

# Estimation of Commercially Off-The Shelf (COTS) Software

<sup>1</sup>Usha Divakarla, <sup>2</sup>Jitin Khurana

Lecturer-IT, LIMAT, student, LIMAT,  
ushachavali@gmail.com, jitinkhurana@yahoo.co.in

## ABSTRACT

*Software estimation provides the development effort and schedules based on the estimated size of the software product, and is calculated using mathematical models based on data from software projects that have already been completed. The estimates generated by these models can be refined using various tuning parameters allowing for factors such as technical complexity of the product, experience of the development team, and maturity of the development process. The results of the estimation process provide project managers with a range of options and probabilities for the likely project timelines, and levels of effort required to achieve them. A COTS (commercial off-the-shelf) product is one that is used "as-is" and are designed to be easily installed and to interoperate with existing system components. Almost all software bought by the computer user fits into the COTS category: operating systems, office products suites, word processing and e-mail programs are among the myriad examples. One of the major advantages of COTS software is its relatively low cost.*

## 1.INTRODUCTION TO SOFTWARE ESTIMATION

Software Cost Estimation is defined as Predicting the resource required for Software Development Process

Software Cost is measured by

- Manpower Loading
- Effort
- Duration

Software Cost Estimation Process is done by Set of techniques and procedures that an organization uses to arrive at a software cost estimate. Managing the cost of software development and its maintenance are major concerns of companies today. The process of calculating the cost of software development starts with estimating how much work needs to be done to successfully complete the project, what resources are required and how long will it take. Quality of the software has a significant impact on the cost of the software as finding and fixing bugs is a major cost element in the development process. Once the application is delivered, ease of maintenance and enhancement of the application considerably affects the overall cost of the software to the organization.

Software development organizations are regularly challenged to provide early and precise estimates for the work they are setting out to do. However statistics given by the United States government show that these organizations have not adequately handled this area as

- 60% of projects are behind schedule
- 50% are over cost
- 45% of delivered projects are unusable

Given the above statistics it is fair to conclude that 50% of the work undertaken by software organizations fails to meet the expectations of their customers in terms of productivity, quality and cost.

### 1.1 Software Estimation

Software estimation has been an issue ever since the first project was scheduled and budgeted. In order to address this task adequately the first challenge is to define and understand the problem domain of the project, also referred to as the scope of the project. Scope determines the boundaries of the software. Without well-defined boundaries, accurate estimates are unattainable. The complexity in defining the scope of a project arises from lack of information or domain expertise.

In order to provide a viable estimate, the software organization needs to capture the business requirements that are normally in the heads of the customer rather than on paper. To add to the complexity a reasonable size project involves multiple stakeholders having pieces of information that affect the project. Capturing and providing a unified vision to all involved, i.e. project stakeholders and the development team, is a key pre-requisite in accurately determining the size and complexity of a project. Hence a software development team that has a keen grasp on the customers' business issues and is inherently familiar with the environment the software needs to integrate in reduces the risk of inaccurate estimates significantly.

Estimating software is also dependant upon the skill set available and development processes within the software organization. It is imperative for a software organization to understand the capability of its resources and follow processes that ensure successful delivery. Thus enabling the software organization to determine effort needed for delivering a quality

solution in a given timeframe. Opting for a development team whose delivery model has been put to the fire and can be trusted greatly reduces the risk of undue heartburn in the future.

## 2.COMMERCIALLY-OFF-THE-SHELF (COTS)

### 2.1 Definition

A COTS package[1] is a software product that is:

- Supplied by a vendor
- Sold, leased, or licensed to an acquirer
- Used without modification of its code
- That may be tailored to be integrated into the acquirer environment
- Supported and evolved by the vendor, who retains the intellectual property rights

### 2.2 Characteristics of COTS

Based on this definition, we can identify a list of unique characteristics of COTS packages[2]

- The marketplace, not the needs of a single system, drives COTS package development and evolution.
- COTS packages and the marketplace undergo frequent, almost continuous, change.
- Frequency and context of COTS package releases are determined at the discretion of the vendor.
- 

- COTS packages are built based on unique architectural assumptions that may not be applicable to the target organization.
- At best, consumers have limited visibility into COTS package internals and behavior.
- COTS package assumptions about end-user processes may not match those of a specific organization.
- A COTS vendor is not a subcontractor. Acquirers seeking to influence package changes require a different type of relationship to the software supplier.
- COTS package components often have unsuspected dependencies on other COTS package components.

## 3. PROPOSED METHOD FOR ESTIMATING COTS

### 3.1 Introduction to Use Case Estimation

Working in Ericsson in the late 1960s, Ivar Jacobson devised Use-Case Documents. Thanks to Ivar Jacobson to come out with such a wonderful way of communication by using Use Case Documents. Later, Use Case Documents became a subset of UML[9]. In 1994, Alistair Cockburn constructed the 'Actors and Goals conceptual model' while writing Use Case guides for the IBM Consulting Group. It provided guidance as to how to structure and write Use Cases. It's the document which can stand not only for programmer or architect, but also for the stake holders. It's a document which stands between the Customer and Programmers/Architects/Business analysts/etc. It also serves as handover when any new programmer comes in the project. Use Case Document also serves as a valuable input to the design of software. In short, it serves in the whole life cycle of software development. Karner identified that this Method can also be used to measure and estimate effort.

**Definition:** Use Case Point is software sizing and measurement based on "Use Case Point" by Gustav Karner in 1993. It was written as a diploma thesis at the University of Linköping. This work is a modification of work by Allen Albrecht on function points.

### Use Case orientation Analysis Process steps

1. Requirements Use Case Analysis
2. COTS Functional Evaluation
3. Use Case Orientation
4. Project Estimates
5. Make/Buy analysis

### 3.2 Estimating COTS using Use Case Estimation

Use case models are increasingly being used to capture and describe the functional requirements of a software system. There are different approaches and methods to successfully estimate effort using use cases. A few researchers have tested the use case points method and analyzed their findings. The results, though not conclusive, indicate that the use case points method has potential to be a reliable source of estimation, much like the function point method, and it can have a strong impact on estimating the size of software development projects, especially when it is used along with expert estimates. Also, since use case modeling is increasingly being utilized as the method of choice to describe the software and system requirements and as a basis of design, development, testing, deployment, configuration management and maintenance, it makes sense to have an estimation method that makes use of them.

### 3.3 Step-1: Requirements Use Case analysis

1. Determine the UAW (Unadjusted Actor weight): The first step is to classify all the actors in to the following classification. Table 4.0 will give you a clear idea of how to classify the actors. Second column is the litmus test for making a decision of which type of actor falls in which category. The last column provides the factor of complexity:

Table 1: Classify Actors

Classification	Litmus test to recognize classifications	Value/Factor
Simple actors	Simple actors are those which communicate to system through API.	1
Average actors	Average actors are recognized if they have the following properties: <ul style="list-style-type: none"> <li>Actors who are interacting with the system through some protocol (HTTP, FTP, or probably</li> </ul>	2

	some user defined protocol). <ul style="list-style-type: none"> <li>Actors which are data stores (Files, RDBMS).</li> </ul>	
Complex	Complex actor is interacting normally through GUI.	3

2. Determine number of UUCW (Unadjusted Use case Weight): The second step is to count Use Cases and assign weights depending on number of scenarios and number of transactions.

Table 2: use case points

Use Case Type	Litmus test to decide the Classification	Value/Factor
Simple	Greater than or equal to 3 transactions	5
Average	Between 4 to 7 transactions	10
Complex	Greater than 7 transactions	15

3. Determine Total UUCP (Unadjusted Use Case Point): Total UUCP = Total UAW + Total UUCW.

Computing technical and environmental factor: Final step is to take into account the technical complexity. All technical factors will be assigned a value from 0 to 5 depending on complexity.

Table 3: Technical Factor		
Technical factor	Weight	Description

T	Distributed System	2	Is the system having distributed architecture or centralized architecture?
T	Response time	1	Does the client need the system to fast? Is time response one of the important criteria?
T	End user efficiency	1	How's the end user's efficiency?
T	Complex internal processing	1	Is the business process very complex? Like complicated accounts closing, inventory tracking, heavy tax calculation etc.
T	Reusable code	1	Do we intend to keep the reusability high? So will increase the design complexity.
T	Installation ease	0.5	Is client looking for installation ease? By default, we get many installers which create packages. But the client might be looking for some custom installation, probably depending on modules. One of our client has a requirement that when the client wants to install, he can choose which modules he can install. Is the requirement such that when there is a new version there should be auto installation? These factors will count when assigning value to this factor.
T	Easy use	0.5	Is user friendliness a top priority?
T	Portable	2	Is the customer also looking for cross platform implementation?

T	Easy to change	1	Is the customer looking for high customization in the future? That also increases the architecture design complexity and hence this factor.
T	Concurrent	1	Is the customer looking at large number of users working with locking support? This will increase the architecture complexity and hence this value.
T	Security objectives	1	Is the customer looking at having heavy security like SSL? Or do we have to write custom code logic for encryption?
T	Direct access to third parties	1	Does the project depend in using third party controls? For understanding the third-party controls and studying its pros and cons, considerable effort will be required. So, this factor should be rated accordingly.
T	User training facilities	1	Will the software from user perspective be so complex that separate training has to be provided? So this factor will vary accordingly.

4. Equation for Tfactor = sum(T1...T13)
5. TCF (Technical Complexity Factor):  $TCF = 0.6 + (0.01 * Tfactor)$ .
6. EF (Environmental Factor): There are other factors like trained staff, motivation of programmers etc. which have quiet a decent impact on the cost estimate.

7. Efactor = SUM(e1...e8).
8. Calculating Environmental Factor = EF = 1.4 + (-0.03 \* Efactor).
9. AUCP (Adjusted Use Case Points). Finally, calculating the Adjusted Use case points: AUCP = UUCP \* TCF \* EF
10. Multiplying by Man/Hours Factor: AUCP \* Person/Hours/AUCP.

**Example:**

Effort Estimate	
TCF (Technical Complexity Factor)	0.91
Environmental Factor	0.83
Adjusted Use Case Points (AUCP)	45.318
Productivity(In Hours per UCP)	10
Effort Estimate in Person hours:	453.18
Number of working days in a month	20
Total Project Effort (Person Months)	2.83

**3.4 Step-2: COTS Functional Evaluation**

Review COTS product Utilizing Use Case Points and count from Step-1. Identify the following Use Cases.

- A. Exist in COTS with no change required
- B. Exist in COTS but require modification
- C. Need to be Added to COTS Product
- D. Need to be removed from COTS Product

**3.5 Step-3: Use Case Orientation Analysis**

Determine what the percentage of out of box functionality is  
 Determine percentages of enhancement requirements (i.e. Add changes and deletes etc)

**3.6 Step-4: Project Estimates**

In this Step the following activities are performed

- Estimate effort, staff, and schedule for the

“Develop from Scratch Alternative”[10].

- Estimate effort, staff, and schedule for the COTS Alternative

**3.7 Step-5: Make / Buy analysis**

The following are the observations derived from above calculations

Buy As is If:

- Prepared to live with it
- Willing to change business to adapt to application
- Schedule sensitivity is overriding factor
- Development and future maintenance funding is limited

Customize if:

- Cost is more viable than building (Development and ongoing support)
- Minor Customization.
- Relatively schedule sensitive

Develop If

- Specific Requirements are not available
- Initial cost of COTS [11] (With support) is higher than developing and supporting
- Ongoing upgrade and support costs
- User questions and concerns

The following are considerations to be made for Make / Buy analysis

- Evaluation costs
- Return on Investment (ROI)[12].
- Cost of Purchase (Initial Purchase cost)
- Site Licenses (Per site Costs)
- Annual Maintenance Support (Scope of this support)
- Access To package Specifics (Data Model, Process Model)
- Ownership of customized Software
- Vendor would do the customization or can the buyer do the customization
- Use the analysis information to discuss alternatives, proposed time frames and budget constraints.
- Existing Database structures (Does it meet standards? and maintainability?)
- Existing Architecture (Learning curve and compatibility)

**4.CONCLUSION**

This proposed work reports results to aid the selection of components for a real world Materials Module and Supply chain application. The results allow the user to answer the potential to aid selection between components in a medium-sized application.

The observed benefit is that this work encourages and supports wider stakeholder involvement in selection by allowing them to define and reason with requirements. Likewise the model allowed architects, developers and integrators to communicate and develop a shared understanding of the new application.

Use Case Points have the potential to produce reliable results because its estimates are produced from the actual business processes – the use cases - of a software application. Additionally, in many traditional estimation methods, influential technical and environmental factors are often not adequately given enough consideration. Use Case Points includes and abstracts these subjective factors into an equation. When tweaked, over time, Use Case Points can provide estimates that are very reliable.

## REFERENCES

- [1]. Ibid.
- [2]. P. Oberndorf, L. Brownsword, and C. Sledge. "An Activity Framework for COTS-Based Systems," (CMU/SEI-2000-TR-010 ADA383836). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
- [3]. See P. Oberndorf, L. Brownsword, and C. Sledge. "An Activity Framework for COTS-Based Systems," (CMU/SEI-2000-TR-010 ADA383836). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2000.
- [4]. Albert and Brownsword, Op cit. Estimation Using Use Case Points by Mel Damodaran Computer Science Program, University of Houston-Victoria [http://interactive.sei.cmu.edu/Features/1998/June/Applying\\_COTS/Applying\\_COTS.htm](http://interactive.sei.cmu.edu/Features/1998/June/Applying_COTS/Applying_COTS.htm)
- [5] L.Brownsword, T.Oberndorf, C.Sledge. "Developing New Processes for COTS-Based Systems". IEEE Software July/August 2000, pp. 48-55
- [6] Constructive COTS Model (COCOTS) Status.by Chris Abts.
- [6] C.Abst, B.Boehm, E.Clark. "COCOTS: A COTS Software Integration Lifecycle Cost Model - Model Overview and Preliminary Data Collection Findings", Technical report USC-CSE-2000-501, USC Center for Software Engineering, 2000.
- [7] <http://sunset.usc.edu/research/COCOMOII/index.htm>
- [8] [www.liemur.com/Articles/UseCasePointEstimation.html](http://www.liemur.com/Articles/UseCasePointEstimation.html)
- [9] Use case estimation - the devil is in the detail. By Vinsen, K. Jamieson, D. Callender, Murdoch Univ., Perth, WA, Australia.
- [10] Effort estimation of use cases for incremental

large-scale software development by [Bente Anda](#), [Reidar Conradi](#)

[11] V.Basili, B.Boehm. "COTS-Based Systems Top 10 List" IEEE Computer 34(5), May 2001, pp 91-93

[12] How to Estimate a software project in man-hours by [RedSunBeer](#).

Table 4: Environmental Factors

Added, Changed and Unchanged COTS by increment				
<b>Increment -1</b>	<b>Added</b>	<b>Changed</b>	<b>Unchanged</b>	<b>Total UCP</b>
Onsite	9444	200	258	9902
Development Center	2862	418	167	3447
<b>Total</b>	<b>12306</b>	<b>618</b>	<b>425</b>	<b>13349</b>
<b>Increment -2-4</b>	<b>Added</b>	<b>Changed</b>	<b>Unchanged</b>	<b>Total UCP</b>
Onsite	6887	18	9	6914
Development Center	1182	0	0	1182
<b>Total</b>	<b>8069</b>	<b>18</b>	<b>9</b>	<b>8096</b>
<b>Total Project</b>	<b>Added</b>	<b>Changed</b>	<b>Unchanged</b>	<b>Total UCP</b>
Onsite	16331	218	267	16816
Development Center	4044	418	167	4629
<b>Total</b>	<b>20375</b>	<b>636</b>	<b>434</b>	<b>21445</b>
<b>Percent of total F</b>	<b>95%</b>	<b>3%</b>	<b>2%</b>	<b>100%</b>