Sizing Agile Software Development Programs

Authors

Bob Hunt, NSI
Heather Meylemans, NAVAIR
Denton Tarbet, NSI
Chad Lucas, NSI
Rainey Southworth, NSI

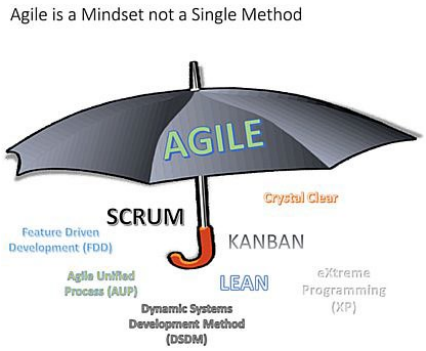
Abstract

The most significant factor impacting software development cost is its scope and complexity.[1] Scope, i.e. features and functionality of the software, implies size or volume of software to be developed. Therefore, Software size estimation is a critical element in software cost estimation. Source Lines of Code (SLOC) has been a predominate metric for estimating software size. As Agile has become more prevalent, the use of SLOC as a software size metric has come under more scrutiny. This paper compares Physical, Relative, and Functional size alternatives including SLOC, Tee Shirt, and Functional size alternatives. The paper will discuss advantages and disadvantages of each alternative. The presentation will provide some emerging metrics for assessing size metrics. Since many automated models convert functional size to physical size, the presentation will address techniques to accomplish "backfiring."

[1] https://fullscale.io/blog/12-factors-that-affect-software-development-cost/

Paper Outline

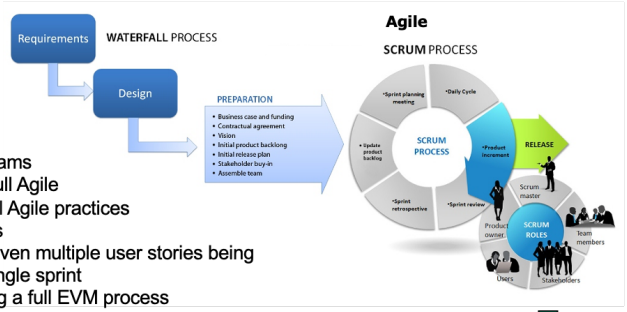
1. AGILE DEVELOPMENT OVERVIEW
    a. In this paper when we use the term "agile, we include all forms of Agile and iterative development. Stories, features, story points, and feature points to reflect the same concept, recognizing that a "feature" typically may be used in a different context than a "story." Specifically, in large federal programs, "features" generally represent a larger concept than "stories." We do believe that the application of estimating, management, and tracking practices can significantly and positively impact the success and cost of federal programs.
    b. Agile is a mindset and it refers to the methods and best practices for organizing projects based on the values and principles documented in the Agile Manifesto. There is no



one way to implement Agile. Kanban, Scrum, Extreme Programming (XP), Feature-driven development, Dynamic Systems Development Method, Crystal, Lean, and Adaptive Project Framework are all examples of agile software development. (The illustration is taken from David DeWitt of Galorath.)

c. There are two classes of federal agile software development programs. Programs that are evolving on an incremental basis that generally follow the commercial Agile practice, and large "transformational" programs creating completely new capabilities. In these "transformational" programs a "Hybrid-Agile" approach is often applied with longer sprints and larger conceptual stories/features. Figure 1. Practical Applications of Agile Full or Hybrid Agile (Water-Scrum-Fall) Development, presents this concept.

Practical Applications of Agile
Full or Hybrid Agile (Water-Scrum-Fall) Development



Two classes of Federal programs
- Incremental programs – Full Agile
  - Follow the commercial Agile practices
    - Small user stories
    - Single sprint, or even multiple user stories being completed in a single sprint
  - Generally, not applying a full EVM process
- Transformational programs – Hybrid Agile
  - Creating completely new capabilities
  - "Hybrid-Agile" approach applied
    - Longer sprints
    - Larger conceptual stories/features
    - Full EVM process.

Testing and Sustainment (sometimes in the Sprint sometimes a separate activity)

Successfully estimating the cost and schedule of a software development program remains a challenge. In Jack Flynn's internet article, 16 Amazing Agile Statistics (2023), He presents the following:

- Agile is an increasingly popular software building methodology,
- At least 71% of U.S. companies are now using Agile,

- Agile projects have a 64% success rate,
- Waterfall only has a 49% success rate,
- Agile projects are nearly 1.5X more successful than waterfall, and
- Scrum is the most popular Agile framework, with 61% of respondents from 76 countries reporting that they use it.

2. FUNDAMENTALS OF SOFTWARE ESTIMATION

a. In the late 1960's and early 1970's, analytic equations based on Lines of Code data were derived by Putnam, Jenson, Boehm, Galorath, and others. There was general agreement that effort was a function of size.

Effort (months)=A*(size of the program)**B

b. The early COCOMO formula was E=3.2*(KSLOC)**1.05. Today the exponent varies in commercial models from about 0.9 to 1.2.

c. Over time databases, software tools, productivity factors, and complexity factors have significantly affected the fundamental estimation equations and the models have become more complex. Most automated models are adjusted to account for Agile practices. SEER by Galorath and TruePlanning by Unison are two examples. A summary of these models is included in the Appendix.

d. With respect to Agile there have been five fundamental cost estimation approaches. The approaches are similar in that all require estimate of scope, i.e. size, estimate

i. Many Agile programs are awarded as fixed price contracts, so the simple methodology is labor rates times quantity. While this approach can be applied for budgetary purposes, it does not tell the cost analyst much about the final cost or probability of success.

ii. From "scrum" based approaches, a simple build-up approach based on averages can be applied as; Sprint Team Size (SS) x Sprint length (Sp time) x Number of Sprints (# Sprints).

iii. A structured approach based on established "velocity" (the average rate at which stories are completed) can be applied. This approach is most often used internally by the developer since detailed/sensitive data are available to them. The cost analyst generally needs several iterations of this data to apply a "velocity" approach.

iv. An automated model such as NEMO, SEER, COCOMO, TruePlanning, SLIM, or … can be applied. This approach based on a size metric (Physical, Relative, or Functional size) and a historical database. The models assume there is a fixed relationship between size and effort, e.g. Effort =*A**(Size Metric) ^*B***C* where A is a constant, B is the non-linear scaler, and C is a combination of Environmental factors. Results are then modified by current trends and analyses. Total effort can be distributed by applying a probability distribution. The advantage of this approach is that it is based on a historical database.

v. Many developers will utilize an analogy/Factor/Complexity approach based on data generated from their professional experience or actual iterations. The use of Tee-Shirt sizing (discussed in detail later in this paper) is an example. This methodology has been successfully applied, but it is based on expert opinion.

vi. As you can see, the approaches are significantly based upon an estimate of software size.

3. TYPES OF SIZING

a. There are three broad categories of software sizing.

i. Physical Size – Source Lines of Code (SLOC). SLOC is an objective measure highly dependent on language and programmer skill. SLOC is generally rejected by Agile developers since it was developed and designed for the old data, languages, and development methods such as Waterfall. SLOC counts can be automated reliably for historical data collection.

ii. Relative Effort Size - Story Points, Tee Shirt Sizing, …. These relative measures are determined by Software Developers. These concepts/measures are generally familiar to Agile development teams.

iii. Functional Size/Function Points. This objective size measure is standardized and can be independently estimated. There are multiple Functional Sizing Metrics.

All size metrics are normally estimated as low, most likely, and high number. Therefore, a probability distribution can be developed to estimate at the desired confidence level, e.g., the 70% level. A wide range in the low, most likely, and high estimate would result in a significant variation in cost as the confidence level of the cost estimate is increased.

4. PHYSICAL SIZING

a. Source Lines of Code (SLOC) is the total number of lines of source code in a project. It is sometimes measured as KSLOC or ESLOC. The analyst must be sure the code measure is the same one used in the model. C code counters like USC's UCC or the Government (UCC-G) can be utilized to automatically count code lines. The DoD's SRDR is a good data source.

b. Physical Size Advantages

i. Accepted and is used in many automated models like COCOMO.
ii. SLOC is easily quantified.
iii. SLOC is being used today to successfully estimate and manage agile programs.

c. Physical size Disadvantages
i. Different programming languages, programmer experience, and automated tools effect the code count.
ii. When platforms and languages are different, LOC can be difficult to normalize.
iii. For new programs, SLOC must be estimated, usually by analogy to similar programs.

5. RELATIVE SIZING
a. Relative Effort Size is determined by the development team. Common relative measures are Story Points, Feature Points, Epics, Tee Shirt Size, Use Cases, User Stories, …The effort associated with each of these measures is based on expert opinion or analogy from previous work.
b. Relative Size Advantage
i. These are metrics that software developers are familiar and comfortable with.
ii. They are typically project or team specific.
iii. They have been successfully applied (examples in the following pages).
b. Relative Size Disadvantage
i. There is often no “formal”/consistent methodology.
ii. They often only reflect effort directly related to coding.
iii. It is difficult to extrapolate from project to project and especially from organization to organization due to process improvements and old data.

iv. Limited historical databases.

c. Relative sizing should not be discounted. In a recent large Federal agile development program, the Tee Shirt hour estimate was compared to a SEER-SEM hour estimate. When the estimates were reconciled there was a 5% difference in the total hours estimated.

c. The key here is a consistently applied methodology. The chart below in Figure 5, Successful Relative Effort Size Example, presents a successful and consistent application of Tee Shirt sizing.

Sample Tee Shirt Estimate								
Full Agile Development for a real-time radar ststem								
100 Identified Requirements								
Sprint Team Size	4		Assumptions:	a Small Tee Shirt requirement can be completed in one sprint				
Sprint length	2 weeks			An XS can be done in 1/2 Sprint				
Sprint hours	320			Use Fibonicci Series to distribute hours				
Tee Shirt Size	XS	S	M	L	XL	XXL		
	160.00	320.00	480.00	800.00	1,280.00	2,080.00		
Requirements	3	6	21	40	20	10	100	
Hours	480.00	960.00	1,440.00	2,400.00	3,840.00	6,240.00		
Total Hours	15,360.00							

6. FUNCTIONAL SIZING

a. Function Point Analysis (FPA) was introduced at IBM in the late 1970s to measure functional requirements of software. Over time the emergence of new languages, improved tools, and new development strategies (especially Agile) have caused developers to question the value of the SLOC

database and SLOC based models since they were based on the waterfall software development process.

b. Functional Size Measurement (FSM) provides a consistent technique for measuring software in terms of the functionality it delivers. Functional Size is especially valuable in the planning stage for input into project resource estimation calculations for cost, effort, and schedule. There are multiple Functional Sizing Metrics – COSMIC, IFPUG/SFP, and NESMA. There is a minimal difference in the actual count between the various methods.

c. Advantages
   i. independent of the technology,
   ii. estimated from statements of early requirements,
   iii. objective, repeatable and verifiable, and
   iv. enables benchmarking.

d. Disadvantages
   i. can require people with the expertise to carry out this activity.
   ii. can take some time and costs,

e. The International Function Point User Group (IFPUG) standard has been the most common functional size measure in the U.S. and is an ISO Standard (20296:2009). Recently, IFPUG adopted Simple Function Points (SFP) and this measure is quickly gaining traction in the estimating community.

f. The SFP method estimates a software's functional size based on quantifying its **transactions** and **logical data groups**. A transaction has a value of 4.6 Function Points, and a logical data group has a value of 7 Function Points. The SFP count can be performed quickly and early in a program's lifecycle

using existing documents. With minimal training, a cost analyst can complete a SFP count.

g. The Simple Function Point method estimates a software's functional size based on quantifying its business functions / transaction types, system interfaces, and other functional requirements from high-level acquisition documentation.

h. The SFP method developed by Italian researchers and acquired by IFPUG in 2019 (https://www.ifpug.org/ifpug-acquires-the-simple-function-points-method). The methodology was subsequently validated by a DHS study, Lets Go Agile: Data-Driven Agile Software Costs and Schedule Models for DHS Projects", ICEAA 2022, Wilson Rosa, Sara Jardine, Kimberly Roye, Kyle Eaton, and Chad Lucas.

i. The SFP count can be performed quickly and early in a program's lifecycle using existing documents. SFP 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). The SFP process was approved and evaluated by IFPUG. The figure below illustrates this mapping between the IFPUG components, and their Function Point counts and the SFP components and weightings. The chart below comes from the DHS Simple Functional Sizing document.

Field Code Changed

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

Mapping between IFPUG and SFPA Components and Weightings

j. 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 SFP components. Work done by Function Point counting experts produced a lexicon of over 140 action verbs and their associated components. The appropriate weighting factors are then applied to these components to produce a total SFP count. Recognizing these action verbs is how the manual or automated function point count is derived.

k. Once the Function Point Count (FPC) is established, the count must be converted in effort hours/person months, etc. International Software Benchmarking Standards Group. (ISBSG) offers a good commercial database (it may contain only successful projects). Currently no large Federal/National Security database is available. (The SRDR is beginning to collect this data.) An analogy to similar programs is often applied. Most Commercial models "backfire" the FPC into SLOC. Uison Global and others are developing unique parametric conversion methodologies.

l. Some requirements will have a zero SFP count since these action verbs may not be utilized. These non-functional requirements receive a zero SFP count. Items such as documentation or meeting a certain developmental standard do not require end user interaction and, as such, are not functional. There is, however, effort associated with these

requirements as they add complexity to the overall work effort. Most software cost estimating models account for these hours from the parametric estimating equations derived from their historical data base(s). Hours for these elements are included in the total. Non-model users might utilize tools like SNAP to account for these non-functional hours.

m. There are four useful methods to address these zero functional count items.
    i. Complete a manual review and apply human logic.
    ii. Develop an average functional count for the program and apply that to these requirements.
    iii. Utilize an estimating tool that internally accounts for these requirements.
    iv. Apply the SNAP process for these requirements.

n. Automated Function point counters use AI (Natural Language Processing (NLP)) and a robust set of rules to reduce the time it takes to inspect and estimate the functional size of each requirement. Two examples are ScopeMaster and Cadence. The actual lists are product sensitive and are revised annually. These models do the initial heavy lifting, but some manual review is required. The DHS CADE Simple FP model, SiSE, has a list of 143 keywords.

o. Often many military requirements show a “zero” function count when there is obviously work required. This “zero” functional count can be due to there being no actual functional requirement, a poorly written requirements statement, or an unrecognized “verb” due to the unique nature of the subject matter. In any case a manual review of the “zero” functional requirement is advised.

p. A recent large Federal program had a manual function count of over 25,000. and the initial automated count was about

18,000. When the "zero" elements were evaluated the two counts came within 8%.

7. CONCLUSIONS

a. All size metrics can provide meaningful estimates when used appropriately.

b. The team documented the successful utilization of SLOC, FP, and Tee Shirt sizing in recent Agile Development Programs.

c. Automated function point counters can do much of the initial heavy lifting.

d. It is important to ensure Apples-to-Apples comparisons are made to analyze results.

e. No generally accepted "direct" FP to hours model exists today, although Boehm Center for Systems and Software Engineering (BCSSE) at USC is working on A COCOMO III release to address the conversion from Function Points to hours. And Unison has developed a unique conversion process and will implement the COCOMO III as an add on to the TruePlanning Software Model.

f. The development of a model with a formulation something like; Effort = A(FP)^b*C would greatly benefit the software cost estimating community/

Where: A = a "complexity" modifier

FP = number of function points

b = a derived exponent

C = a "productivity" modifier

The overall goal is to develop/validate a National Security FP database.

Appendix

SEER by Galorath

Model is based on the Jensen SEER/Sage equation

$$S_E = C_{TE} * (K)^{1/2} * t_d$$

Where: S_E **= Product Size in ESLOC**
C_{TE} **= Effective Technology Constant (based on cost driver inputs)**
K = total software life-cycle effort in person-years
t_d**= development time in years**

Software development effort = 0.3945 * K

SEER Effort Formula: $K = (S_E / (C_{TE})(t_d))^2$
SEER Schedule Formula: $t_d = D^{-0.2} (S_E / C_{TE})^{0.4}$

Where:
D = Staffing Complexity Constant (i.e., How hard is it to get the staff required to code this type of software?). Note that the default value for D = 15

Unison Cost Engineering TruePlanning for Software

TruePlanning® CER:

$$Effort\ (Hours) = A * Size\ ^B$$

Where:
A = influence of the drivers in the model (Functional Complexity, Technology, People, Reuse, Organizational Productivity, etc. - 33 numerical cost drivers + 10 nominal cost drivers)
Size = software size in specified Size Units (SLOC, IFPUG FP, COSMIC FP, etc.)
B = economy or diseconomy of scale as a function of Organizational Productivity (B ranges between 1.077 and 1.117)
Example: Military Avionics in SLOC with default values: $Effort$ = 0.252 * $Size$ 1.107

$$Schedule\ (Months) = C * Effort\ ^0.33$$

Where:
C = efficiency/inefficiencies in developing the software based on model inputs (Functional Complexity, Technology, People, Organizational Productivity, etc., 33 numerical drivers and 19 nominal drivers – C ranges from 0.41 to 0.9)

Author Biographies

Bob Hunt

Bob Hunt has over 40 years of cost estimating and analysis experience. He received his Society for Cost Estimating and Analysis (SCEA) Certification in 1991. He is a Subject Matter Expert supporting NSI. He has served in senior technical and management positions at Dulos Incorporated (President), Galorath Federal Incorporated (President), Galorath Incorporated (Vice President for Services), CALIBRE Systems (Vice President), CALIBRE Services (President), SAIC (Vice President), the U.S. Army Cost and Economic Analysis Center (Chief of the Vehicles, Ammunition, and Electronics ICE Division), U.S. Army

Directorate of Cost Analysis (Deputy Director for Automation and Modeling), and other Army analysis positions. He is the author of multiple technical papers published for ICEAA, IEEE, DCAS, and other professional journals. His published works include Price To Win, Should Cost, Earned Value, and Agile Estimating. He has served as a track chair and technical presenter for multiple SCEA/ISPA/ICEAA Conferences.

Rainey Southworth

Rainey Southworth (NSI) has been part of the PMA-281 cost team for the past 4 years as a contractor through Naval Systems, Inc (NSI). In this multifaceted role, Ms. Southworth has provided support across the IPT, primarily navigating the effective use of Agile, software sizing, and associated metrics in a DoD setting.

Chad Lucas

Chad Lucas has 21 years of experience in Cost Analysis working CEI Management Consulting, The MITRE Corporation, MTSI, and currently work for Naval Systems Incorporated (NSI). He holds an MBA from George Mason University and has been a CCE/A since 2005. Over the course of his career Mr. Lucas has supported multiple agencies with the DoD, HHS, and DHS. He has authored or co-authored multiple white papers on a wide range of cost related topics, completed estimates for various technology platforms, and has an extensive range of knowledge spanning multiple Federal Agencies.

He is a co-author of two previous papers on the topic of Agile Software Estimation:

"Let's Go Agile: Data-Driven Agile Software Cost and Schedule Models for DHS projects" Wilson Rosa, Sara Jardine, Kimberly Roye, Chad Lucas, and Kyle Eaton, ICEAA Conference 2022

"Applying Simple Function Point Analysis to an 804 Rapid Acquisition Program Cost" Robert Hunt, Rainey Southworth, Chad Lucas, and Esteban Sanchez, ICEAA Conference 2023

Heather Meylemans

Heather is a research chemist turned operations research analyst for NAVAIR in Pt Mugu, CA. I hold a PhD in chemistry from the University of Colorado at Boulder and a BS in Biochemistry and Math from Regis College in Denver, CO. I have authored more than 25 papers in peer-reviewed journals, and I am an inventor on more than 20 US patents. I currently serve as the Cost Department division head for acquisition and O&S cost estimating at NAWCWD. My main interest is striving to live as sustainably as possible. In my spare time I enjoy doing puzzles and crafts. I am a mom of two teens and spend time as a baseball mom.

Denton Tarbet

Mr. Tarbet has an advanced degree in Systems Engineering and Optimal Control has over 40 years of Project Management and cost estimating and analysis experience. He is a Senior Systems/Software SME at NSI. He served as SW director at major aerospace firms, VP for development of FAA certified commercial aircraft systems, and CTO for a major technology company. He has consulted internationally in software

estimation and project cost and planning for both Engineering and IT applications.