Technomics
Better Decisions Faster

Natural Language Processing: A New Approach to Simple Function Point Estimation

Dave Brown-dbrown@technomics.net

International Cost Estimating & Analysis Association

5/16/2023

Abstract

The need for data-driven methods for predicting software size has never been greater. This is especially true for agencies that rely on SME opinion or T-shirt based sizing as the primary method for estimating the size and cost of agile-developed software. Recent research at NSA and DHS provides a promising approach, known as Simplified Software Estimation, which generates a simple function point sizing estimate based on functional requirements. This simple function point sizing method offers significant advantages to expert-based sizing because it is a repeatable process that provides defensibility and traceability to either a high-level requirements document or requirements written as user stories. This presentation details a method for using Natural Language Processing (NLP) to automate the estimation of simple function point counts. The advantages of this approach are (1) it provides complete consistency among multiple counts and (2) it offers a way to quickly generate a count from a large dataset spanning multiple projects. Initial research shows that a NLP-derived simple function point count is an effective approach that offers great potential for improving data-driven estimates for small and large agile software development programs. In particular, the requirements documentation delivered in Lean Business Cases (LBC) at the onset of a go/no-go decision is readily available, standardized across developers/agencies, and an effective source for NLP-generated simple function point counts.

Keywords: *Software Estimation, NLP, Function Point Sizing*

Table of Contents

Introduction

This paper presents a method for estimation and analysis of software development cost using Simple Function Point (SFP) sizing and Natural Language Processing (NLP). First, SFP and NLP are defined in a context that describes their purpose and usage in this approach. Next, the details of SFP analysis performed using NLP are presented and lessons-learned for refining this estimation approach are discussed. Lastly, the paper provides an evaluation of the approach, including its strengths and weaknesses.

Overview of the Problem

The analysis and methods presented in this paper are driven by the cost estimator's need for a high-quality estimate of software development cost. This is not a new problem, but as software and IT programs continue to become more prevalent within the federal sector, the need increases. Furthermore, constrained IT budgets have only heightened the importance of software development estimates that are:

- **Defendable**. Cost estimators must be able to support their findings. Every element of a cost estimate should be accompanied by its underlying data, ground rules, assumptions, and estimating methodology. Cost estimators should always be able to answer questions such as "where did that number come from?", or "how precise is the number?".
- **Data driven**. Estimates that are grounded in historical data are not only more defendable, but also less subject to bias. The contrast to a data-driven estimate is one based on subject matter expert (SME) opinion. While complete elimination of any SME opinion within an estimate may not be practical, it is a best practice to minimize this method.
- **Available early in the acquisition life-cycle**. Analysts are often asked to create an estimate early in the acquisition cycle. This means that any meaningful actual cost history within the program is generally not available. It also means

that a detailed technical design is rarely available for either software or for the program as a whole. Therefore, estimators must work with available data, which for software is often limited to a high-level requirements document.

The value of quality software cost estimates should be self-evident to most in the cost community, but is worth summarizing here. Improved software cost estimates will allow:

- Better budget justification. Cost estimates are often used as the basis for budget formulation. Better software cost estimates will allow stakeholders to better justify and defend their budgets.
- Better understanding of cost. Even outside of the budget process, it is valuable for stakeholders to fully understand the cost of software development, including the drivers of cost. In cases where software cost is dependent on functional requirements, it is valuable to fully understand that relationship – how do requirements influence cost. A fully established connection between cost and requirements allows both analysts and stakeholders to make better decisions.
- Better understanding of risk and uncertainty. As with any cost estimate, software cost estimating is imperfect. It is impossible to know the exact cost of software development. Therefore, it is often valuable to express estimates as a range rather than as a single point. This is especially valuable if the range is based on the uncertainty of the underlying data and assumptions. The process for creating this range, known as cost risk and uncertainty, should be applied for software estimates just as it should be applied for other types of estimates.

Software Sizing

For software cost estimators, it is critical to obtain an estimate of the proposed software size. Software size is a metric that quantifies how much software needs to be developed. A well-accepted method for quantifying size is Source Lines of Code (SLOC). The appeal of SLOC as a sizing metric is ease of countability for delivered software. The downside of SLOC is it's difficult to predict, meaning accurate SLOC

estimates are elusive. This reality makes SLOC a less than ideal candidate for estimates that are required early in the software lifecycle, when inadequate software definition is available to predict SLOC. Further limiting the utility of SLOC is that fact that modern software development continues to use visual tools, where the size and complexity is often not correlated with SLOC.

A popular alternative to SLOC is functional sizing. Functional sizing is the category of metrics that measure functional size. Unlike SLOC, which is a metric of physical size, a functional size metric is based on the functionality delivered by the software.

Function Point Analysis

The most widely used functional size metric is function point analysis (FPA), as defined by the International Function Point Users Group (IFPUG). This metric addresses some of the deficiencies with SLOC discussed in the previous section. In particular the FPA method offers:

- Estimation earlier in the life cycle. An FPA size can be created as soon as detailed functional requirements are documented.
- Direct connection to requirements. Because the FPA size is derived from functional requirements, there is an explicit linkage between requirements and cost.
- Technology and platform independent. Because FPA is based on requirements and not a specific technical architecture, an FPA count does not depend on any specific programming language or technology.

There are, however, some significant drawbacks to the FPA sizing metric:

- Functional requirements must be detailed. FPA sizing relies on requirements documentation that is detailed enough to fully understand the size and complexity of data stored within the system, calculations and queries on the data,

as well as all of the interfaces to external systems. These requirements may not be fully available at the time a cost estimate is first needed.

- Need for skilled an experienced FPA function point counters. In addition to publishing FPA standards, IFPUG offers a certification program that requires an exam as well as a recommended 6+ months of FPA experience.[1] For organizations that don't have access to a certified function point counter, this limits the utility of FPA.

To address these limitations, a second method of functional sizing that is also recognized by IFPUG is offered, namely Simple Function Point (SFP) sizing.

Simple Function Points

SFP sizing is a lightweight version of FPA that borrows some of the high-level concepts of FPA, but simplifies the process as well as the level of detail required from requirements documentation.

The following graphic shows a comparison between an FPA sizing metric and an SFP sizing metric:

IFPUG Components	Low	Average	High
External Inputs	3	4	6
External Outputs	4	5	7
External Inquiries	3	4	6
Internal Logical Files	7	10	15
External Interface Files	5	7	10

SFP Components	Weighting Factor
Transactions (Create, Update, Delete, Report, Read)	4.6
Logical Data Groups (Saves)	7

Figure 1: FPA versus SFP Sizing

The left side of the graphic shows the five components of an FPA count. For each component, the analyst must also assess the complexity on a scale of low, medium, and high. The IFPUG process provides definitions and standards for how each component is counted, and how complexity is assessed. The numbers in the table

[1] IPFUG Certifications and Requirements: https://ifpug.org/certification/cfps-cfpp

represent the number of function points counted for each instance of the component counted.

The right side of the graphic shows what must be counted using SFP. This technique only requires counting of two components and eliminates the need to assess complexity. The numbers in this table also represent function point counts and are based on an average of the complexity levels.

Simplified Software Estimation

The Department of Homeland Security (DHS) has further developed the SFP process by publishing a process known as Simplified Software Estimation (SiSE)[2]. This technique generates a SFP count based on key verbs that appear in a high level requirements document, such as a Concept of Operations. A list of key verbs is associated with an assumed number of SFP Transactions and SFP Logical Data Groups. The list of verbs is then expanded using synonym verbs. The following graphic shows an example from the SiSE verb table:

Keyword	SFP TX (4.6 SiFP)	SFP Data (7 SiFP)	TOTAL FOR VERB SiFP	*Synonyms*
Allow	2	1	16.2	*grant, accept*
Create	2	1	16.2	*build, conceive, constitute, construct, devise, establish, initiate, invent, make, set up, start, inception, origination*
Detect	1		4.6	*determine, discover*
Search	1		4.6	*hunt, inquire, investigate, research, study, filter, find*
Store	2	1	16.2	*keep, stow*

Figure 2: SiSE Verb Table

The SiSE process requires the analyst to separate functional from non-functional requirements. It further requires that any redundancy in functional requirements is

[2] Let's SiSE Agile Program Requirements: https://www.dhs.gov/sites/default/files/publications/2021_jitscf_lets_sise_agile_software_development_mann_hoang_lucas_dekkers.pdf

eliminated. After these manual process steps are completed, a SFP count is created by counting verbs and assigning the SiSE weight to each verb instance.

Recent experience at the Department of Homeland Security (DHS)[3] as well as the National Geospatial-Intelligence Agency (NGA) shows that a SFP count can successfully be created using SiSE from a single document, such as a Concept of Operations or Lean Business Case.

Natural Language Processing

While all the sizing methods discussed have their own merits, they require some level of expertise in software sizing, as well as the time and manpower to create the metric. In situations where only a single sizing number is needed (e.g., for a single program), the schedule and effort requirement is likely not an issue. But for organizations that need a software sizing estimate on a portfolio of multiple programs, an automated solution offers some benefit. Additionally, an automated solution to software sizing is based on a single algorithm that can be applied consistently to many estimates. As such, consistency can be achieved without the need for published standards or a certification program. For these reasons, a solution based on Natural Language Processing (NLP) offers some significant advantages.

NLP Overview

Though rarely applied in cost estimating, NLP is not a new concept. It is used today in many different applications, such as: classifying emails as spam, autocorrect, spellcheck, and foreign language translation. For data analysts, NLP offers a compelling capability: converting unstructured text into structured data. When narrative text can be expressed as numbers, it opens the door to a wide variety of statistical and data-driven approaches that would otherwise be unavailable.

---

[3] Let's Go Agile: Data-Driven Agile Software Cost and Schedule Models Derived from DHS Projects, https://www.iceaaonline.com/wp-content/uploads/2022/06/SA09-Rosa-Lets-Go-Agile.pdf

Although a full exploration of NLP is beyond the scope of this paper, there are a few key terms which will help introduce the concept and facilitate further discussion:

- Corpus. The body of text that is fed into the NLP process. A corpus may be as short as a single sentence, or as long as an entire collection of documents.
- NLP Data Cleaning. The process of removing or replacing special characters and symbols (e.g., line breaks, @, #, %); replacing contractions with their full text; expanding acronyms; removing punctuation; and converting to lower case.
- Stopwords. Common words that can be removed from NLP analysis without losing meaning (e.g., “a”, “the”, “to”).
- Tokenization. The process of breaking the corpus down into smaller chunks (often individual words) called “tokens”.
- Annotation. The process of noting where in the corpus the word (token) is found, and the part of speech (e.g., identifying verbs and noting their location).
- Stemming/Lemmatization. The process of standardizing a word to its root. In the case of verbs, this process converts verbs to present tense.

NLP in R and R Studio

The R programming language and the RStudio Integrated Development Environment are tools commonly used in NLP. For the analysis presented in this paper, all of the NLP was performed using R and RStudio. These tools are appealing for a number of reasons. R offers a wealth of open source packages to facilitate NLP that are easy to install and easy to use. Additionally, the R language as well as RStudio are both available as open source. The specific packages used in this analysis are: dplyr, stringr, textclean, tm, udpipe, writexl, readxl, and openxlsx.

While R and the listed packages provide all of the basic NLP capability, analysts wishing to perform analysis discussed in this paper also need tailored R code that reads their specific corpus and performs the problem-specific data cleaning, tokenization, annotation, and stemming. R code is also needed to implement the SFP counting

process using the DHS SiSE or a similar algorithm. These steps were accomplished using R code developed internally by Technomics and implemented in a tool that provides the following functionality:

- Read text data from any corpus. Text can be read from a MS Word file, Excel spreadsheet, or scraped from a requirements repository such as Confluence or JIRA.
- Use NLP to identify verbs and convert them to their lemma.
- Align the converted verb list against a standard SFP list, such as the list found within SiSE.
- Score each verb according to a pre-defined SFP weight, and add the results to generate an overall SFP estimate.

Using NLP to Generate a SFP Size Estimate

The analysis and results discussed in this section are the result of applying NLP to generate a SFP size estimate. It is based on research conducted for an agency in the intelligence community. All of the data have been anonymized, with all specific dollar values, program names, and identifying characteristics removed. However, all data shown are representative and consistent with actual analysis and results.

Source Data

The source for corpus text used in this analysis is requirements documentation in the form of a Lean Business Case (LBC). An LBC is a standard product generated by many organizations that follow an agile development methodology. For this analysis, use of the LBC was chosen because programs could be decomposed into multiple Program Epics (PE's), each of which contained their own LBC. Furthermore, a portfolio of LBC's within the same program reflects the fact that each LBC is written to the same standards and level of detail.

The analysis also made use of estimated costs for each PE. The purpose of the analysis was to validate existing estimates, and to support an independent cost estimate using alternative methods. Therefore, estimated cost was not assumed to be completely accurate, nor was it treated in the way that an estimator would handle actual cost data.

The analysis discussed in the following sections reflects an iterative process, specifically: results are generated and analyzed, the method is refined, and then new results are generated and analyzed. There are lessons learned at each step of the process that not only result in conclusions about the program being analyzed, but also help to refine and improve the NLP/SFP approach. Each iteration is discussed in a separate subsection.

Results Using SiSE Verbs and Weights

The first application of this approach was to input a set of 10 PE's from the same program into the NLP / SFP algorithm using the SiSE verbs and SiSE verb weights. An automated SFP count was produced for each PE. Results where then regressed against estimated costs to determine how strongly an SFP count generated with NLP can predict cost. The following figure shows the result:

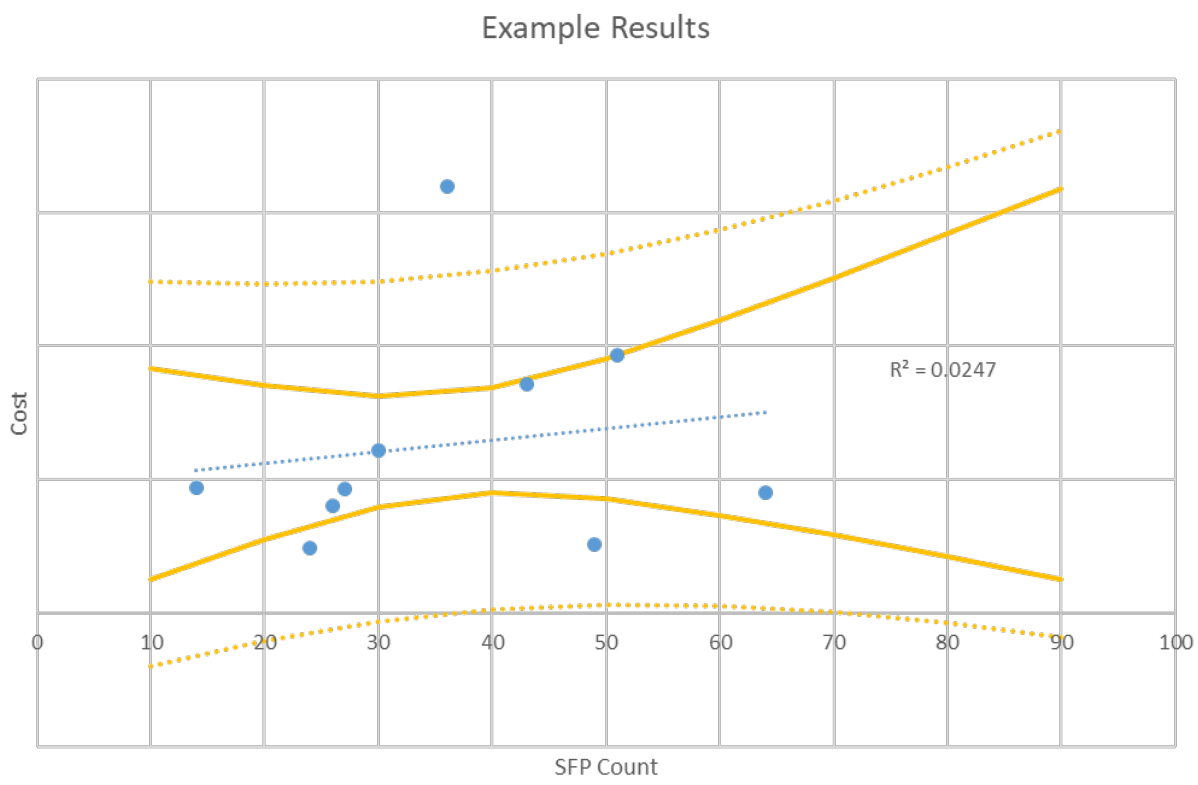

Figure 3: Results Using SiSE Verbs and Weights

Each data point is plotted as a blue dot. The linear regression line is plotted as a dashed blue line, and annotated with the R-squared correlation of 0.0247. The solid yellow and dashed yellow lines represent a 90% confidence interval and 90% prediction interval, respectively. As expected with any linear regression, the confidence interval and prediction interval are the most narrow near the mean of the data, and widen when viewed at levels either above or below the data set.

The relatively low R-squared value of 0.0247 indicates a very weak correlation. The positive slope of the regression line and the positive Y intercept are expected based on a logical assumption that a higher SFP count should be associated with higher cost, and that there may be fixed a cost with respect to SFP.

These results were not strong enough to generate any definitive conclusions. Therefore the next step was to analyze the underlying algorithm, and determine if anything could be modified to provide additional insight. The finding of this analysis was that the 10

LBC's contained many verbs not counted in either the SiSE verb list or its defined synonyms. Many of these verbs appear to be significant, but were more often seen in documentation from the intelligence community than in other agencies. Examples of such verbs include "exploit", "subscribe", and "structure".

Results Using Agency-Specific Verbs

Based on the results of the previous section, the next step was to analyze whether the results could be improved by using an agency-specific verb list. To generate such a list, another NLP algorithm was used to identify verbs and their lemmas, using a higher level requirements document within the same program. The Intelligence Capability Baseline Description (ICBD) was chosen for this purpose. The result was a verb list that better aligns with verbs frequently used within intelligence community programs. As a first cut, a weighting of 1.0 was assigned to all verbs. The results are shown below:

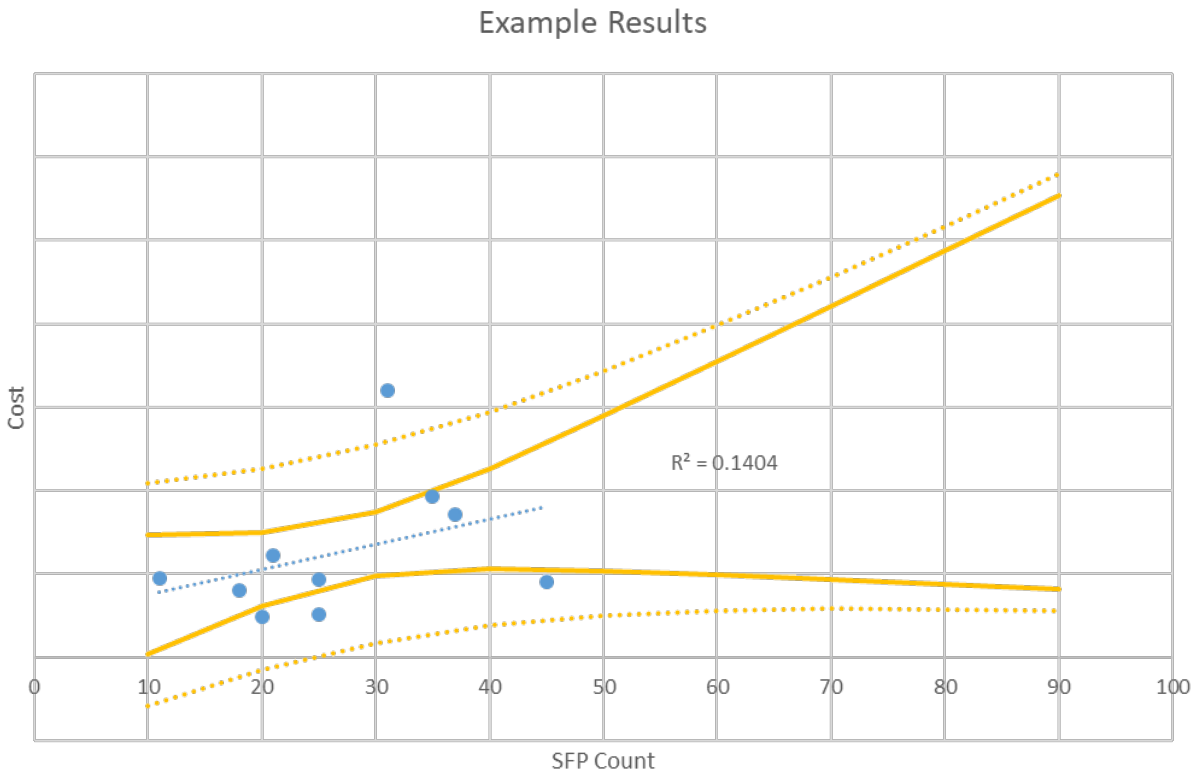

Figure 4: Results Using Agency-Specific Verbs

These results also have the expected positive correlation and positive Y intercept. Additionally, the improved R-squared statistic of 0.1404 seems to indicate that an agency-specific verb list might serve as a better predictor. The fact that improved results are observed even using a verb list with equal 1.0 weights underscores how important the agency-specific verb list is to the process.

The next step in the analysis was to examine the outlier data points to glean insight into why some data points do not follow the general trend. The prediction interval served as a useful indicator for the purpose of identifying outliers. The data point that most stands out is the one at the top of the graph with the highest cost. This is the only data point that falls outside of the prediction interval.

Exploration of this outlier yielded some interesting findings. It was discovered that this PE was created by combining requirements that were originally captured in three separate PE's. Further analysis of the verbs within this PE revealed that it contained several verbs that were rarely seen in the remaining nine PE's. These results indicated that the algorithm used in this analysis may have under estimated the SFP count for this particular data point. At this point in the analysis, some valuable conclusions could be made:

- An outlier data point was identified. This allowed analysts to ask specific questions about this particular PE.
- Exploration and additional data collection related to the outlier data point revealed some issues with the data that might otherwise have gone unnoticed.
- The *reason* this data point is an outlier may be attributable to one of two situations: either the SFP was inaccurate or the cost estimate was inaccurate. The former indicates that the error can be viewed horizontally on the graph. That is, correction of the error would involve a horizontal increase in the SFP count. Alternatively, if the cost estimate was inaccurate, then the error could be viewed as the vertical distance on the graph between the data point and regression line.

As discussed above, the data indicated that it was more likely a horizontal error for this data point.

Results Using Agency-Specific Verbs and Custom Weights

In order to correct one of the deficiencies of the previous analysis, a verb weighting scheme was created. Verbs were given weights on a 3-point scale: either 1.0, 3.0, or 5.0. Emphasis and higher weighting was given to verbs that were unique to the identified outlier. It should be noted that this method of adjusting SFP results based on an outlier data point will inevitably improve the correlation of the results. Just as any regression can be improved by removal or adjustment of outliers, this technique will guarantee a better fitting regression line. With this caveat in mind, the results were re-generated using the modified verb weights. No data points were removed in this analysis and the modified verb weights, although disproportionally impacting the outlier, were equally applied to all data points. The following figure shows the result:

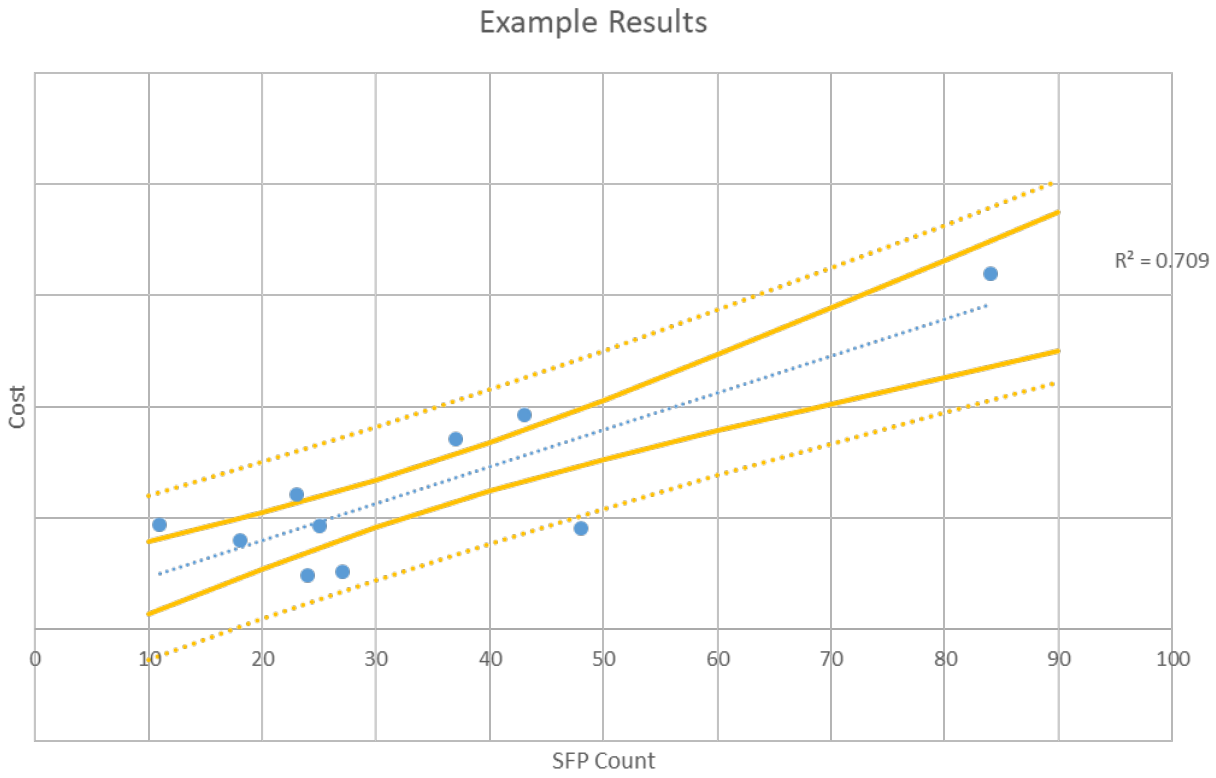

Figure 5: Results Using Agency-Specific Verbs and Weights

As expected, the correlation, as measured by R-squared, dramatically improved.

Using these results, the process of identifying and analyzing outliers was repeated. By viewing data points that fell above, within, or below the confidence interval lines, it can be seen that approximately 1/3 of the data lies within each band. This result stood out because the cost estimates for this program used a "T-shirt" sizing method that is known to be accurate to plus or minus one T-shirt size. It was acknowledged that T-shirt sizing within this estimate would be correct approximately one third of the time, off by one size too low one third of the time, and off by one size too high one third of the time. Further, the scaling of T-shirt sizes was such that the next higher size was double in size, while the next lowest size was one half the size. These known characteristics of T-shirt sizing seemed to align well with the observed results in the above figure. This was especially true for the three data points that are found below the confidence interval

line. In all three cases, these data points have cost values that are approximately one half the predicted value on the regression line.

Again, this intermediate step in the analysis produced some insightful conclusions:

- Outlier data points were identified, with the three data points below the confidence interval line being the largest outliers.
- Again, consideration was given to whether the outliers were more likely due to horizontal error or vertical error. As before, a horizontal error would be due to an inaccurate SFP estimate. A vertical error would be due to an inaccurate cost. These results indicate that the error is more likely vertical—a result of an inaccurate cost estimate. This conclusion is based on the known limitations of T-shirt sizing, the overall pattern and banding of the results, and the consistent pattern of the low side outliers being one half their predicted value.

The next step in the analysis was to hypothesize that the low side outliers should have their cost adjusted upward by a factor of two. The high side data points were not adjusted because they were not as extreme as the low side outliers, and they didn't follow the pattern where a T-shirt size selected as one size too large should be double the actual value.

Final Results

The final results were produced by adjusting the cost for the low side outliers, as discussed in the previous section. As before, this technique ensures a result that will have an improved correlation. This is shown in the following figure:

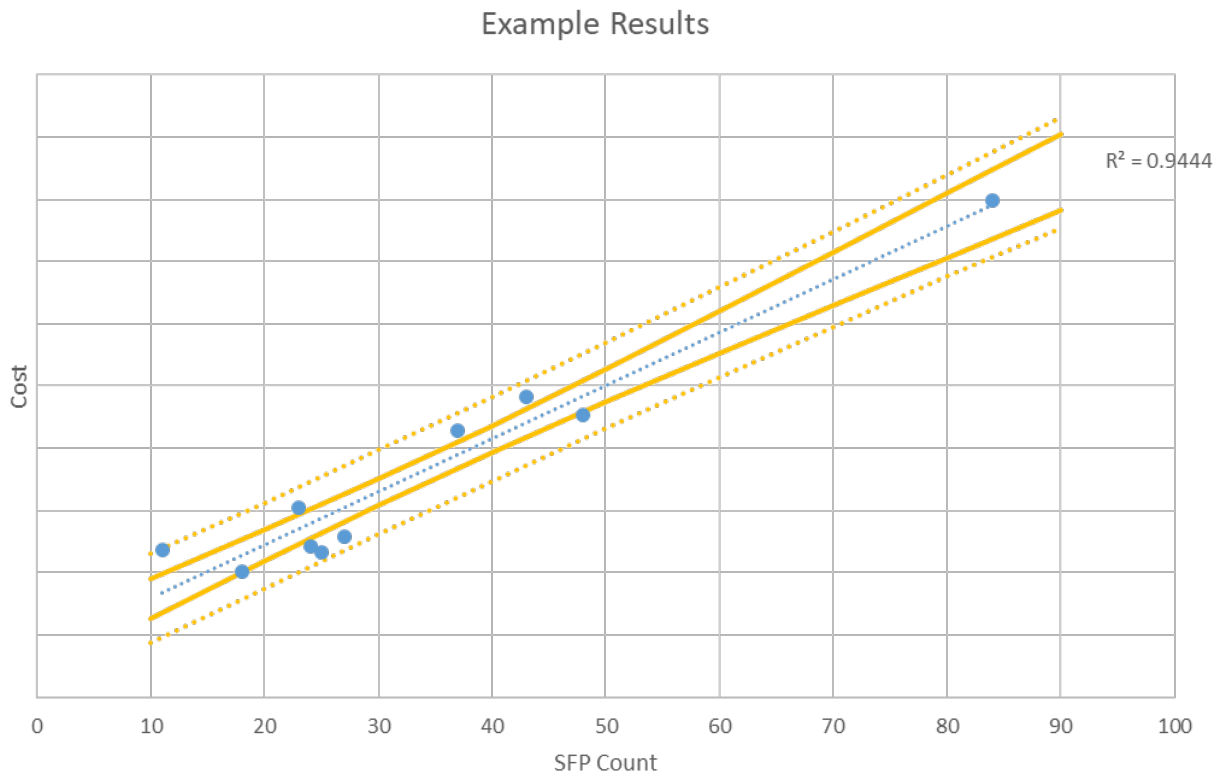

Figure 6: Results with Adjusted Outliers

Not surprisingly, the correlation and R-squared are very high. But, because this is a direct result of modifying only outlier data, the correlation is not viewed as a significant statistic nor should it suggest that the true correlation between SFP and cost is this strong. However, even with these limitations and caveats, the analysis provided here and in the previous sections offered valuable insight:

- One data point was identified as a high side outlier. No cost adjustment was suggested for this PE because the data indicates that the initial SFP count was under sized.
- Three low side outliers were identified. These data points likely carry increased cost risk and should possibly be increased by a factor of two.
- The remaining seven data points look reasonable when compared against their requirements, as measured using a NLP / SFP technique.

The results from each iteration of this analysis are summarized in the following table:

Approach	Results	Conclusions
DHS SiSE Verbs and Weights	Low correlation	Some agency-specific verbs were not counted
Agency-Specific Verbs with Equal Weighting	Improved correlation; one high-side outlier	Analysis of the outlier indicated SFP under-estimation
Agency-Specific Verbs and Custom Weights	Much improved correlation; three low-side outliers	Analysis of outliers indicated under-estimation of cost
Agency-Specific Verbs, Custom Weights, and Adjustment of Low-Side Outliers	Greatly improved correlation	SFP estimation using NLP is a valuable tool; some promising results that can be improved with additional data and research

Table 1: Summary of Results

Conclusions

Before stating any conclusions, it is important to acknowledge the limitations of this analysis. Although the results shown in the previous section appear to have a strong correlation, this method is not yet refined enough to produce a re-usable cost estimating relationship (CER). This limitation and caveat is necessary for two reasons:

- The strong correlation was achieved by adjustment of outlier data points. Although each adjustment was made for data-driven reasons, the process guarantees a strong correlation, which may not entirely be a result of the underlying relationship. The R-squared values in this analysis should be used as a way of evaluating the relative strength of each approach, and not interpreted as an unbiased measure of correlation.
- The results are based on a weighting methodology that doesn't align with SiSE or any other established SFP counting method. Therefore the sizing metric produced is not a standard one that can be compared against other SFP results.

With these limitations in mind, this analysis produced some valuable conclusions:

- For SFP analysis, agency-specific verbs improve the results.
- The analysis described in this paper can be a valuable tool in assessing an existing estimate.
- This analysis is an effective way of identifying and exploring outliers.
- This method is best viewed as a way to determine relative not absolute size. The SFP weights that generated the best results are not consistent with any other industry standard way of measuring SFP. However, because the NLP algorithm treats each data point in a consistent way, the results can be viewed as an effective measure of relative size.
- The results indicate that an NLP generated SFP count can be a useful tool that will only improve as additional data and additional analysis becomes available.

Next Steps

This analysis produced some useful results, but it also indicated that much work needs to be done in order to improve the process and make it more generally applicable. In particular, the following next steps are suggested:

- Use this approach on a larger data set that includes additional programs. A larger data set might allow the analyst to develop the verbs and weighting values on a training set of data. A customized method could then be tested on a separate set of data.
- Apply this approach using actual cost history as the Y-axis variable.
- Continue to refine the agency-specific verb list and weights. Additional data and analysis will allow calculation of verb weights that have more utility than the 1, 3, 5 scale used in this analysis.