Evaluating COTS Using Function Fit Analysis

This article presents function fit analysis (FFA), a methodology proven to be successful in evaluating commercial off-the-shelf (COTS) products as they relate to meeting customer requirements. The steps of this function point-based process are discussed to show the benefits of providing this information to the decision-making process for the use of COTS as a development solution.

Today many organizations are attempting to reduce software development effort and schedule by purchasing off-the-shelf solutions rather than building them in-house. This strategy can be very cost effective if the COTS solution meets customer requirements. Unfortunately, COTS solutions have often proven to be a great disappointment. This is largely due to poor *fit* in meeting the required functionality. The result is a major COTS enhancement project comparable to a custom-developed solution in terms of overall project schedule and cost. In one such situation, it was found that the COTS solution fit only 2 percent of the requirements. Ninety-eight percent of the solution would have to be delivered as new development and enhancements to the COTS package. Luckily, in this situation the problem was identified early in the process and the proposed COTS solution was rejected.

The situation described above and similar experiences highlight how important it is to conduct a complete and accurate assessment of how a COTS solution fits the requirements and does not rely on vendor claims of high compatibility. This article discusses a technique that has been successful for projects in evaluating the compatibility of COTS products to customer requirements. The technique, called function fit analysis, is based on function point analysis. Function point analysis is the decomposition of an existing or planned system based on the user's perspective of functional requirements. Function points can be used to evaluate various COTS solutions, select the best solution, and determine the degree of enhancement work necessary to meet customer requirements.

Function fit analysis provides the ability to:

- Document functional requirements in terms understandable to users and technicians.
- Identify the functional gap of the COTS products.
- Quantify the effort necessary to enhance the COTS package.
- Provide input into the *make vs. buy* decision making process.

Since function points are a key element to this process, it is important to understand their definition. "Function points are a measure which represents the functional size of application software" [1]. Function points are a unit of measure that represent the work products of software developers. They quantify the deliverables of the software development process. When combined with other data, such as effort and defects, metrics can be developed to aid in planning and managing software projects. Function points were originally developed as a communication tool for defining functional requirements in nontechnical terms. To accomplish this they describe functionality from the end user's perspective of how it supports one's business functions.

To count function points it is necessary to understand the counting rules as well as the user requirements of the project or system being assessed. For that reason, it is important to have knowledgeable participants and supporting documentation available when conducting the count. The function point process as defined by International Function Point User Group follows.

Function Point Analysis Process

There are three types of function point counts:

1) Development project count.
2) Enhancement project count.
3) Application count.

Determining which count to be produced is the first step in the process. The different types of counts and how they are used in the function fit analysis process are described below.

Development project counts are used to size projects with totally new functionality and include all functions being developed. In function fit analysis, the development project count would compare to the *custom-developed* alternative. Enhancement projects are modifications to existing systems and include application functions that are added, changed, or deleted. This is the type of count used when evaluating specific COTS alternatives in the function fit analysis process. It identifies enhancements necessary in the COTS product for it to meet customer needs. The final type of count is an application count. This is the function point count of any installed system. Once a system exists, this is the function point count of all functions provided to the user regardless of how they were delivered (i.e. developed vs. COTS).

To complete a function point count, user recognizable functions are identified and evaluated. From a user's perspective, a computer application assists him in doing his job by providing five basic functions. Two of these capabilities address the data

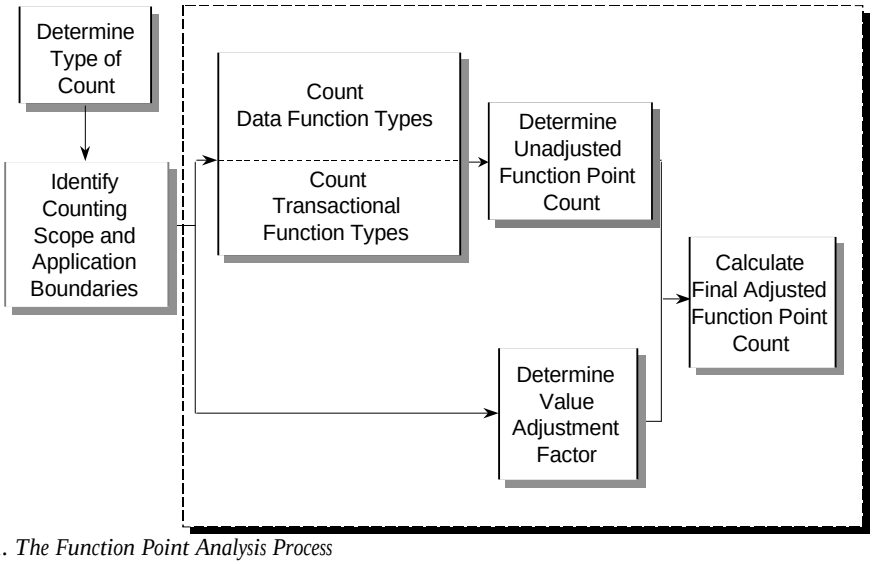


Figure 1. *The Function Point Analysis Process*

requirements of the business and are referred to as data functions. Data functions consist of internal logical files and external interface files. Three of these capabilities address the user's need to access and manipulate data stored on the files and are referred to as transactional functions. Transactional functions consist of external inputs, external outputs, and external inquiries.

The Five Components of Function Points

Data Functions

1. Internal Logical Files
2. External Interface Files

Transactional Functions

3. External Inputs
4. External Outputs
5. External Inquiries

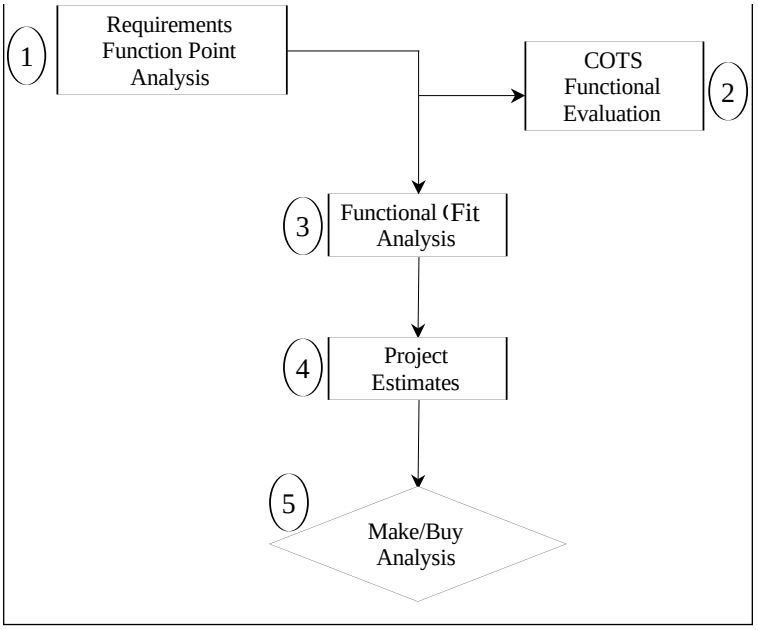
The first data function allows users to utilize data they are responsible for maintaining. For example, a user may be responsible for adding, changing, and deleting employee information on the employee master file. Therefore, the user is responsible for maintaining the file. Logical groupings of data that are maintained by an end user of an application are referred to as internal logical files (ILFs).

The second function of an application provided to an end user is also related to logical groupings of data. In this case, the user is not responsible for maintaining the data, which resides in another application and is maintained by another user. The user of the application being counted requires the other application's data for reference purposes only. For example, a user may require the ability to access security information from the security application. The user does not have the responsibility for updating security information but must reference it to complete his or her job. Groupings of data from another application used only for reference purposes are defined as external interface files (EIFs).

The remaining functions address the user's capability to access the data contained in ILFs and EIFs. This capability includes maintaining, inquiring, and outputting of data, referred to as transactional functions.

The first transactional function allows a user to maintain ILFs through the ability to add, change, and delete the data. For example, a user can add, change, or delete employee information on the employee master file. In this case, the user is utilizing a transaction referred to as an external input. An external input gives the user the capability to maintain data on ILFs through adding, changing, or deleting its contents.

Figure 2. *Function Fit Analysis Process*



The next function gives the user the ability to produce outputs that contain calculations. For example, a user may require a report that contains expense data with derived information. The report is produced using maintained and referenced information. In function point terminology, the resulting report is called an external output.

> . . . [Function fit] analysis should only focus on quantifying user requirements, not on potential COTS solutions.

The final capability provided to users through a computer application addresses the requirement to display specific data from files. In this situation there is no manipulation of the data, it is a direct retrieval of information contained on the files. For example, a user can inquire on employee data by inputting a Social Security number and retrieving information. These transactions are referred to as external inquiries.

Two adjustment factors are applied to calculate the function point count. The first adjustment, functional complexity, assigns weights to each functional component based on data elements and processing logic. The second adjustment, the value adjustment factor, evaluates the operational complexity of the application.

Function points are just one piece of information that is used in the function fit analysis process. The function fit analysis process consists of five primary steps.

Function Fit Analysis Process

Step 1: Requirements Function Point Analysis

This step is used to identify and document exactly what functionality is necessary to meet the customer needs. This assessment should be completed early in the development life cycle to provide insight into the scope of the project. It can be completed early because it is based on *what* functions are to be delivered, not *how* they are to be delivered. During this step, a function point count is completed based on documented functional requirements, or by analyzing the functions of an existing system to be replaced. It is important to focus on the *to be* version of the system and to identify all of the functions necessary to meet the users' needs.

The analysis should only focus on quantifying user requirements, not on potential COTS solutions. The result is a development function point count and a detailed listing of the functions necessary to meet the requirements. For example, a course registration application may contain the following user functions:

Add training course information
Modify training course information
Display training course information
Establish training sessions
Add participants to training sessions

Step 2: COTS Functional Evaluation

This involves reviewing the various COTS choices and comparing their functionality to the requirements documented in Step 1. From a function point perspective, this is an enhancement project count, as we are starting with a system and identifying changes necessary to meet the functional requirements. Using the function point count from Step 1 as a guide, each database and panel in a COTS alternative is reviewed to identify functions that:

1. Exist in the COTS with no change required [Unchanged].
2. Exist in the COTS but require enhancements to meet the requirements [Change].
3. Need to be added to the COTS product [Add].
4. Exist in the COTS but are unrelated to the requirements and will not be used [Unused].

The resulting enhancement function point count summarizes how well the COTS product matches the functional requirements. Using the example functions from above, the COTS evaluation function point count contains the following:

User Function	Enhancement Activity
Add training course information	Unchanged
Modify training course information	Change
Display training course information	Change
Establish training sessions	Add
Add participants to training sessions	Add

The COTS enhancement project function point count will include functions that are being added and changed. The first analysis of results should compare the function point size of the COTS enhancement project to the function point size of the custom-developed solution. If the enhancement function point size is not significantly less than the custom-developed solution, then the COTS solution might not be the best *fit.* The second step of the analysis is to evaluate the *unchanged* functions since this represents what portion of the solution *fits* the user requirements. The *unused* functions show how much of the COTS product is unusable and unrelated to the requirements.

Step 3: Functional Fit Analysis

This step is used to apply a percentage to the *fit* between the requirements and the COTS product. By comparing the function point counts from the previous two steps a fit can be determined. One of the first tasks is to define what *fit* means in the minds of all those involved in the assessment. There are different ways of looking at fit depending on the approach used. In function fit analysis, *fit* is defined as the amount of out-of-the-box functionality that can be utilized without any modifications. Using this definition, a comparison of the requirements function point count to the COTS function point count will result in calculating the *fit* percentage of the COTS (FFA fit = unchanged FPs ÷ total FPs from Step 1).

In addition to the *fit* calculation, analysis of the percentage of added and changed functions is also useful. These percentages can be calculated by identifying the *added* and *changed* functions from Step 2, calculating a function point count, and evaluating it against the total function point count from Step 1.

The *added* function point count will show the amount of user requirements that are not supported at all in the COTS product. Obviously, if this is a large number, it may be better to look at a different alternative or consider a custom-developed solution.

Identifying *changed* functions is helpful in evaluating whether to enhance the COTS to meet the business requirements, modify the business process to meet the COTS, or a combination of both. This analysis provides an opportunity to revisit certain functions with the users to see if the requirements are flexible.

Table 1 details added, changed, and unchanged function points for a project.

Identifying and quantifying the *unused* functions in the COTS product can be helpful in a couple of ways. They are used to determine how much of the purchased COTS solution will not be utilized. Also, reviewing these functions with users may help to generate ideas on how to improve the business process in the future.

Knowing the *fit* numbers is one piece of information that is helpful in the make/buy decision. The other piece of information is the estimate to deliver the functions in the various options, which leads us to the next step, project estimates.

Step 4: Project Estimates

The function point data from the previous steps is used to develop the project estimates in Step 4, which uses a top-down approach to estimate effort, staff, and schedule for each alternative under review. The first is for the *custom-developed* alternative. Two components are used to develop an effort estimate: project size, which is the function point count from Step 1 of the function fit analysis process, and an evaluation of the project attributes, factors that, in addition to project size, influence software development productivity. Four major attribute categories are evaluated:

1. Personnel and Management—focuses on knowledge and experience of information systems and end user personnel involved in the project.
2. Process and Methods—focuses on what development and project management methods are used and the extent the methods are followed.
3. Technology and Tools—evaluates the technology being utilized and the effectiveness of the tools available to the developers.
4. Environment and Support—evaluates the development environment in terms of computer resources, administrative resources, and overall working environment.

Once the size and the project attributes are determined, an effort estimate can be developed. This can be done using various industry tools, industry benchmark data from external consultants, or organization-specific historical data. The estima-

Total Project		Added	Changed	Unchanged	Total AFP
	System A	750	350	100	1,200
	System B	800	250	50	1,100
	Total	1,550	600	150	2,300
	Percent of Total Project	67%	26%	7%	100%

Table 1. *Added, Changed, and Unchanged COTS*

tor selects the appropriate productivity rate to use based on the development project function point size and the project attributes. To calculate the effort estimate, the function point count is divided by the productivity rate (project effort = custom developed project size ÷ function points/hour). Schedule and staffing estimates are derived from the project effort incorporating organizational constraints, e.g. staff limitations and preset schedules.

A similar process is followed for each COTS option under review. For this analysis, the project size is based on the enhancement-function point count completed in Step 2 of the function fit analysis process. Project attributes should be reviewed to see if the previously completed assessment applies to the COTS alternatives. Enhancement productivity rates differ from development productivity rates depending on project size, so the effort estimates for the COTS alternatives may not use the same productivity rate (function points/hour) as the custom development option.

Now there is enough measurement data about the functional requirements to feed to the make/buy decision step.

Step 5: Make/Buy Analysis

The function point information and estimate data from Steps 1-4 is used in Step 5, the make/buy analysis. The make/buy analysis is the decision-making process to determine whether to implement a COTS solution or build a custom one. The following are some initial decision points based on information from previous steps:

Buy *As Is* if:

1. The users are prepared to live with the COTS functionality.
2. The users are willing to change their business processes to adapt to the COTS application.
3. The sensitive schedule is an overriding factor. This means delivering the functionality in a certain time frame takes precedence over how the users would like the business process to be. The schedule is the highest priority and it drives the decision.
4. Development and future maintenance funding is limited. If there is not going to be any money available to maintain or enhance a modified or developed system, then it may be best to rely on the COTS provider for enhancement and maintenance support.

Customize COTS if:

1. The cost to customize the COTS product is more viable than building. Development costs as well as ongoing support and operation costs should be included.
2. Minor customization is necessary to meet the user requirements.
3. The schedule is relatively sensitive. If developers can take advantage of existing COTS functionality and modify a minimal number of functions, then the development time frame will be shorter. If a short schedule is a requirement, then business functions should be evaluated to determine if they could be changed to utilize the COTS functionality, therefore limiting the customization.

Custom Develop (no COTS) if:

1. A significant number of requirements are not available. If the user requirements are specific and cannot be changed to use the existing COTS functionality, then building them from scratch is the best option.
2. Initial cost of the COTS, including enhancement and support dollars, is higher than developing and supporting the application in-house.
3. Ongoing upgrades are cost prohibitive. This point considers the cost of implementing future upgrades to the COTS product. If installing the upgrades requires rework of the customizations, then it is possible that significant development effort would be incurred each time an upgrade is installed. If this situation occurs, it would not be cost-effective.

In addition to evaluating the development effort the following factors should be included in the make/buy analysis:

COTS costs
Make sure to include all the costs associated with the COTS product. This would include evaluation costs, initial purchase costs, seat or site license costs, and annual support and operation costs.

Access to package specifics
Discuss with the COTS vendor what technical components of the product will be available to the in-house developers. For example, will they have access to the data model or process model? What *hooks* are available to the developers for adding custom code? Is there a coding standard the COTS provides to ensure custom-built components will have the same look and feel as the core product?

Ownership of customized software
Once the COTS solution has been purchased and customization is completed, there may be a gray line defining who owns what. Make sure you understand the rules up front.

Customization responsibility
Determine at the start who will make the changes to the COTS, and what the associated costs will be. This may also dictate who owns the customized code.

Existing database structure
Some organizations have strict guidelines on naming conventions and structures. It is beneficial to examine the COTS database structure to see if it can easily comply with your organization. If it does not, a decision can still be made to change the organization rules to meet the COTS.

Existing computing architecture
This will be important if the COTS product needs to communicate with other in-house, non-COTS, applications. The compatibility should be examined to determine the amount of work necessary to integrate multiple systems.

Once the above information is gathered and analyzed, the function fit analysis process is complete, and an appropriate make/buy decision can be made. The data and findings from the process should be documented and stored in a repository for reference. This information may be helpful in future analyses of this kind.

Summary

There is a great deal of information that is necessary to make an informed decision as to whether to utilize COTS in development efforts. The function fit analysis process provides an excellent framework for information gathering, evaluation, and decision making. It communicates functional requirements in objective terms, so the COTS can be evaluated from a *fit* perspective and iden-

tifies functions to evaluate for possible business process re-engineering efforts.

As function fit analysis provides a size metric for the development options, it enables estimates to be made in terms of effort, staff, and schedule. Without knowing *what* needs to be delivered, it is impossible to determine a good estimate of how long it will take to deliver it. Function fit analysis gives us the *what*.

In today's world of providing things better, faster, and cheaper, it can be difficult to determine the best option. The use of COTS products has been touted as a faster and better method for software development, but that is not always the case. Function fit analysis is a valuable technique to help evaluate purchased/customized solutions and make the best and most appropriate decision for each organization and project.◆

Reference

1. IPFUG Counting Practices Manual v. 4.1.

About the Author

Lori A. Holmes is a Senior Management Consultant with Q/P Management Group, specializing in Total Quality Management, software measurement, and process improvement. Prior to joining Q/P, she spent nine years at First Data Corp. in Omaha, Neb. as Quality Assurance Manager and Applications Programming Manager. Holmes received a bachelor's degree in business administration, business information systems focus, from Illinois State University in 1984, and is a certified function point specialist and a quality analyst.

Q/P Management Group Inc.
15539 Burdette St.
Omaha, Neb. 68116
Voice: 402-493-0228
Fax: 402-493-0506
E-mail: lholmes@QPMG.com

Web Addition

A Ship Cost Agent for Pier and Port Management

Agent-based collaborative decision support is a methodology utilizing a domain specific intelligence system to partner with human decision makers to reach a consensus solution to a complex problem. This paper describes the cost agent and its deployment within a collaborative planning facility management tool designed to support Navy pier and port management, the Collaborative Infrastructure Asessment Tool .

James A. Sena
College of Business, California Polytechnic State University

Find this article on the Internet at http://www.stsc.hill.af.mil/crosstalk/2000/feb/sena.asp

New CMMI Requirements for Risk Management

A new risk management process area is one of the changes proposed in the new Capability Maturity Model Integration® (CMMI) product suite. For some, this is long overdue in the software community; for others, this is a new and potentially difficult requirement.

The purpose of risk management is to identify potential problems before they occur, so that risk-handling activities may be planned and invoked to mitigate adverse impacts on achieving objectives. Risk management is a continuous, forward-looking process, integral to both business and technical management. It encompasses:

- Defining a risk management strategy.
- Identifying and analyzing risks.
- Developing and implementing risk mitigation plans.

In Software CMM v. 1.1, risk management was identified as a key practice in Software Project Planning, Software Project Tracking and Oversight, and Integrated Software Management.

At Maturity Level 2, projects identified, assessed, documented, and tracked software risks. At Maturity Level 3, projects were expected to actively manage the software risks in a more proactive and integrated fashion.

In practice, many current software projects give lip service to risk management in the CMM by listing a vague set of nonspecific risks (e.g., inability to meet schedule). Projects fail to identify or quantify project-specific risks or mitigation plans, or to continuously manage them throughout the project life cycle. Often, practitioners think that risk management is a systems engineering function, outside software's responsibility.

In creating the CMMI-SW/SE v. 0.2 maturity model, risk management was elevated to a process area, with a full set of required goals and expected practices that apply to software. This structure follows the System Engineering maturity models, where risk management has always been handled at the process area level.

In addition to specific project activities, risk management must be institutionalized through written organizational policies, plans, allocated resources, assigned responsibilities, training, configuration management of work products, quality audits, and management reviews.

Some organizations have already reaped the benefits of a mature, proactive, software risk management process. Resources include the SEI's Software Risk Taxonomy and Team Risk Management process, the Software Project Manager Network's Risk Radar tool® , and numerous books and training courses.

Contributed by Rick Hefner, Ph.D, TRW
Co-Chairman
CMMI Assessment Methodology Team
rick.hefner@trw.com