But Wait, There's More!
Using SiSE for your Cost, Schedule & Performance Needs

Katharine Mann
Ryan Hoang

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 piloted the Simple Function Point Analysis (SFPA) methodology, later renamed to Simple Software Estimation (SiSE), tying high level requirements in existing Acquisition documentation to a standard sizing metric. After demonstrating successes with several programs, additional value of SiSE results for program management beyond cost estimating was explored. This paper and presentation will cover the SiSE 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 SiSE 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 SiSE 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, Software sizing, Program Management, Stakeholder Engagement, Requirements, Functional Sizing

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.

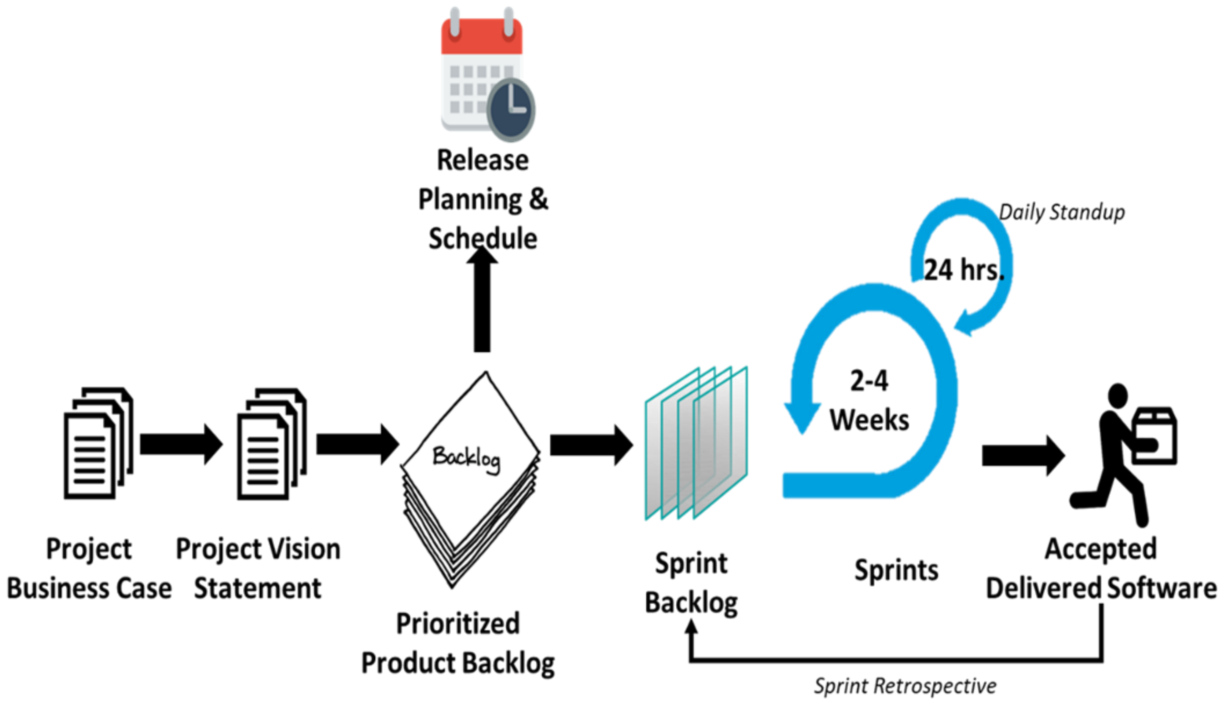


Figure 1: Agile SCRUM Process Chart, adapted from the SCRUM Body of Knowledge (SBoK, 2017)

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 in Figure 1, 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

The Function Point, developed by IBM's Allan Albrecht in 1979, is a standard unit of measurement based on how a system uses information. Capers Jones, leading authority in software estimating, 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] 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.

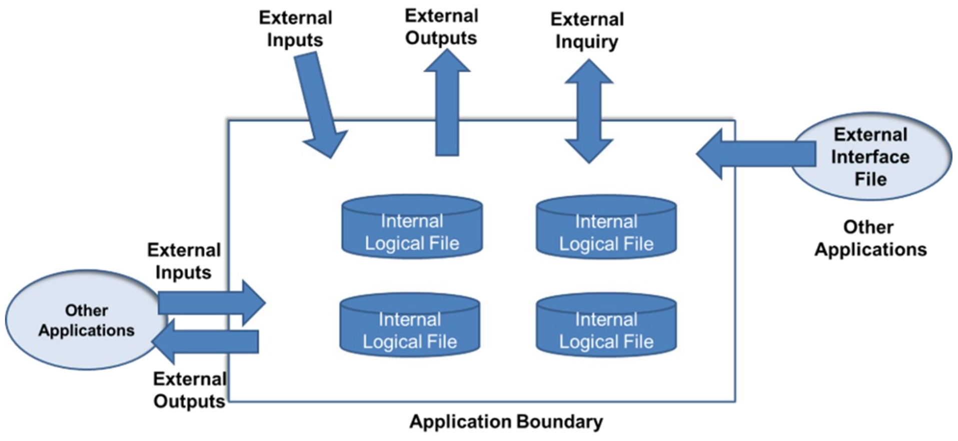


Figure 2: Pictorial representation of Function Point counting components of a system

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. These software systems are used for such purposes as administering Federal Emergency Management Agency (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. 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:

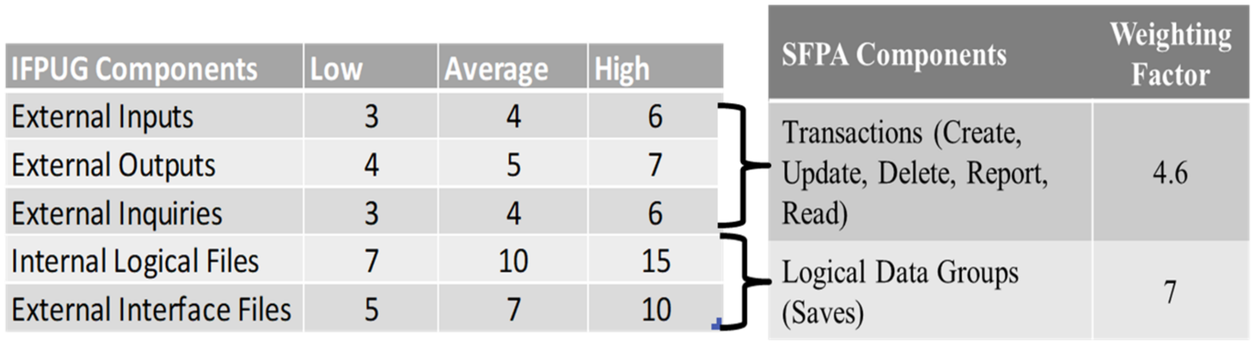
1. Enhance the credibility and accuracy of a software development estimate and
2. Decrease the time required to develop the estimate.

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. The charge by the USM provided a timely opportunity for the Cost Analysis Division to expand its technical knowledge of software development and attempt some novel estimating methods. From discussions with industry and Government partners, the Cost Analysis Division learned of the benefits of functional sizing techniques and identified functional sizing as a promising solution to the current dilemma.

2. SIMPLE SOFTWARE ESTIMATION (SiSE)

2.1 SiSE Process Overview

The Cost Analysis Division developed a custom software cost estimation process called Simple Software Estimation (SiSE), combining the open-source Simple Function Point (SiFP) method as published by Dr Roberto Meli (v1.01) and work pioneered and shared by analysts at the National Security Agency (NSA). [8] The SiSE Estimating



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

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

Figure 3: Mapping between IFPUG and SiSE Components and Weightings

Process combines functional software sizing (i.e., quantifying business function/transaction types, system interfaces, and requirements counts from high-level acquisition documentation) together with software productivity rates (e.g., hours per function point) to determine Agile software development effort and costs. Note that we realize that there are Diseconomies of Scale (DoS) associated with software development estimating; however, we currently assume a simplified linear relationship between software functional size and productivity. 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 underlying methodology. [9]

The SiSE 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. Then, depending on the component type and complexity, a Function Point value is assigned. The SiSE method was developed as an alternative to this lengthy and labor-intensive Function Point counting process.

The SiSE 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 SiSE components and weightings.

The Cost Analysis Division's research illustrates that functional requirements are typically expressed as action verbs (e.g., "submit," "maintain," "receive"). Each requirement can be decomposed into one or more components (groupings of generic transactions and/or data groups) and corresponding weighing factors from the Simple Function Point method. Work done by functional sizing experts produced a lexicon of 140+ action verbs and their associated components. With the associated size for each action verb pre-defined, a functional size estimate for a set of requirements can be produced and totaled quickly.

To understand a software system's business transactions and estimate software requirements, the Cost Analysis Division uses a program's Concept of Operations (CONOPS), a high-level acquisition document developed early in our acquisition lifecycle that describes what functions the completed system will do. 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. The SiSE sizing step leverages the action verbs used in the CONOPS written functional requirements to quickly estimate a Simple Function Point size of the software. Once the initial size estimate is calculated, additional factors and risk may be applied to the estimated size to anticipate software growth, complexity, and program uniqueness.

The program office and the appropriate technical communities should then validate the final size estimate to ensure a consistent interpretation of the requirements used for the estimate. The Cost Analysis Division uses analogous historical and industry data to determine a throughput/ productivity rate used with the estimated Simple FP size to estimate the total software development effort for the program. This is then time-phased across the schedule to estimate the software development cost for a program's Life Cycle Cost Estimate (LCCE).

2.2 What is a Good Requirement?

It should be obvious that successfully conducting SiSE 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** – SiSE 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 SiSE.
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 SiSE 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. The Cost Analysis Division is exploring using other DHS document sources for SiSE sizing to include the FRD, Requirements Traceability Matrix (RTM), and the Software Requirements Document (SRD). Analysts can also derive SiSE from user stories pulled from 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 SiSE with each of these documents will produce different sizing estimates. Work is ongoing to investigate which requirements documents provide the most useful information to accurately estimate functional size.

3. BUT WAIT, THERE'S MORE!

As part of the Simple Software Estimating process, the Simple FP estimate quantifies the size of the functional requirements for a development effort. This number, when used by a cost estimator, provides a justifiable input for estimating cost. But after performing this sizing effort to produce just one number – the Simple FP estimated size, is that it? No! There are many ways that our size estimate, when utilized effectively by a program office, can provide maximum value by influencing many aspects of program management activities.

3.1 Developing Schedules – "When can this be delivered?"

One of the first items that a program needs to have agreement on is the realistic duration of the software development effort. There have been studies conducted that provide metrics on development rates for functional sizing (ex: FP/team-month, etc.). Using a standard approach like Simple FP with an appropriate productivity rate to estimate our software development effort (in hours or person months) we can then estimate, using historical development rates, the schedule duration to complete the software development. If a schedule was already assigned to a program, this schedule estimation can assess the reasonableness of existing development timelines. The program can then justify to decision makers why pre-assigned milestones (or deadlines!) may be unrealistic and should be delayed or re-evaluated.

3.2 Estimating Resources – "What staff is needed?"

If timelines are already established, SiSE can assist program management with an easy way to quantify how many resources will be required to meet those deadlines. Using software development rate metrics together with the Simple FP size estimate the assigned milestone dates, an analyst can estimate the required team size and quantity to meet the desired schedule. If the available team size is insufficient, the analysis provides solid, objective justification to ask for additional program resources and funding.

3.3 Planning Agile Sprints – "What is everyone's workload?"

If based on good requirements (i.e., unambiguous, clearly stated, functional requirements), a cost analyst can use the Simple FP estimate, with a relevant productivity rate to estimate the effort required to develop each of those requirements. Because each requirement is objectively quantified, Agile teams can appropriately divide tasks when planning sprints and minimize potentially over-assigning work. Program managers can use also use this approach to assess team throughput and ensure that they are all producing similar amounts of functionality. This approach is far more objective and applicable across teams than the alternative velocity metric (expressed as Story Points per Sprint) typically used by Agile development teams.

3.4 Reviewing Vendor Proposals – "Is this bid realistic?"

This paper provides an overview on how programs can use SiSE to assess internal schedules and resourcing. This estimating process can also be applied to assessing vendor proposals for software development services, to validate that the scope of work is mutually understood between the Government and contract offerors. The Simple FP sizing 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 proposed to meet those dates. This will allow programs to better evaluate best-value proposals when awarding contracts.

3.5 Tracking Progress – "How is the program performing overall?"

The results of SiSE combines with other noted analyses to produce a baseline for accurate tracking of development progress. An initial cumulative Simple FPs "estimate to complete" chart can be plotted to project completion dates and effort, using assumed development rates and proposed staff. Plotting cumulative, delivered Simple FPs completed after each sprint against this initial projection can provide a program manager with valuable information in the form of a Burn Up chart. Program and project managers can track current development progress and see if the project progress is trending as planned. Deviations will provide an early indication of potential issues and allow the program to react pre-emptively. Establishing a visual representation of such progress also provides an instrument to initiate useful communication with leadership and focus the conversation on issues that require attention. An example of a visual representation can be seen in Figure 4.

3.6 DHS Examples of SiSE Use

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

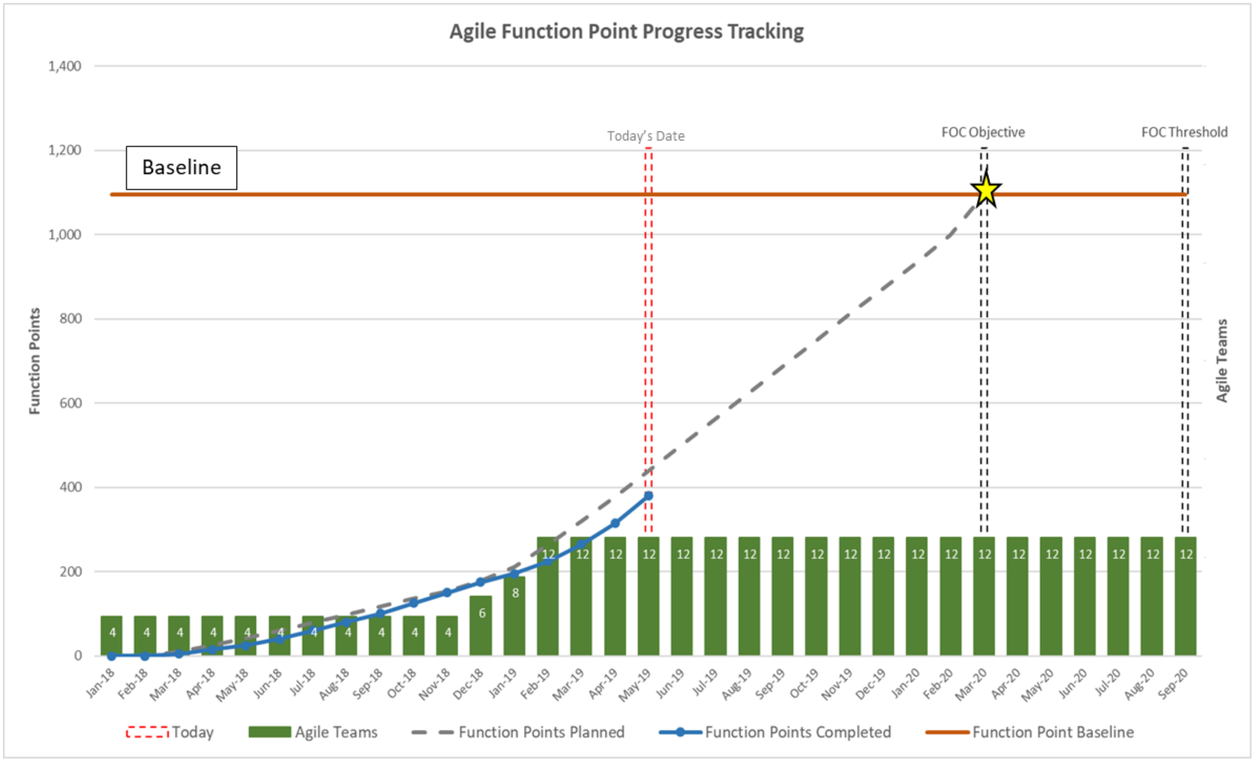


Figure 4. Example Function Point Tracking Chart

3.6.1 Program A

Program A was one of the first pilot programs for testing the Simple Function Point Analysis (SFPA) methodology, the predecessor to SiSE. 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 SiSE to estimate functional size. The estimated SiFPs were adjusted for risk and used with current throughput rates to update the program's LCCE for Department approval. Program A's LCCE helped to prove the methodology's viability for DHS programs; the program's requirements statements are some of the primary examples the Cost Analysis Division uses to educate other programs how to effectively write requirements for SiSE.

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, the Cost Analysis Division used SiSE to size requirements described in one of the program's capability documents. The software development costs calculated through SiSE were within 8% of the program's estimate, a reasonable range for an independent cross-check. The Cost Analysis Division's ICA and the approval of Program B's LCCE demonstrated the value of SiSE 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. The Cost Analysis Division was able to collaborate with the program to apply SiSE on business functions described in the program's CONOPS; the use of SiSE 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. The Cost Analysis Division used the Simple Function Point 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 SiSE's ability to be performed on requirements regardless of development approach, as well as reduce program overdependence on contractors for program management activities.

4. SUCCESSES WITH SiSE IMPLEMENTATION

Over the last few years, there has been a large amount of progress made by the Cost Analysis Division in implementing SiSE in DHS. We highlight several accomplishments, as well as ongoing efforts to improve, refine, and expand the SiSE methodology to provide maximum value to the Department.

4.1 Leadership Support

DHS Leadership has supported the use of SiSE as a methodology to estimate software development sizing. They have recognized the objective nature of Simple Function Points and their standard calculation, as well as the link to functional requirements. Tracking SiFPs 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 functional sizing 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 Functional Sizing
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, the Cost Analysis Division provides SiSE 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 SiSE to improve the development of cost estimates and are excited to implement SiSE in their own organizations. JASI CIPT won the 2019 Team Achievement award through the Washington Capital Area Chapter of the International Cost Estimating and Analysis Association (ICEAA) in recognition of the collaborative efforts of the team.

4.3 Adoption by New Acquisition Programs

The Cost Analysis Division's efforts to promote SiSE 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 SiSE as part of their program planning activities. To date, two DHS programs independently used SiSE to develop their Rough Order of Magnitude estimates.

4.4 Engagement with DHS Stakeholders

The Cost Analysis Division is working with acquisition stakeholders across the department to refine, improve and standardize SiSE. The Cost Analysis Division 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 Simple Function Point-based LCCEs from a technical perspective. The Cost Analysis Division 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

The Cost Analysis Division is undergoing efforts with many DHS agile programs to collect data on completed software. Data being collected includes written requirements and respective Simple FP counts, agile team quantities and composition, effort to develop functional requirements, and actual costs, among others. The Cost Analysis Division intends to use the data to refine the number of Simple FPs assigned to various requirements statements, as well as develop DHS-specific throughput rates to improve size and schedule estimates for future programs. Simple FP 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

The Cost Analysis Division believes SiSE offers many benefits to Agile acquisition programs. SiSE provides a faster, more reliable, and repeatable process for cost estimators to produ63ce credible estimates of 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 can shift development priorities. Lastly, integrating functional sizing into other aspects of program management provides additional value to program managers by tying all activities to the same requirements and can be communicated consistently to leadership and decision makers.

5.2 Future Work

The SiSE methodology is still a "work in progress." We seek to improve this methodology based on data and lessons learned by programs as they progress through software development. All Cost Analysis Division efforts referenced in this paper are ongoing, with the hope that the SiSE process will soon become a standard not only within DHS, but across the U.S. 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/SiSE at DHS. Thank you to Ms. Carol Dekkers for her Function Point SME knowledge and enhancement of the methodology. Thank you to Dr. Wilson Rosa for his guidance and to the Cost Analysis Division and DHS OCFO leadership for their support and encouragement to write this paper.

References:

[1] Government Accountability Office, "Software Development Effective Practices and Federal Challenges in Applying Agile Methods," United States Government Accountability Office, Washington DC, 2012.

[2] Agile Manifesto, "Manifesto for Agile Software Development," 2001. [Online]. Available: http:// agilemanifesto.org/. [Accessed 30 January 2020].

[3] M. Cohn, Agile Estimating and Planning, Upper Saddle River: Pearson Education, Inc., 2006.

[4] Defense Innovation Board, "Defense Innovation Board Metrics for Software Development," 9 July 2018. [Online]. Available: https://media.defense.gov/2018/Jul/10/2001940937/-1/-1/0/DIB_METRICS_FOR_SOFTWARE_DEVELOPMENT_V0.9_2018.07.10.PDF. [Accessed 9 February 2020].

[5] Code.org, "How many lines of code?," 2020. [Online]. Available: https://code.org/loc. [Accessed 30 Janaury 2020].

[6] C. . Jones, "Function points as a universal software metric," ACM Sigsoft Software Engineering Notes, vol. 38, no. 4, pp. 1-27, 2013.

[7] Government Accountability Office, "Progress Made to Implement IT Reform, but Additional Chief Information Officer Involvement Needed," 18 May 2017. [Online]. Available: https://www.gao.gov/products/gao-17-284. [Accessed 31 January 2020].

[8] Simple Function Point Association, "SiFPA," 2020. [Online]. Available: http://www.sifpa.org/en/metodo/. [Accessed 30 January 2020].

[9] International Function Point Users Group, "IFPUG Acquires the Simple Function Point Method," 12 September 2019. [Online]. Available: https://www.ifpug.org/ifpug-acquires-the-simple-function-points-method/. [Accessed 30 January 2020].

[10] D. French and C. Dekkers, "From Point A to Point Estimate: How Requirements Become Function Points," in ICEAA Professional Development & Training Workshop, Tampa, 2019.

[11] COSMIC, "COSMIC-Sizing," 2020. [Online]. Available: https://cosmic-sizing.org/ . [Accessed 18 February 2020].

***Katharine Mann**, CCEA, is an Operations Research Analyst for the Department of Homeland Security, Cost Analysis Division (CAD) IT / Agile Software Development Team. Before joining DHS, Ms. Mann supported the NATO Communication and Information Agency (NCIA) in Brussels, Belgium, and various DoD programs. She is the Outreach Chair of the Washington Capital Area Chapter of ICEAA and Secretary of the Joint Agile Software Innovation (JASI) Government IPT. Ms. Mann has Undergraduate and Master's degrees in Industrial and Systems Engineering from Virginia Tech.*

***Ryan Hoang**, CCEA, is an Operations Research Analyst for the Department of Homeland Security, Cost Analysis Division (CAD), IT / Agile Software Development Team. Since 2015, he has provided cost support to many major IT acquisition programs across DHS. He has B.S. and M.Eng. degrees in Chemical Engineering from Cornell University.*

Submissions:

Prior to writing or sending your manuscripts to us, please reference the JCAP submission guidelines found at

www.iceaaonline.com/publications/jcap-submission

Kindly send your submissions and/or any correspondence to JCAP.Editor@gmail.com

International Cost Estimating & Analysis Association

4115 Annandale Road, Suite 306 | Annandale, VA 22003

703-642-3090 | iceaa@iceaaonline.org