



Sizing Software Modernization; A Real-World Example

Authors

**Chad Lucas, Naval Systems Inc.
Nicholas Taw, PMA 281**

Abstract

The ability to estimate the effort and schedule required for software development programs is challenging. This is even more so for programs in the early phases of development where requirements are not always well defined, and design is in a constant state of flux. Also challenging are program modernization efforts where the design of the program is changing but its operations and purpose are not. In this paper we present a real-life example of how our cost team took the initiative to come up with an estimation plan for mission critical program beginning the process of modernization, and executed it, for a customer that was unused to applying some of the accepted techniques of Agile software estimation.

Contents

1. Introduction.....	3
2. The Case for Modernization	3
3. What We Knew	3
4. The Estimation Plan	4
5. Executing the Estimation Plan.....	5
6. Estimate Results	7
7. Program Benefits and the Path Forward.....	8
8. Conclusions and Lessons Learned.....	9
9. Acronyms.....	10
Author Biographies.....	11
Chad Lucas.....	11
Nicholas Taw	11

1. Introduction

Theatre Mission Planning Center (TMPC) is a software-intensive system of systems, providing precision targeting, mission planning, strike planning, strike execution and coordination, post-launch execution, missile control and mission reporting of the Tomahawk cruise missile and the Conventional Prompt Strike (SCPS) weapon. TMPC has been installed at over 180 sites globally.

TMPC has three developers: one for, Weaponing (designing and attack with weapons) and Navigation tools and services; another for the Tomahawk Mission Planning tools and services; and a third for the Task Management, Strike Planning & Strike Execution Software and other related tools and services.

TMPC is undergoing an effort to modernize the system and requires a credible estimate of cost and schedule for submission of the funding request. Previous attempts to estimate the modernization effort centered around Software Lines of Code (SLOC). These estimates lacked credibility due to the massive growth in effort year over year and produced unreasonable costs relative to what is known about the programs needs. In the past TMPC estimates relied heavily on input from the system developers and lacked independence and were not useful as program management tools.

2. The Case for Modernization

The current TMPC software architecture was designed in line with industry best practices of 25 to 40 years ago, before the existence of modern cybersecurity, software engineering techniques, and human systems integration (HSI) practices/TMPC regularly deploys cybersecurity patches, but the process is cumbersome and inefficient due to the medley of architectures implemented as Tomahawk capability has expanded. Redesigning to modernize the architecture of TMPC implements DoD-mandated HSI and NSA-required cyber standards, streamlines “zero-trust” operations, and enables continuous development/delivery system upgrades.

The TMPC program needed to create an estimate of the cost of modernization to support a funding request to accomplish the work. Design is not completely finalized and there is uncertainty about the effort required.

Previously the program had completed cost estimates of the software development using estimated Software Lines of Code (SLOC) provided by the engineering team. NAVAIR NEMO and SEER-SEM estimating tools were used to analyze SLOC. These SLOC counts included estimates of SLOC that could be reused. Over time the cost of software development grew exponentially, leading the cost team to think that a new approach was required.

3. What We Knew

This redesign will result in coding new and existing capabilities in micro-services versus the current large component structure. Micro-services is composed of small independent services that communicate across the Application Programming Interface (API), reducing code interdependencies. These micro-services would be “containerized” to allow for easier and faster updates in the future. To minimize risk to existing operations and reduce the risk of failure the process of software strangler migration will be used.

Software Strangler Migration “is a software design and architectural pattern used in the context of legacy system modernization or application migration. The pattern’s name comes from the way certain plants, like strangler figs, grow around host trees, eventually replacing or “strangling” them entirely. Similarly, the Strangler Pattern aims to replace or evolve an existing system gradually, without causing any significant disruptions to the overall system functionality.” For more information on the Strangler Pattern, you can visit <https://www.techtarget.com/searchapparchitecture/tip/A-detailed-intro-to-the-strangler-pattern>

Sizing the full modernization effort was problematic. Many design decisions had not yet been made and full scope of what a containerized solution would look like had not been flushed out. The design of DevSecOps is an area of uncertainty and the costs were incorporated by adding cloud services, annual license costs for current baseline ATO approved software, and adding six more cybersecurity personnel. The DevSecOps implementation plan is to have an iterative rollout, which the DevSecOps environment could change once the Continuous Authority to Operation (CATO) is developed, cloud infrastructure is developed, full Agile is embraced, and DoD Software Factories are established. All that was known was the all the existing functionality would have to be reallocated into a new design and that changes in that functionality would be required to accommodate the new design.

The program itself is in the process of transitioning to full Agile development from a more Hybrid-Agile/Waterfall approach. Programming languages to be used had been previously identified. Most importantly, this is a fully functional system as is and the developers under contract have expert level knowledge of all facets of the program.

4. The Estimation Plan

It was decided that, since the modernization effort would require changes to the entire TMPC system, at a minimum, the requirements for the system “as is” could be used as a baseline for sizing. Some level of reuse would have to be accounted for, but all requirements will have some effort required to complete the redesign. Software Lines of Code (SLOC) counts to size the effort have resulted in unreasonable results in the past and were not available for the entire system. It was decided the Simple Function Points (SFP) would provide the best path to an objective sizing measure and in a timely fashion. Assumptions for some aspects of the effort would be informed by John’s Hopkins Applied Physics Lab (JHU APL), who had been tasked with completing a comprehensive study of the modernization effort by PMA leadership.

Simple Function points were selected as a sizing measure for several reasons. First, they are an object measure of size easily applied to programs in their early phases of development. Second, the counting conventions around SFP make it easy for the counts to be validated. Finally, because they have been proven one of the most accurate predictors of development effort as illustrated in figure 1 below.

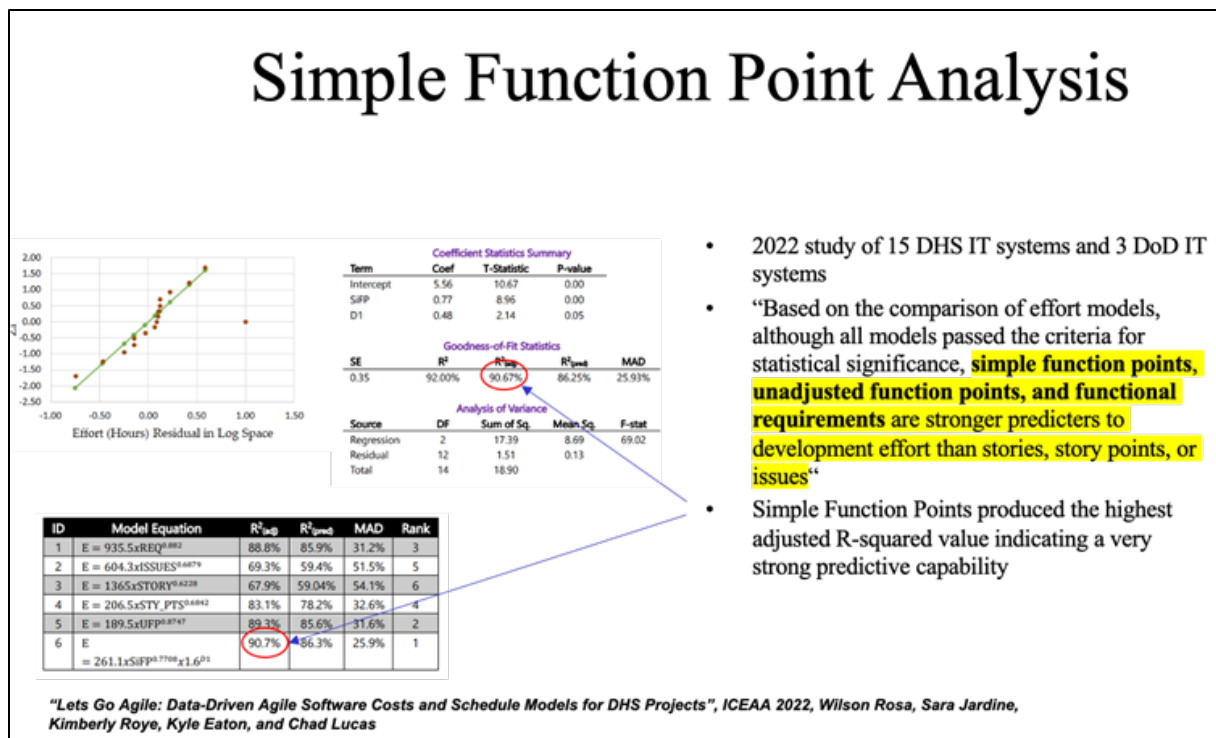


Figure 1. Why Simple Function Points

The topic of SFP is a paper onto itself. For more information on how SFP visit International Function Point Users Group (IFPUG) website on Simple Function Points at [Introducing Simple Function Points \(SFP\) - IFPUG - International Function Points Users Group](#)

5. Executing the Estimation Plan

A full listing of the System/Subsystem Specifications (SSS) by component was obtained from the prime integrator. These requirements were detailed enough to conduct an initial SFP count by component. To enhance the accuracy of the count, the cost team engaged the systems engineering team for help in validating the assumptions used. TMPC Engineers assisted the cost team in validating requirements initially labeled as non-functional and verifying whether duplicate requirements were in fact duplicates. The engineering team also spot-checked the actual counts to see if there was disagreement with the initial results and to ensure the cost team had full understanding of the requirements as written. This resulted in a valid SFP count that could be used as basis for estimating the development effort. The results of the SFP count can be seen in Figures 2 and 3 below.

	Component 1	Component 2	Component 3	Component 4	Component 5	Component 6	Component 7	Component 8	Component 9	Total
Total Functional Requirements	1,637	197	154	244	289	56	340	262	93	3,272
Non-functional listed as functional	185	6	14	44	17	1	8	0	0	275
Duplicate Requirements	123	4	17	10	20	0	9	3	3	189
Actual Functional Requirements	1,329	187	123	190	252	55	323	259	90	2,808
Total EPs	1,455	336	179	230	326	60	339	294	118	3,337
Total LFs	236	57	39	31	59	7	67	46	26	568
Total SFP	8,345.0	1,944.6	1,096.4	1,275.0	1,912.6	325.0	2,028.4	1,674.4	724.8	19,326
SFP per requirement	6.28	10.40	8.91	6.71	7.59	5.91	6.28	6.46	8.05	6.88
LF/EP Ratio	16.22%	16.96%	21.79%	13.48%	18.10%	11.67%	19.76%	15.65%	22.03%	17.02%

Figure 2. Results of the Simple Function Point Count

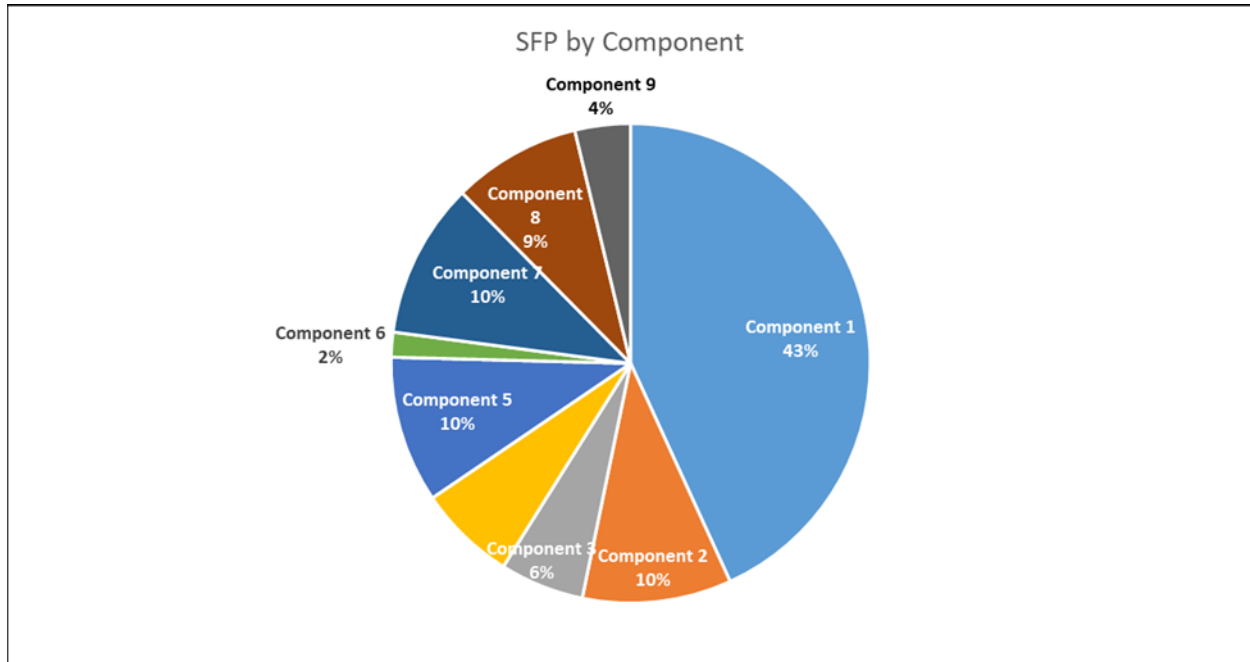


Figure 3. Simple Function Points by Component

The final SFP count was input into SEER-SEM as “existing functions, not designed for reuse”. The model had preprogrammed factors, based on the knowledge bases selected, that assigned a level of reuse appropriate for each component.

Ordinarily, the cost team would use a triangular distribution using Joint Cost and Risk Handbook guidance around the SFP count to assign an appropriate level of uncertainty. However, with the assumption that software strangler migration was being used and the fact that the requirements are stable and complete, a limited distribution of +/- 10% was applied.

JHU APL assisted the cost team in establishing appropriate assumptions for the SEER-SEM model. First was determining the appropriate acquisition method knowledge base for the effort. After some debate, Concept Reuse was selected as the most appropriate knowledge base to leverage. Concept reuse is defined in the figure below as:

<p>Concept Reuse</p>	<p>This knowledge base should be used for situations where the software approach is being appropriated from a well-defined basic concept, including architectural definitions. This work may have been done previously and shelved or possibly under a separate contract. This knowledge base assumes that full coding and testing will be required but also that some savings in basic design, likely from 10 to 20 percent, are likely. The programming languages need not be the same.</p>
-----------------------------	---

Figure 4. Concept Reuse

Other assumptions and information required to run the parametric model are included in the table below. Some, such as programming language and application type, were carried over from previous estimates done in SLOC. Others were validated with the JHU APL team to ensure the best possible representation of the situation as it stood at that time. Each major component of the

system, while somewhat of a system itself, was treated as a component of overarching TMPC in order to capture correlation in the estimate calculations.

SEER Settings/Inputs by Component					
Component	Language(2)	Application(2)	Acquisition Method (5)	Development Method(5)	Software Phase at Estimate
Component 1	C#	Command/Control	Concept Reuse	Full Agile	Design
Component 2	SQL	Database App	Modification, Average	Full Agile	Code
Component 3	Javascript	Mission Planning & Analysis	Concept Reuse	Full Agile	Design
Component 4	C++	Communications	Concept Reuse	Full Agile	Design
Component 5	3rd Gen	Data Mining	Concept Reuse	Full Agile	Design
Component 6	3rd Gen	Database App	Concept Reuse	Full Agile	Design
Component 7	3rd Gen	Simulation	Concept Reuse	Full Agile	Design
Component 8	Visual Basic	Mission Planning & Analysis	Concept Reuse	Full Agile	Design
Component 9	3rd Gen	Mission Planning & Analysis	Modification, Average	Full Agile	Code

Assumptions:
1) All function points are pre-existing functions, not designed for reuse. Assumes some reduction in effort vs "green space" coding.
2) Assumed the same as previous estimate based on SLOC, verified with IT leads on 7/20/23.
3) Development Standard is IEEE full.
4) Due to move to microservices, may result in some functionality and design changes. Requirements are assumed stable.
5) APL provided assumptions as a result of their modernization study

Figure 5. Other Assumptions

These assumptions, combined with the validated SFP count were input into the SEER-SEM model to produce and estimate of hours and schedule for software modernization effort.

6. Estimate Results

The model produced an estimate of effort of 1,211,686 hours in total and schedule of approximately 9 years to complete. Results of the estimate were validated with JHU APL as part of completing the study and briefed to PMA leadership. The estimate of software effort was accepted with no change and used to formulate the request for budget to fund the modernization initiative. Figures 6 shows the results of the SEER-SEM “quick estimate”, which displays results at the 50% confidence interval of the distribution. Figures 7 and 8 show the breakdown of estimated hours by system component and by anticipated activity.

Quick Estimate		Person Hours by Labor Category
Item	Estimate	
PROJECT: TMPC Modernization		
Development Schedule Months	107.37	
Development Effort Months	7,971.61	
Development Effort Hours	1,211,685	
Development Labor Cost	0	
Maintenance Schedule Months	0.00	
Maintenance Effort Months	0.00	
Maintenance Effort Hours	0	
Maintenance Labor Cost	0	
Delivered Defects	143	

Figure 6. SEER-SEM “Quick Estimate” results

Component	Hours
Component 1	605,951
Component 2	4,130
Component 3	117,309
Component 4	72,715
Component 5	19,750
Component 6	119,967
Component 7	139,176
Component 8	127,258
Component 9	5,430
	1,211,686

Figure 7. Hours by Component

Activity	Hrs
System Requirements design	17,048
Software Requirements analysis	50,032
Preliminary design	95,079
Detailed design	190,798
Coding and unit test	358,371
Component integration and test	178,858
Program Testing	24,226
System Integration through OT&E	297,272
	1,211,686

Figure 8. Hours by Activity

7. Program Benefits and the Path Forward

The estimate resulted in an independent assessment of the effort required to modernize TMPC software rather than leaning on vendor proposals. It is defensible, rational, and repeatable given a solid underlying data set and methodology.

The program can also now add and remove requirements from the estimate at the SSS level, if desired, to assess impacts of requirements changes to the effort. As the move to micro-services takes shape, requirements can be grouped by those services, providing insight in the costs of those functions. The ability to do these things gives program leadership maximum flexibility and insight for charting alternative courses of action and future planning.

As of the writing of this paper, the program has already been able to leverage the estimate for POM submissions and what-if drills. This has made meeting short turn reporting requirements easier and helping to ensure consistency between all cost products related to the modernization effort.

TMPC is now also positioned to build a data set based on actual results. The program should now be able to track delivered requirements and associate them to their SFP count and actual hours. The long-range goal is to build a data set that can be used going forward to either

calibrate parametric models or develop their own data based Cost Estimating Relationships (CERs)

8. Conclusions and Lessons Learned

Software Requirements Specifications are a better source of data for SFP counts than SSS. PMA engineering staff indicated that, had the initial count been conducted with the SRS vs SSS, there would have been far less need to reconcile with them.

In an ideal world, it would be optimal to have engineering staff complete the SFP count independent of the cost team. This would require training the engineering team in the conventions for counts SFP and require the engineering team to have the bandwidth to conduct the counts on their own. The latter is often challenging, given the short staffing of government engineering teams on most IT programs.

Lastly, like all cost estimates, these types of efforts cannot be done in a vacuum. Estimators should admit what they do not know, seek expert inputs, and involve the entire team in the process as much as feasible. This will also help with buy-in of the estimate results once completed and makes acceptance easier as well as reducing rework. Time was lost due to rework and being connected late with a government Subject Matter Expert engineer, to help us with verification and validation, that there was not enough time to gather inputs from other functional teams. It would have been optimal to have the DevSecOps team provide input on the future DevSecOps pipeline and process to include costs that may have not been considered. The SFP count took about four months to complete. If resources allowed, purchasing a Commercial Off-The-Shelf (COTs) SFP automation counting software license could have helped decrease the time in half and the shifting of personnel for support.

Utilization of SFP provides a reasonable method for estimating the costs of programs in their early phases or for programs where there is some design uncertainty. Use of parametric estimation models is appropriate, in the absence of actual historical data.

9. Acronyms

CERs – Cost-Estimating Relationships

DoD – Department of Defense

HIS – Human Systems Integration

IFPUG – International Function Point Users Group

JHU APL – Johns Hopkins University Applied Physics Lab

NSA – National Security Agency

PMA – Program Management Activity

SLOC – Software Lines of Code

SSS - System/Subsystem Specifications

SRS – Software Requirements Specifications

SFP – Simple Function Points

TMPC – Theatre Mission Planning Center

Author Biographies

Chad Lucas

Chad Lucas has 25 years of experience in Cost Analysis. 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 within the DoD, HHS, and DHS working for companies such as CEI Management Consulting, the MITRE Corporation, MTSI, Galorath Federal, and now Naval Systems Incorporated. He has authored or co-authored multiple white papers on a wide range of cost related topics and completed estimates for a wide range of technology platforms.

Nicholas Taw

Nicholas Taw is a Test & Evaluation Engineer turned Operations Research Analyst at Naval Air Systems Command (NAVAIR) at Naval Air Warfare Center Weapons Division (NAWCWD) in Point Mugu, CA. He holds a Bachelor of Science in Electrical Engineering from California State Polytechnic University, Pomona. He served 5 years with the Test & Evaluation Department supporting the F/A-18 Electronic Warfare program and has been with the Cost Acquisition and Analysis Department for 3 years. He currently serves as the Cost Team Lead for PMA-281.