ABSTRACT
Software cost estimating is a challenging effort when there is little information known about a program early in the lifecycle. Historically, the Department of Homeland Security (DHS) has had difficulty in properly sizing software development efforts for Life Cycle Cost Estimates (LCCEs). In 2017, the DHS Cost Analysis Division (CAD) piloted the Simple Function Point Analysis (SFPA) methodology, tying high level requirements in existing Acquisition documentation to a standard sizing metric. After demonstrating successes with several programs, additional value of SFPA results for program management beyond cost estimating was explored. This paper and presentation will cover the SFPA methodology, requirements definition and analysis, and how functional size can be used to effectively manage program cost, schedule, and performance. We will use real DHS programs, policies, and lessons learned to demonstrate the benefits of SFPA as a secret ingredient for program management success and describe how to engage with program managers to employ this tool in their programs. Finally, we will discuss our future research efforts and initiatives to implement SFPA across federal acquisitions. We believe this to be an innovative and exciting way to estimate and manage software development programs in any organization.

Keywords
Software cost estimating, Agile, Simple Function Points Analysis, Software sizing, Program Management, Stakeholder Engagement, Requirements, Functional Sizing

1. INTRODUCTION
1.1 Agile Requirements
Developing agile requirements is a different process than the traditional waterfall approach used in U.S. Government software acquisitions [1]. Gone are the days of defining and finalizing hundred-page requirements documents before typing a single line of code; gone are the months of development that may or may not deliver software that functions as originally intended. Instead, the Agile Manifesto prioritizes continuous delivery of working software, which often requires changing requirements late in development to suit the customer’s needs. [2] This is a momentous paradigm shift in the acquisition of new IT software systems.

There have been difficulties in implementing Agile development approaches across the Government, as many Agile principles conflict with established processes. For example, Agile only plans work over weeks or months; however, the federal budget process requires funding requests to be prepared for submission to Congress nearly two years before those funds would be
received. As a result, acquisition programs must be able to estimate future software development work based on vague or unknown requirements. As seen in Figure 1 the agile scrum development process is incremental and iterative, the key premise on delivering working software to the customer for continuous feedback and refinement. To address the need for flexible, user-centric requirements that still meet the federal acquisition regulations associated with taxpayer funding, cost estimators need a way to estimate software development from flexible high level agile requirements.

![Agile SCRUM Process Chart](image)

**Figure 1: Agile SCRUM Process Chart, adapted from the SCRUM Body of Knowledge (SBoK, 2017)**

### 1.2 Software Cost Estimating
The basics to estimating the cost of software development break down to the simple equation:

\[
Effort = Size \times Throughput
\]

where Size is equal to the scope of working software produced to meet the customer’s needs, expressed through a standard unit of measurement, and Throughput is the rate at which software developers can design, code, test, and deliver working software to the client. Effort can then be easily converted to cost as a build-up from labor rates and quantities.

Of the two components depicted above, throughput is easier to quantify. Throughput is typically presented as a “per time” metric, such as Lines of Code per Hour, Function Points per Month, or Story Points per Sprint, and can be determined from historical development rates. Size is much more difficult to measure, as there is a myriad of ways to calculate the size of the software system being developed; below is a brief discussion of three common size measurements.

#### 1.2.1 Story Points
In *Agile Estimating and Planning*, Cohn wrote: “Story points are a unit of measure for expressing the overall size of a user story, feature, or other piece of work. When we estimate
with story points, we assign a point value to each item. The raw values we assign are unimportant. What matters are the relative values. A story that is assigned a two should be twice as much as a story that is assigned a one. It should also be two-thirds of a story that is estimated as three story points.” [3] Story points, as stated by Cohn, are a subjective unit of measure. An individual Agile team assigns them to identify the relative difficulty of various tasks, and usually require a prioritized product backlog of user stories, which is not developed until much later in the acquisition process; therefore, it is difficult to aggregate and compare between development teams and different programs. Additional technical information is necessary to derive a normalized relationship between the effort performed by different teams and their assigned story points. While we believe that story points are an important and indispensable part of the individual agile team planning process for sprints and backlog burn-down, story points do not lend themselves to long-term program management.

1.2.2 Software Lines of Code (SLOC)
Most commonly used within the Department of Defense, Software Lines of Code (SLOC) is the physical count of lines of text in the source code. As stated in the 2019 Defense Innovation Board Metrics for Software Development “The current state of practice within the DoD is that software complexity is often estimated base on the number of source lines of code (SLOC), and its rate of progress is measured in terms of programmer productivity. While both of these quantities are easily measured, they are not necessarily predictive of cost, schedule, or performance.” [4] According to code.org, “Of course, every engineer knows that ‘lines of code’ is a silly measure…No software engineer measures the value of their work in lines of code. In fact, the best-designed programs often have the simplest designs and the fewest lines of code.” [5] SLOC may provide some general idea of the scope of a development effort for an Rough Order of Magnitude estimate, but variations in code length due to type of programming language and the coding efficiency of individual developers means accuracy of SLOC estimates are likely inconsistent.

1.2.3 Function Points
Capers Jones stated: “Function Point metrics are the most accurate and effective metrics yet developed for software sizing and also for studying software productivity, quality, costs, risks, and economic value. Unlike the older ‘lines of code’ metric Function Points can be used to study requirements, design, and in fact all software activities from development through maintenance.” [6] The Function Point, developed by IBM’s Allan Albrecht in 1979, is a standard unit of measurement based on how a system uses information. Function Points are agnostic of programing language or development methodology (e.g., waterfall, agile). Since its inception, the methodology governed by the International Function Point User’s Group (IFPUG) established in 1986 is an International Organization for Standardization (ISO) standard. A Function Point is consistent regardless of who performs the count or what the system does. While the counting process involves some interpretation, experienced Function Point counters can produce counts for a system within 5% of each other. Figure 2 is a pictorial representation of system components that need to be understood when calculating Function Points. Drawbacks
to Function Points can include the time required to learn the counting practice, time to conduct a full count, and the effort required to obtain certification.

1.3 Challenges to the Department of Homeland Security

Each year, DHS invests billions of taxpayer dollars into everything from helicopters for Customs and Border Protection (CBP), vessels for the U.S. Coast Guard, baggage screening equipment for the Transportation Security Administration (TSA), and complex software systems for such purposes as administering FEMA grants, processing U.S. citizenship applications, and monitoring the enforcement of illegal immigration. In a recent report by the Government Accountability Office (GAO) in May of 2017 it was noted that “in fiscal year 2016, the department’s IT budget of approximately $6.2 billion was the third largest in the federal government.” [7] Like many other federal agencies in the U.S. Government, the Department of Homeland Security (DHS) has struggled with estimating the cost of and establishing realistic schedules for large IT programs.

One of the primary challenges experienced by DHS is accurately estimating the size of software development efforts. Many of these efforts result in public-facing systems and have many stakeholders with various needs, leading to complex sets of requirements. In the cost estimating field, developing an estimate is not conceptually difficult, as estimates are often just build-ups of labor; the justification of those inputs is what presents the major challenge. As mentioned earlier, Agile development principles often conflict with established processes in the traditional acquisition lifecycle framework. Development teams will continually shift or add requirements as directed by the customer to deliver working software, but how can a program tell that it has completed what it originally set out to do? Understanding the true scope of programs is the missing piece to improving program management practices.
1.4 Charge by the Under Secretary for Management
In 2017, the DHS Under Secretary for Management (USM) charged the Cost Analysis Division (CAD) under the DHS Office of the Chief Financial Officer (OCFO) to find a way to improve cost estimates for Agile software development programs. There were two primary objectives:

1. Enhance the credibility and accuracy of a software development estimate and
2. Decrease the time required to develop the estimate.

At the time, DHS had designated five software development programs as pilots for implementing agile processes and best practices and providing lessons learned for other DHS endeavors. In addition to being highly visible major acquisitions, these programs were at various stages of the acquisition lifecycle and had experienced common challenges with cost, schedule, and performance. This provided a timely opportunity for CAD to expand its technical knowledge of software development and attempt some novel estimating methods. From discussions with industry and Government partners, CAD learned of Simple Function Points and identified them as a promising solution to their current dilemma.

2. SIMPLE FUNCTION POINT ANALYSIS (SFPA)
2.1 SFPA Process Overview
The SFPA methodology combines functional software sizing (i.e., quantifying Function Points based on business function/transaction types, system interfaces, and requirements counts from high-level acquisition documentation) and functional software productivity (e.g., hours per function point) to determine Agile software development costs. The Simple Function Point Association, an international non-profit association formed in Italy to evolve and promote the method of Simple Function Points (SFPs), has published a SFP measurement manual, SFP counting templates, and examples of SFP counting, which CAD used as the basis for the SFPA pilot. [8] The Simple Function Point method was acquired by the International Function Point User’s Group (IFPUG) in September of 2019, which indicates growing interest in the methodology. [9]

The SFPA functional sizing methodology leverages the process of IFPUG’s ISO certified counting practices manual. The IFPUG counting practice estimates the size of software based on an understanding of the system’s lowest-level business transactions (External Inputs, External Outputs, and External Inquires) and data storage interfaces (Internal Logical Files and External Interface Files) as seen in Figure 2. The IFPUG counting method requires the counter to quantify the complexity of each transaction or data storage component based on a set of criteria and depending on the component type and complexity a Function Point value is assigned. The SFPA method was developed as an alternative to this lengthy and labor-intensive Function Point counting process.

The SFPA method maps the IFPUG components to two groups – Transactions (i.e., Create, Update, Delete, Report, and Read), which map to External Inputs (EI), External Outputs (EO), or External Queries (EQ), and Logical Data Groupings (i.e., Saves), which map to Internal Logical
Files (ILF) and External Interface Files (EIF). Figure 3 illustrates this mapping between the IFPUG components and their Function Point counts and the SFPA components and weightings.

When functional requirements are documented, they are expressed as action verbs (e.g., “submit,” “maintain,” “receive”), which can be decomposed to one or more of the SFPA components. Work done by Function Point counting experts produced a lexicon of 140+ action verbs and their associated components. The appropriate weighting factors are then applied to these components to produce a total SFP count. To understand system business transactions and estimate software SFP requirements, CAD uses a program’s Concept of Operations (CONOPS), a high-level acquisition document developed early in the acquisition lifecycle that describes how the completed system should function. The CONOPS is reviewed and validated by the DHS requirements and technical communities to ensure all required capabilities are captured before a program moves further through the acquisition lifecycle. SFPA leverages the action verbs used in the CONOPS to quickly estimate a function point count from the written functional requirements.

After the initial SFP estimate is calculated, additional factors and risk may be applied to account for software growth, complexity, and program uniqueness. The final SFP count should be validated by the program office and the appropriate technical communities to ensure a consistent interpretation of the requirements used for the estimate. CAD uses historical and industry data to determine an SFPA throughput rate to be used with the final SFP count to calculate the total software development effort required for the program.

2.2 What is a Good Requirement?
It should be obvious that successfully conducting SFPA depends on a solid understanding of the functionality of the system being developed. It is impossible to assess the accuracy of any sizing estimate without understanding what a developed system does, and much of this understanding comes from written program documentation. With the shift to Agile methods also comes the mindset that documentation is secondary to developed software due to constantly changing requirements; therefore, it is crucial that the early, high-level program requirements are well written. There are many factors to consider when writing requirements [10]:

1. **User’s perspective** – SFPA focuses on functional size, i.e. the actions that the system performs when it is operational. Good requirements should capture those actions. Non-
functional requirements such as availability, maintainability and reliability, while important considerations during development, do not factor directly in SFPA.

2. **Unique / One action per requirement** – A good requirement should only describe one individual action. Including multiple actions in a requirement may cause confusion and lead to effort being underestimated.

3. **Clear and concise actions** – In the Agile spirit, requirements should be direct to keep documentation minimal. Keeping the written requirement concise also helps ensure that the functions are easily recognized.

4. **Consistent level of detail** – Good requirements for SFPA should be described at similar levels of detail. If a requirement is overly detailed or broken into multiple smaller actions, it may lead to that effort being overestimated relative to others.

5. **Testable / Verifiable** – Good requirements should have criteria to determine if the requirement has been developed properly and the capability met. This allows for development progress to be accurately tracked.

Various artifacts such as the CONOPS or a Functional Requirements Document (FRD) are produced as a program increases in maturity and describe requirements at differing levels of detail. CAD is exploring using other DHS document sources for SFPA sizing to include the FRD, Requirements Traceability Matrix (RTM), and the Software Requirements Document (SRD). Analysts can also derive SFPA from user stories pulled from a project management tools such as JIRA for sizing. Other federal agencies can utilize similar high-level requirements documentation if the organization does not employ CONOPS to the detailed software business function level. Performing SFPA with each of these documents will produce different sizing estimates. Work is ongoing to investigate which of documents provides the most useful information to accurately estimate functional size.

3. **But Wait, There’s More!**

After SFPA is conducted, it produces a number to quantify the functional requirements for a development effort. This number, when used by a cost estimator, provides a justifiable input for estimating cost. But after the effort to produce just one number, is that it? No! There are many ways that a SFP count, when utilized effectively by a program office, can provide maximum value by influencing many aspects of program management activities. We highlight several questions that program managers need to answer, and ways we have explored using SFPA to address them.

3.1 **Developing Schedules – “When can this be delivered?”**

One of the first items that a program needs to have agreement on is when the software needs to be delivered. There have been studies conducted that provide metrics on development rates (FP/team-month, etc.). Since SFPs are a standardized sizing for development effort, they can be used with historical rates to project the time required to complete a set of requirements given a known set of resources. If a schedule is already assigned to a program, the schedule analysis can also be performed to assess the reasonableness of development timelines. The program can then
justify to decision makers why assigned milestones may be unrealistic and should be delayed or re-evaluated.

3.2 Estimating Resources – “What staff is needed?”
If timelines are already established, SFPA can provide a program with a way to identify the resources required to meet those deadlines. Using software development rate metrics with the SFP estimate and assigned milestones, an analyst can calculate the staff quantity required to meet the desired schedule. If current quantities are insufficient, this provides solid justification for additional program resources and funding.

3.3 Planning Agile Sprints – “What is everyone’s workload?”
If SFPA is conducted on good requirements, it provides the estimated effort required to develop each those requirements. Since each requirement is objectively quantified, Agile teams can appropriately divide tasks when planning sprints and minimize over-assigning work. Program managers can use this to assess team throughput and ensure that they are all producing similar amounts of functionality.

3.4 Reviewing Vendor Proposals – “Is this bid realistic?”
We have described how programs can use an SFP count to assess internal schedules and resourcing. This analysis can also be applied to vendor proposals for software development services to see if the scope of work is mutually understood between the Government and contract offerors. An SFP estimate can provide a quick cross-check to the overall amount of effort proposed, as well as gauge reasonableness of the delivery timeline and the staffing used to meet those dates. This will allow programs to better evaluate best value when awarding contracts.

3.5 Tracking Progress – “How is the program performing overall?”
When programs utilize SFPA in the ways outlined above, they can combine the analyses to produce a baseline for accurate tracking of development progress. Figure 4 illustrates how this information can be organized. An initial cumulative SFPs completed line can be projected using assumed development rates and proposed staff. Plotting cumulative SFPs completed after each sprint against this initial projection can provide a program manager with valuable information. They can identify current development progress and see if this is trending as planned. Deviations will provide early indication of potential issues and allow the program to react preemptively. Establishing a visual representation of progress also provides an instrument to initiate useful communication with leadership and focus conversation on issues that require attention.
3.6 DHS Examples of SFPA Use

The pilot of SFPA methodology on DHS programs has resulted in successes in various aspects of program acquisition processes. Three examples are highlighted in this section.

3.6.1 Program A

Program A was one of the first pilot programs for testing the SFPA methodology. The Program was a Level 2 ($300M-$1B total lifecycle cost) agile development program for a public facing web-based system. The program’s CONOPS clearly detailed the user requirements via functional capabilities statements, which made it very easy to apply SFPA to estimate functional size. The estimated SFPs were adjusted for risk and used with current throughput rates to update the program’s LCCE for Department approval. This helped to prove the methodology’s viability for DHS programs; the program’s requirements statements are some of the primary examples CAD uses to educate other programs how to effectively write requirements for SFPA.

3.6.2 Program B

Program B is a Level 1 ($1B+ total lifecycle cost) program in the obtain phase of the acquisition lifecycle for a very complex, critical system with large computing/storage requirements and interfaces with systems both internal to DHS and external to stakeholders and partners. The program estimated Function Points for the system using the COSMIC sizing methodology [11]. As part of an Independent Cost Assessment (ICA) of the program’s LCCE, CAD used SFPA to size requirements described in one of the program’s capability documents. The software development costs calculated through SFPA were within 8% of the program’s estimate, a
reasonable range for an independent cross-check. CAD’s ICA and the approval of Program B’s LCCE demonstrated the value of SFPA for developing a software size estimate quickly and with similar accuracy to other standardized Function Point counting methods.

In addition to using Function Points in the development of the LCCE, Program B also implemented a progress tracking chart as described in Section 3.5. The chart is presented to stakeholders whenever the program meets with the DHS Acquisition Review Board for milestone decisions or program reviews. Trends projected in the chart have been consistent with progress observed as the program continues development activities.

3.6.3 Program C
Program C is a Level 2 program in the obtain phase of the acquisition lifecycle for a system that streamlines many unique process workflows into a single management platform. The program recently updated their LCCE to reflect a shift in acquisition approach to agile software development. CAD was able to collaborate with the program to apply SFPA on business functions described in the program’s CONOPS; the use of SFPA did not require Program C to create any new acquisition documents specifically for the LCCE update. Part of the program’s updates also included re-baselining schedule milestones due to lower staffing levels than planned. CAD used the SFP estimate along with throughput data and agile team quantities to project system development and identify a new date to reach Full Operational Capability. The recommended milestone dates were consistent with the schedule provided by the program’s development contractor. The work done with Program C showed SFPA’s ability to be performed on requirements regardless of development approach, as well as reduce program overdependence on contractors for program management activities.

4. CAD Successes with SFPA Implementation
Over the last few years, there has been a large amount of progress made by CAD in implementing SFPA in DHS. We highlight several accomplishments, as well as ongoing efforts to improve, refine, and expand the SFPA methodology to provide maximum value to the Department.

4.1 Leadership Support
DHS Leadership has supported the use of SFPA as a methodology to estimate software development sizing. They have recognized the objective nature of SFPs and their standard calculation, as well as the link to functional requirements. Tracking SFPs has begun to focus discussions of development progress on capabilities delivered rather than deadlines promised. In a May 2019 memo from the DHS Acting Chief Financial Officer, all new or re-baselining Major Acquisition programs are now required to use SFPA for estimating software development effort.
4.2 Joint Agile Software Innovation Cost IPT (JASI CIPT)
The Joint Agile Software Innovation (JASI) Cost Integrated Product Team (IPT) was founded in 2018 by representatives from DHS, the National Security Administration (NSA), and the National Geospatial-Intelligence Agency (NGA) with three objectives:

1. Develop a pragmatic and defendable approach to estimate and measure software development through SFPA
2. Improve data availability to enhance the credibility of estimates
3. Investigate new approaches to track, measure, and report progress of an agile program throughout its development lifecycle

Through JASI, CAD provided SFPA training to over a dozen different audiences, with more sessions planned. JASI has also expanded to include membership from thirteen federal agencies in the Defense, Intelligence, and Civilian cost communities. These agencies all recognize the potential for SFPA to improve the development of cost estimates and are excited to implement SFPA in their own organizations.

4.3 Adoption by New Acquisition Programs
CAD’s efforts to promote SFPA and provide training in the methodology to current acquisition programs have spread awareness across the department. Several early phase acquisitions have indicated that they are attempting to use SFPA as part of their program planning activities. To date, two DHS programs independently used SFPA to develop their Rough Order of Magnitude estimates.

4.4 Engagement with DHS Stakeholders
CAD is working with acquisition stakeholders across the department to refine, improve and standardize SFPA. CAD is collaborating with the Offices of the Chief Information Officer and Chief Technology Officer to improve and standardize written requirements and develop processes to validate SFP-based LCCEs from a technical perspective. CAD is also engaging with the Office of the Chief Procurement Officer to facilitate collection of valuable performance metrics as part of future Agile development contracts.

4.5 Data Collection
CAD is undergoing efforts with many DHS agile programs to collect data on completed software. Data being collected includes written requirements and respective SFP counts, agile team quantities and composition, effort to develop functional requirements, and actual costs, among others. CAD intends to use the data to refine the number of SFPs assigned to various requirements statements, as well as develop DHS-specific throughput rates to improve size and schedule estimates for future programs. SFP estimates from different requirements documents will also be examined to determine which document type provides the most accurate and appropriate basis of estimate.
5. CONCLUSIONS

5.1 Benefits and Summary
CAD believes SFPA offers many benefits to Agile acquisition programs. SFPA provides a faster, more reliable and repeatable process for cost estimators to produce credible estimates functional size and development effort. The methodology leverages high-level documents created early in the acquisition lifecycle, allowing long-term analysis of system capabilities without being impacted by agile processes that shift development priorities. Lastly, integrating SFPs into other aspects of program management provides additional value to program managers by ensuring all activities are tied to the same requirements and can be communicated consistently to leadership and decision makers.

5.2 Future Work
The SFPA methodology is still a “work in progress.” We seek to improve this methodology based on data and lessons learned experienced by programs as they progress through software development. All CAD efforts referenced in this paper are ongoing, with the hope that SFPA will soon become a standard not only within DHS, but across the Federal Government.

6. ACKNOWLEDGMENTS
The authors would like to thank Mr. David Seaver, Mr. Lyle Patashnick, Mr. Tyrese Johnson, and Mr. Collin Brooks for their mentorship and early work piloting SFPA at DHS. Thank you to Dr. Wilson Rosa for his guidance and to the CAD and DHS OCFO leadership for their support and encouragement to write this paper.

7. REFERENCES


