s own purposes. The purpose of Analytics is to allow organizations to access information on how their users are accessing and utilizing the organization’s website. If Google were then to hijack this relationship by also processing this data they would be potentially gaining vast amounts of information of companies financial and operational details. According to Google the account admins own their Analytics data and Google only accesses information that the admin has given them permission to. The data is regarded as confidential and subject to the confidentiality provisions of Google’s Privacy Policy and employee access is limited to need-only basis. [26][27]

An Analytics admin can choose which type of data Google is allowed to utilize for specific services. The following options can be enabled or disabled by the admin.

- Google products & services
- Benchmarking
- Technical support
- Account specialists

The Google products & services option decides whether Google is allowed to access and analyze data to improve their own products and services. The Benchmarking option allows the admin to compare their own organization’s data with aggregated industry data from other companies who also have enabled this option. Enabling this option will also share the organization’s data to this aggregated pool though all of the shared data will be anonymous. The Technical support option allows Google support employees to access the data to provide for troubleshooting of technical problems. The Account specialists option allows all Google specialists within sales and marketing to access the data so that they can offer guidance and optimization suggestions for the organization. [28]

Aside from protecting their own data, admins may also choose to protect their visitor’s data as well. A website owner may give the visitor’s the choice to opt-out of Analytics tracking entirely by giving them the option to set the following property to true in the website’s code:

```javascript
window['ga-disable-UA-XXXXX-Y'] = true;
```
Where the value “UA-XXXXXX-Y” is to be replaced by the accounts property ID. [29] 

The admin can also protect their user’s IP address by enabling the following code:

```javascript
ga('set', 'anonymizeIp', true);
```

This sets the last octet of the user IP address to zero as soon as technically feasible at the earliest possible stage of the collection network as seen in figure 4. [30]

![Diagram](image-url)

**Figure 4:** Illustration of the anonymizeIP function [30]

### 2.2 Software Patterns

#### 2.2.1 Model-view-controller

Model-view-controller (MVC) is a software architectural pattern that can be utilized to structure graphical user interface (GUI) source files. The pattern divides the software into three separate parts.

- **Model**
- **View**
- **Controller**

The model contains the data of the system and the algorithms that manipulate that data. It is often structurally similar to the abstract process of the system which simplifies the complexity of understanding how the concrete process operates in the software. The view is responsible for displaying the data in the model to the user and accurately represent any changes in the data. The controller is the intermediary between the model and view components and it manipulates both the data in the model and the visual representation of the view. [31]

The MVC pattern simplifies the development of GUI elements and the data manipulation controls by separating both from the existing data structure. In the first published article describing the pattern the authors Krasner and Pope state that:

> When building interactive applications, as with other programs, modularity of components has enormous benefits. Isolating functional units from each other as much as possible makes it easier for the application designer to understand and modify each particular unit, without having to know everything about the other units. [31]
2.2.2 Facade pattern
The facade pattern is part of the group of patterns described in the book Elements of Reusable Object-Oriented Software whose authors are commonly referred to as the Gang of Four (GoF). The purpose of the pattern is to provide a high level interface to the underlying systems in order to simplify their implementation.[32] This facade interface promotes loose coupling which is a common principle in software development which can vaguely be described as how strongly connected different subsystems are connected to each other. [33]

The empirical evidence for the effectiveness of the GoF patterns is not sufficient to make a strong case for the use of any specific pattern. Therefore deciding which patterns to use is up to the developer’s discretion. In a systematic literature review of design patterns published in the IEEE Transactions on Software Engineering (TSE) called “What Do We Know about the Effectiveness of Software Design Patterns?” the authors in the conclusion write: “Our study indicates that we are currently far from having the necessary degree of knowledge for making evidence-based judgments about when to employ individual patterns.”. [34]

2.3 NetBeans Project Template
The NetBeans integrated development environment (IDE) has a feature to create customized project templates. This can help developers reduce the amount of time and resources required to create new products that are similar to previously completed projects. This is done by defining the structure of the project and creating the basic source code files necessary for the type of software that is being developed. [35]

2.4 Software Product Line
A Software Product Line (SPL) is a system that incorporates various parts of the production process and whose main feature is to define the procedure for developing a specific type of software product. By following the predefined production plan for the system and utilizing a core set of assets developed to aid this process a form of assembly line can be created for software products that share similar feature specifications. While software patterns can aid the software production process on a compartmental level the SPL methodology structures the development process on a larger scale. [37]

According to the Software Engineering Institute (SEI) at Carnegie Mellon University the reported benefits that have been attained from implementing an SPL are: [37]

- Improved productivity by as much as 10x
- Increased quality by as much as 10x
- Decreased cost by as much as 60%
- Decreased labor needs by as much as 87%
Customized Analytics Software –
Investigating efficient development of an application
Tomas Altskog
2016-06-13

• Decreased time to market (to field, to launch) by as much as 98%
• Ability to move into new markets in months, not years

Various case studies on the implementation and efficacy of SPLs have been conducted by the SEI and can be found at their website. [38]

2.5 Software Estimation

There are several methods for estimating the amount of development hours required to complete a software project. In the following sub-chapters a few of these methods are explained.

2.5.1 Source Lines of Code

The Source Lines of Code (SLOC) measurement is a size metric of a software product defined as how many lines of code the system is built of. Since the true amount of SLOC needed is unknown until the completion of a project this measurement can not by itself be used to accurately predict the amount of work hours required for a specific project. A project might have an estimate of the amount of SLOC required which then combined with some cost estimation model can be used to give a rough estimate of the development time. One of the difficulties with measuring SLOC and comparing the data to other projects is the lack of consensus on a SLOC counting standard. This is further complicated when comparing measured data between projects written in different programming languages. [39]

2.5.2 Function point

One way of approximating the size of a program is to calculate Function Points. (FP). A FP analysis includes identifying the user requirements of the system and categorizing them into five different types: Inputs, Outputs, Inquiries, Internal Files, and External Files. These types are then given approximated degrees of complexity for the specific system and using predefined weights for said types a rating of the amount of FPs can be given. [40]

Since the FP analysis can be done before the project is completed an estimate of the amount of SLOC required can be given by using a ratio for the amount of SLOC expected to be used per FP. For JAVA projects the mean and median amount of SLOC per FP is 53, according to the QSM Software Almanac: 2014 Research Edition published by QSM, Inc. Their SLOC measurement counts the amount of new and modified amounts of lines of code. [41]

2.5.3 COCOMO

The Constructive Cost Model (COCOMO) was originally developed in 1981 by Barry Boehm and published in the Software Engineering Economics journal. The original model (known as COCOMO 81) has been subsequently updated over the years and the COCOMO II model was released in the year 2000. The function for describing the amount of development hours is shown in the following formula:
where PM is the amount of effort in person-months (In COCOMO, one PM equals 152 hours of working time); A which is a constant that approximates productivity; S is the size of the project in kilo-SLOC (KSLOC); E is the scale factor as an exponent, and the EM product function is the product of the individual effort multipliers. [42]

The scale factor E can vary as shown by the formula:

\[ E = B + 0.01 \sum_{j=1}^{5} SF_j \]  

where B is a constant that can be calibrated but by default set to 0.91 according to the COCOMO II calibration and SF is the five different scale factors. The possible values for the SF variables are given by table 62 in the COCOMO II Model Definition Manual and can be seen in appendix A. [42]

According to the COCOMO II Model Definition Manual the average large project’s effort multipliers are equal to 1.0 and E is equal to 1.15 (1.0 for small projects). For COCOMO II the constant A is set to 2.94 prior to calibration according to the local development environment. [42]
3 Methodology

The following chapters describe the methodology of this thesis and the various tools and methods being used. The expected prior knowledge of the reader is a basic understanding of programming and software engineering, particularly within the context of the Java programming language. The development of this project is performed on the Windows 10 platform and thus will use the corresponding tool versions.

3.1 Tools

3.1.1 Java

The chosen programming language for this project is Java. The main reasons for this is that it is well supported by the Google API libraries and also a common development language for application software. More specifically the Java Standard Edition 8 will be used as it is the latest supported version at the time of writing.

Alternative programming languages that could be considered are Python, PHP, and JavaScript since they are all supported by the Google APIs.

3.1.2 NetBeans

The NetBeans 8.1 IDE will be used for the development of the prototype because it is a free, open-source platform with built-in support for Java. A customized NetBeans project template will be created that conforms to the proposed structural design of the software.

3.1.3 Google Analytics

The work of this thesis requires a Google Analytics account in order for the prototype to be developed and tested. For the purpose of enabling the prototype to utilize data from Analytics an account which is connected to a website will be created. The website will be a locally hosted temporary service containing data suitable for testing purposes and created solely for the purpose of testing the prototype.

The prototype will be developed using the Google Analytics Core Reporting API for retrieving and reading the data from Analytics. An extensible markup language (XML) parser could be used for this purpose as well but since the Core Reporting API supports Java it is deemed to be the simplest solution.

3.2 Literature study

Prior to the development of the prototype a literature study was done in order to find information on the subjects described in chapter 2. The sources gathered can be divided into four separate types; one type for the required background information on the subject; another for the technical details required for the de-
development of the prototype; the third type for the theory and methods behind software development and design, and the last for software estimation models and theory.

The reliability of the background information and technical sources can be considered high since most of them are related to Google Analytics and the sources are from Google’s own websites. Although it might be important to confirm technical details in the future since they might become deprecated. The sources for the theory and methods for software development are taken from the original articles describing said theories and methods. This validates the description given for these entities though it does not validate the efficiency of them. Lastly the sources for software estimation models, specifically COCOMO, are from credible and/or official sources.

3.3 Prototype

The prototype will be a Java application displaying information taken from Google Analytics in the form of custom reports. The focus of the prototype will not be to develop GUI elements for all different kinds of reports and visualizations in Analytics, but rather to use an example from which one could extrapolate an assessment of how much work is required to implement any GUI element. The prototype will be structured according to the MVC and facade patterns described in the following sub-chapters; in theory this will allow for a loosely coupled program where implementing another GUI element is a simple task of creating the interface and using the facade to connect it with Analytics.

The GUI will display a custom report taken from the test data in the form of tables. A feature for choosing which dimensions and metrics to build a report of will be available in the GUI as well as additional functionality such as choosing the dates the report should gather data from.

The Java GUI library JavaFX will be used to create the interface elements of the prototype. The reason for choosing JavaFX instead of other libraries such as Swing or AWT is that JavaFX is a more modern and up to date library. The GUI files will be programmed in the JavaFX Extensible Markup Language (FXML) since this further helps achieve the goal of separation of the application logic and interface.

3.3.1 MVC

The MVC pattern will be followed when constructing the prototype in order to separate the GUI components from the internal functionality which implements the Analytics API. The motivation for doing this is that whenever a new GUI element needs to be implemented or a completely new application is to be delivered the internal functionality can be reused. This should lead to a shorter deployment cycle for new products since less programming will be required.

The MVC pattern will be implemented following the structure seen in figure 5. The views will be constructed using the Java libraries mentioned in the previous chapter; these will then be connected to a controller that mediates the interac-
tion between them and the model. The model will handle the data received from Analytics using the Google API as well as other internal logic functions.

![MVC pattern example](image)

Figure 5: MVC pattern example [43]

### 3.3.2 Facade pattern

The facade pattern will be used for creating an interface to the Google APIs in order to simplify the development of new features as well as applications. By hiding the implementation details of the API and just providing functions through the facade that the developer needs access to, the development of new applications should become less complex and require fewer lines of code. The facade's internal logic could also be changed or extended without needing to change the implementation on the clients side.

The facade will be utilized by the model described in the previous chapter since the model will be the component interacting with the Analytics data. This should simplify the implementation of the model by making it more readable and have fewer lines of code. The facade could then also be reused for future projects.

### 3.3.3 Project Template

A project template in the NetBeans IDE will be created in order to allow for the quick setup for the development of new applications. Since the applications will be following the MVC and facade patterns, the project template could facilitate the implementation of these patterns by determining the structure of the source files. This could promote consistency of design across several applications and help the implementation of an SPL for the development of similar applications.

### 3.4 SPL

Even though this thesis will not be producing more than one prototype, the concept of SPLs can be relevant when trying to determine how efficient mass production could become. The patterns implemented are not specifically chosen for simplifying the development of the prototype but to aid in the potential de-
development of several products of similar nature. By examining the complexity and size of the prototype and extrapolating the results to a theoretical SPL implementation, a rough estimation could be given for how efficient the production of these applications is.

The reliability of this estimation is unfortunately not certain and the recommended way of determining the benefits of an SPL for the implementation of customized Analytics applications would be to conduct a large case study where an SPL is utilized to create several applications. For the purposes of this thesis though, the estimation will be sufficient to determine a range of possibilities for the complexity and efficiency of the development process.

3.5 Software Estimation

3.5.1 Size

The size metric will be measured using SLOC where the counting rules used follow the guidelines given by the COCOMO II Model Definition Manual [42] in table 64. The main reason for using SLOC instead of FPs is that SLOC is an actual measurement while FPs is just a size estimation. SLOCs can be difficult to use for comparisons between different programming languages and applications since it is subject to the languages’ syntax and can’t measure the algorithmic and structural complexity of the software. For this thesis however it is deemed a sufficient measurement for estimating the amount of effort it would require to develop similar applications. Specifically since they would be developed using Java and implementing the same source code structure as the prototype.

There will be two SLOC measurements taken from the prototype; the first will be counting lines from all of the source code files that are written for the prototype; the other will only be counting lines that are expected to be rewritten for future applications. This means that the first measurement will be used for estimating how long it could take to deploy a system similar to the prototype without copying any code from the prototype while the second measurement will be used to estimate how long it could take when reusing code from the prototype. This means that the second measurement will only count the lines that need to be edited or written for the new application. The code that can be copied over from the prototype such as the facade will not be considered.

Another reason for not using FPs as a size metric is that it is a more time consuming process which could require a lot of effort if it were to be made every time a new application is developed. The process of counting SLOCs could be automated for future applications and these measurements could then be utilized when trying to estimate how long it would take to develop similar programs. By using knowledge gained from past projects the estimations could be improved as described in the next chapter.
3.5.2 Estimation model

The estimation model used for estimating the amount of development hours required will be COCOMO II. The reasoning behind choosing this model instead of some other estimation method is that it is relatively simple to use depending on how much the parameters are calibrated. It is also a quite well known method which has existed for a long time and is widely used within the industry.

The possible ranges of the parameters for the productivity constant, scale factor, and effort multipliers in equation (1) will be calculated in order to allow for a minimum and maximum estimation. These estimations will enable a rough prediction of how long it could take to develop the prototype or a new application. This range of values could be narrowed down by examining how future projects implementing an SPL and additional software engineering methods could affect the parameters.

As discussed in the previous chapter there will be two different estimations made; one for reimplementing the entire system and another that would reuse code from the prototype. These estimations will have different size values applied in equation (1) but the other parameters could also be affected since the two projects would have differing characteristics. While this does not change how the calculations are made since the entire range of possibilities for the parameters is being used it does mean that the values could be calibrated differently in the future.

By doing comparisons with similar projects the estimation for a specific type of project could be refined and improved according to past experience. The estimation of how much effort it would require to create a new application while reusing code from past projects could be suitable for this adaptation. However this will not be done within the scope of this thesis since only a single prototype will be developed.
4 Implementation

4.1 Tools

4.1.1 Java

The Java libraries used for the NetBeans development of the prototype are: JDK 1.8, Java EE 7, Google Analytics API v3, and Jetty 9.3.7. For the web development HTML and JavaScript are used to create the website that is being tracked by Analytics as well as web pages that are integrated into the NetBeans application.

4.1.2 Google Analytics

A free Google Analytics account is used to track and store data from the website created for testing purposes. The website is hosted on the local host using an Apache HTTP server with the following JavaScript snippet in order to send data to Analytics:

```javascript
(function(i,s,o,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
}(window,document,'script','https://www.google-analytics.com/analytics.js','ga'));
```

After the above function the following code is used to send the specified data when a user visits the web page:

```javascript
ga('create', 'UA-XXXXX-Y', 'none');
ga('send', 'pageview');
```

where ‘UA-XXXXX-Y’ is replaced by the property ID of the website as found on the Analytics admin page. The parameter ‘none’ specifies that no cookie domain is being used which is the case when hosting a website on localhost. The second line of code sends a pageview data-hit to the Analytics property.

In order to utilize the Google API services for the application an authentication and authorization is made using the OAuth 2.0 protocol. Two types of credentials from the Google Developer Console are used; the first to authorize the NetBeans application and the second to authorize the web development. A client secret file in JavaScript Object Notation (JSON) format is stored locally and accessed by the NetBeans application using the following code:

```javascript
GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(
  JSON_FACTORY, new InputStreamReader(
    AnalyticsFacade.class.getResourceAsStream("client_secret.json")));
```
The web pages authorize using the Google Embed API with the following JavaScript code:

```javascript

gapi.analytics.auth.authorize({
  container: 'embed-api-auth-container',
  clientid: 'XXXXXXXX'
});
```

where the clientid parameter is retrieved from the credentials created in the Google Developer Console.

### 4.2 Prototype

The prototype is developed in the NetBeans IDE as a JavaFX application and the source code structure follows a MVC pattern. The GUI functionality is controlled by an FXML controller class that manages and responds to user input and program logic. A facade interface is used to connect the Google Core Reporting API to the application model.

#### 4.2.1 GUI

The user interface is created using the JavaFX GUI library and a FXML file which contains the code that builds the GUI elements. The base container for the GUI elements is a tab panel as shown in figure 6, where each tab shows data from Analytics in different formats.

![Figure 6: GUI Tab panel](image)
The Custom Report tab as shown in figure 7 is built using two ComboBoxes, two DatePickers, two ListViews and a button along with labels headers for each element.

![Custom Report tab](image)

Using the ComboBoxes the user can choose a Metric and Dimension that they wish to construct a custom report from. The DatePickers’ values determines during what time period the data retrieved will be from and the button executes the logic that retrieves the data and displays it in the lists.

The Overview tab as seen in figure 8 contains a WebView container which allows for the embedding of a web page inside the application using the following code:

```java
WebEngine overviewEngine = overviewWebView.getEngine();
overviewEngine.load("Website URL here");
```

The web page embedded in the Overview tab is built using JavaScript with the Google Embed API as well as a widget contained inside an iframe.
Similarly the Realtime data tab also utilizes a WebView container to display a website that contains widgets inside an iframe as seen in figure 9.

Figure 8: Overview tab

Figure 9: Realtime data tab
4.2.2 **MVC**

The prototype is developed according to the MVC pattern where the source code is separated into three different components; Model, View, and Controller. The project structure is shown in figure 10 along with the source code files associated with the different components.

![File structure diagram](image)

Figure 10: File structure

The AnalyticsView file contains the programming code that describes the prototype’s GUI. The code is written in the FXML language, which is similar to XML, instead of regular Java. A GUI structure is described in the format as seen in figure 11 where the `<children>` tag contains the interface elements of the container.

```xml
<Tab fx:id="tabPanel1" text="Custom report">
  <content>
    <AnchorPane prefHeight="180.0" prefWidth="200.0">
      <children>
        ...
      </children>
    </AnchorPane>
  </content>
</Tab>
```

Figure 11: FXML GUI container

An example of an interface element is the dimensions ComboBox whose code can be seen in figure 12.

```xml
<ComboBox fx:id="dimensionComboBox" layoutX="34.0" layoutY="160.0"
  items>
  <FXCollections fx:factory="observableArrayList">
    <String fx:value="Continent" />
    <String fx:value="Country" />
    <String fx:value="City" />
  </FXCollections>
</ComboBox>
```

Figure 12: FXML GUI element
The AnalyticsController file contains a Java class that acts as an intermediary between the view and model. The main purpose of this class is to update the view- and models data as well as define the various event handlers required for interfacing with the system. Specifically for the Custom report tab the controller implements the ‘Get Report’ button’s event handler. This handler then uses the class’ private instance of the model to retrieve a report using the method ‘getTable’.

The model is described by the AnalyticsModel file which is a Java class that maintains the internal data of the system and utilizes the facade (described in the next chapter) to retrieve Analytics data. The ‘getTable’ method that the controller accesses in its event handler is written as follows:

```java
public List<List<String>> getTable(String startDate, String endDate, String metric, String dim) {
    table = facade.getTable(startDate, endDate, metric, dim);
    this.fillRows();
    return table;
}
```

### 4.2.3 Facade pattern

The facade pattern is implemented using the AnalyticsFacade Java class. This class contains the methods for retrieving data from Analytics using the Google Core Reporting API v3. The facade also manages the authorization of the application with Analytics using the client secret. The Core Reporting API method for retrieving a custom report in the prototype is the following:

```java
analytics.data().ga().get(tableId, startDate, endDate, pre + metric)
    .setDimensions(pre + dim)
    .execute();
```

where `tableId` is the report profile’s ID retrieved from Analytics; `startDate` and `endDate` are the reports time parameters; `metric` and `dim` are the metrics and dimensions chosen to construct the report. The `pre` variable is a string used as a prefix which contains “ga:” which is necessary for the method’s format.
4.2.4 **Project Template**

The project template is created using a NetBeans module and creating a project sample from a project with the desired configurations. By deploying the module and creating a new project, choosing NetBeans Modules under categories, then selecting the Analytics Template sample as seen in figure 13; a preconfigured project is created. This new project is structured similarly to the structure seen in figure 5.

![Creating new project using template.](image)

**Figure 13:** Creating new project using template.

4.3 **Software Estimation**

4.3.1 **Size**

The measurement of the size metric of the software is counted using SLOC and the standard mentioned in chapter 3.5.1. The code is counted manually while following the guidelines and a total amount of SLOC for each source code file is retrieved. The source code files that contribute to the SLOC count are:

- AnalyticsFacade.java
- AnalyticsController.java
- AnalyticsModel.java
- AnalyticsPrototype.java
- AnalyticsView.fxml
- Testing.html
- Realtime.html

The SLOC counting for the FXML and HTML files also count the declarations within tags as separate lines of code. The measurement which takes into account reusable code excludes the AnalyticsFacade and AnalyticsPrototype files.
5 Results

In this chapter the results from the software estimation made using the prototype are presented.

5.1 Size

The result from counting the amount of SLOC of the individual source code files can be seen in table 3.

<table>
<thead>
<tr>
<th>File name</th>
<th>SLOC count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalyticsFacade.java</td>
<td>83</td>
</tr>
<tr>
<td>AnalyticsController.java</td>
<td>84</td>
</tr>
<tr>
<td>AnalyticsModel.java</td>
<td>44</td>
</tr>
<tr>
<td>AnalyticsPrototype</td>
<td>18</td>
</tr>
<tr>
<td>AnalyticsView.fxml</td>
<td>208</td>
</tr>
<tr>
<td>Testing.html</td>
<td>84</td>
</tr>
<tr>
<td>Realtime.html</td>
<td>46</td>
</tr>
</tbody>
</table>

The first size measurement which includes the SLOC from all source code files is retrieved by adding the individual file SLOC counts together; using the data seen in table 3 the summation result for the first measurement is 567 SLOC.

The second measurement which excludes the AnalyticsFacade and Analytics Prototype files is retrieved by subtracting the two file SLOC counts from 567, the result of which is 466 SLOC.

5.2 Estimation

The effort estimation is done using equation (1) with the scale factor variable E derived from equation (2). The constant B in equation (2) is set to 0.91 and the range of the SF variable summation is calculated using the table seen in appendix A. This calculation is done by setting all the individual SF variables to their lowest value to get the minimum and setting them to their highest value to get the maximum. The sum for the maximum values is thus given by equation (3):

\[ \sum_{j=1}^{5} SF_j = 6.2 + 5.07 + 7.07 + 5.48 + 7.8 = 31.62 \]  

(3)
while the sum for the minimum values is 0 since the lowest value for each SF variable is 0. By implementing the sums for the minimum and maximum values in equation (2) the lower and upper bounds for $E$ are given by equation (4) and (5) respectively.

$$|E| = 0.91 + 0.01 \times 0 = 0.91$$  
(4)

$$|E| = 0.91 + 0.01 \times 31.62 = 1.2262$$  
(5)

From these the interval of the scale factor $E$ is shown to be $[0.91, 1.2262]$.

The constant $A$ in equation (1) is set to 2.94 and the EM product is set to 1, thus the estimation equation used for calculating the amount of development hours in person-months is:

$$PM = 2.94 \times S \times E \times 1$$  
(6)

where the $E$ variable can vary between the interval given above.

For the first estimation which includes the SLOC count from all of the source code files the size variable $S$ is set to 0.567. The upper bound for the estimation is given by using the minimum value for the $E$ while the lower bound is given by using the maximum for $E$ as shown in equation (7) and (8) respectively:

$$PM = 2.94 \times 0.567^{0.91} \times 1 \approx 1.7543$$  
(7)

$$PM = 2.94 \times 0.567^{1.2262} \times 1 \approx 1.4662$$  
(8)

Thus the approximated estimation for how many person-months the development of the prototype would take is $1.61025 \pm 0.14405$.

For the second estimation the size variable $S$ is set to 0.466 and the bounds are calculated in the same way as seen in equation (9) and (10)

$$PM = 2.94 \times 0.466^{0.91} \times 1 \approx 1.4675$$  
(9)

$$PM = 2.94 \times 0.466^{1.2262} \times 1 \approx 1.1527$$  
(10)

Thus the approximated estimation for how many person-months the development of a software with code reused from the prototype would take is $1.3101 \pm 0.1574$.  

26
6 Discussion

In this chapter the results from the work of the thesis are analysed and conclusions that could be made from them are explored. Additionally a recommendation is given for what further research could improve upon the work done in this thesis.

6.1 Prototype

The prototype that was developed for this thesis is limited in scope and does not contain the functionality to be expected from a fully developed software product. However the prototype does meet the criteria that were set in the beginning of this thesis and thus has served its purpose. Most of the development time for the prototype was spent on researching various solutions required for the integration with Analytics. Specifically the authentication with Google and the processing of the received data consumed a large part of the development time.

The main functionality of the prototype is contained in the Custom report tab and several alternative interfaces such as interactive graphs were considered during the development. Ultimately those were abandoned since they would have taken too much time to implement.

The prototype’s safety in terms of safeguarding the data available in Analytics is limited since no secure login is required. The recommendation for a secure implementation is to create an internal authentication system for the potential users of the software and after verification be allowed to access the Analytics data.

There is additional possible functionality from Analytics that could be developed for the prototype which a real world implementation would benefit from. Therefore the eventual usability of future work isn’t limited to the functionality of the prototype. Given enough development time a software could re-implent all of the interface elements available in Analytics and also integrate this with other external software data. In theory this customized software could not only be more suitable for the specific users needs but also allow for functionality that is not possible solely within the context of Analytics.

It is unknown what kind of maintenance would be required once a software similar to the prototype is deployed in a real work environment. However due to the separation of the internal logic and the GUI it should not be a difficult task to update the facade whenever the Google APIs are updated.
6.2 Conclusions

The results from the estimation of how many person-months it would take to develop the prototype and another similar program with reused code have several weaknesses. The primary weaknesses are that the estimation might not properly take into account the complexity of the code and the person-month metric does not accurately predict how much work an individual developer could accomplish. Furthermore the estimation only estimates how long it would take to develop a similar program without the implementation of methods such as SPL which would likely reduce the development time. In order to estimate this reduced development time a study would have to implement the SPL framework and produce several prototypes but this was beyond the scope of this thesis.

In COCOMO one person-month is defined as how much work the average developer could carry out in 152 working hours. Since this thesis was developed by a single individual whose relative work-efficiency is unknown compared to the average developer; it is difficult to compare the estimation result with the actual time it took to develop the prototype. This problem is expanded even further since this individual also had to do research and other work tasks unrelated to the development of the prototype at the time of development. However if this work would be replicated on a larger scale by more than one individual the average person-month metric would likely better represent the actual efficiency of the development. It is also possible that working in collaboration with more people and having code quality control could increase the workload overhead.

The complexity of the code is supposed to be accounted for by varying the factors in the COCOMO estimation formula. Even though the result that was given included the possible range of estimations it only did so by varying the scale factor. For a more accurate estimation the effort multipliers would have to be calibrated according to the projects parameters along with the productivity constant. Another issue with the estimation is that the KSLOC count for the prototype was below one which means that the scale factor exponent, which is supposed to increase the estimation the higher it gets, actually did the opposite. This might not have affected the result severely since it conversely increased the estimation the lower it got so the final range should be reasonable.

Following the results from this thesis; the viability of replacing Analytics with customized software is deemed a valid alternative worth exploring further. Any more rigorous conclusion as to how the two solutions compare to each other in terms of efficiency and economy would require a larger case study. The sources used for the theory chapter were taken from mostly official Google websites at the time of writing and are likely to become deprecated in the future. This could mean that the relevance of the thesis will decline over time, however the main concepts and techniques used could still be translated over to future platforms since the principles they are based upon are sound. In conclusion the overall aim and goals of this thesis have been fulfilled but further research is recommended.
6.3 Further research

In order to improve the accuracy of the result, more iterations of similar software could be made and through comparison of these iterations’ actual development time a more refined estimation might be developed. The difference between the two measurements could also be explored further in a similar way. The reused code measurement gives a roughly 20% shorter estimation than the full prototype measurement, whether this relation is translatable to larger projects is unknown. Since this prototype is a small program, the facade might represent a larger percentage than it would in a larger program and thus the relative size difference could change.

Unfortunately the limited scope of this thesis made it difficult to incorporate more than one software estimation model. In future research the implementation of several different estimation methods could be used in order to improve the accuracy of the result.

If the development of the facade continued alongside improvements gained from an SPL implementation the work required to create a new software might mainly consist of creating the GUI. This could affect how much work effort would be required to create programs similar to the prototype since a developer with expertise in interface development could be employed and would likely be more efficient at creating the GUI. If no back-end development was required when a new software order was placed the development time could therefore be substantially reduced.

Therefore, in order to fully analyse the financial differences between a bespoke software product and the default Google Analytics interface, a more rigorous case study is recommended which further explores the possibility of implementing an SPL along with multiple iterations of prototypes.

6.4 Ethical Aspects

Some of the ethical concerns that could be raised in relation to the work done in this thesis are the safeguarding of the data that Google is given access to and how, in a hypothetical future, the replacement of Analytics courses with customized software might affect the industry and what positive or negative effects it could have.

As mentioned in chapter 2.1.8 the data protection of both the Analytics admin and the website users is safeguarded by the Google Privacy Policy however there is no guarantee that Google will at all times respect this policy. As long as Google has the means to access the private data of their users there is a potential security risk which should be considered. If the website does not contain highly sensitive information, this likely small risk is probably not a concern. However if the data protection of the website is a high priority then even this small risk should be taken into account before giving Google access to the data.
In regards to how the replacement of Analytics courses could affect the industry there is only speculation. If the customized software alternative proved to be vastly more economical and efficient then it is reasonable to assume that most companies would opt out of sending their employees on courses. However, the company would still need in-house expertise to manage their Analytics account so there would still be a demand for courses that are meant for the more advanced users. The biggest shift would happen amongst the entry-level employees whose work assignments only require a limited interaction with Analytics.

If reducing the amount of work related courses in the industry in the end would yield a positive or negative result is difficult to know and would likely require further research on its own. It would likely reduce the amount of man-hours required to sustain the knowledge level of employees within the industry. Something which many would argue is of economical benefit to society since it frees up labor to pursue other tasks. While there certainly is room for discussion on whether or not that is true, most of the work done within all industries is heading in the direction of freeing up labor. Therefore the work within this thesis is likely in line with the overall progress the industry is making and should not be considered ethically different.
References


https://developers.google.com/analytics/devguides/reporting/core/dims
mets
Retrieved 2016-03-16.

https://developers.google.com/analytics/devguides/collection/analyticsjs
/custom-dims-mets
Retrieved 2016-03-16.

https://developers.google.com/chart/interactive/docs/gallery/motionchart
Retrieved 2016-03-16.

https://developers.google.com/analytics/devguides/collection/analyticsjs
/events
Retrieved 2016-03-16.

https://support.google.com/analytics/answer/1012040?hl=en
Retrieved 2016-03-16.

https://support.google.com/analytics/answer/1320491?hl=en
Retrieved 2016-03-16.

https://developers.google.com/analytics/devguides/config/mgmt/v3/user
-management
Retrieved 2016-03-17.

https://developers.google.com/analytics/devguides/reporting/realtime/v3
Retrieved 2016-03-17.

https://support.google.com/analytics/answer/2519986
Retrieved 2016-03-17.

http://analytics.blogspot.se/2011/10/introducing-flow-visualization.html
Retrieved 2016-03-17.

https://support.google.com/analytics/answer/4553001?hl=en
Retrieved 2016-03-18.
[22] Utbildning.se, “Utbildning i webbanalys med Google Analytics”
http://www.utbildning.se/kurs/google-analytics
Retrieved 2016-03-18.


https://developers.google.com/analytics/devguides/reporting/embed/v1/
#introduction

https://developers.google.com/identity/protocols/OAuth2

https://support.google.com/analytics/answer/6004245
Retrieved 2016-03-18.

[27] Google, “Privacy Policy”,
Retrieved 2016-04-02.

https://support.google.com/analytics/answer/1011397?hl=en
Retrieved 2016-03-18.

https://developers.google.com/analytics/devguides/collection/analyticsjs/
/user-opt-out
Retrieved 2016-03-18.

https://support.google.com/analytics/answer/2763052?hl=en
Retrieved 2016-03-18.

[31] Glenn E. Krasner, Stephen T. Pope, “A Cookbook for Using the Model-
View-Controller User Interface Paradigm in Smalltalk-80”, *Journal of

[32] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, *Design

https://en.wikipedia.org/wiki/Loose_coupling
Retrieved 2016-03-22.


34
## Appendix A: COCOMO II 2000

Calibrated Post-Architecture Model Values

<table>
<thead>
<tr>
<th>Driver</th>
<th>Symbol</th>
<th>VL</th>
<th>L</th>
<th>N</th>
<th>H</th>
<th>VH</th>
<th>XH</th>
</tr>
</thead>
<tbody>
<tr>
<td>PREC</td>
<td>SF_a</td>
<td>6.20</td>
<td>4.96</td>
<td>3.72</td>
<td>2.48</td>
<td>1.24</td>
<td>0.00</td>
</tr>
<tr>
<td>FLEX</td>
<td>SF_a</td>
<td>5.07</td>
<td>4.05</td>
<td>3.04</td>
<td>2.03</td>
<td>1.01</td>
<td>0.00</td>
</tr>
<tr>
<td>RESL</td>
<td>SF_a</td>
<td>7.07</td>
<td>5.65</td>
<td>4.24</td>
<td>2.63</td>
<td>1.41</td>
<td>0.00</td>
</tr>
<tr>
<td>TEAM</td>
<td>SF_a</td>
<td>5.48</td>
<td>4.38</td>
<td>3.29</td>
<td>2.19</td>
<td>1.10</td>
<td>0.00</td>
</tr>
<tr>
<td>PMAT</td>
<td>SF_a</td>
<td>7.50</td>
<td>6.24</td>
<td>4.88</td>
<td>3.12</td>
<td>1.56</td>
<td>0.00</td>
</tr>
<tr>
<td>RELY</td>
<td>EM_a</td>
<td>0.82</td>
<td>0.92</td>
<td>1.00</td>
<td>1.10</td>
<td>1.26</td>
<td></td>
</tr>
<tr>
<td>DATA</td>
<td>EM_a</td>
<td>0.90</td>
<td>1.00</td>
<td>1.14</td>
<td>1.28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPLX</td>
<td>EM_a</td>
<td>0.73</td>
<td>0.87</td>
<td>1.00</td>
<td>1.17</td>
<td>1.34</td>
<td>1.74</td>
</tr>
<tr>
<td>RUSE</td>
<td>EM_a</td>
<td>0.95</td>
<td>1.00</td>
<td>1.07</td>
<td>1.15</td>
<td>1.24</td>
<td></td>
</tr>
<tr>
<td>DOCU</td>
<td>EM_a</td>
<td>0.81</td>
<td>0.91</td>
<td>1.00</td>
<td>1.11</td>
<td>1.29</td>
<td>1.63</td>
</tr>
<tr>
<td>TIME</td>
<td>EM_a</td>
<td>1.00</td>
<td>1.11</td>
<td>1.29</td>
<td>1.63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STOR</td>
<td>EM_a</td>
<td>1.00</td>
<td>1.05</td>
<td>1.17</td>
<td>1.46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PVOL</td>
<td>EM_a</td>
<td>0.87</td>
<td>1.00</td>
<td>1.15</td>
<td>1.30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACAP</td>
<td>EM_a</td>
<td>1.42</td>
<td>1.19</td>
<td>1.00</td>
<td>0.85</td>
<td>0.71</td>
<td></td>
</tr>
<tr>
<td>PCAP</td>
<td>EM_a</td>
<td>1.34</td>
<td>1.15</td>
<td>1.00</td>
<td>0.88</td>
<td>0.76</td>
<td></td>
</tr>
<tr>
<td>PCON</td>
<td>EM_a</td>
<td>1.29</td>
<td>1.12</td>
<td>1.00</td>
<td>0.90</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>APEX</td>
<td>EM_a</td>
<td>1.22</td>
<td>1.10</td>
<td>1.00</td>
<td>0.88</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td>PLEX</td>
<td>EM_a</td>
<td>1.19</td>
<td>1.09</td>
<td>1.00</td>
<td>0.91</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>LTEX</td>
<td>EM_a</td>
<td>1.20</td>
<td>1.09</td>
<td>1.00</td>
<td>0.91</td>
<td>0.84</td>
<td></td>
</tr>
<tr>
<td>TOOL</td>
<td>EM_a</td>
<td>1.17</td>
<td>1.09</td>
<td>1.00</td>
<td>0.90</td>
<td>0.78</td>
<td></td>
</tr>
<tr>
<td>SITE</td>
<td>EM_a</td>
<td>1.22</td>
<td>1.09</td>
<td>1.00</td>
<td>0.93</td>
<td>0.86</td>
<td>0.89</td>
</tr>
<tr>
<td>SCED</td>
<td>EM_a</td>
<td>1.43</td>
<td>1.14</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td></td>
</tr>
</tbody>
</table>