Applying an Activity Based Counting Technique
to System Design Efforts

Jeffrey K. Shupp
Lockheed Martin
Management and Data Systems
4000 Geerdes Boulevard
King of Prussia, PA 19406

Abstract. Established costing techniques applied to software based systems provide a reasonably accurate technique in predicting the implementation costs of projects. For projects with a firm design baseline, driven by well defined requirements, the system engineering cost is typically based on a “tax” of x% of the total development hours.

When the software system being proposed is large, or when the customer’s needs are not well articulated in the specifications, a simple multiplier against the total development effort is insufficient to predict the systems engineering effort. Even predictions based comparing design artifacts from one project to another cannot be successfully defended during cost negotiations.

This paper proposes a technique that assesses the scope of Systems Engineering work by examining the executing features of the design and their associated difficulty. Examining how the system runs early in the development cycle helps the project better understand where the unknowns and uncertainties are in the design, providing insight into technical risk management, as well as providing more defensible costs.

PREDICTING SYSTEM DESIGN EFFORTS

At a time when Systems Engineering is being recognized as a critical part of why developed systems succeed or fail, the ability to estimate the System Design effort has not kept pace. With the movement away from managing requirements, and towards the development of architectures by examining the executing nature of the system (Frank 1998; Insight 1998), there is a need to find a way to predict the Systems Engineering effort in a better way than is currently found.

Classically, the SE efforts are derived as a percentage of the overall development costs. Even commercially available costing products want the SE effort as a percentage multiplier as an input to the model, while the software effort utilizes LOC as its input. With development environments emphasizing COTS solutions as opposed to developed capabilities, and the infusion of emerging technologies, such as those associated with distributed computing architectures, treating the Systems Engineering efforts as a flat tax multiplier is insufficient and indefensible.

Using an experiential approach has yielded only limited success. M&DS operates in a varied set of domains, so predicting the SE effort for one project based on effort expended on a “similar” effort does not always yield accurate results. This experiential technique requires that the estimator be intimately familiar with both the new project and the comparing project in order to apply the appropriate “uppers” or “downers”.

Therefore, what is needed is an approach that can be applied to diverse domains, that accentuates what is common across projects, that can be accumulated into a database, and used more confidently on new projects. A bottoms up technique would help to differentiate the salient features of one system design from another: a technique that isolates work to be done in one project that may or may not be necessary in other projects would also be of benefit.

To meet the above objectives, two previously published approaches have attacked the problem by directly examining the tasks of the System Engineers themselves. The technique explored by (Langston 1996) looks at the SE effort from an engineering task perspective, asserting that the effort varies not because of intrinsic complexities in the respective tasks, but rather on whether certain tasks are performed. By breaking down the engineering process to a fine enough granularity, knowing which subset of process steps are appropriate for the project, and knowing how much effort each individual step takes, one can determine the cost of the SE effort without having to deal with the intrinsically subjective term of complexity. This technique holds hope for predicting costs; it does require that the individual subset of tasks employed in a Systems Engineering sense be implemented in a repeatable fashion from one project to another, with performance metrics collected at that level.
Another approach, published by (Miller 1994), employs a counting technique based on the design artifacts (number of requirements, number of tests, etc.) produced by the SE effort. In order for this technique to be accurate on future projects, the nature of the artifacts must be consistent from one project to another. In an environment where specifications range from hundreds to thousands of requirements for similarly sized (in LOC terms) systems, based on the customer’s needs and whether the project is commercial or governmental, these kinds of artifacts, as evidenced in our business, do not provide the needed predictability.

**SYSTEM FUNCTION POINTS**

Rather than looking directly at the effort of SE, as these other techniques do, the technique described here provides a robust examination of the tasks, or activities, that the system performs in meeting its objectives. There is ultimately one tangible output of the Systems Engineering tasks: the System Design. The System Design defines the executing nature of the system, and more importantly shows how the components interrelate through various “wiring diagrams” from the many architectural views that make up the system architecture. Between 60 and 80% of systems engineering work revolves around the analysis and design of, first, the system, and then, the components of the system, to meet (and sometimes, to push back against) mission and user objectives. The artifacts of the systems engineering effort are requirement specifications, concepts, analysis reports, equations reports, and validation and verification plans, procedures, and reports.

The cost estimation method proposed for defining the systems design is fashioned after the Function Point Analysis technique used for estimating software development costs, and is referred to here as the System Function Point technique. This technique, like FPA, counts the items of interest related to the system. Unlike FPA, whereas the influential factors reflecting the “environmental influences” on a software development effort are applied against the sum of the Unadjusted Function Points, the System Function Point technique embeds the Systems Engineering “environment” into the difficulty factors directly. The System Function Point technique contained in this paper captures the difficulty applied to each item counted based on the following four areas: Pervasiveness; Ambiguity; Technology; and Analysis / Evaluation Difficulty.

A final factor, known as Key Development Item, applies a multiplier to account for the driving features of the design.

Figure 1 shows a sample template for capturing the topic items and the difficulty factors associated with each of them. The example shown in the figure is a Satellite Mission Planning System; the items shown are a subset of those identified for the total system.

<table>
<thead>
<tr>
<th>Topic Items</th>
<th>Difficulty Factors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pervasiveness</td>
</tr>
<tr>
<td></td>
<td>L(low), A(average), H(high)</td>
</tr>
<tr>
<td>Mission Functionality / Concepts</td>
<td>Broker Comm Traffic</td>
</tr>
<tr>
<td></td>
<td>Notify User of Changes in Plan</td>
</tr>
<tr>
<td></td>
<td>Generate Requests for Comm</td>
</tr>
<tr>
<td>Mission Scenarios / Threads</td>
<td>Extend Existing Plan</td>
</tr>
<tr>
<td></td>
<td>Status and Reporting - External</td>
</tr>
<tr>
<td>Interfaces</td>
<td>Plan Publication</td>
</tr>
<tr>
<td></td>
<td>Comm Requests</td>
</tr>
<tr>
<td></td>
<td>Comm Granted / Denied</td>
</tr>
<tr>
<td>Operator Interactions</td>
<td>Monitor Plan Generation</td>
</tr>
<tr>
<td></td>
<td>View Recent Changes</td>
</tr>
<tr>
<td></td>
<td>Edit Current Plan</td>
</tr>
<tr>
<td>Data / Control Processing</td>
<td>Data Access - Ops Data</td>
</tr>
<tr>
<td></td>
<td>Initialize Test with Ops Data</td>
</tr>
<tr>
<td></td>
<td>IT Configuration Setup</td>
</tr>
<tr>
<td>Design Stability</td>
<td>Plan Generation Timeline</td>
</tr>
<tr>
<td></td>
<td>Shared DB Performance</td>
</tr>
<tr>
<td></td>
<td>Plug and Play Architecture</td>
</tr>
</tbody>
</table>

The information derived and published in a template such as Figure 1 is valuable throughout the
Mission Functionality. This topic examines the system design in terms of the functionality derived from the requirements allocated to the system from the customer specifications and interfaces to outside systems.

Beginning with the requirements, a sufficient level of functional decomposition is performed to understand the functionality and performance expectations of the design. Functional decomposition also highlights the perishability of data input to / generated within the design, and the need for control gates in the design to ensure that processing occurs in an appropriate order, or continues under certain error conditions. The functionality derived from interfaces in this context is NOT the functionality to accept data from and output data to interfaces; rather, it is oriented around the work that is derived in support of the accepted data or that is derived to produce the expected output data. A separate topic, Interfaces, addresses the work of the data itself.

Companion to the functional decomposition of the design component are a set of concepts that describe the behavior of the components with respect to the mission. These concepts address such issues as the extent of operator control and/or monitoring required over the process flow through the design components, the resilience of the design to keep processing when part of its functionality cannot be performed, etc. Therefore, this task centers around the systems engineering effort required to arrive at these “kingpins” of mission functionality and behavior.

Mission Scenarios / Threads. Mission scenario / thread development is very important in understanding the data processing and control flow among the components within the system design. It helps to identify the events (e.g. the receipt of a message, a detected error condition, the passage of time, an operator action) that initiate the various processing functions that were identified above. Scenarios and threads are the vehicle for expressing the processing timeline assignments to, and defining the interface relationships between, the various components or sub-components.

Scenario development is not restricted to situations that require multiple processing entities to express the response to an event. Sometimes a single component will be responsible for the entire reaction to the event. Scenarios are a way of exploring and understanding the processing reactions to events that affect the system or component design. Therefore, it is not so important how many components contribute to the scenario; rather, it is more important to capture the understanding of how the design responds in the scenario.

Interfaces. The term “interfaces” refers to the data or control relationship between the entity being designed to its peers. Interfaces exist between the system and data sources and sinks outside the system and among components within the system. Since the functionality derived from the interfaces has been addressed previously under the topic of Mission Functionality, this view is concentrated at the data and control information level of the interface itself, e.g. the format, content, and protocols that enables messages to be sent to and read by recipient processes. Therefore, this task defines the amount of systems engineering effort needed to achieve this level of detail.

Operator interactions. Operator interactions represent a special case of interfaces. In this case, software or hardware only exists on one side of the interface. Since people don’t interact at the bit level, an HMI “bridge” needs to be created. Coupled with the mission function and concepts, this effort encompasses the work necessary to define the purpose of the HMI bridge (process control, monitoring, assessment), and how big the bridge is (displays to be built). Because of the intrinsic complexities of certain operator interactions, many times the HMI bridge is engineered through prototyping activities, with extensive analysis by the prospective operators of the system being developed.

Another side of operator interaction involves the examination of the operator tasks and an analysis of how “busy” the operators are over the course of their shift, along with an assessment of the operators’ skill level required to accomplish those tasks. This is also where operations staffing is examined, with discussions of shift versus day positions staffing,
training, and decision hierarchy being emphasized. Several day-in-the-life scenarios are constructed to ensure that operations perspectives are accommodated in the design.

The systems engineering group(s) must be able to articulate which of the identified mission functions have an operator interaction aspect to them, either as a processing control gate, data entry, or for solution assessment and approval. Using these mission functions as a basis, a set of operator functions is defined. Examples of operator functions include examining or editing an activity plan, receiving a notice that processing has commenced, initiating or holding equipment commands, approving the schedule just produced, and monitoring the progress of a processing task sequence.

**Data and Control Processing.** The Data and Control Processing topic examines the design from the aspect of the underlying processing architectures and paradigms, as opposed to examining how it responds to the mission and system requirements, which was covered in the first four topics discussed previously. This topic covers functions and concepts regarding how data and processing capabilities are made available to personnel inside and outside the system who are responsible for analysis of the system - as in information system - performance. It is concerned with how processing is initiated, terminated, quiesced, and resumed after being quiesced. It deals with how data is managed in its repository (creating new data, purging expired data, maintaining integrity among related data) and made available to the process from the data repository, as well as how it is passed from process to process. It deals with the decisions to use executive functions or peer to peer messaging to manage processing flow, and decisions about how static data updates are incorporated into the system. Finally, this topic covers concepts for test and maintenance, e.g. the ability of the component to support discrepancy identification and workoff, the ability to stop time / quiesce processing in a test mode, etc.

**Design Stability.** For any project, certain aspects are less mature than others. This is especially true in the performance arena - both in timeline and in product quantity, quality or throughput. These uncertainties represent technical risks to the system design, and if not addressed become Program risks to successful completion of the project. As the project progresses, effort must be expended to mitigate those uncertainties. Therefore, this topic addresses the uncertainties and unknowns found in the system design at the time of the costing activity, usually just before the proposal is generated. It also captures those performance requirements that require effort to resolve the values stipulated in the specification. Further, it captures the special trade areas for functional or architectural uncertainties with the proposed design that must be worked before the components can be finalized.

**DETERMINE EFFORT DIFFICULTY**

With the count of the functions and concepts known, the next step is to determine the extent of engineering effort that will be required to analyze, synthesize, decompose and associate the functions, concepts, and requirements among the components in the System Design. Therefore, factors regarding pervasiveness, ambiguity, technology, and evaluation difficulty are examined for each concept and function identified. The final factor, whether the concept or function is considered “key” is applied. Like FPA, the System Function Point technique discriminates the difficulty factors along three bins, Low, Average, and High. Employing any further granularity is going beyond the accuracy of the technique.

**Pervasiveness.** The pervasiveness factor is related to the overall impact the topic item has to the System Design. This factor is considered Low if the topic has a highly localized influence on the design, High if the topic’s influence is pervasive across the design, and Average represents influences in-between. For example, a scenario to process a message from an interface that affects only one component within the system design would be assigned a difficulty factor value equal to Low. Conversely, a processing scenario to a message that affects most of the components of the system design would be assigned a difficulty factor value of High.

For the sample project shown in Figure 1, a full spectrum of indexes were derived under this difficulty factor category. As a general rule, because mission events that trigger processing are usually associated to distinct actions, mission scenarios and interfaces tend to be less pervasive than data and control processing items. This rule is evident in this sample. Specific Operator actions / reactions tend to be less pervasive than monitoring tasks, again, as evidenced in the sample system.

**Ambiguity.** The Ambiguity difficulty factor is applied to reflect the amount of interpretability there is to the data or control input or produced for a given topic
item. An ambiguity factor value of Low indicates that the information has only one interpretation. An ambiguity value of High indicates that the information may be convoluted to have multiple meanings depending on other data or control conditions present or absent at the time this information is known. A value of Average indicates that the information present, although not uniquely defined, has only a limited number of possibilities.

In general messages whose contents are defined in fixed formatted fields are valued as Low or Average. Messages in which multiple fields must be read to know the exact nature of the data, and in which the interpretation is significantly different depending on subtle differences in those fields, would be assigned an ambiguity value of High.

Functions, scenarios and operator interactions also have an associated ambiguity factor. In these instances, the data or control is not being examined; rather, it is the processing necessary to properly act on the data or control. Functionality performed unconditionally, or operator interaction performed independent of processing conditions would be assigned a value of Low. Functionality and scenarios which involve coordinated processing among several components, but in a well orchestrated fashion, would be assigned an ambiguity factor of Average. Functionality or operator interaction that is highly dependent on conditions present at the time of processing would be assigned a value of High.

Ambiguity is significantly higher for systems characterized by decision points. The mission planning system, as illustrated by Figure 1, is a classic example of such a system. In this system, a request for comm support can be granted, denied, or granted initially only to be denied some time later. The systems engineering effort is focused on understanding what all those conditions are, and more importantly, how the system reacts or anticipates such decisions.

**Technology.** The application of technology to solve system engineering problems is the most significant difficulty factor to be considered in the costing. The amount of engineering effort expended to solve capability and performance issues that lie at the current state of the technology can sometimes overwhelm the system engineering effort.

Technology factor values of Low would be found in follow-on programs, where major portions of the system are brought forward untouched by new requirements, interfaces, and services. Technology factor values of Average would constitute the bulk of new development programs, where technology is pushed forward in a marginal fashion - to use an existing technology in a new way. Factors valued at High indicate a state of the art level of technology is needed, and would likely be found in IR&D projects, or where a major change of system characteristics is called for - as in moving from a Mainframe processing architecture to a fully distributed one.

Technology is usually a wild card. If it can be avoided, projects should stay away from items exhibiting high technology difficulty factors. In the sample, the technology associated with both meeting the mission and the infrastructure objectives are Low to Average, indicating that the system being developed is borrowing from known capabilities, maybe exercised in a new way, more than developing state of the art solutions.

**Analysis / Evaluation Difficulty.** This difficulty factor addresses the effort of the systems engineering group(s) to meet the objectives imposed by the various function points identified. These efforts are most often coupled to functions and requirements that have high complexity or technology issues with them. Included in this factor are analyses related to processing timeline performance and mission performance against the objectives, as well as basic services (message flow, database access) performance. The System Design address processing timeline performance through processing models incorporated into tools, where the execution processing is modeled against a given stimulus, with delays built into the processing for data retrieval from a database and message routing to and from other processes. Some operator functions may need to be prototyped to ensure that operator tasks can be met.

The Analysis / Evaluation Difficulty factor is considered to be Low if the topic item is well defined, a reuse item, or whose performance attributes are known to already exceed the expectations for the program being developed. A factor value of Average indicates that the item requires a paper or spreadsheet level of analysis, or utilizes an already existing analysis tool, requiring straightforward changes to the data that drives the tool. A value of High demonstrates that the analysis or evaluation of that topic item will be extensive. Requiring a new prototype, creating a new model, creating a significant number of use cases, or simply maintaining and updating the data driving the model over an extended period of the development cycle are all characteristics of a High analysis / evaluation difficulty factor. New algorithms created to operate in environments with very high quality or
throughput, or very short timelines are typically considered to exhibit High factor values.

As shown in Figure 1, the sample system has some special areas in which significant effort will be required to ensure that the design put in place meets the objectives. These efforts show up in both functional areas, as well as timing and performance areas. Looking across the overall topical areas, the theme associated with requests for comm service represent the place in the design where the most analysis needs to be done. When coupled with the columns about technology, the message is that existing technology is being applied to a new application, but there is a confidence issue as to whether those solutions will in fact work in the new system. This issue can only be addressed by a focused effort to analyze the design in its new system environment.

**Key Development Item.** This difficulty factor comes from the extra effort required to ensure that a key function or concept is properly addressed to the satisfaction of the customer. After applying the criteria by (Kaslow et al, 1996), if an item falls into the category for a Key Development Item (KDI), the total function points computed through the preceding factors is multiplied the KDI factor (if value = Y, multiply the first four terms by 2, for example) to arrive at the total System Function Points for the topic item. This extra effort is required to support detailed design reviews by internal and external panels, more detailed analysis required to prove the design meets the defined objectives, and more extensive testing to ensure that the completed product performs as the design expects. If the item of interest does not warrant Key Development Item status, the System Function Point total is based on the first four factors only.

For the sample system of Figure 1, there are only three items that meet the criteria as Key Development Items. These three items are the areas where the design drives the architecture decisions the most, and pose the highest risk to the baseline design defined in the proposal could be undone as development progresses. Therefore, efforts in these areas are going to get a lot more internal and external attention, and hence more effort will be expended to prove that the system design is sound in these areas.

**REPEATABILITY OF THE TECHNIQUE**

For this technique to become a useful tool for analyzing and predicting a project’s engineering efforts, it has to have enough stability that two people similarly trained would produce the same counts. This is obviously not unique to System Function Points; this very same issue arises with standard Software FPA. Like FPA, there are two general areas where the application of a System Function Point technique could yield widely different results among estimators: number of items, and the subjectivity of assessing the difficulty factors.

On a recent pilot project, a team of engineers examined the same design documentation and independently performed the SFP analysis. Two of the three people received no extensive training into FPA or SFP counting techniques, only a few hours of reading material equivalent to what is found in this paper. As feared, one engineer had identified items that were fairly high level, which exhibited higher average pervasiveness and ambiguity. A second engineer decomposed the items further, creating more individual items, but each item now exhibited a lower index for the same two difficulty factors. Even though the number of items arrived at was significantly (up to 50%) different, the total SFP scores were within 10%. Therefore, the technique seems to have a built in compensation for overzealous counters.

To avoid the subjectivity normally associated with complexity, as cited in (Langston 1996), the difficulty factors are arrived at through a technique similarly employed for the SE-CMM ratings. The factor is viewed through a series of questions; the rating of Low, Average, or High is derived from the responses to the questions. In this way, the subjectivity is minimized. Again, on the same pilot project, there was a clear sense of common interpretation exhibited in the application of the difficulty factors. On average, only about 20% of the previously agreed to list of items showed a difference in difficulty interpretation. In about half of these instances, the “odd” score was determined to be wrong. Therefore, the technique is not nearly so sensitive to subjectivity as had been anticipated.

**GETTING TO A COST PROFILE**

Once the template has been filled in for the project, the next step is to translate the information to an equivalent function point total. Since there is no industry level database of System Engineering level function point analysis to draw from, the scores associated with each topic and difficulty factor must be drawn from within the organization.

Effectively, the organization must reverse engineer from historical projects the metrics necessary to construct a system function point database. Through examination of the design, requirements specifications, and interviews of contributing personnel, Figure 1 can
be filled in for each of the topics for the now-completed project. With the cost sheets from the project, aggregates of the topics can be directly correlated. Judgment is required to break the aggregates down into the individual contributions of each of the difficulty factors. M&DS is in the formative stages of examining projects of similar features to derive values for the weighting terms for each of the difficulty factors, as well as the Key Development Item multiplier. As more confidence is gained with the technique, we hope to test the premise of being applicable across diverse domains by examining dissimilar projects.

CONCLUSION

Applying a more rigorous technique to estimating the systems engineering design effort of a project has many immediate benefits over a more traditional “flat tax” technique. First and foremost, it forces the systems engineering organization to recognize what is unique about this project compared to “like” projects that are used as costing benchmarks. These uniquenesses, once understood, allow great credibility in projecting cost differences between projects. The technique also provides rigor in the process of stating assumptions about various facets of the design going into the design step, so that management can assess the risks that the system design activity entails. Finally, the technique allows projects of disparate domains to be compared from a common perspective, based on the design effort difficulties, rather than the resulting artifacts of the design.

REFERENCES

Humphrey, Watts S., Managing the Software Process, Addison-Wesley, MA, 1989
INSIGHT, A publication of the INCOSE, Volume 1 Issue 1, Spring 1998
Miller, William D. “Systems Engineering Metrics ++”, Fourth Annual NCOSSE Symposium (San Jose, CA) August 1994

BIOGRAPHY

Jeffrey K. Shupp is a Senior Systems Staff Engineer with the Technical Operations group at Lockheed Martin’s Management and Data Systems operation in Valley Forge, Pennsylvania. He has 19 years experience developing software systems, first in the Nuclear industry with GE’s Nuclear Business Group, then in the Aerospace industry with M&DS. Mr. Shupp has been a member of INCOSE since 1996. He co-authored and presented a paper in the 1996 symposium based on material he co-authored for a book dealing with ground mission operations. Mr. Shupp has a Bachelor’s degree in Engineering from the Pennsylvania State University, and a Master’s degree in Engineering from the University of California at Berkeley.