

From ConOps to ROM Estimate:

Six Easy Steps for Custom Software Development

Carol Dekkers, Quality Plus Technologies

Daniel B. French, Cobec consulting

International Cost Estimating Analyst Association (ICEAA) 2023 Workshop

Author Note

1. *This case study was originally published in *The IT Measurement Compendium: Benchmarking and Estimating Success with Functional Size Measurement* by Manfred Bundschuh and Carol Dekkers, 2008, Springer Publications, Germany, chapter 18 as a comparison of FP counts.*
2. *For the CEBoK-S, this case study was corrected updated, abridged, and adapted to include a demonstration of how to ESTIMATE the functional size, using high-level early-estimating methods, rather than using the original, detailed full methods including International Function Point User Group (IFPUG) Simple Function Points (SFP)*

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

2

Table of Contents

Abstract 3

From ConOps to ROM Estimate: Six Easy Steps for Custom Software Development..... 4

Background 4


 History of Function Points..... 4

Case study: Course Registration ConOps/High-level requirements 10

Conclusion 13

References.....**Error! Bookmark not defined.**

Appendix A-Original Case Study**Error! Bookmark not defined.**

 CEBoK-S abridged Case Study first start f 14

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

3

Abstract

Estimating the cost and schedule of custom software development from preliminary requirements or a Concept of Operations (ConOps) document can be challenging to even the most seasoned cost estimator. It is well-established that software size and productivity are major drivers of software development costs, however, both can be elusive when creating early estimates. When considering software size, both Source Lines of Code (SLOC) and Functional Size (Function Points) are not easy to estimate given that solid details for each are unavailable until after software delivery. Additionally, barriers to entry exist including mastering the IFPUG Function Point Analysis rules and then adapting them to apply to vague high-level requirements. This presentation overcomes these barriers to entry and outlines a streamlined six-step process to develop a data-founded and defensible cost and schedule ROM estimate from high-level requirements/ConOps documents. A sample ConOps case study will be used to demonstrate the six-step process.

This paper covers:

- :• The newest IFPUG function point standard, Simple Function Points, for sizing software from preliminary or high-level project requirements.
- A six-step process for a cost analyst to develop a defensible ROM cost and schedule estimate from a ConOps document (or high-level agile user stories/requirements)
- Publicly available (historical) productivity and size data sources in lieu of your own historical data
- Recommendations for cross-checking and strengthening the ROM estimate

Keywords: Software Measurement, Simple Function Points (SPF), Function Points, Software Estimating

From ConOps to ROM Estimate:

Six Easy Steps for Custom Software Development

Background

Historically, it's been a struggle for software development organizations to create accurate and defensible software estimates from early requirements documents. The challenge is independent of the software development methodology and arises because of the lack of specificity of requirements details that are needed to formulate a realistic software size estimate. For example, Epics, Use Cases, Concept of Operations (ConOps) documents, or other documents that describe software requirements are often the only information on which an analyst has available to develop a software size estimate, on which a rough order of magnitude (ROM) cost and schedule estimate can be created. This paper demonstrates how to create a reliable software size estimate using the <new> IFPUG Simple Function Point (SFP) method given a sample set of ConOps high-level requirements. From this size estimate, a ROM cost and schedule estimate can be created.

History of Function Points.

Function points date back to the mid-1970's when they were developed at International Business Machines (IBM) by a team researching alternatives to the only available software size measure at that time, Source Lines of Code (SLOC). Problems arose when new programming languages complicated homogenizing SLOC and created a productivity paradox whereby higher-level languages (and less SLOC) appeared to be less productive. Software size based on function points eliminated both the complexities of multi-language development and the productivity paradox, and a formal function point sizing methodology became public in 1979.

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

5

The International Function Point User Group (IFPUG), established in 1984, codified and maintains the official counting rules in its Counting Practices Manual (CPM). In 1998, IFPUG Function Points (FP) were formally approved as an international standard by a joint technical committee of the International Organization for Standardization (ISO) and the International Electrotechnical Committee (IEC).

A decade later in 2009, to address the needs of an emerging estimating community, Dr. Roberto Meli of Italy introduced the Early and Quick function point method based on IFPUG FP. While this helped to enable earlier size estimates without the need for detailed requirements, there were still challenges to applying this method. In 2016, a newer version became the Simple Function Point (SiFP) method, and the basis of the IFPUG branded version: Simple Function Points Counting Practices Manual v.2.1. IFPUG SFP was released in 2021 following acquisition of the method by IFPUG.

Challenges with estimating software projects early in the software development lifecycle (SDLC).

The greatest challenges to estimating software size on development projects, whether new development or enhancement, are a poorly defined scope and the lack of requisite requirements details (report layouts, data schema, interface definition documents, et al) to apply the traditional IFPUG function point counting method. Traditionally, to overcome this lack of specificity, analysts would either assume an average complexity for all identified functions or infer the IFPUG functional complexity (low, average, or high) based on the professional judgment of subject matter experts (SMEs).

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

6

The Early and Quick (E&Q) method, based on the IFPUG counting rules and developed in Italy, was one of the first extensions of the IFPUG method that allowed for identification of IFPUG standard functions (when the details were available), and introduced simplified concepts of “unclassified” functions and data groups that could be estimated from high-level requirement documents. E&Q outlined four levels of software requirements and set out guidelines to determine small, medium, and large typical processes based on generic functions such as Create, Read, Update, Delete (CRUD.) At the end of the analysis, requirements were categorized by level, and function point values assigned based on a derivation of values from the traditional, detailed IFPUG FP method. E&Q introduced Typical Processes, Macro Processes, and the four levels of requirements, which helped analysts to overcome the lack of requirements detail and scope issues, but it also introduced ambiguity and challenges to analysts trying to identify and determine the relative size of functions they were estimating. Subsequently, the next version: the Simple Function Point (SiFP) method, streamlined the levels to one, and reduced the functional components further into two categories: Elementary Processes (analogous to IFPUG transactional functions EI, EO, & EQ) and Data Groups (analogous to IFPUG data functions ILF and EIF). Simple Function Points was acquired by the IFPUG from its creator Dr. Roberto Meli, by the IFPUG in 2019. Subsequently, the method was standardized, rebranded, and published as IFPUG Simple Function Points (SFP) v.2.1 in October, 2021.

The Six Step Process to Creating a ROM estimate from ConOps / high-level software requirements

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

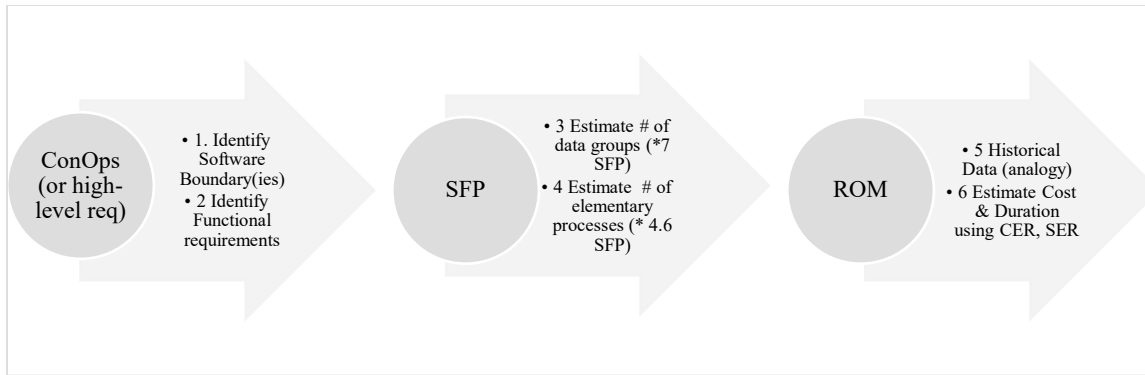


Figure 1. The Six Step Process to Creating a ROM estimate from ConOps / Early Requirements

As depicted in Figure 1, there are six steps, starting with a CONOPS document on the left, to creating a ROM cost and schedule estimate. Each step is described below.

Step 1: Identify Software Boundary(ies)

The initial, and most crucial step in the process is the identification of the software application boundary, or if there are more than one application included in the CONOPS, boundaries. In functional sizing terminology, the “boundary” is the conceptual line between the software under analysis and its users, through which data passes into and out of the application(s).

Misplacement of the boundary can result in significant overcounting or undercounting of the size of the software. To enable the correct identification and placement of the application boundary, an analyst can refer to system documentation, defined areas of user data responsibility,

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

8

application tier(s) and other factors. This is easy to do when there is already software in place (as in software enhancement projects), but can be more difficult for new software development.

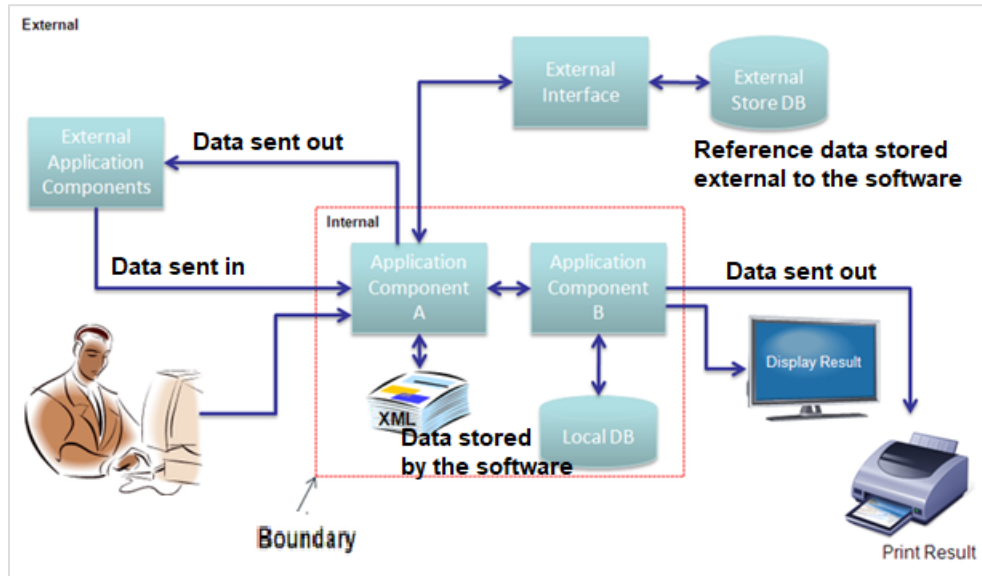


Figure 2: Illustration of the Software Application Boundary in the context of functional sizing. (Adapted from 1

<https://sourceforge.net/p/functionpoints/wiki/Function%20Point%20Analysis/>)

Step 2: Identify functional requirements

Once the boundary is established, the next step is to review the ConOps and/or other documentation to identify the functional requirements to be estimated. Functional requirements are those software requirements that are user-identifiable and specify what the system shall do (input, output, reports, interfaces, screens, et al) to meet the users' needs. Functional

requirements do not include how the software will perform the functions and therefore do not include specifics of business rules, or technical, quality or performance requirements.

Steps 3 and 4: Using the Simple Function Point method, estimate the number of Data Groups and Elementary Processes to determine the Functional Size

After identification of the functional requirements, each is analyzed to determine its functional size. In traditional IFPUG FP methodology, this step included having to determine the type of function (transactional type: input or output or query; or data type: internal or external) and the relative functional complexity (based on matrices for Low, Average or High complexity), and then assigning a preset number of function points to the function. As stated previously, simple function points streamline this step by having only two functional components (transaction or data) with a singular Simple Function Point value assigned to each type. Transactional functions are called elementary processes (worth 4.6 SFP each), while data functions are called Data Groups (worth 7 SFP each). Once the number of data groups and elementary processes are estimated, the functional size is calculated.

Step 5: Gather historical data

Given the functional size from the prior steps, the estimator should gather any available historical project data, preferably organizational, for use in analogous estimating if that is the method being used. If historical data are not available, the analyst can resort to publicly available project data such as the Application Development and Enhancement repository owned

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

10

by the not-for-profit International Software Benchmarking Standards Group (ISBSG) or other available data sources.

Step 6: Identify appropriate Cost Estimating Relationships (CERs) and Schedule Estimating Relationships (SERs).

Using the historical data from step 5, the Functional Size from step 4, and the CER and SER, generate Cost and Schedule (Duration) estimates for the software development. It is an estimating best practice to develop software estimates using more than one estimating technique to verify that the estimates are within an acceptable range, 10-20% of each other, to increase the confidence and defensibility of the estimate.

Case study: Course Registration ConOps/High-level requirements

The functional requirements below describe a project to develop software to replace the front-end of the existing course registration system with a new system. The new course registration system will allow students and professors to access the system through personal computers (PC's). The current registration system is a legacy system used for the past 20 years, but it lacks the capacity to handle the current and future student and course load projections. In addition, the current system is outdated mainframe technology, and only supports access through Registration Office clerks. The new system will enable all professors and students to access the system through portals connected to the college computer network, and through any personal computer connected to the Internet. The new system will bring the college to the leading edge in course registration systems and improve the image of the college, attract more students, and streamline administrative functions.

Step 1: Identify Software Boundary(ies) – Single Course Registration Software

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

Step 2: Identify Functional requirements – High Level Use Cases

Use case #	Use case for Course Registration System
1.	Logon (by all users)
2.	Maintain professor information (by the registrar)
3.	Select courses to teach (by professors)
4.	Maintain student information (by the registrar)
5.	Register for course(s) (by students)
6.	Close registration (by the registrar)
7.	Submit grades (by professors)
8.	View report card (by students)

Table 1: Use case requirements for the new Course Registration System

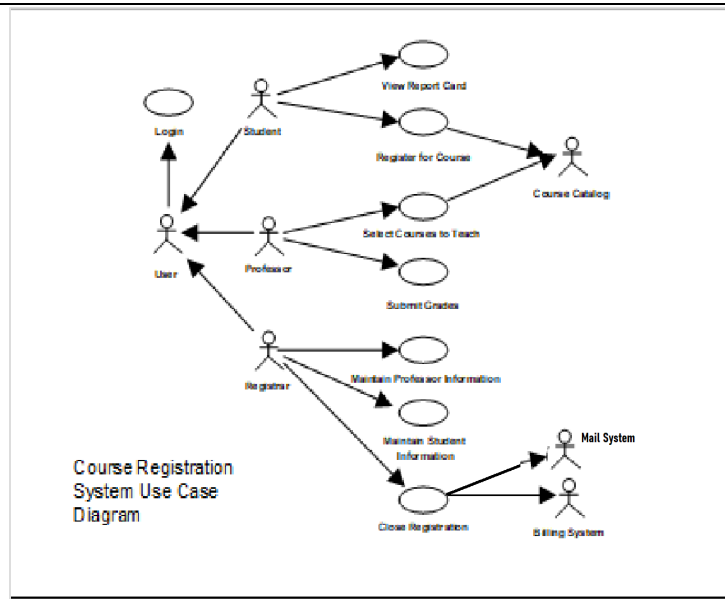


Figure 3: Object model of the Use Case Requirements

Step 3: Identify and estimate the size of data groups:

- 5 distinct data groups

	Data Group	Description	# Of SFP
1	Course	A standard series of lectures, etc. on a specific subject from the College Course Catalog	7
2	Course-offering	A Course that is available for students to enroll during a particular Semester.	7
3	Professor	A person who may register to deliver a Course-offering in the current Semester, for a Course which he is eligible to teach.	7
4	Student + schedule item(s)+ schedule item history (see note 2 in section 3)	A person who can register to attend a Course-offering	7
5	User	Any person (Registrar, Professor or Student) that is authorized to use the Course Registration system	7

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE DEVELOPMENT PROJECTS

12

		<i>Total data group SFP</i>	35
--	--	-----------------------------	-----------

Step 4: Identify and estimate the size of Elementary Processes

- 21 Elementary processes are identified:

	<i>Use Case 1: Logon</i>	
1	Logon (validate userid and password)	4.6
	<i>Use Case 2: Maintain professor</i>	
2	Registrar: add professor	4.6
3	Registrar: retrieve and display professor information	4.6
4	Registrar: modify professor	4.6
5	Registrar: delete professor	4.6
	<i>Use Case 3: Select courses to teach</i>	
6	Professor: Display course offerings available for this professor	4.6
7	Professor: Select /de-select courses and save (update) course offerings.	4.6
	<i>Use Case 4: Maintain student information</i>	
8	Registrar: add student	4.6
9	Registrar: display student information	4.6
10	Registrar: update student	4.6
11	Registrar: delete student	4.6
	<i>Use Case 5: Register for courses</i>	
12	Student: Display available course offerings	4.6
13	Student: Maintain schedule	4.6
14	Student: Display schedule	4.6
	<i>Use Case 6: Close registration</i>	
15	Registrar: Close Registration	4.6
16	System: Send billing system notice of courses closed	4.6
17	System: Notify all students by mail of any changes to their schedule	4.6
	<i>Use Case 7: Submit grades</i>	
18	Professor: List of course offerings taught in previous semester	4.6
19	Professor: List of all students registered for course offering and grades	4.6
20	Professor: Enter student grades	4.6
	<i>Use Case 8: View report card</i>	
21	Student: View Report Card	4.6
	<i>Total elementary process SFP</i>	96.6

$$\begin{aligned} \text{Total Functional Size} &= \text{Sum of (total Data SFP + total Elementary process SFP)} \\ &= 35 + 96.6 = 132 \text{ SFP} \end{aligned}$$

Step 5: Gather Historical Data --- ISBSG in lieu of own actual data

If there is comparable project actual data available internally to use as an analogous estimate,

then identify the project(s) that will be used and conduct the requisite analysis of the

comparisons between the project being estimated and the analogous projects to determine what

adjustments need to be made to the actual data being used to estimate the project more accurately. Ideally, the projects selected are similar in platform, language, business functionality and size. If no suitable in house candidate projects exist with the requisite actual data, then external industry data sources such as the International Software Benchmarking Standards Group (ISBGS) (<https://www.isbsg.org/>) can be used. Under no circumstances, however, should the data be used without proper analysis and adjustment, regardless of source. If not the primary method to estimate the project, analogous is a quick, relatively simple method to validate estimates using other methods.

Step 6: Estimate cost and schedule based on analogous size and linear assumption

Once the appropriate analogous estimate has been completed, the result is a ROM estimate providing at a minimum, a cost, schedule, and effort for the project being estimated. As mentioned earlier, use of other estimating techniques such as parametric estimating or expert judgment, can be used to provide a check against the estimate and increase its confidence and defensibility. The key is the proper use of actual data, analysis of the data to make adjustment to account for differences between the project being estimated and the projects providing the data used, to ensure a reasonable ROM software project estimate.

Conclusion

Development of ROM estimates for software projects early in the SDLC, regardless of development and estimating methodologies, uses present the estimator/cost analyst with significant challenges. However, using the IFPUG SFP software sizing method, in combination

FROM CONOPS TO ROM ESTIMATE: SIX EASY STEPS FOR CUSTOM SOFTWARE
DEVELOPMENT PROJECTS

14

with the analogous estimating technique based on completed project data, provides the estimator/cost analyst the means to create a reasonably high confidence cost, schedule and effort estimate based on high level requirements such as those in a ConOps, in a short period of time.