Software Effort & Schedule Measurement:

A Framework for Counting Staff-hours

and Reporting Schedule Information

Wolfhart B. Goethert

Elizabeth K. Bailey

Mary B. Busby

Software Effort & Schedule Measurement:
A Framework for Counting Staff-hours
and Reporting Schedule Information

Wolfhart B. Goethert

Elizabeth K. Bailey

Mary B. Busby

with the Effort and Schedule Subgroup of the Software Metrics Definition Working Group
and the Software Process Measurement Project Team

Software Engineering Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Table of Contents

List of Figures

Preface

In 1989, the Software Engineering Institute (SEI) began an effort to promote the use of measurement in the engineering, management, and acquisition of software systems. We believed that this was something that required participation from many members of the software community to be successful. As part of the effort, a steering committee was formed to provide technical guidance and to increase public awareness of the benefits of process and product measurements. Based on advice from the steering committee, two working groups were formed: one for software acquisition metrics and the other for software metrics definition. The first of these working groups was asked to identify a basic set of measures for use by government agencies that acquire software through contracted development efforts. The second was asked to construct measurement definitions and guidelines for organizations that produce or support software systems, and to give specific attention to measures of size, quality, effort, and schedule.

Since 1989, more than sixty representatives from industry, academia, and government have participated in SEI working group activities, and three resident affiliates have joined the Measurement Project staff. The Defense Advanced Research Projects Agency (DARPA) has also supported this work by making it a principal task under the Department of Defense Software Action Plan (SWAP). The results of these various efforts are presented here and in the following SEI reports:

- *Software Size Measurement: A Framework for Counting Source Statements* (CMU/SEI-92-TR-20)
- *Software Quality Measurement: A Framework for Counting Problems and Defects* (CMU/SEI-92-TR-22)
- *Software Measures and the Capability Maturity Model* (CMU/SEI-92-TR-25)
- *Software Measurement Concepts for Acquisition Program Managers* (CMU/SEI-92-TR-11)
- *A Concept Study for a National Software Engineering Database* (CMU/SEI-92-TR-23)
- *Software Measurement for DoD Systems: Recommendations for Initial Core Measures* (CMU/SEI-92-TR-19)

This report and the methods in it are outgrowths of work initiated by the Effort and Schedule Subgroup of the Software Metrics Definition Working Group. Like the reports listed above, this one contains guidelines and advice from software professionals. It is not a standard, and it should not be viewed as such. Nevertheless, the frameworks and recommendations it presents give a solid basis for constructing and communicating clear definitions for some important measures that can help all of us plan, manage, and improve our software projects and processes.

We hope that the materials we have assembled will give you a solid foundation for making your effort and schedule measures repeatable, internally consistent, and clearly understood by others. We also hope that some of you will take the ideas illustrated in this report and apply them to other measures, for no single set of measures can ever encompass all that we need to know about software products and processes.

Our plans at the SEI are to continue our work in software process measurement. If, as you use this report, you discover ways to improve its contents, please let us know. We are especially interested in lessons learned from operational use that will help us improve the advice we offer to others. With sufficient feedback, we may be able to refine our work or publish additional useful materials on effort and schedule measurement.

Our point of contact for comments is

Kathy Cauley
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Acknowledgments

The SEI measurement efforts have depended on the participation of many people. We would like to thank the members of the Effort and Schedule Subgroup of the Software Metrics Definition Working Group who contributed to the content and structure of this document. The SEI is indebted to them and to the organizations that sponsored their participation in the efforts to improve the measurement of effort and schedule. Without the contribution of these professionals, we could not have completed this task:

This report builds on the *Software Project Effort and Schedule Measurement* report that was presented and distributed for review at the SEI Affiliates Symposium in August 1991. A first draft of the current document was distributed to a large number of reviewer in June 1992. More than 140 comments and suggestions for improvement were returned. All have received careful consideration, and most have been either incorporated or addressed through the development of new materials. We are indebted to those who took the time and care to provide so many constructive recommendations:

John Salasin
Software Engineering Institute

Norman Schneidewind
Naval Postgraduate School

Marie Silverthorn
Texas Instruments

Al Snow
AT&T Bell Laboratories

S. Jack Sterling
Logicon Eagle Technology, Inc.

Irene Stone
AIL Systems Inc.

Bob Sulgrove
NCR Corp.

Susan Voigt
NASA (Langley Research Center)

Software Effort & Schedule Measurement: A Framework for Counting Staff-Hours and Reporting Schedule Information

Abstract. This report contains guidelines for defining, recording, and reporting staff-hours. In it we develop a framework for describing staff-hour definitions, and use that framework to construct operational methods for reducing misunderstandings in measurement results. We show how to employ the framework to resolve conflicting user needs, and we apply the methods to construct specifications for measuring staff-hours. We also address two different but related aspects of schedule measurement. One aspect concerns the dates of project milestones and deliverables, and the second concerns measures of progress. Examples of forms for defining and reporting staff-hour and schedule measurements are illustrated.

1. Introduction

1.1. Scope

This report presents an approach to obtain operational methods for defining and recording staff-hours and related schedule information. It provides the following:

- A checklist-based framework for increasing clarity and consistency and reducing misunderstanding in derived measures by making the staff-hour and schedule measures exact and unambiguous.

- A checklist form that enables project managers to identify the issues and choices they must address to avoid ambiguity and to communicate precisely what is included and excluded in the staff-hour and schedule measurements.

- Examples of how to use the checklist to construct specifications that will meet differing objectives.

- Examples of forms for recording and reporting measurement results.

Refer to Appendix B for our rationale for using staff-hours to measure software project effort.

1.2. Objective and Audience

Our goal is to reduce ambiguities and misunderstandings in measures that use staff-hour and schedule by giving organizations a foundation for specifying and communicating clear definitions of the staff-hour and schedule measurements. There are three primary reasons for collecting staff-hours:

- To pay individuals (payable hours)

- To charge for hourly services (billable hours)

- To use in productivity and quality studies (actual hours)

We provide operational methods to help organizations implement clear and consistent recording, reporting, and use of staff-hours and schedule information. In many cases the hours that should be reported vary with the purpose for which they will be used. Staff-hour measurement and schedule information are key elements in software project estimating, planning, and tracking.

This report is appropriate for managers, developers, maintainers, estimators, and process improvement teams who want to use measurement to help plan, control, and improve their processes for acquiring, building, and supporting software systems.

1.3. The Software Measurement Environment

The framework presented in this report is based on the notion that a software organization has or will create a software measurement environment structured along the following points:

1. Goals and objectives are set relative to the software product and software management process.
2. Measurements are selected to ascertain the degree to which the goals and objectives are being met.
3. Data collection processes and recording mechanisms are defined and used.
4. A data analysis and corrective action process is defined and used.
5. Measurements and reports are part of a closed-loop system that provides current (operational) and historical information to technical staff and management.
6. Post-software product life measurement data is retained for analysis leading to improvements for future product and process management.

These points are prerequisites for all measurement environments, and are stated here to emphasize that their implementation is essential for the successful use of the framework described in this report [IEEE Std 982.1988].

2. Defining a Framework for Software Effort Measurement

The framework proposed in this report uses checklists extensively. These checklists provide the following:

- An operational mechanism for creating and communicating explicit definitions.
- A means for extending an organization's understanding of its products and processes as process maturity increases, without having to change underlying definitions.
- A means for structuring definitions to meet the different information needs of various organizations.

We used two principal criteria in preparing the framework and checklists:

- *Communication*: They must precisely communicate what is being measured and what has been included and excluded in the numbers presented.
- *Repeatability*: They must enable others to repeat the measurements and get the same results.

The framework we use to satisfy these criteria consists of the following steps:

1. Identify the principal attributes that characterize the object we want to measure.
2. Identify the principal classes of values within each attribute that users may want to include in their measurements and ensure that these classes are mutually exclusive.
3. Identify the principal classes of values within each attribute that users may want to exclude from their measurements.
4. Prepare a checklist of principal attributes and their values, so that values included in and excluded from measures can be explicitly identified.
5. Select the values to be included in the measure and exclude all others. Record the selections and exclusions on the checklist.
6. Specify project-specific information to provide the context for the data and to allow its comparison across projects.
7. Make and record measurements according to the definition and data specifications.
8. Attach the measurement definition and data specifications to each set of measurement reports.

These criteria and tactics have led us to the Staff-Hour Definition Checklist, the Schedule Definition Checklist, the Supplemental Information Forms, and the recording forms for both staff-hours and schedule. Appendix E contains blank forms that can be used as

reproduction master. In the following sections, we will describe the forms and explain how they are related and how they work together.

2.1. Staff-Hour Definition Checklist

The process we propose for constructing a staff-hour definition starts with a checklist. The checklist identifies the principal attributes of staff-hours the software engineering community wishes to measure and also identifies the values that each attribute can take on. Values listed for an attribute should be both exhaustive (complete), to provide a means of recording each possible value of the attribute, and mutually exclusive (non-overlapping), to avoid ambiguity as to which recorded value is to be used. After we list principal attributes and their values and arrange them into a checklist, the process for constructing a definition becomes relatively straightforward. We use the checklist in constructing a definition by simply checking all attribute values we want to include in and exclude from our definition. We construct a supporting form to record any special situations that are not amenable to checklist treatment. Forms help to ensure robust and repeatable measurement.

Initially the Staff-Hour Definition Checklist may be used not to require a specific definition for staff-hours, but rather be used in conjunction with staff-hour reports to communicate what attributes and values were included in a specific staff-hour measurement.   In this case, the checklist provides a structured approach for dealing with the details that must be resolved to reduce misunderstandings when using the data from staff-hour measurement reports.

Since staff-hour definitions help to make staff-hour reports more meaningful, we have included in the Staff-Hour Definition Checklist a means to construct report specifications that contain the attribute values for which individual subtotals are desired. Our Staff-Hour Definition Checklist makes it possible to construct numerous report specifications based upon the same definition of staff-hours. Figure 2-1 illustrates three report specifications based upon the same staff-hour definition. For each report specification, the reporting details may be different, but the staff-hour definition will be the same.
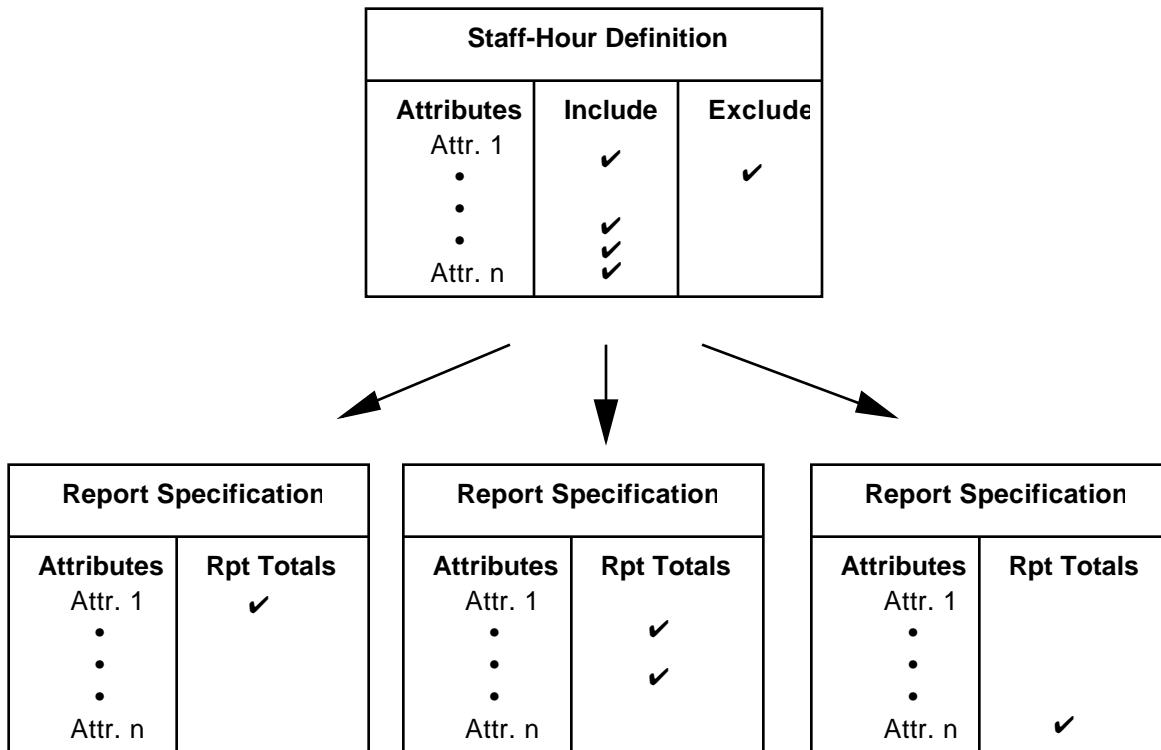
Figure 2-1 Example of Multiple Report Specifications

For ease of use we have combined the staff-hour definition and the report specification into a single Staff-Hour Definition Checklist as illustrated in Figure 2-2.

Staff-Hour Definition		
Attributes	**Include**	**Exclude**
Attr. 1	✔	✔
•		
•	✔	
•	✔	
Attr. n	✔	

Report Specification	
Attributes	**Rpt Totals**
Attr. 1	
•	
•	✔
•	
Attr. n	✔

Staff-Hour Definition Checklist			
Attributes	**Include**	**Exclude**	**Rpt Totals**
Attr. 1	✔	✔	
•			
•	✔		✔
•	✔		
Attr. n	✔		✔
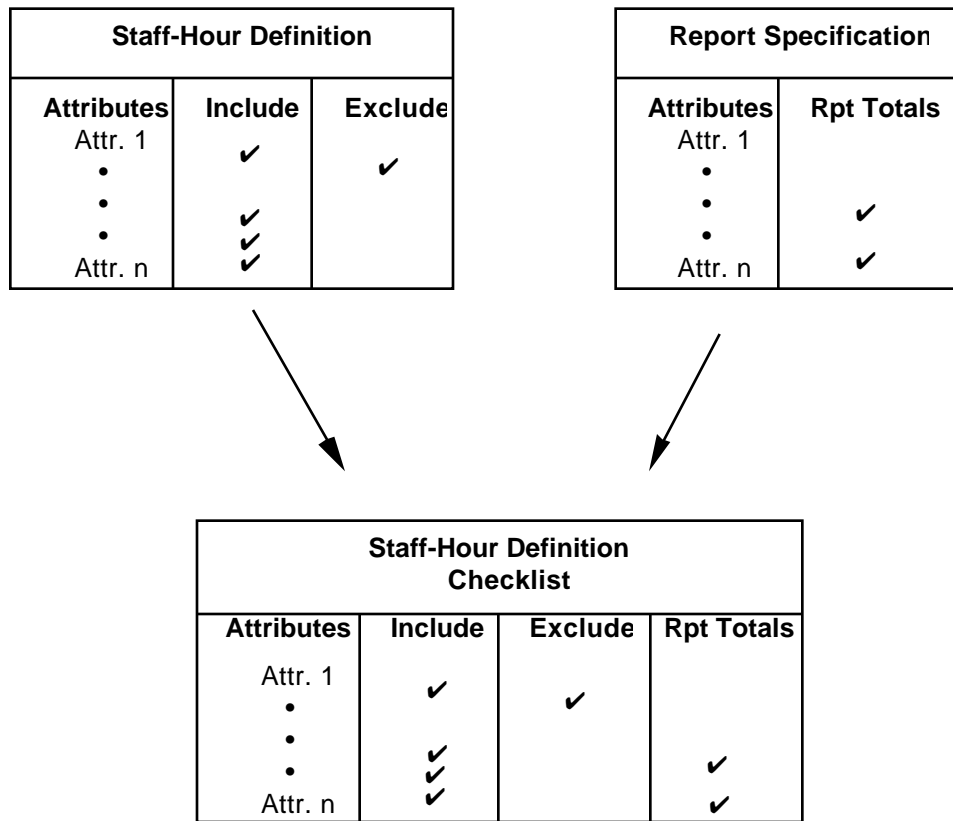
Figure 2-2 Interrelationship of Staff-Hour Definition and Report Specification

We have designed the checklist so that when a particular staff-hour is counted, it will have one and only one value per attribute. Values within an attribute must be both mutually exclusive and collectively exhaustive. We show the attributes as bold-faced attribute headings, each followed by a list of the classes of values that the attributes may take on. Chapter 3 discusses in detail the full set of attributes and values for the definition.

The checklist uses seven attributes to describe and bound the kinds of effort included in a measure of staff-hours.   These attributes are: Type of Labor,   Hour Information, Employment Class, Type of Pay, Labor Class, Activity, Product-Level Function (CSCI-Level Functions, Build-Level Functions, System-Level Functions).   Values within an attribute do not overlap. When creating a definition, users have only to check the attribute values that they will include and those they will exclude when measuring and recording staff-hours. We have also included blank lines that may be used to expand partitioning of values if required to meet local needs.

The user of the checklist constructs a definition of staff-hours by checking off all attribute values included in and excluded from the definition. The Report Total column is used to construct report specifications that contain the attribute values for which individual subtotals are desired.

The checklist we have designed helps to provide a structured approach for:

- Dealing with details that must be resolved to reduce misunderstanding.
- Communicating unambiguously just what is included and what is excluded from the measurement.
- Specifying attribute values for which individual reports are wanted.

Figure 2-3 is an example of how the first page of a completed checklist might look; it shows requests for a number of reports for one particular definition of staff-hours. A blank Staff-Hour Definition Checklist is located in Appendix E.

Staff-Hour Definition Checklist

Definition Name:	**Total System Staff-Hours**	Date:	**6/27/92**
	For Development	Originator:	**SEI**
		Page:	1 of 3

Type of Labor	Totals include	Totals exclude	Report totals
Type of Labor			
Direct	✔		
Indirect		✔	
Hour Information			
Regular time			✔
Salaried	✔		
Hourly	✔		
Overtime			✔
Salaried			
Compensated (paid)	✔		
Uncompensated (unpaid)	✔		
Hourly			
Compensated (paid)	✔		
Uncompensated (unpaid)	✔		
Employment Class			
Reporting organization			
Full time	✔		
Part time	✔		
Contract			
Temporary employees	✔		
Subcontractor working on task with reporting organization	✔		
Subcontractor working on subcontracted task	✔		
Consultants	✔		
Labor Class			
Software management			
Level 1	✔		
Level 2	✔		
Level 3		✔	
Higher		✔	
Technical analysts & designers			
System engineer	✔		
Software engineer/analyst	✔		
Programmer	✔		
Test personnel			
CSCI-to-CSCI integration	✔		
IV&V	✔		
Test & evaluation group (HW-SW)	✔		
Software quality assurance	✔		
Software configuration management	✔		
Program librarian	✔		
Database administrator	✔		
Documentation/publications	✔		
Training personnel	✔		
Support staff	✔		

Figure 2-3 Example of Completed Staff-Hour Definition Checklist

In practice, checklists turn out to be very flexible tools. An organization may include a particular attribute in its definition and report a subtotal for each of several particular values for the attribute. For example, an organization may report the subtotal for design as a sum of preliminary design and detailed design. A checklist can easily address this option.

One word of caution: There are cost implications that users must keep in mind when constructing report specifications (Report totals column). The level of detail depends on the needs, goals, and objectives of an organization and project. Each organization and project will be different. If the information requested is part of a contractor's normal time reporting system, the cost associated with providing the requested subtotals may be minimal. However, if the information required depends on manual collection and collation, the cost may be very high. For small or exploratory projects, the cost of collection may exceed the value of the data, making detailed measurement collection undesirable for such projects.

2.2. Supplemental Information Forms

Sometimes definition checklists cannot adequately describe all the information that must be addressed to avoid ambiguities and misunderstandings in measurement data. When this occurs, we recommend construction of supplemental information forms to record and communicate the additional information. The combination of a completed checklist and its supporting forms becomes a vehicle for communicating the meaning of measurement results to others, both within and beyond the originating organization.

Our Supplemental Information Form for staff-hour measurement includes the following:

- Work period length: The number of staff-hours in a work day, work week, and labor month varies between organizations. This information needs to be recorded so that derived measures in terms of these work periods can be compared.

- Labor class clarifications: Not all organizations use the same labor class terms or include the same responsibilities in various labor classes. These differences need to be described so that valid data comparisons can be made.

- Product-level function clarifications: Different organizations count the staff-hours for some functions at different product-levels (major functional element, customer release, or system level). This information needs to be recorded so that valid data comparisons can be made.

Chapter 4 discusses in detail the form we have constructed for communicating supplemental information about staff-hours.

2.3. Reporting Forms

The checklist must be supported by reporting forms that record and communicate measurement results, providing a vehicle that can be used by those who enter the results into a database. These forms must be consistent with the attributes and values designated for measurement, and should capture all information needed to track the data back to both the definition and the entity measured. In many cases the data that should be reported vary with the purpose for which they will be used. The reporting forms should be designed to allow the user to communicate precisely what was included in the measurement, as well as what was excluded. Each organization should determine its primary objective before completing any of the reporting forms.

Chapter 5 discusses in detail the form we have constructed for reporting staff-hours and schedule information.

3. Understanding Staff-Hour Checklist Attributes and Values

In this chapter, we define and illustrate the attributes and attribute values used in the definition checklist and reporting forms. This chapter also addresses why the issues that the checklist seeks to resolve are important. The sequence of discussion follows the order of items on the checklist so that information on specific elements can be readily located. We have used boldface type to highlight the attributes and their values. The sequence of attributes is as follows:

- Type of Labor

- Hour Information

- Employment Class

- Labor Class

- Activity

- Product-Level Functions

    - CSCI-Level Functions (major functional element)
    - Build-Level Functions (customer release)
    - System-Level Functions

The list of attributes and attribute value documents what constitutes staff-hour measurement data for a given software development project. This helps to ensure that those who receive the data know exactly what it contains. It also helps to avoid oversights when collecting the data for later analysis.

Not all software development projects use the terms that we present in the checklist. However, we have attempted to describe the terms in sufficient detail below so that their meaning is clear. Also, not all of the attribute values may be included in the staff-hour measurement definition for a given project. The detailed list is meant to be used initially as a "memory jogger" for employees who are asked to develop the staff-hour definition for specific projects. This ensures that significant elements of staff-hours in a given definition were excluded consciously and not merely overlooked. Because some software development projects may have additional staff-hour attributes or values, we have included blank lines where you may add project-unique elements.

Reminder: The format of the checklist is not meant to imply that you should collect separate subtotals for every individual value of every attribute. The costs associated with software measurement data collection cannot be ignored. The benefits gained from collecting more detailed data must be weighed against the costs of that data collection.

3.1. Type of Labor

	Totals include	Totals exclude	Report totals
Type of Labor			
Direct			
Indirect			

Figure 3-1 The Type of Labor Attribute

Labor on software development projects consists of two types—direct and indirect hours. Direct hours are those that are charged directly to the project or contract, and indirect hours are those that are not. Even though indirect hours are not charged directly to the contract, the costs associated with these hours are frequently covered, at least in part, in the burden or overhead rates that are often applied as multipliers to the direct contract or project charges.

You may include all of the attributes described in the following sections in direct staff-hours, if such a definition is legitimate for the individual project or contract. The staff-hours associated with development of a software product for an internal or external customer are usually direct project charges. However, organizations and contracts vary on how software process development staff-hours are charged. Software process activities directly related to development of a specific software product (such as defining the procedures, standards, and conventions to be used in developing that product) are usually directly charged to the project. Other software process activities, such as software quality assurance, may be charged directly to the projects being covered or may be included in indirect charges. Still others, such as organization software engineering process improvement team activities, are usually not specific to a given project, but are applicable to numerous projects. Therefore, the costs associated with staff-hours for these activities are frequently included in indirect charges. The same can be said for secretarial support, internal training, and computer operations functions. Overhead rates applied to the direct charges cover these indirect charges.

An organization can use direct and indirect staff-hour counts for various planning and tracking purposes. Counts of direct staff-hours are useful in tracking actual hours expended versus planned hours. These counts are also frequently used in conjunction with source line counts to determine productivity rates. These rates in turn are useful in estimating staffing needs on future software development projects. You may use counts of indirect staff-hours in calculating burden rates to be applied to direct charges so that overhead costs are covered. However, since indirect charges are usually made for activities that are not specific to a given software product, we recommend that you include only direct staff-hours in a project's staff-hours measurement definition.

3.2. Hour Information

	Totals include	Totals exclude	Report totals
Hour Information			
Regular time			
Salaried			
Hourly			
Overtime			
Salaried			
Compensated (paid)			
Uncompensated (unpaid)			
Hourly			
Compensated (paid)			
Uncompensated (unpaid)			

Figure 3-2 The Hour Information Attribute

Staff-hours, whether direct or indirect, may be regular time or overtime hours. Regular time consists of the hours in the usual work day for a given software development organization. The 8-hour work day remains the norm for most organizations; however, some have work days of 9, 7.5, or 7 hours. The work shift—first, second, or third—is not a factor in measuring work effort. For true shift work, each shift has a regular work day. Staff-hours beyond an employee's regular shift are considered overtime. For some organizations, overtime hours do not accrue until an employee has worked the number of hours in the regular work week. Any number of hours may be worked in a given day, but none are considered overtime until the 40 (or 45 or 37.5 or 35) hours in the regular work week have been accumulated. Depending on company or organization policy, the contract, and the employee's job position, overtime hours may or may not be compensated; and if uncompensated, the overtime hours may not even be recorded.

This attribute describes the wage or pay type for the employees working on a given software development project. It is further subdivided into the salaried and hourly types of pay. Whether an employee is considered salaried or hourly depends on individual company policy regarding the type of pay for a given job position. Usually, salaried employees are in more professional job positions, such as the fields of engineering, law, and medicine, which require more education than do hourly positions. These employees are generally paid an annual salary, although they are not paid on an annual basis. That is, they receive a paycheck on some periodic basis: monthly, semi-monthly, weekly. On the other hand, hourly employees are paid at an hourly rate. However, they may receive their paychecks on the same schedule as do the salaried employees in the same company or they may receive them more frequently.

Salaried positions are frequently considered *exempt* under federal government labor law. That is, employees in higher level salaried positions are usually not compensated for

overtime. Employees in entry and low-level salaried positions may or may not be compensated for overtime. Hourly positions are usually considered *non-exempt*. Hourly employees are generally paid for overtime and at a higher hourly rate than for their regular time.

Even though salaried employees are not paid at an hourly rate, it is necessary to record the hours in some fashion, probably using the same timekeeping system as for the hourly employees. It should not matter whether the hours have been worked by salaried or hourly employees as long as those hours are applicable, i.e., they fall under the **Type of Labor** definition.

For clear insight into the effort expended on a software development project, all staff-hours must be counted, both regular time and overtime. This is true regardless of whether the overtime hours have been compensated and regardless of the type of pay. (Obviously, the staff-hours associated with unrecorded overtime cannot be included.) The many unknowns involved can make it extremely difficult to estimate in advance the number of staff-hours that will be required to complete a project. As a result, cost overruns and schedule slips can occur. However, if historical data from previous, similar projects are available, including both regular and recorded overtime hours, planners can more accurately estimate the resources required to complete a new project.

3.3. Employment Class

	Totals include	Totals exclude	Report totals
Employment Class			
Reporting organization			
Full time			
Part time			
Contract			
Temporary employees			
Subcontractor working on task with reporting organization			
Subcontractor working on subcontracted task			
Consultants			

Figure 3-3 The Employment Class Attribute

The employment class attribute includes the reporting organization and contractor categories. The reporting organization is the prime developer of a software product. This may be an in-house software development organization or it may be the prime contractor if a contract has been established to develop a given software product. In turn, the prime developer may have subcontractors with responsibility for providing support functions or for developing portions of the software product.

The full-time staff consists of those employees who are hired to work at least a full work week, whatever number of hours the work week is defined to be. Part-time staff

members are hired to work some number of hours less than a full work week. Full-time and part-time employees may be hired on a permanent or temporary basis. Temporary employees may be supplementals such as college co-ops and interns, retirees who return to work temporarily to give the benefit of their expertise, or term employees hired for a prescribed period of time. It should not matter whether the staff-hours counted are from full-time or part-time employees if those hours are included under the **Type of Labor** definition described above.

Contracted employees may be temporaries hired for short periods of time to cover for absent permanent employees of the prime developer or to work on short-term tasks so the prime developer need not hire additional staff members. Contractors may also be hired to work on long-term tasks, either participating directly on tasks being worked on by teams of employees of the prime developer or working semi-independently on a subcontracted task. Consultants are contracted employees who may provide assistance on either a short-term or long-term basis. As with the full-time and part-time staff-hours, it should not matter whether the hours counted are from the prime developers organization or from a subcontractor, as long as they fall under the **Type of Labor** definition. All of these hours need to be counted for clear insight into the effort expended on a software development project.

It is useful to count the full-time/part-time and prime developer/subcontractor hours separately to be able to determine the amount of work being done by each employment class. Turnover is frequently higher among part-time and subcontracted employees. If a high percentage of the work on a software development project is being performed by part-time and/or subcontractor employees, the schedule risk may be greater.

3.4. Labor Class

Labor Class	Totals include	Totals exclude	Report totals
Labor Class			
Software management			
Level 1			
Level 2			
Level 3			
Higher			
Technical analysts & designers			
System engineer			
Software engineer/analyst			
Programmer			
Test personnel			
CSCI-to-CSCI integration			
IV&V			
Test & evaluation group (HW-SW)			
Software quality assurance			
Software configuration management			
Program librarian			
Database administrator			
Documentation/publications			
Training personnel			
Support staff			

Figure 3-4 The Labor Class Attribute

The **Labor Class** attribute consists of the various classes of functional job positions on a software development project. The list of labor classes in the Staff-Hour Definition Checklist covers the functional job positions associated with most software projects. However, not all organizations use the same terms for labor classes as are in the checklist, nor do all companies organize software project efforts into exactly these classes of job positions. Some classes may be combined, and others may be broken out differently. Therefore, valid comparisons of individual labor class subtotals may not be possible between different organizations. However, where the labor class definitions are consistent within an organization or between organizations, you will be able to compare multiple projects. The classes listed below cover a broad range of categories; however, you may need additional classes for a specific project. If so, blank lines are available at the end of the labor class section of the definition checklist to allow for additional classes.

As stated before, all hours should be counted if they fall under the **Type of Labor** definition for a given software development project. It should not matter what classes of labor are involved. However, it is useful to accumulate separate subtotals for some of

the specific classes of labor. For example, the totals hours accrued by the software designers and programmers are frequently used with software size counts to calculate productivity rates. These rates are useful in planning schedules for later builds (releases) for a software development project or for future projects. Similar productivity rates may be calculated for other labor classes as well. Labor classes included are described in the following sections.

3.4.1. Software management

This class covers employees in supervisory and managerial positions. These employees are responsible for the business and administrative planning, organizing, direction, coordination, control, and approval of the activities necessary to accomplish the objectives of a given software development project. Level 1 managers are the first line of supervision. In addition to their business and administrative tasks, they may be responsible for numerous personnel functions for the employees reporting to them, such as hiring, performance evaluations, and determining pay rates. They may have technical responsibilities as well. However, this class does not include lead designers and programmers and other functional leads that do not have personnel responsibilities; these employees should be included with their functional labor class. Level 2 managers have one or more departments or groups headed by Level 1 managers reporting to them. Level 3 managers have one or more departments or groups headed by Level 2 managers reporting to them. This reporting hierarchy is similar for higher level managers. The checklist allows for the inclusion of any of these levels of management in the staff-hour measurement definition if desired for a given software development project. However, the list of management levels in the checklist is meant more as a reminder to the developer of a projects staff-hour definition to consider all levels of management and not just Level 1. Because higher levels of management may charge directly to the project as well, they should not be overlooked.

3.4.2. Technical analysts and designers

This class included the individuals responsible for developing the detailed requirements and designing a given software product. They are the system engineers or requirements analysts, performance analysts, software architects, and software engineers or software designers. The system engineers or requirements analysts develop the detailed requirements and generate the Software Requirements Specification and Interface Requirements Specification documents [DOD-STD-2167A]. They are also responsible for reliability engineering, maintainability, engineering, and human factors engineering functions. They may also develop the detailed performance requirements for the system and analyze the actual performance of the system as it is developed, if these tasks are not done by independent performance analysts. The software architects develop the high-level software system architecture based on the system architecture and the detailed requirements provided by the system engineers. Software engineers or software designers develop the high-level and detailed software design, including design

documentation. Depending on the size of the software development project, these roles may overlap, with one individual performing multiple functions in this class.

3.4.3. Programmer

The individuals in the programmer class are responsible for the implementation of an element or group of elements of a software product. That is, based on the design provided by the software engineers or software designers, the programmers write the code and usually perform the initial testing of the code. This class may overlap with the previous one: programmers may also develop the detailed and even the high-level design of the software elements they are responsible for implementing. Overlap with the Technical Analyst/Designer class may also occur, especially when a requirement is implemented in a fourth-generation language (4GL). Unless a separate group performs all testing, the programmers are also responsible for generating and developing test procedures for the unit, intermediate functional element (computer software component, or CSC [DOD-STD-2167A]) and sometimes even for the major functional element (computer software configuration item, or CSCI) and may also run the tests. Software maintenance activities such as problem analysis and the design, coding, and testing of fixes are also performed by programmers. In addition, programmers may be responsible for software reuse activities such as locating candidate objects and adapting them for reuse.

3.4.4. Test personnel

The individuals in this class perform various levels of independent testing of the software and hardware/software systems. They include high-level test engineers responsible for generating test documentation such as test plans and test procedures, as well as line testers who run the tests. In some organizations, the same personnel perform these roles The programmers are almost always responsible for the initial, detailed functional testing, and the staff-hours associated with this testing are usually combined with the rest reported for the Programmer labor class. However, on some projects, even this testing is performed by a separate test team, and the associated staff-hours could be reported in the Test Personnel labor class. This distinction needs to be documented on the Supplemental Information Form. Large software development projects may have major functional element-to-major functional element (CSCI-to-CSCI) integration test departments within the organization responsible for implementing the software. If so, the personnel performing this testing would be included in this class. The individuals who perform independent testing after release of a software product from the implementing organization are definitely included. This testing includes software independent verification and validation (IV&V), software system integration testing, hardware/software system test and evaluation, and any other testing performed prior to delivery to the customer.

3.4.5. Software quality assurance

Some software development projects include SQA functions within the software development organization. However, this class covers those individuals who perform SQA functions and who report to management outside of the software development organization. The functions of these independent SQA personnel vary depending on the project. They frequently include analyzing software measurement data and auditing the software development organization's adherence to its documented procedures. The staff-hours associated with SQA functions performed within a software development organization, such as design and code inspections or peer reviews, are frequently not available separately from the time reporting system but are collected with the specific labor class of the personnel performing them (for example, Programmer and Test personnel).

3.4.6. Software configuration management

The personnel in this class are responsible for linking together the various software elements to create the software system for testing and, ultimately, the system delivered to the customer. Involved in this is the creation and management of the software libraries that contain the software, and possibly electronic copies of the associated software documentation. On some large software development projects, the organization responsible for implementing the software may include an internal team that performs CM functions prior to release to independent test. This function is sometimes called *informal CM*, and is followed by *formal CM*, which is performed by a group external to the implementing organization after release to independent test. On other software projects, only one group—either the implementing organization or an external group—performs all CM functions. In either case, all staff-hours should be counted. However, if the staff-hours associated with software CM are mixed with those of other labor classes in the time reporting system, such as Programmer or Test Personnel, it may not be possible to obtain a separate subtotal for this labor class.

3.4.7. Program librarian

This class includes individuals responsible for maintaining the library of all documentation for a given software development project. This documentation may include hard copies of code listings and associated software documents, manuals, reports, and correspondence. On federal government software development contracts, this labor class is responsible for maintaining and possibly distributing the data requirements deliverables (DRDs) specified in the Contract Data Requirements List (CDRL). In some software development organizations, this function may be called Data Management.

3.4.8. Database administrator

The DBA is responsible for creating and maintaining the electronic databases associated with a software development project. These databases may be used for internal, non-deliverable functions, such as inspection action item and problem tracking systems, or may be created as part of the actual deliverable products of a software development project.

3.4.9. Documentation/publications

This class covers individuals who support the generation of the documentation associated with a software development project. It usually includes at least copy support functions. Depending on the level of support defined for a given project, this may or may not include technical writers and editors. However, as workstation documentation tools become more widely used, much of the work previously performed by individuals in this class is now being done by the members of other labor classes described in this section. This is often the case with tools that generate software documentation automatically from commentary in the code.

3.4.10. Training personnel

This class includes the individuals involved in the development and/or delivery of training. However, this does not include the hours worked by outside vendors who supply commercial training courses. This class includes the personnel who develop or deliver internal training courses, i.e., training for the employees involved in a software development project, as well as the individuals who develop or deliver training courses to customer personnel, if the courses are required for a specific software development contract and if the hours are charged directly to the contract. Frequently, the staff-hours associated with internal training course delivery or development are considered indirect charges. Also, many contracts specify whether or not training may be included in the development charges. Therefore, it depends on the contract as well as the **Type of Labor** definition whether or not these staff-hours are included in the measurement data.

3.4.11. Support staff

This class covers individuals performing support functions not covered in the above classes. These functions include secretarial and clerical support and software development environment support personnel, such as computer operators, internal and customer help desk personnel, and technicians responsible for installing workstations and associated software development tools. Also included are employees responsible for creating, installing, and checking out the operational software product packages for each customer or user set.

3.5. Activity

	Totals include	Totals exclude	Report totals
Activity			
Development			
Primary development activity			
Development support activities			
Concept demo/prototypes			
Tools development, acquisition, installation, & suppc			
Non-delivered software & test drivers			
Maintenance			
Repair			
Enhancements/major updates			

Figure 3-5 The Activity Attribute

A software development organization may have one or more types of activities occurring over time. *Development* generally means new software development projects and frequently covers most, if not all, of the software development life cycle—requirements analysis, design, code, development test, independent verification, and system test. *Maintenance* includes activities that occur after a new software product has been released—problem repair and functional enhancement. However, maintenance may include the same activities that are performed for new software products. Sometimes, an organization or company that did not develop the initial product may perform the maintenance of a software product. A further decomposition of the maintenance activities—such as reverse engineering and examining side effects of the code fixes— have not been included at this time. We have left blank lines on the checklist so you may include them if appropriate for your organization.

Individual totals for development staff-hours and maintenance staff-hours are useful as historical data for the planning of future new software development projects and software maintenance projects. In fact, individual totals within the maintenance activity for repair staff-hours and enhancement staff-hours give further insight into the resources needed for future similar activities.

3.6. Product-Level Functions

This attribute addresses the various functional levels of a software development or maintenance project by **CSCI-Level Function** (Major Functional Element), by **Build-Level Functions** (Customer Release), and by **System-Level Functions**. Different organizations collect the staff-hours for management, software quality assurance, configuration management, and documentation at different levels. Some may collect

these staff-hours at the system level, while others may collect them at more detailed levels, that is, per customer release or even for each major functional element. Where a project collects these staff-hours—or the staff-hours for any other functions collected differently than indicated in the definition checklist form—should be indicated on the associated Supplemental Information Form.

3.6.1. CSCI-level functions (major functional element)

	Totals include	Totals exclude	Report totals
Product-Level Functions			
CSCI-Level Functions (Major Functional Element)			
Software requirements analysis			
Design			
Preliminary design			
Detailed design			
Code & development testing			
Code & unit testing			
Function (CSC) integration and testing			
CSCI integration & testing			
IV&V			
Management			
Software quality assurance			
Configuration management			
Documentation			
Rework			
Software requirements			
Software implementation			
Re-design			
Re-coding			
Re-testing			
Documentation			

Figure 3-6 The CSCI-Level Functions Attribute

The functions of a software development project occur at different levels–the system level, the customer release level, and the major functional element level. On federal government software development projects using the methodology outlined in DOD-STD-2167A, a major functional element corresponds to a computer software configuration item (CSCI). This level is where the bulk of the actual software development and maintenance efforts occur. These efforts include the activities described below.

- **Software requirements analysis.** This function includes analysis of the system requirements and their decomposition into major functional elements, as well as documentation of the detailed software and interface requirements for each major

functional element. In addition, internal inspections (walkthroughs) of the requirements documentation may be included. This function usually concludes with a formal review of the detailed requirements with the customer.

- **Design.** Included here are analysis of the detailed requirements and their decomposition into intermediate and low-level functional elements, as well as generation of the high-level (preliminary) and low-level (detailed) design and associated test documentation. Generation of program design language (PDL) descriptions of the design, internal inspections of the design, and test documentation may also be included. This function usually concludes with formal reviews of all levels of design with the customer. Representative sample activities are as follows:

  - Create and maintain the software development files/folders.

  - Analyze the preliminary software design.

  - Derive and map out software design specifications.

  - Define and describe interface design specifications.

  - Generate input to software test planning.

  - Prepare and conduct design reviews.

- **Coding.** This function includes analysis of the design and generation of the coding language instructions that implement the design. Also included are compilation and debugging (if necessary) of the code and generation of detailed test procedures. In addition, internal inspections or peer reviews of the code and test procedures may be included. Representative sample activities are as follows:

  - Code and compile.

  - Conduct code walk-throughs.

  - Generate test and integration procedures.

- **Development testing.** Included in this function are execution of the detailed test procedures to test the low-level functional elements (units) and integration of the low-level functional elements into intermediate functional elements (CSCs). This is followed by the execution of test procedures to exercise the intermediate elements to verify that algorithms and data employed in interfacing each CSC or object are correctly specified and implemented (sometimes called computer software component integration and testing). Representative sample activities are as follows:

  - Perform unit testing.

  - Perform CSC integration and analysis.

  - Perform CSC build and lower level thread testing.

- **Major functional element integration and testing.** This function includes integration of the intermediate level functional elements into major functional elements (CSCIs) and execution of test procedures to exercise the major functional elements (sometimes called CSCI integration and testing). This is usually followed by a formal project review to determine if the major functional elements are ready for release to the independent test organization and release after approval.

- **Independent testing.** Included here are generation of independent (from the software development organization) test plans, specifications, and procedures and execution of these test procedures to ensure the code implements the requirements (sometimes called independent verification and validation or IV&V).

- **Management.** Different organizations collect their management staff-hours at the specific major functional element level or aggregate all of their management hours at the system level. If the management staff-hours are collected at the specific major functional element level, they are included in this activity. The Supplemental Information Form should indicate were the staff-hours are collected.

- **Software quality assurance.** This function includes the SQA activities associated with a specific major functional element. For DoD software projects, this consists of those tasks that ensure compliance with the government requirements for development and implementation of the contractor's software quality program [MIL-STD-881B, (Draft)]. Depending on the project, staff-hours for SQA may be collected at a higher level. Again, the Supplemental Information Form should indicate the level that the SQA hours are collected.

- **Configuration management.** Included in this function are software CM activities associated with a specific major functional element. Depending on the project, staff-hours may be collected at a higher level which should be indicated on the Supplemental Information Form.

- **Documentation.** This function includes generation and update of documentation associated with a specific major functional element. It may be desirable to decompose this category into sub-categories for each of the major formal documents at some future time. We decided for the initial version of the checklist not to do this at this time. Depending on the project, staff-hours may be collected at the CSCI-level or may be collected with the corresponding functional activity (for example, design). This should be indicated on the Supplemental Information Form.

- **Software rework.** Included here are analysis and rework of all appropriate software development products, including documentation, to (1) fix problems when errors are discovered during any software development activity or (2) make the necessary software changes whenever the customer changes existing requirements or adds new ones. Rework may require changes only to the code, in which case only re-coding and re-testing will be necessary. However, changes to the design or to the detailed software requirements may be required, in which case re-design and rework of the associated design and requirements documentation will also be necessary.

3.6.2. Build-level functions (customer release)

	Totals include	Totals exclude	Report totals
Build-Level Functions (Customer Release)			
(Software effort only)			
CSCI-to-CSCI integration & checkout			
Hardware/software integration and test			
Management			
Software quality assurance			
Configuration management			
Documentation			
IV&V			

Figure 3-7 The Build-Level Functions Attribute

For DoD contractors, a software build is an aggregate of one or more computer software configuration items that results in the satisfaction of a specific set or subset of requirements based on development of software as defined in DOD-STD-2167A [MIL-HDBK-171, (Draft)]. A build is a separately tested and delivered product. Build-level functions are those that are performed for each release of a software product to the customer. On federal government contracts using DOD-STD-2167A, a customer release corresponds to a build.   One or more major functional elements are included in a customer release.   In addition to the major functional element-level functions described above, customer release-level functions include the following activities:

- **CSCI-to-CSCI integration and checkout**. This function includes integration of the major functional elements, if more than one, into the software system and execution of test procedures to exercise the system. This activity is sometimes called CSCI-to-CSCI integration and checkout.

- **Hardware/software integration and test.** This function includes integration of software system with the operational hardware platform and execution of the test procedures to exercise the entire HW/SW system.

- **Independent testing.** Included here are the generation of independent (from the software development organization) test plans, specifications, and procedures and execution of the test procedures to ensure the code implements the requirements (sometimes called independent verification and validation or IV&V).   Also included is the generation of problem reports when errors in the code are discovered.   This function usually ends with a formal review with the customer of the readiness of the software system for release to system testing and release after   approval. Depending on the project, staff-hours may not be collected at this level.

- **Management.** This function includes software management activities associated with a specific customer release. Depending on the project, staff-hours may not be

collected at this level. The level where these staff-hours are collected should be indicated on the Supplemental Information Form.

- **Software quality assurance.** Included in this function are SQA activities associated with a specific customer release. Depending on the project, staff-hours may not be collected at this level. The level where these staff-hours are collected should be indicated on the Supplemental Information Form.

- **Configuration management.** This function includes software CM activities associated with a specific customer release. Depending on the project, staff-hours may not be collected at this level. The level where the staff hours are collected should be indicated on the Supplemental Information Form.

- **Documentation.** Included here are generation and update of documentation associated with a specific customer release. Depending on the project, staff-hours may not be collected at this level or may be collected with the corresponding functional activity (for example, independent testing). The level where these staff-hours are collected should be indicated on the Supplemental Information Form.

3.6.3. System-level functions

	Totals include	Totals exclude	Report totals
System-Level Functions			
(Software effort only)			
System requirements & design			
System requirements analysis			
System design			
Software requirements analysis			
Integration, test, & evaluation			
System integration & testing			
Testing & evaluation			
Production and deployment			
Management			
Software quality assurance			
Configuration management			
Data			
Training			
Training of development employees			
Customer training			
Support			

Figure 3-8 The System-Level Functions Attribute

The development of a system consists of one or more customer releases (builds), which in turn consist of one or more major functional elements (CSCIs). In addition to the

major functional element-level functions and the customer release-level functions, the system-level functions include the following activities:

- **System requirements and design.** This function includes analysis of the customer's requirements and generation of the system requirements documentation as well as analysis of the system requirements and generation of the system design documentation. These usually conclude with formal reviews of the system requirements and system design with the customer.

- **Software requirements analysis.** Included here are analysis of the system requirements and their decomposition into major functional elements, which are inputs to software requirements analysis at the major functional element-level.

- **Integration, test and evaluation.** This function includes integration of the software system with other software systems and execution of test procedures to test the combined software system environment (sometimes called system integration and testing). Also included are integration of the overall (multi-system) software system with the hardware system and execution of test procedures to test the overall environment (sometimes called test and evaluation). This usually ends with a formal review with the customer of the readiness of the entire operational system for customer acceptance testing.

- **Production and deployment.** Included in this function is packaging of the operational system for distribution to customer sites. This may also include installation at customer sites and user help desk (customer support) activities.

- **Management.** This function includes software management activities associated with a specific software system, a combined (multi-system) software system, or an overall hardware/software system. Depending on the project, staff-hours may not be collected at this level. The level where these staff hours are collected should be indicated on the Supplemental Information Form.

- **Software quality assurance.** Included here are SQA activities associated with a specific software system, a combined (multi-system) software system, or an overall hardware/software system. Depending on the project, staff-hours may not be collected at this level. The level where these staff-hours are collected should be indicated on the Supplemental Information Form.

- **Configuration management.** This function includes software CM activities associated with a specific software system, a combined (multi-system) software system, or an overall hardware/software system. Depending on the project, staff-hours may not be collected at this level. The level where these staff hours are collected should be indicated on the Supplemental Information Form.

- **Data.** Included in this function are copying, distribution, and library management of all required software-related documentation. This includes project documents, reports, manuals, and correspondence, as well as the generation and update of documentation associated with a specific software system, a combined (multi-

system) software system, or an overall hardware/software system. Depending on the project, documentation staff-hours may be collected at this level or may be collected with the corresponding functional activity (for example, system requirements and design). The level where documentation staff-hours are collected should be indicated on the Supplemental Information Form. The staff-hour data collected here is associated only with the preparation and review of documentation and does not include the effort required in the development of the software, such as design, requirements analysis, and coding.

- **Training.** This function includes development and/or delivery of training courses for software development or maintenance project employees or for the customer.

- **Support.** Included here are additional support activities associated with a software development or maintenance project such as secretarial support, computer facilities and operations support, internal help desk, and staff support to higher level management and program office (for example: software process definition, measurement, and improvement; contract change and budget coordination; customer action item coordination).

4. Using Supplemental Staff-Hour Information Form

The Supplemental Information Form used in conjunction with the Staff-Hour Definition Checklist provides a means to document project-specific information. This supplemental information helps us to avoid ambiguities and misunderstandings when comparing staff-hours measurement data for different projects. We capture the following information on these forms:

- Hour information
- A description of the labor class
- A description of the product-level functions

4.1. Hour Information

As discussed in Section 3.2, different organizations have different lengths of work day, work week, and labor month. The standard length for each of these should be listed on a Supplemental Information Form as shown in Figure 4-1.

If measurements are reported in terms of work days, work weeks, or labor months, it's necessary to know how many staff-hours are included in the units of measure. It is then possible to make accurate comparisons among different projects. These comparisons are useful for applying the historical data against future estimates so you can map "hours needed" to "staff required."

4.2. Labor Class

Not all organizations use the same terms for the various labor classes as we listed in the Staff-Hour Definition Checklist form. Even those that do use the same terms may include different responsibilities in a given labor class. (The usual responsibilities for each labor class are described in the Section 3.4.) When your classes differ in any way from ours, record it on the Supplemental Information Form.

- **Software management.** One area especially requiring clarification is the management level. Primarily, the first level is where differences occur among software development organizations. For some companies or organizations, the first-level manager is responsible for technical management tasks but does not have personnel management tasks other than possibly supplying employee performance evaluation inputs to another level of management. In other companies or organizations, the first-level manager is responsible for all the personnel management activities as well, tasks such as interviewing, making hiring/firing

decisions, planning employee development activities, evaluating employee performance, and determining pay rates. The Supplemental Information Form includes a section where you can list the position titles and job descriptions for each level of management included in the staff-hours measurement for a given project.

- **Technical analysts and designers vs. programmer.** These two labor classes occasionally overlap. That is, on some projects the design and the code are developed by two different teams of employees; on others, the detailed design—and even the high-level design—may be developed by the programmers, and there may not be a separate design team. On the Supplemental Information Form, you can explain which labor class develops each level of the software design.

- **Programmer vs. test personnel.** These two labor classes overlap because the programmers usually perform the initial testing on the software they've developed. However, in some cases, once the programmers achieve an error-free compilation, they hand the software over to another team to perform the unit testing and the intermediate functional element and major functional element integration testing. You can include an explanation of which labor class performs these tests in the Supplemental Information Form.

4.3. Product-Level Functions

Different software development and maintenance projects count staff-hours for management, software quality assurance, configuration management, and documentation at different product-levels. That is, depending on the project, the staff-hours associated with these functions may be counted at the major functional element (CSCI), customer release (build), or system level–or a combination of these levels. On the Supplemental Information Form, you can explain where the staff-hours are counted for these or any other functions that differ from the delineation in the Staff-Hour Definition Checklist form.

```
┌─────────────────────────────────────────────────────────────────────┐
│                  Supplemental Information Form                        │
│                                                                       │
│                   Staff-Hours Measurement                             │
│                                                                       │
│           Definition Name:          _____               │
│                                                                       │
│           Project Name:             _____               │
├─────────────────────────────────────────────────────────────────────┤
│ Hour Information                                                      │
│     Indicate the length of the following:                            │
│                                            Hours                      │
│                         Standard work day    ┌──────┐                 │
│                         Standard work week   ├──────┤                 │
│                         Standard labor month ├──────┤                 │
│                                              └──────┘                 │
├─────────────────────────────────────────────────────────────────────┤
│ Labor Class Information                                              │
│     Describe the typical responsibilities and duties for the labor    │
│     categories indicated.                                             │
│                                                                       │
│     Labor Class                        Description                    │
│     Software Management                                               │
│              Level 1                                                  │
│                                                                       │
│              Level 2                                                  │
│                                                                       │
│              Level 3                                                  │
│                                                                       │
│              Level 4                                                  │
│                                                                       │
│     Technical analysts and designers                                 │
│                                                                       │
│     Programmer                                                        │
│                                                                       │
│     Test personnel                                                    │
│                                                                       │
│     Others                                                            │
│                                                                       │
├─────────────────────────────────────────────────────────────────────┤
│ Product-Level Functions                                              │
│     Describe at what level(s) (major functional element, customer     │
│     release, and/or system) staff hours are counted for the functions │
│     indicated.                                                        │
│                                                                       │
│     Function                           Level                          │
│     Management                                                        │
│                                                                       │
│     Software quality assurance                                        │
│                                                                       │
│     Configuration management                                          │
│                                                                       │
│     Documentation                                                     │
│                                                                       │
│     Other                                                             │
└─────────────────────────────────────────────────────────────────────┘
```

Figure 4-1 Supplemental Information Form

5. Using Forms for Collecting and Reporting Staff-Hour Measurement Results

We have prepared examples of forms that can be used for reporting and transmitting staff-hour information. They are consistent with the definitions and data specifications in Chapter 3. They include information that tracks the data back to the definitions and to the entities measured. Our purpose in presenting these example forms is not to say, "This is the way." Rather, it is to suggest ideas as to the kinds of forms that can be helpful in ensuring that the details requested by measurement users are reported and communicated precisely.

In principle, you should use one reporting form for each functional element measured. Thus, you may use several (or even many) reporting forms for a given project or product. For example, if the reporting specification requests staff-hours for the system as well as for each build and CSCI within each build, generate one form for the system, one form for each build, and another form for each CSCI within each build. Figure 5-1 shows this overall scheme. This figure illustrates that all reporting forms are based upon the same staff-hour definition, and separate copies of the reporting forms are used to report the staff-hours at the CSCI, build, and system-level.

The primary purpose of the reporting form is to ensure correctly labeled data is entered into the database and to communicate what was included and excluded in the staff-hour measurement.

Figure 5-2 illustrates an example form that can be used to report staff-hours during the development of a CSCI. Appendix E contains forms that may reproduced for use for the entire system as well as for each build.
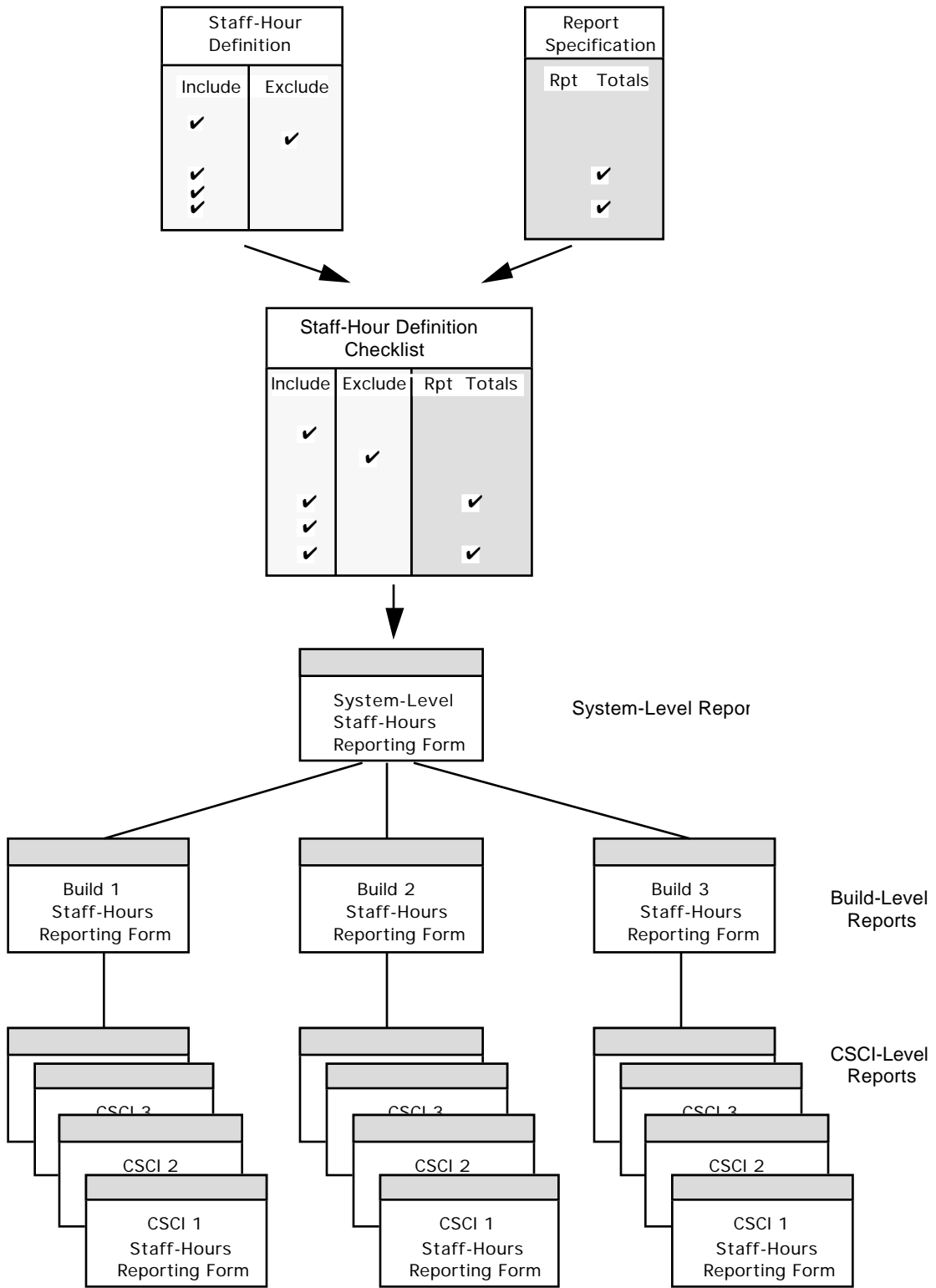
Figure 5-1   Reporting Concept

Direct Staff-Hours Report
CSCI (Major Functional Element) Development

System Name: _____   Build ID: _____
CSCI Identification: _____   Version : _____

Direct Staff-Hours

	Total	Compensated	Uncompensated
Regular Time =	[_____]	[_____]	
Overtime =	[_____]	[_____]	[_____]
Total =	[_____]	[_____]	[_____]

Work Performed Time Frame

Beginning Date: _____   Ending Date: _____

CSCI (Major Functional Elements) Level Function

	Included	Excluded	Don't Know	Staff-Hours (If requested)
Software requirements analysis	☐	☐	☐	_____
Design	☐	☐	☐	_____
Preliminary design	☐	☐	☐	_____
Detailed design	☐	☐	☐	_____
Code & development testing	☐	☐	☐	_____
Code & unit testing	☐	☐	☐	_____
Function (CSC) int. & testing	☐	☐	☐	_____
CSCI integration & testing	☐	☐	☐	_____
IV&V	☐	☐	☐	_____
Management	☐	☐	☐	_____
Software quality assurance	☐	☐	☐	_____
Configuration management	☐	☐	☐	_____
Documentation	☐	☐	☐	_____
Rework	☐	☐	☐	_____
Software requirements	☐	☐	☐	_____
Software implementation	☐	☐	☐	_____
Re-design	☐	☐	☐	_____
Re-coding	☐	☐	☐	_____
Re-testing	☐	☐	☐	_____
Documentation	☐	☐	☐	_____

Figure 5-2 Example Reporting Form for CSCI Development

---

6. Defining a Framework for Schedule Definition Measurement

The framework presented here addresses two different but related aspects of schedule measurement. One aspect concerns the **dates** of project milestones and deliverables. The second concerns measures of **progress**, specifically the rate at which work is accomplished in order to meet any given milestone or complete a deliverable. We include checklists that enable us to specify and communicate both aspects. The checklists allow us to specify the following:

- The reviews and deliverables associated with a project.

- The work units tracked to measure progress.

- Exit/completion criteria for both of the above.

- Frequency of reporting.

- Whether the dates represent planned values, actuals, or both.

We intend the checklists to be tailored by individual organizations and projects. For DoD contractors, we present checklists for dates and for progress measures that are compatible with DOD-STD-2167A. An additional set of checklists for specifying progress measures reflect those included in the Army Software Test and Evaluation Panel set (STEP) [Betz 92], in Air Force Pamphlet 800-48 ("Acquisition Management Software Management Indicators") [AFSC 90], and in the MITRE metrics ("Software Management Metrics" by Schultz) [Schultz 88]. The purpose of including these as examples is to show how the checklists can help people who are implementing any of these sets of measures to better specify the data to be collected, especially the completion criteria.

We also provide report forms that are derived from the checklists and a set of recommendations for acquisition program managers, for cost analysts, and for personnel involved in gathering measures to facilitate process improvement.

6.1. Why Include Schedule in the Core Set?

More often than not, schedule is the primary concern of project management. A timely delivery may be as important as functionality or quality in determining the ultimate value of a software product. The situation is complicated by the fact that the delivery date may have been determined by external constraints rather than by the inherent size and complexity of the software product. The result can be an extremely ambitious schedule.

Given that schedule is such a key concern, it is critical for project management to monitor adherence to intermediate milestone dates; early schedule slips are often a precursor to future problems. It is also critical to have objective and timely measures of progress that

provide an accurate indication of current status and that can be used for projecting the dates of future milestones.

In addition to acquisition and project managers, there are other users of schedule information. Cost estimators and cost model developers are one such group. Project duration is one of the key parameters used to construct new cost models or calibrate existing ones. The model developer must understand what the duration includes and excludes. If we are told that a project took three and half years, a reasonable response is to ask exactly what was included in that time period. Does it include system requirements analysis and design or just the software activities? Does it include hardware-software integration and testing or just the software integration?

Another group of users of schedule information are personnel involved in process improvement. They need to understand the basic time dependencies of the project and to identify bottlenecks in the process.

There are two different but related aspects of schedule measurement. One aspect concerns the **dates** (both planned and actual) of project milestones and deliverables. The second concerns the **rate at which work is accomplished** (again planned and actual) in order to meet any given milestone or complete a deliverable. Section 6.2 contains a checklist for project dates. Section 6.3 presents a checklist for measuring the rate of work accomplished. The checklists are vehicles for describing or specifying the schedule information to be reported. We provide a set of report forms as well.

6.2. Dates of Milestones and Deliverables

A checklist for specifying the dates to be reported for a given project is shown in Figure 6-1 and Figure 6-2. The checklist has two major parts. The first part covers the major milestones (reviews and audits) associated with the project; the second part covers the project deliverables. When the checklist has been filled out, it will convey precisely which reviews and deliverables are included and which are excluded. For projects with incremental builds, the checklist will convey which reviews and deliverables are part of each build.

Schedule Checklist
Part A: Date Information

Date: _____

Originator: _____

Project will record planned dates:	Yes _____	No _____	
If Yes, reporting frequency:	Weekly _____	Monthly _____	Other: _____
Project will record actual dates:	Yes _____	No _____	
If Yes, reporting frequency:	Weekly _____	Monthly _____	Other: _____

Number of builds _____

Milestones, Reviews, and Audits	Include	Exclude	Repeat each build	Relevant dates reported*
System-Level				
System requirements review				
System design review				
CSCI-Level				
Software specification review				
Preliminary design review				
Critical design review				
Code complete				
Unit test complete				
CSC integration and test complete				
Test readiness review				
CSCI functional & physical configuration audits				
System-Level				
Preliminary qualification test				
Formal qualification test				
Delivery & installation				
Other system-level: Delivery to prime contractor				

*Key to indicate "relevant dates reported" for reviews and audits
  - 1 - Internal review complete
  - 2 - Formal review with customer complete
  - 3 - Sign-off by customer
  - 4 - All high-priority action items closed
  - 5 - All action items closed
  - 6 - Product of activity/phase placed under configuration management
  - 7 - Inspection of product signed off by QA
  - 8 - QA sign-off
  - 9 - Management sign-off
  - 10 - _____
  - 11 - _____

Figure 6-1 Schedule Definition Checklist, Page 1

Part A: Date Information (cont.)

Deliverable Products	Include	Exclude	Repeat each build	Relevant dates reported*
System-Level				
Preliminary system specification				
System/segment specification				
System/segment design document				
Preliminary interface requirements spec.				
Interface requirements specification				
Preliminary interface design document				
Interface design document				
Software development plan				
Software test plan				
Software product specification(s)				
Software user's manual				
Software programmer's manual				
Firmware support manual				
Computer resources integrated support doc.				
Computer system operator's manual				
CSCI-Level				
Preliminary software requirements spec(s)				
Software requirements specification(s)				
Software preliminary design document(s)				
Software (detailed) design document(s)				
Software test description(s) (cases)				
Software test description(s) (procedures)				
Software test report(s)				
Source code				
Software development files				
Version description document(s)				

*Key to indicate "relevant dates reported" for deliverable products
  1 - Product under configuration control
  2 - Internal delivery
  3 - Delivery to customer
  4 - Customer comments received
  5 - Changes incorporated
  6 - Sign-off by customer
  7 - _____
  8 - _____

Figure 6-2 Schedule Definition Checklist, Page 2

Note that some of the dates apply to an entire system. The system design review (SDR) and the system design document (SDD) are examples in that there is a single SDR and SDD encompassing the system.   Other dates apply to individual CSCIs. The software specification review (SSR) and the software requirements specification (SRS) are examples.   There are separate reviews held and SSRs written for each CSCI. It is highly recommended that, when applicable, dates be reported to at least the CSCI level.   Thus, for planned dates and actuals, there will be one set of dates for the system-level reviews

and deliverables but as many dates as there are CSCIs for reviews and deliverables at that level.

Note also that milestones have been added between the critical design review (CDR) and the test readiness review (TRR). These include code complete, unit test complete, and CSC Integration & test complete. DOD-STD-2167A leaves a large gap between the formal review of the detailed design and TRR. If code complete and unit test complete are tracked for each CSCI, the date recorded should represent the day that the last CSU from a given CSCI has been coded or unit tested.

The first item on the checklist asks whether planned dates, actuals, or both are to be provided. The second item asks about the frequency of reporting. An acquisition manager is most likely to want planned and actual values provided at monthly intervals. A cost analyst, on the other hand, may want only actual values provided on a one-time basis at the conclusion of the project.

The first major section is used to indicate the scope of the project in terms of formal reviews and audits. Note that there is a column to characterize each build. Formal reviews typically encompass a series of steps or exit criteria rather than being a simple one-time event. Organizations differ in the step that marks the successful completion of the review. An example set of criteria for major project milestones might include the following:

- Hold internal review.

- Hold formal review with customer.

- Close high-priority action items.

- Close all action items.

- Obtain customer sign-off.

The right-most column contains space for filling in one or more numbers that correspond to the exit criteria listed below this section of the checklist. Fill in the numbers corresponding to all criteria which are tracked (i.e., their planned and actual dates are reported). **These criteria are intended to be tailorable to individual projects and organizations.** List those that apply to your project. In this way, anyone looking at the checklist definition for your project will know what dates are reported and exactly what the dates refer to.

Figure 6-3 shows an example of how one might fill out this section of the checklist for a project that begins with software requirements analysis and runs through the CSCI functional and physical configuration audit (FCA & PCA). The example project has a total of four different builds. Each build includes a cycle of detailed design through FCA & PCA.

Schedule Checklist
Part A: Date Information

Date: _____
Originator: _____

Page 1 of 3

Project will record planned dates:	Yes ✔	No _____	
If Yes, reporting frequency:	Weekly _____	Monthly ✔	Other: _____
Project will record actual dates:	Yes ✔	No _____	
If Yes, reporting frequency:	Weekly _____	Monthly ✔	Other: _____

Number of builds _____

Milestones, Reviews, and Audits	Include	Exclude	Repeat each build	Relevant dates reported*
System-Level				
System requirements review		✔		
System design review		✔		
CSCI-Level				
Software specification review	✔			2,3,6
Preliminary design review	✔			2,3,6
Critical design review	✔		✔	2,3,6
Code complete	✔		✔	1
Unit test complete	✔		✔	6
CSC integration and test complete	✔		✔	5
Test readiness review	✔		✔	3
CSCI functional & physical configuration audits	✔		✔	3
System-Level				
Preliminary qualification test	✔			3
Formal qualification test	✔			3
Delivery & installation		✔		
Other system-level: Delivery to prime contractor	✔			3

*Key to indicate "relevant dates reported" for reviews and audits
  1 - Internal review complete
  2 - Formal review with customer complete
  3 - Sign-off by customer
  4 - All high-priority action items closed
  5 - All action items closed
  6 - Product of activity/phase placed under configuration management
  7 - Inspection of product signed off by QA
  8 - QA sign-off
  9 - Management sign-off
  10 - _____
  11 - _____

Figure 6-3 Example of Completed Schedule Definition Checklist, Page 1

We can see from Figure 6-3 that the reviews from the software specification review (SSR) through FCA & PCA are checked in the Include column. For the sake of completeness, the earlier reviews are checked in the Exclude column because they are not part of the project. The critical design review (CDR) and all subsequent reviews are repeated with each build. Completion criteria 2, 3, and 6 are to be reported for SSR, PDR, and CDR; only a single criterion is to be reported for the remaining ones.

The second major section is used to indicate the deliverable products associated with the project. As with the section on reviews and audits, there is a column to characterize

each build. As with reviews, deliverables typically encompass a series of completion criteria, the planned and actual dates of which may or may not be formally tracked. An example set of exit criteria for project deliverables might include the following:

- Document entered under configuration management.

- Internal delivery and review.

- Delivery to customer.

- Customer comments received.

- Changes incorporated.

- Obtain customer sign-off.

The right-most column contains space for entering the number or numbers that correspond to these criteria. The criteria are listed below this part of the checklist. Figure 6-4 shows a filled out example.

Note that there is no section in the checklist for specifying project activities or "phases." The reason for this omission is two-fold:

1. There is a great deal of ambiguity associated with the beginning and end of most activities, making it difficult to define them in a precise, unambiguous way. Most activities (e.g., requirements analysis, design, code) occur to some extent throughout the project. Whereas one project may consider requirements analysis complete with the software specification review, another may consider it to be ongoing throughout development.

2. A second source of ambiguity stems from the fact that some activities start and stop and start again, making it very difficult to pin down any meaningful dates.

In contrast, project reviews and deliverables are associated with specific dates. For these reasons, the checklist on dates is limited to reviews and deliverables. Activities are reflected in the progress measures discussed in Section 6.3.

Part A: Date Information (cont.)

Deliverable Products	Include	Exclude	Repeat each build	Relevant dates reported*
System-Level				
Preliminary system specification		✔		
System/segment specification		✔		
System/segment design document		✔		
Preliminary interface requirements spec.		✔		
Interface requirements specification		✔		
Preliminary interface design document	✔			3
Interface design document	✔			1,3,5,6
Software development plan	✔			3,5,6
Software test plan	✔			3,5,6
Software product specification(s)		✔		
Software user's manual		✔		
Software programmer's manual		✔		
Firmware support manual		✔		
Computer resources integrated support doc.		✔		
Computer system operator's manual	✔			1,6
CSCI-Level				
Preliminary software requirements spec(s)	✔			3
Software requirements specification(s)	✔			1,3,5,6
Software preliminary design document(s)	✔			1,3,5,6
Software (detailed) design document(s)	✔		✔	1,3,5,6
Software test description(s) (cases)	✔		✔	1,3,5,6
Software test description(s) (procedures)	✔		✔	1,3,5,6
Software test report(s)	✔		✔	3,7
Source code	✔		✔	1,2,3,6,7
Software development files		✔		
Version description document(s)		✔		

*Key to indicate "relevant dates reported" for deliverable products
  1 - Product under configuration control
  2 - Internal delivery
  3 - Delivery to customer
  4 - Customer comments received
  5 - Changes incorporated
  6 - Sign-off by customer
  7 - IV&V sign-off
  8 - _____

Figure 6-4 Example of Completed Schedule Definition Checklist, Page 2

It is worth repeating that the purpose of the checklist is to indicate which milestones and which deliverables are associated with the project and the exit criteria that are tracked. The actual dates are given on the report form.

Figure 6-5 shows a report form for system-level milestones, reviews, and audits. The report form has been tailored to reflect the milestones and completion criteria checked in Figure 6-3. Note the column labeled **Changed**. A checkmark in this column indicates that the value shown is different from the previous report (either changed or newly added). This is intended to make it easy for the receiver of the report form to update only those values which have changed.

Schedule Reporting Form Date:		
Date Information Originator:		
System-Level Information Project: Example from Figure 6-3		
Period ending:		

Milestones, Reviews, and Audits*	**Planned**	**Changed**	**Actual**
Contract award/project start			
Preliminary qualification test			
3 - Sign-off by customer			
Formal qualification test			
3 - Sign-off by customer			
Delivery to prime contractor			
3 - Sign-off by customer			

*Only those completion criteria specified on the checklist appear below each deliverable.
Enter a check mark in the "Changed" column if Planned date has changed since last reporting.

Figure 6-5 Example of a Report Form for System-Level Milestone Dates

Figure 6-6 shows a report form for CSCI-level milestones which has been tailored to the example shown in Figure 6-3. Note that there will be a separate report form for each CSCI.

Schedule Reporting Form Date Information *CSCI-Level Information*				Date: _____ Originator: _____ Project: Example from Figure 6-3 Period ending: _____ CSCI: _____ Build: _____

Milestones, Reviews, and Audits*	**Planned**	**Changed**	**Actual**
Software specification review			
2 - Formal review with customer complete			
3 - Sign-off by customer			
6 - Products under configuration management			
Preliminary design review			
2 - Formal review with customer complete			
3 - Sign-off by customer			
6 - Products under configuration management			
Critical design review			
2 - Formal review with customer complete			
3 - Sign-off by customer			
6 - Products under configuration management			
Code complete			
1 - Internal review complete			
Unit test complete			
6 - Products under configuration management			
CSC integration and test complete			
5 - All action items closed			
Test readiness review			
3 - Sign-off by customer			
CSCI functional & physical config. audits			
3 - Sign-off by customer			

*Only those completion criteria specified on the checklist appear below each deliverable.
 Enter a check mark in the "Changed" column if Planned date has changed since last reporting.

Figure 6-6 Example of Report Form for CSCI-Level Milestone Dates

Figure 6-7 shows a report form for system-level project deliverables that is consistent with the example checklist from Figure 6-4.

Schedule Reporting Form Date Information System-Level Information			Date: _____ Originator: _____ Project: Example from Figure 6-4 Period ending: _____
Deliverable Products*	**Planned**	**Changed**	**Actual**
Preliminary interface design document			
3 - Delivery to customer			
Interface design document			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software development plan			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test plan			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Computer system operator's manual			
1 - Product under configuration control			
6 - Sign-off by customer			

*Only those completion criteria specified on the checklist appear below each deliverable.
 Enter a check mark in the "Changed" column if Planned date has changed since last reporting.

Figure 6-7 Example of Report Form for System-Level Deliverables

Figure 6-8 shows a report form for CSCI-level deliverables. Note that there will be a separate report form for each CSCI.

Schedule Reporting Form **Date Information** CSCI-Level Information			
Date: _____			
Originator: _____			
Project: Example from Figure 6-4			
Period ending: _____			
CSCI: _____			
Build: _____			

Deliverable Products*	**Planned**	**Changed**	**Actual**
Preliminary software requirements specification			
3 - Delivery to customer			
Software requirements specification			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software preliminary design document			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software (detailed) design document			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test description (cases)			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test description (procedures)			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test report			
3 - Delivery to customer			
7 - IV&V sign-of			
Source code			
1 - Product under configuration control			
2 - Internal delivery			
3 - Delivery to customer			
6 - Sign-off by customer			
7 - IV&V sign-of			

*Only those completion criteria specified on the checklist appear below each deliverable.
 Enter a check mark in the "Changed" column if Planned date has changed since last reporting.

Figure 6-8 Report Form for CSCI-Level Deliverables

It is worth emphasizing that the forms are intended to be tailored. The checklists and report forms presented here are compatible with 2167A. List those milestones and deliverables that are relevant to your project. For each milestone and deliverable, list your project's exit criteria.

Tracking milestone dates and deliverables provides a macro-level view of project schedule. As noted earlier, slips in the early reviews and deliverables are often precursors of future problems. Much greater visibility can be gained by tracking the progress of activities which culminate in reviews and deliverables. By tracking the rate at which the underlying units of work are completed, we have an objective basis for knowing where the project is at any given point in time and a basis for projecting where it will be in the future. Section 6.3 covers these types of measures.

6.3. Progress Measures

This section contains a checklist for the other aspect of schedule management, that of progress measurement. The checklist provides a means of describing or specifying or communicating the units to be tracked during each of the following activities:

- Software requirements analysis

- Software preliminary design

- Software detailed design

- Code and unit test

- Software integration and test

- Formal qualification test

In order to effectively measure progress, we must have an estimate of the total number of units to be developed and a planned rate of completion. The latter typically takes the form of an S-curve. Figure 6-9 shows an idealized picture of such.

Idealized Progress by Phase

As Revealed Through Rate of Completion of Constituent Work Un

Units:	Eng. Req's. Specified	CSCs Designed	CSUs Designed	CSUs Tested	FQT Test Cases Completed Successfully
Activity/ Phase:	S/W Requirements Analysis	High Level S/W Design	Detailed S/W Design	S/W Implementation	CSCI Integration & Test
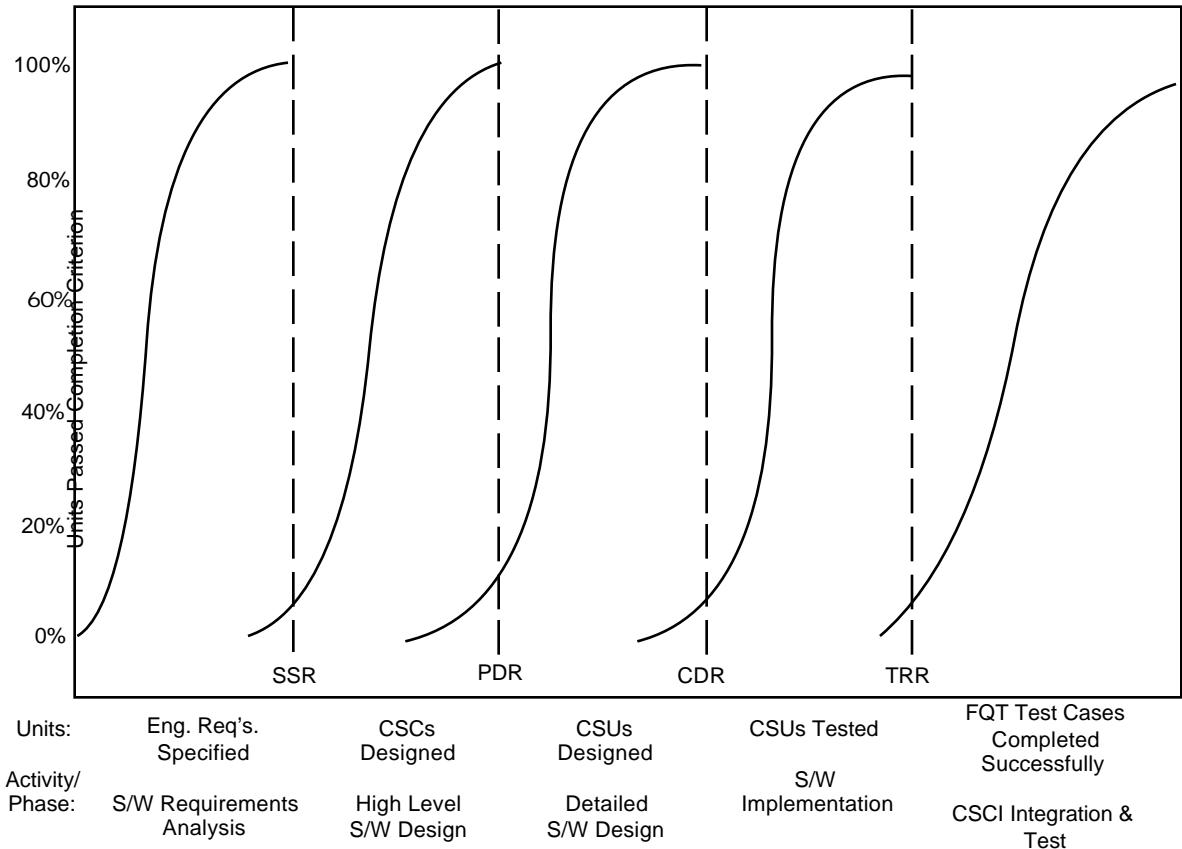
* Duration of phases not shown to scale

Figure 6-9 Idealized Rate of Unit Completion

In order for the progress measures to have any real meaning, there must be objective criteria for counting a unit as complete. The precise set of criteria for completion are addressed by the checklist. The checklist does not go into any detail about how the units are to be defined. That is beyond the scope of the current effort. As long as a consistent definition is used within a project for estimates and actuals, there is no problem. Units *cannot* be compared across projects unless they use a common definition of units. It does not make sense, for example, to compare the number of CSUs designed per labor-month for projects using different definitions for CSUs.

In terms of the users of schedule information, the progress measures will be of most interest to the software acquisition manager as well as to management on the contractor's side because it is these measures which provide the most objective view of project status and an objective basis for schedule projections. Progress measures can also point to potential bottlenecks (e.g., one CSCI lagging behind the others).

Figure 6-10 shows a checklist for specifying progress measures. As with the checklist for dates, there is space at the top to indicate whether planned and/or actual values are to be provided and the frequency of reporting. The work units listed are organized by activity. Note the list of completion criteria at the bottom. Fill in any and all that apply to the units to be tracked and/or add those that apply to your project.

Schedule Checklist, cont.
Part B: Progress/Status Information

Project will record planned progress: Yes _____ No _____
    If Yes, reporting frequency: Weekly _____ Monthly _____ Other: _____
Project will record actual progress: Yes _____ No _____
    If Yes, reporting frequency: Weekly _____ Monthly _____ Other: _____

Activities	Work Units	Tracked	Completion Criterion*
CSCI requirements analysi	Requirements documented or specified		
CSCI preliminary design	Requirements allocated to CSCs		
	CSCs designed		
CSCI detailed design	CSUs designed		
CSU coding and unit testing	Lines coded		
	Lines unit tested		
	Number CSUs coded		
	Number CSUs unit tested		
	Number lines unit tested		
CSCI integration	Number of CSUs integrated		
	Number of lines integrated		
CSCI testing	Number of tests passed		

*Key to indicate "Work Unit Completion Criterion"
    1 - None specified
    2 - Peer review held
    3 - Engineering review held
    4 - QA sign-off
    5 - Manager or supervisor sign-
    6 - Inspected
    7 - Configuration controlled
    8 - Entry in employee status report
    9 - No known deficiencies
  10 - Reviewed by customer
  11 - All relevant action items closed
  12 - _____
  13 - _____

Figure 6-10 Schedule Definition Checklist, Progress/Status Information

Appendix D contains an instantiation of the progress checklist shown in Figure 6-10 that is tailored to DOD-STD-2167A. Appendix D also contains instantiations that are specific to the descriptions provided in some well-known documents describing sets of software measures. These include the progress and completion criteria that are called for in the

Army's STEP set of measures [Betz 92], Air Force Pamphlet 800-48 [AFSC 90], and also for the MITRE set [Schultz 88].

It is worth noting that all three sets have at least one vague completion criterion (e.g., design packages closed, being actually and logically connected with all required modules, no known deficiencies). If you are implementing any of these sets, tailor your own version of the checklist to better define the completion criteria.

Figure 6-11 shows a generic report form for planned and actual work units completed.

	Progress Report Form		Date: _____

Progress Report Form
Periodic Summary by Work Unit

Date: _____
Originator: _____
Project: _____
CSCI: _____
Build: _____

Frequency of reporting: _____

Work unit kind: _____

Estimated total number of units: _____

Period	Ending date*	Planned completed units	Actual completed
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			

•
•
•

*Ending dates could be preprinted.

Figure 6-11 Report Form for Progress Information

7. Meeting the Needs of Different Users

The effort and schedule measurement framework developed in this document can be used by a software development or maintenance project at any point in the software lifecycle. At inception, it can be used to *prescribe* exactly what the staff-hour and schedule measurement data will consist of, what data will be recorded, and how the data will be reported for a given project. The Staff-Hour Definition Checklist and the Schedule Definition Checklist define and communicate unambiguously what is included in and what is excluded from the definitions of staff-hours and schedule for the project.

After a project is underway, and even after it is completed, the framework forms can be used to *describe* the content of the data that has been collected and reported so it can be compared with data from other ongoing or completed projects or used to assist in the estimation and planning for new projects.

Figure 7-1 illustrates the inter-relationship of the different forms we have discussed in this report and how they may be used.
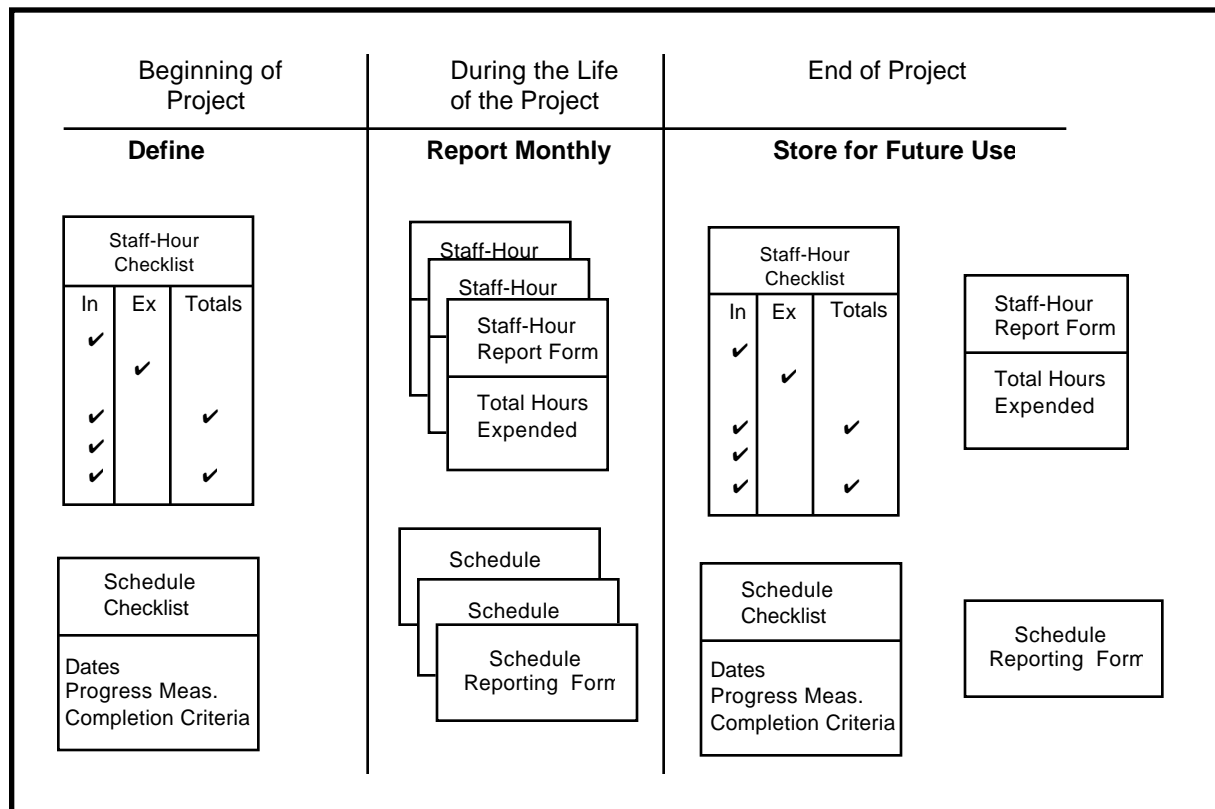
Figure 7-1 Use of Forms

7.1. To Prescribe

At the beginning of a software project, we need to determine and document what the measurement goals are and what data needs to be collected and reported; this work enables us to track the project's status in achieving those goals. To accurately assess the status from one report to the next or even between different software projects, the measurement data must be collected and reported consistently, and what the data consists of must be well-understood.

7.1.1. To specify

Whenever a software measurement program is begun, we can use the Staff-Hour Definition Checklist and the Schedule Definition Checklist to clearly and unambiguously define what the staff-hour and schedule measurement data will cover. The "Totals include" and "Totals exclude" columns in the Staff-Hour Definition Checklist are used to specify what will be included in the staff-hour measurement data and what will be excluded from it. Any additional information that you must communicate to avoid ambiguities and misunderstandings in the definitions, as well as project-specific information, should be recorded on the Supplemental Information Forms.

The schedule checklist can be used to specify the precise set of completion criteria to be tracked for milestones and deliverables and to specify the criteria that must be passed before we can count a given work unit as complete.

7.1.2. To request data elements to be reported

The Staff-Hours Definition Checklist and the Schedule Definition Checklist can also be used to specify what data elements are to be reported. The "Report totals" column in the Staff-Hour Definition Checklist is used to specify what subtotals of staff-hours are to be reported. Report requests can be made for different levels of granularity. In the case of staff-hour measurement data, you can request reports at the system level or broken down further by each build for a system. The reports may be divided even further by each major functional element (CSCI) for each build for a system.

For the schedule checklist, the "Completion Criteria Reported" can be used to request the report of a specific date related to milestones and deliverables or to request a count of work units completing a specified criteria.

7.2. To Describe

We expect that the Staff-Hour Definition Checklist and Schedule Definition Checklist will be used initially in conjunction with existing contractually required reports for documenting the total staff-hours expended and the status of the activities and milestones on a

project. The Staff-Hour Definition Checklist and the Schedule Definition Checklist communicate what attributes and values were included in specific staff-hour and schedule measurements. The checklists provide a structured approach for dealing with the details that must be resolved to reduce misunderstandings when reporting staff-hour and schedule data.

7.2.1. Ongoing projects

During the life of a project, an organization's time reporting/accounting and schedule tracking systems normally capture staff-hour and schedule data. For organizations and contractors dealing with federal government contracts, staff-hours and schedules are part of normal government monthly fiscal reports. The Staff-Hours Report Form and the Schedule Reporting Form can be used to supplement the formally established procedures for documenting the staff-hours expended and the activities and milestones achieved thus far on the project.

By attaching the associated Staff-Hours Definition Checklist and Schedule Definition Checklist to the reports, we can communicate precisely what is included in or excluded from the reported data.

7.2.2. After the fact

At the conclusion of a project, it is extremely important to collect and retain the total resources expended on the project, schedule information both planned and actuals, as well as a number of other data items. For this data to aid in estimating and planning future software projects, it must be well understood. The Staff-Hour Definition Checklist and Schedule Definition Checklist describe and define what attributes were included in and excluded from the staff-hour and schedule measurement data. We suggest retaining the following information to aid in estimating and planning future software projects at the conclusion of the project:

- Staff-Hour Definition Checklist.

- Schedule Definition Checklist.

- Any Supplemental Information Forms for both staff-hours and schedule.

- The final Staff-Hour Report Form and Schedule Reporting Form that capture the total staff-hours expended on the project, how the total staff-hours were expended, and when the milestones and deliverables were planned and actually completed for this project.

8 . Recommendations

This chapter presents our recommendations for using the Staff-Hour Definition Checklist developed in this document for both ongoing projects and for new projects in the future. It also presents our recommendation for the definition of staff-hours as well as specific recommendations for use of the schedule checklist by the acquisition program manager, the cost analyst, or the administrator of a central measurement database.

8.1. Ongoing Projects

For ongoing projects, we recommend that you use the checklist in conjunction with the current contractually required status reports to communicate what attributes and values are included in a specific staff-hour measurement. Use the Supplemental Information Form to communicate the additional project-specific information that does not lend itself to being handled via a checklist.

8.2. New Projects

For new projects, our recommendations have been broken down into a number of sub-elements to enhance communication of the concepts.

At the beginning of all new projects, we recommend the following:

1. Use the Staff-Hour Definition Checklist to create the definition of staff-hours for the project. Figure 8-1 shows our recommended definition.

2. Construct report specifications for the project via the Staff-Hour Definition Checklist. The level of detail depends on the needs, goals, and objectives of the organization and project. Care must be taken that the reports requested do not overburden the management system.

3. Request status reports on a monthly basis if not otherwise specified by the contract.

4. Specify on the Supplemental Information Form the following:

   • Length (in staff-hours) of the standard project work day, work week, and labor month.

   • Levels of management to be included and their description.

   • Labor class (Technical analysts and designers or Programmer) responsible for software design.

   • Labor class (Programmer or Test personnel) responsible for initial testing (unit testing and intermediate functional element and major functional element integration testing).

- Product level (major functional element, customer release, or system) where various staff-hours are counted.
- Any other information required to avoid ambiguities and misunderstandings.

5. Attach the Supplemental Information Form to the Staff-Hour Definition Checklist.

During the development process, we have two recommendations:

1. Use the Staff-Hour Definition Checklist in conjunction with the current contractually required status reports to communicate the resources expended on the project on a periodic basis. If not specified in the contract, we recommend that these reports be submitted on a monthly basis. The reporting forms described in Chapter 5 may be used to supplement the status reports.

2. Attach completed copies of the project Staff-Hour Definition Checklist and associated Supplemental Information Forms to all status reports as a reminder to remain consistent with the staff-hour definition.

8.3. At the End of All Projects

We recommend the following:

1. Use the Staff-Hour Definition Checklist in conjunction with the current contractually required status reports to communicate the resources expended on the entire project.

2. Attach completed copies of the project Staff-Hour Definition Checklist and the final version of the associated Supplemental Information Forms to the final status reports.

8.4. Recommended Staff-Hour Definition

Figure 8-1 presents our recommendation for the definition of staff-hours. From this definition, you can construct a number of report specifications depending on your needs.

Our rationale for our recommended staff-hour definition is as follows:

- **Type of Labor.** Only direct staff-hours should be collected. Not all organizations are able to collect the indirect staff-hours associated with a specific software project without significant manual effort. The customer is most interested in the staff-hours directly charged to the project or contract.

- **Hour Information.** All regular time should be collected as well as all recorded overtime, regardless of whether the overtime is compensated or not. Obviously, organizations that do not record overtime cannot include overtime staff-hours in software project staff-hour definitions. However, for projects that require significant

---

amounts of overtime to meet schedules, measurement results can lead to misleading conclusions if the overtime staff-hours are not included.

- **Employment Class.** Our definition includes all employment classes in our recommended definition since any of these types of employees can be responsible for significant parts of major functional elements, parts of customer releases, or even parts of entire systems.

- **Type of Pay.** Both salaried and hourly employees can be responsible for significant parts of a project. Therefore, we have included both types of pay in our recommended definition.

- **Labor Class.** Our definition of staff-hours includes all labor classes except Level 3 and higher software management. Level 3 and higher level managers frequently charge overhead funds and oversee several projects simultaneously and thus spend only small amounts of time on a given project. However, if they charge directly to a contract, their staff-hours should be included. All other labor classes contribute directly to the development of a software product; however, those which do not charge directly will not be included since they do not fall under the **Type of Labor** definition above (that is, direct staff-hours only). So some labor classes may still not be included or may be only partially included (for example, only those support staff members that charge directly).

- **Activity.** Our recommended definition includes all activities so that all staff-hours that contribute to both the development and maintenance of a software product may be tracked. Obviously, if a given project does not have all of these activities, the staff-hour definition for that project should exclude them. An example of this is a project where one contractor has the development responsibility and a different contractor has the maintenance responsibility.

- **Function.** Our definition includes all functions at all levels, except for production and deployment functions and the customer training function at the system level. We excluded these because they do not contribute to the actual development of a software product, but to follow-on activities or to other products such as non-developer training classes.

Staff-Hour Definition Checklist

Definition Name: **Total System Staff-Hours**   Date: **7/28/92**

For Development   Originator:

Page: 1 of 3

Type of Labor	Totals include	Totals exclude	Report totals
Direct	✔		
Indirect		✔	
Hour Information			
Regular time			✔
Salaried	✔		
Hourly	✔		
Overtime			✔
Salaried			
Compensated (paid)	✔		
Uncompensated (unpaid)	✔		
Hourly			
Compensated (paid)	✔		
Uncompensated (unpaid)	✔		
Employment Class			
Reporting organization			
Full time	✔		
Part time	✔		
Contract			
Temporary employees	✔		
Subcontractor working on task with reporting organization	✔		
Subcontractor working on subcontracted task	✔		
Consultants	✔		
Labor Class			
Software management			
Level 1	✔		
Level 2	✔		
Level 3		✔	
Higher		✔	
Technical analysts & designers			
System engineer	✔		
Software engineer/analyst	✔		
Programmer	✔		
Test personnel			
CSCI-to-CSCI integration	✔		
IV&V	✔		
Test & evaluation group (HW-SW)	✔		
Software quality assurance	✔		
Software configuration management	✔		
Program librarian	✔		
Database administrator	✔		
Documentation/publications	✔		
Training personnel	✔		
Support staff	✔		

Figure 8-1 Recommended Staff-Hour Definition

Definition Name: **Total System Staff-Hours** **For Development**	Page:	2 of 3	
	Totals include	**Totals exclude**	**Report totals**
Activity			
Development			
Primary development activity	✔		
Development support activities			
Concept demo/prototypes	✔		
Tools development, acquisition, installation, & suppc	✔		
Non-delivered software & test drivers	✔		
Maintenance			
Repair		✔	
Enhancements/major updates		✔	
Product-Level Functions			
CSCI-Level Functions (Major Functional Element)			✔
Software requirements analysis	✔		
Design			
Preliminary design	✔		
Detailed design	✔		
Code & development testing			
Code & unit testing	✔		
Function (CSC) integration and testing	✔		
CSCI integration & testing	✔		
IV&V	✔		
Management	✔		
Software quality assurance	✔		
Configuration management	✔		
Documentation	✔		
Rework			
Software requirements	✔		
Software implementation			
Re-design	✔		
Re-coding	✔		
Re-testing	✔		
Documentation	✔		
Build-Level Functions (Customer Release)			✔
(Software effort only)			
CSCI-to-CSCI integration & checkout	✔		
Hardware/software integration and test	✔		
Management	✔		
Software quality assurance	✔		
Configuration management	✔		
Documentation	✔		
IV&V			

Figure 8-1 Recommended Staff-Hour Definition, Page 2

For Development

Product-Level Functions continued	Totals include	Totals exclude	Report totals
System-Level Functions			✔
(Software effort only)			
System requirements & design			
System requirements analysis	✔		
System design	✔		
Software requirements analysis	✔		
Integration, test, & evaluation			
System integration & testing	✔		
Testing & evaluation	✔		
Production and deployment		✔	
Management	✔		
Software quality assurance	✔		
Configuration management	✔		
Data	✔		
Training			
Training of development employees	✔		
Customer training		✔	
Support	✔		

Figure 8-1 Recommend Staff-Hour Definition, Page 3

8.5. Schedule Recommendations for the Acquisition Program Manager

8.5.1. Dates of reviews/audits/deliverables

We recommend the following:

- Require planned and actual dates for milestones and deliverables.

- Use the checklist to specify the exact dates to be reported. A good starter set includes the date of baselining any and all products developed as part of a given activity, the date of formal review, the date of delivery of interim products to your office, and the date of formal sign-off.

- Some dates will apply to the entire project. In some cases, there will be dates for each CSCI. Track schedule information to at least the CSCI level. For critical CSCIs, you may want to track dates for individual CSCs and CSUs.

- Require that all planned and actual dates be updated monthly. Keep all plans. A great deal can be learned by looking at the volatility of plans over time and the extent to which they are based on supporting data (like the progress measures).

8.5.2. Progress measures

- Use the checklist on progress measures to specify the measures to be tracked.

- Require a plan from the contractor showing the rate at which work will be accomplished. There should be a plan for each CSCI. Require that the plan and actuals be reported monthly.

- The progress measures are meaningless without objective completion criteria. Make sure that these criteria can be audited. It is your way of being assured that progress is real.

- At a minimum, require that the following be planned for and tracked:

  - the number of CSUs completing unit test

  - the number of lines of code completing unit test

  - the number of CSUs integrated

  - the number of lines of code integrated

DOD-STD-2167A leaves a huge gap between the critical design review which precedes coding and the test readiness review which precedes testing for a complete CSCI. If there are to be problems in meeting integration and test schedules, the earlier you know about it the better. These simple measures have been found to be extremely useful.

Schultz presents an example in which counts of the number of CSUs completing unit test were plotted weekly [Shultz 88]. A simple linear extrapolation of the plot provided a remarkably accurate projection of when unit testing would be complete for all CSUs .

- Require that lines of code estimates be updated monthly. A significant increase (>10%) or a significant change in the composition of code (e.g., decreasing COTS or reuse) is likely to affect the schedule and should be accompanied by replanning.

- Keep track of problem reports by CSCI during all activities/phases. Those that are error-prone early are likely to be so throughout development. Closely monitor integration and testing progress for those CSCIs.

8.6. Schedule Recommendations for the Cost Analyst or the Administrator of a Central Measurement Database

We recommend the following:

- If possible, use the checklist before any data is reported to specify what dates you'd like to see (e.g., "For all reviews, report the date the review began and the date that the last document to be included in that review was signed off").

- Require a filled-out date checklist from anyone submitting schedule data (so that you'll at least understand and can document what the dates represent).

- For project begin and end dates, make sure that it is clear what activities are included and whether the dates are planned or actual.

- Be sure to collect all the core measures at the same level. We recommend the CSCI level. You will need to distinguish between system-level activities (e.g., system engineering at the beginning and integration and test activities at the end) and CSCI-level activities. Thus you will have one set of dates for system-level reviews and deliverables but as many dates as there are CSCIs for CSCI-level reviews and deliverables. If someone reports a single set of dates for the latter, you need to understand what this date represents (the day that review began for the first CSCI, the day review began for the last CSCI, or something else.)

8.7. Schedule Recommendations for Process Improvement Personnel

We recommend the following:

- You'll want to know as much about the process as possible but you're probably limited to whatever data is already being collected. If possible, gather information on the dates of all exit criteria associated with reviews and deliverables. This will allow you to track the detailed sequence of events associated with completing reviews and deliverables. This can be extremely useful for identifying bottlenecks.

- Gather progress measures as well.

References

[AFSC 90]   *Software Management Indicators* (AFSC Pamphlet 800-48). Andrews Air Force Base, D.C.: Headquarters Air Force Systems Command, 1990.

[Baumert 92]   Baumert, John H. *Software Measures and the Capability Maturity Model* (CMU/SEI-92-TR-25). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Betz 92]   Betz, Henry P.; & O'Neill, Patrick J. *Army Software Test and Evaluation Panel (STEP) Software Metrics Initiatives Report*. Aberdeen, Md.: U.S. AMSAA, 1992.

[Boehm 81]   Boehm, Barry W. *Software Engineering Economics*. Englewood Cliffs, N.J.: Prentice-Hall, 1981.

[DOD-STD-2167A] *Military Standard, Defense System Software Development* (DOD-STD-2167A). Washington, D.C.: United States Department of Defense, 1988.

[Grady 87]   Grady, Robert B.; & Caswell, Deborah L. *Software Metrics: Establishing a Company-Wide Program*. Englewood Cliffs, N.J.: Prentice-Hall, 1987.

[Humphrey 89]   Humphrey, Watts S. *Managing the Software Process*. Reading, Mass.: Addison-Wesley, 1989.

[IEEE 90]   *IEEE Standard Glossary of Software Engineering Terminology* (IEEE Std 610.12-1990). New York, N.Y.: The Institute of Electrical and Electronics Engineers, 1990.

[IEEE 92]   *Standard for Software Productivity Metrics [draft]* (P1045/D5.0). Washington, D.C.: The Institute of Electrical and Electronics Engineers, 1992.

[Jones 86]   Jones, C. *Programming Productivity*. New York, N.Y.: McGraw-Hill, 1986.

[MIL-STD-881B]   *Work Breakdown Structures for Defense Material Items* (MIL-STD-881B, draft). Air Force System Command, 18 February 1992.

[MIL-STD-171]   *Military Handbook Work Breakdown Structure for Software Elements* (MIL-STD-171, draft). US Army CECOM Software Engineering Directorate, 29 May 1992.

[Rozum 92]   Rozum, James A. *Software Measurement Concepts for Acquisition Program Managers* (CMU/SEI-92-TR-11). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1992.

[Schultz 88]   Schultz, Herman P. *Software Management Metrics* (ESD-TR-88-001). Bedford, Mass.: The MITRE Corporation, 1988.

Appendix A: Acronyms and Terms

A.1. Acronyms

C/SCSC	cost/schedule control systems criteria
C/SSR	cost/schedule status report
CDR	critical design review
CCDR	contractor cost data reporting
CDRL	contract data requirements list
CFSR	contract funds status report
CMU	Carnegie Mellon University
CPR	contract progress report
CSC	computer software component
CSCI	computer software configuration item
CSU	computer software unit
DID	data item description
DoD	Department of Defense
FCA	functional configuration audit
FQT	formal qualification test
HW	hardware
IEEE	Institute of Electrical and Electronics Engineers, Inc.
IV&V	independent verification and validation
OT&E	operational test and evaluation
PCA	physical configuration audit
PDR	preliminary design review
PM	project manager
QA	quality assurance
SDD	system design document
SDR	system design review
SEI	Software Engineering Institute
SRS	software requirements specification
SSR	software specification review
SWAP	Software Action Plan
S/W	software

TRR	test readiness review
WBS	work breakdown structure
WBS.SW	work breakdown structure for software
4GL	fourth-generation language

A.2. Terms Used

Attribute - A quality or characteristic of a person or thing. Attributes describe the nature of objects measured.

Computer software component (CSC) - A distinct part of a computer software configuration item (CSCI). CSCs may be further decomposed into other CSCs and computer software units (CSUs) [DOD-STD-2167A].

Computer software configuration item (CSCI) - A configuration item for software [DOD-STD-2167A].

Computer software unit (CSU) - An element specified in the design of a computer software component (CSC) that is separately testable [DOD-STD-2167A].

Contract work breakdown structure (WBS) - The DoD-approved work breakdown structure for reporting purposes and its discretionary extension to the lower levels by the contractor, in accordance with MIL-STD-881B and the contract work statement [MIL-STD-881B].

Direct staff-hour - The amount of effort directly expended in creating a specific output product [P1045/D4.0].

Granularity - The depth or level of detail at which data is collected [P1045/D4.0].

Measure - *n*. A standard or unit of measurement; the extent, dimensions, capacity, etc. of anything, especially as determined by a standard; an act or process of measuring; a result of measurement. *v.* To ascertain the quantity, mass, extent, or degree of something in terms of a standard unit or fixed amount, usually by means of an instrument or process; to compute the size of something from dimensional measurements; to estimate the extent, strength, worth, or character of something; to take measurements.

Measurement - The act or process of measuring something. A result, such as a figure expressing the extent or value that is obtained by measuring.

Staff-hour - An hour of time expended by a member of the staff [P1045/D4.0].

Appendix B: Background

B.1. Origins of the Report

In 1989, the Software Engineering Institute (SEI) began an effort to promote the increased use of objective measurement in software engineering, project management, and software acquisitions. As part of this effort, the SEI Measurement Steering Committee was formed to provide technical guidance and increase public awareness of process measurements. Based on the advice of the steering committee, two working groups were created: Software Acquisition Metrics and Software Metrics Definition. This report and the methods in it are outgrowths of work initiated by the Effort and Schedule Subgroup of the Software Metrics Definition Working Group. At the SEI Affiliates Symposium in August 1991, a preliminary draft was released for review and field testing . More than eight hundred copies of the draft were subsequently distributed. All comments received were addressed. A revised draft was completed in May 1992, and reviewed by senior industry and government executives and the SEI Measurement Steering Committee. Suggestions from these reviewers were evaluated, and the document was modified as required.

B.2. Why Staff-Hours?

Effort is frequently the largest element of cost. Some candidate units for accumulating effort data are labor-months, staff-weeks, and staff-hours.

The concept of labor-month is well understood. However using labor-months to record and report effort data presents a number of problems:

- A labor-month does not provide enough granularity when attempting to report in fractions of a month.

- The number of hours per labor-month varies widely among contractors and within the government. Each definition may have a different value. It is even possible for a contractor to have varying definitions for a labor-month across internal projects because of government requirements or because the contractor is working as a subcontractor to another prime contractor. Still, labor-months can easily be calculated from staff-hours, if preferred.

Another candidate unit for accumulating effort data is staff-weeks. The basic assumption is that a calendar week is five working days. However, the issue of holidays falling within a week or employees working weekends must be addressed when staff-week units are used.

Using staff-hours as the basic unit for recording and reporting effort data overcomes all of the above problems.

In this document, we will use the term "staff-hour" to mean an hour of time expended by a member of the staff [P1045/D4.0].

One must recognize that no single level of measurement may be applicable to all projects. For extremely small projects, or those of an exceedingly exploratory nature, staff-hours may not be meaningful enough to deserve collection. For extremely large projects, the sheer bulk of staff-hours may conceal vital trends, so other measures of effort may need to be used. Even these projects may use staff-hours as a basis for the aggregated measurement used to actually track the project. Thus, we feel that staff-hours will be meaningful for the vast majority of software projects.

B.3. Source of Staff-Hours

The basic source for obtaining and collecting staff-hour data is an organization's time reporting system. Organizations and contractors that deal with federal government contracts normally collect staff-hours as a part of the contractor's way of doing business. They are part of normal government monthly fiscal reports such as the Contract Funds Status Report (CFSR), Cost/Schedule Status Report (C/SSR), and the Contract Progress Report (CPR). Since these reports are subject to government audit, contractors have formally established procedures for collecting the data.

Personnel record their time according to established charge accounts and procedures. Unless an organization captures the information in an organization's time reporting system, the information may not be available.

The granularity of the charge accounts determines the levels or amounts of information that may be collected about a project. This will vary from project to project, within organizations, and among organizations.

The basic information that time cards typically contain for each staff-hour is the following:
- Personnel identification: This information may be in the form of the employee's name and/or a unique identification number.
- Type of hour information: Regular time or overtime.
- Type of employee: Salaried (exempt) or hourly (non-exempt).
- Date: The day that the work was performed.
- Charge account: Project-specific charge code. Organizations use a number of coding schemes such as codes that correspond to a specific function performed, objective worked on, phase of the project, organizational element performing the work, or some other company or site-specific coding scheme.

For federal government contracts, one way to code charge account numbers is to key them to elements of work breakdown structures (WBS). One may organize work breakdown structures by either products or activities. MIL-STD-881B provides guidance on establishing a Work Breakdown Structure that *strongly* favors a product-oriented approach. Within its general guidance, a contractor can extend the contract WBS (the DoD-approved work breakdown structure for reporting purposes and its discretionary extension to lower levels by the contractor) to an appropriate level that satisfies critical visibility requirements and does not overburden the projects's management system.

Government contractors may organize their personnel along program, function, natural work team, or matrix lines to facilitate effective management. When assigning specific work tasks to the project team, the organizational structure must be linked with the work breakdown structure. A contractor establishes a system of charge accounts to reflect specific work tasks and uses the time reporting system to collect work against this established system of charge accounts.

Figure B-1, extracted from the February 1992 draft of MIL-STD-881B (Figure II-14 in MIL-STD-881B), illustrates how a cost management system with job coding and the work breakdown structure can provide needed detail and visibility without extending the WBS to extremely low levels.

-Figure B-1 Cost-Account-to-Contract-WBS Relationship
(from MIL-STD-881B, draft of Feb. 1992)

Appendix C: Using Measurement Results— Illustrations and Examples

In this appendix, we illustrate a few of the ways in which we have seen that effort data has been used to help plan, manage, and improve software projects and processes. Our purpose is not to be exhaustive, but rather to highlight some interesting uses that you may find worth trying in your own organization. We also want to encourage organizations to seek other new and productive ways to put effort measures to work, and we would very much like to hear from those who succeed.

Information pertaining to staff-hour expenditure enables you to obtain insight into the development process as well as how resource usage compares to the planned values. You can use effort measures to display expended resources over time with the intent of providing current status and forecasting actual effort expended at completion.

Because effort measures can display the staff-hour expenditures per time period, managers can track trends in the effort expended. You can use effort measures to reflect actual versus planned staff-hours expended for the current and past time period. Effort measures help you address the following questions:

- Is the rate at which effort is being expended going to overrun the planned budget?
- Is enough effort being planned and applied to the project to achieve the desired schedule?

The following sections contain some simple examples using staff-hour information.

C.1. Noncumulative Effort Distribution Example

Some organizations use effort distribution graphs to track the resources expended (staff-hours) per time period. When you collect actual expended staff-hours monthly, or some other periodic time period, and compare them to the previously estimated staff-hours for that period, you can gain some insight into the development process.

The following sections will discuss effort distribution graphs per month and cumulative graphs.

C.1.1. Effort profile for total staff-hours only

Figure C-1 illustrates the use of the checklist to specify a definition of staff-hours and the collection of staff-hours for an entire project at the project or system level only. It communicates unambiguously what has been included in and excluded from the measurement of staff-hours.

Staff-Hour Definition Checklist			

Definition Name:	**Example: Total staff-hours**	Date:	**6/27/92**
	for development, report at System	Originator:	**SEI**
	level only.	Page:	1 of 3

Type of Labor	Totals include	Totals exclude	Report totals
Type of Labor			
Direct	✔		
Indirect		✔	
Hour Information			
Regular time			
Salaried	✔		
Hourly	✔		
Overtime			
Salaried			
Compensated (paid)	✔		
Uncompensated (unpaid)		✔	
Hourly			
Compensated (paid)	✔		
Uncompensated (unpaid)		✔	
Employment Class			
Reporting organization			
Full time	✔		
Part time	✔		
Contract			
Temporary employees	✔		
Subcontractor working on task with reporting organization	✔		
Subcontractor working on subcontracted task	✔		
Consultants	✔		
Labor Class			
Software management			
Level 1	✔		
Level 2	✔		
Level 3		✔	
Higher		✔	
Technical analysts & designers			
System engineer	✔		
Software engineer/analyst	✔		
Programmer	✔		
Test personnel			
CSCI-to-CSCI integration	✔		
IV&V	✔		
Test & evaluation group (HW-SW)	✔		
Software quality assurance	✔		
Software configuration management	✔		
Program librarian	✔		
Database administrator	✔		
Documentation/publications	✔		
Training personnel	✔		
Support staff	✔		

Figure C-1 Example of a Completed Staff-Hour Definition Checklist

Definition Name:	**Example: Total staff-hours**	Page:	2 of 3
	for development, report at System		
	level only.		

	Totals include	Totals exclude	Report totals
Activity			
Development			
Primary development activity	✔		
Development support activities			
Concept demo/prototypes	✔		
Tools development, acquisition, installation, & suppc	✔		
Non-delivered software & test drivers	✔		
Maintenance			
Repair		✔	
Enhancements/major updates		✔	
Product-Level Functions			
CSCI-Level Functions (Major Functional Element)			
Software requirements analysis	✔		
Design			
Preliminary design	✔		
Detailed design	✔		
Code & development testing			
Code & unit testing	✔		
Function (CSC) integration and testing	✔		
CSCI integration & testing	✔		
IV&V	✔		
Management	✔		
Software quality assurance	✔		
Configuration management	✔		
Documentation	✔		
Rework			
Software requirements	✔		
Software implementation			
Re-design	✔		
Re-coding	✔		
Re-testing	✔		
Documentation	✔		
Build-Level Functions (Customer Release)			
(Software effort only)			
CSCI-to-CSCI integration & checkout	✔		
Hardware/software integration and test	✔		
Management	✔		
Software quality assurance	✔		
Configuration management	✔		
Documentation	✔		
IV&V			

Figure C-1 Example of a Completed Staff-Hour Definition Checklist, Page 2

Definition Name: **Example: Total staff-hours for development, report at System level only.**			
Product-Level Functions continued	**Totals include**	**Totals exclude**	**Report totals**
System-Level Functions			✔
(Software effort only)			
System requirements & design			
System requirements analysis		✔	
System design		✔	
Software requirements analysis	✔		
Integration, test, & evaluation			
System integration & testing	✔		
Testing & evaluation	✔		
Production and deployment		✔	
Management	✔		
Software quality assurance	✔		
Configuration management	✔		
Data	✔		
Training			
Training of development employees	✔		
Customer training		✔	
Support	✔		

Figure C-1 Example of a Completed Staff-Hour Definition Checklist, Page 3

You can display the staff-hours expended per time period as an x-y line graph with both actual and planned curves on the same graph. The x-axis shows the calendar time period increments and the y-axis shows staff-hours expended. Showing the actual staff-hours expended per time period illustrates the spikes and drops in the effort expended on the contract. Any significant deviations between planned and actual expenditures can be used as an indicator for further investigation of possible causes.

Even if only total staff-hours expended by the project for the current time period are available, you can gain some insight by plotting the actual and planned total staff-hours on the same graph as Figure C-2 illustrates.

The plan curve typically builds up slowly through the early portion of the development effort, which is devoted to requirements definition and analysis and design, peaks around integration and test, and shows an orderly decrease through the latter part of the development, which is devoted to system testing. For maintenance projects, on the other hand, the curve tends to more flat (level-loaded).

In Figure C-2 you can see that in the early months, more effort was expended than planned. This could be an indicator that staff was allocated to this project before they could be effectively utilized or that more analysis effort was required than estimated. To obtain a clear view of the progress implications, staff-hours must be correlated with measures of size and schedule.
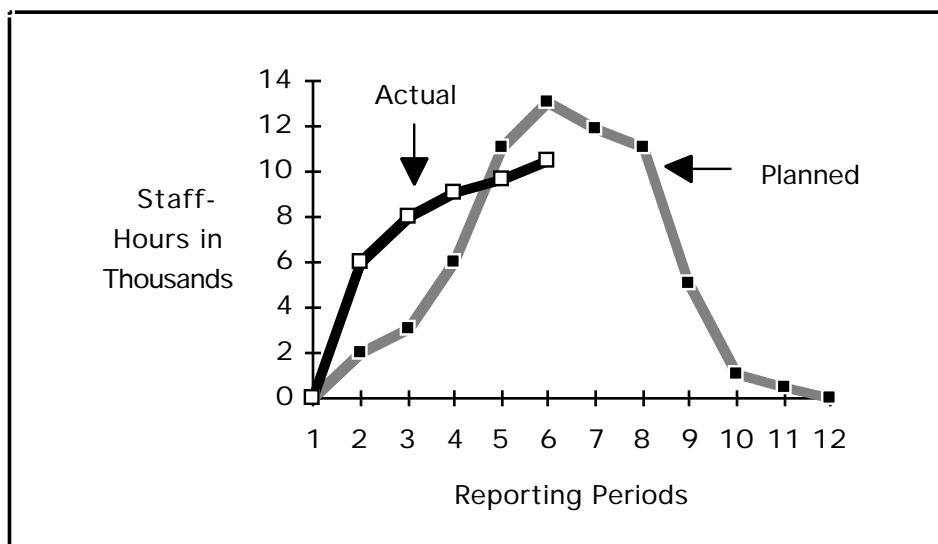
Figure C-2 Example of an Effort Profile for Total System Expenditure by Month

If you add the monthly staff-hour expenditures, you can obtain a cumulative effort profile. The cumulative effort profile can be displayed as an x-y line graph with both actual and plan curves on the same graph, as described previously, to determine if the number of actual staff-hours expended corresponds with the number of staff-hours planned for a

particular point in time. The x-axis shows calendar time period increments and the y-axis show total staff-hours expended. For development efforts, the graph usually resembles a flattened S-curve as in Figure C-3. The flattened S-curve reflects a smaller staff early in the project (a smaller slope in the curve), a larger staff during the heart of the project (a steeper slope in the curve), and a reduction in staff toward the end of the project (another smaller slope in the curve).
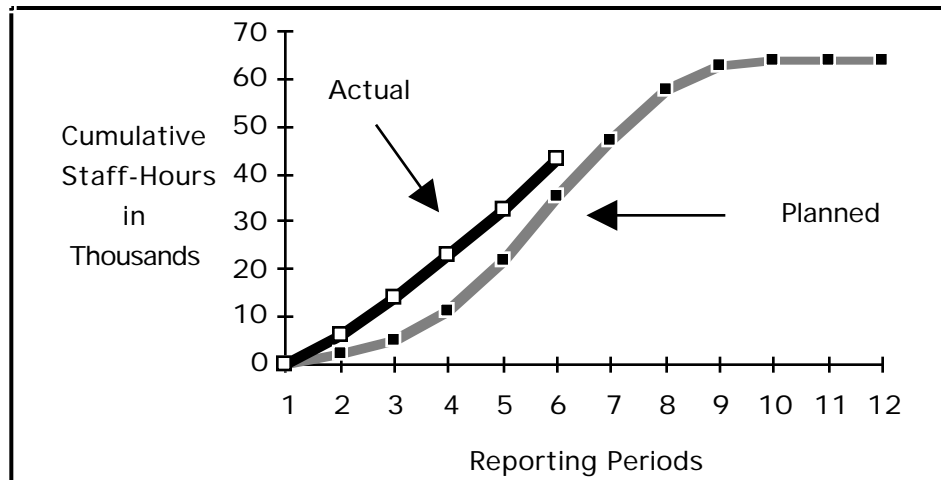
Figure C-3 Example of a Cumulative Effort Profile

The project manager can compare the total number of staff-hours planned to be expended to the total number of staff-hours actually expended for a particular point in time. In the example shown, one can readily see that more effort is being expended than planned. Analyzing the relationship of actual staff-hours being expended to the planned staff-hours can do the following:

- Provide some early indicators of potential problems. For example, if the actual expenditure is starting to deviate above or below the planned expenditure, this should be an indicator for management to ask some pointed questions as to the cause of the deviation.

- Assist the project manager in judging whether the planned amount of effort will be sufficient to complete the project.

C.1.2. Effort profile for each build and CSCI

You can obtain additional insight into the development process by requesting staff-hour utilization, not just for the entire project, but also for each build, or customer release and each CSCI (major functional element) within each build or release. In Figure C-4 we show only the specific entries of the checklist required to specify the collection of staff-hours for the entire project, for each build, and for each CSCI. The remainder of the

checklist is exactly as Figure C-1 shows. Again, use the checklist to communicate unambiguously what has been included in and excluded from the measurement of staff-hours.

Product-Level Functions	Totals include	Totals exclude	Report totals
CSCI-Level Functions (Major Functional Element)			✔
Build-Level Functions (Customer Release)			✔
System-Level Functions			✔

Figure C-4 Example of a Staff-Hour Definition Checklist for System, Builds, and CSCIs

For illustrative purposes, we will use a system development effort consisting of one build with two CSCIs. Just as discussed in the previous section, the staff-hours expended per time period can be displayed in an x-y line graph with both actual and plan curves on the same graph. We show the calendar time period increments on the x-axis and staff-hours expended on the y-axis.

Figures C-5 and C-6 illustrate the planned effort profile for each CSCI for each reporting period and the planned cumulative effort profile for the total system respectively.
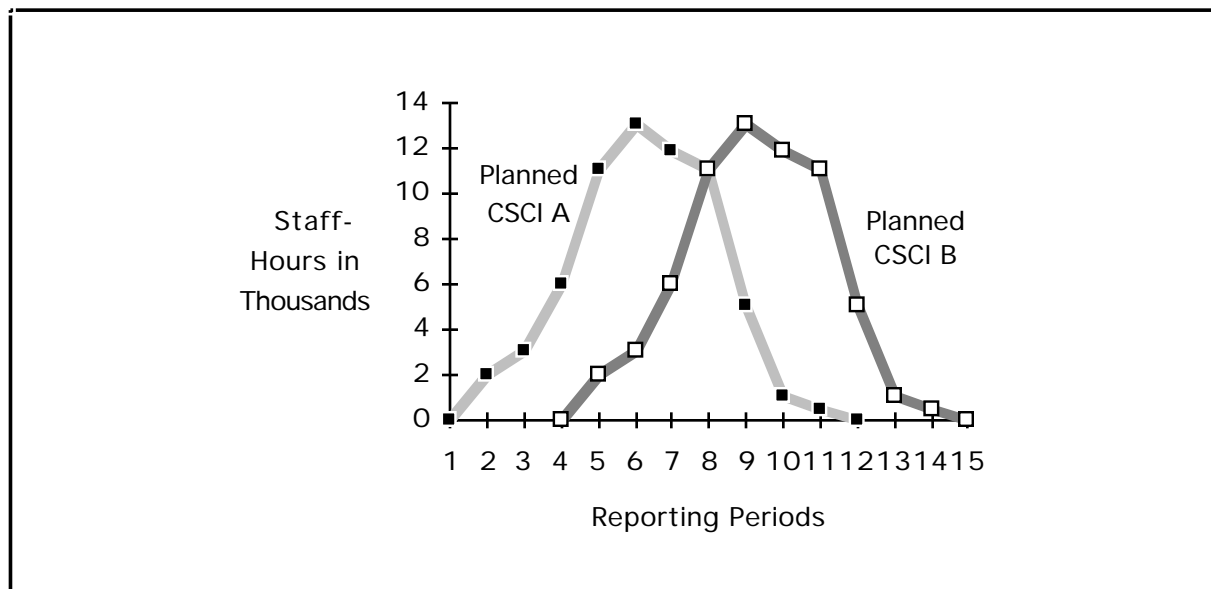
Figure C-5 Example of a Planned Effort Profile by CSCI
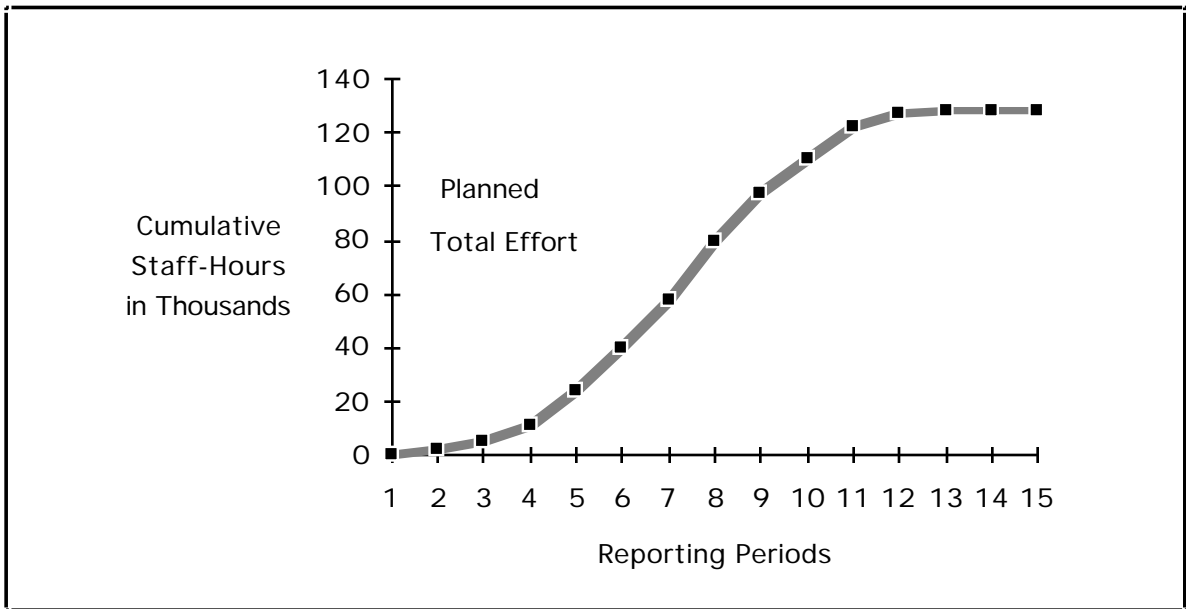
Figure C-6 Example of a Planned Cumulative Effort Profile

From the monthly status reports, you can extract and plot the total effort expended on this project as well as the effort expended on each CSCI on the same graph as the planned expenditures. Figure C-7 plots the actual cumulative effort   on the same graph as the planned cumulative effort for our example system development effort.
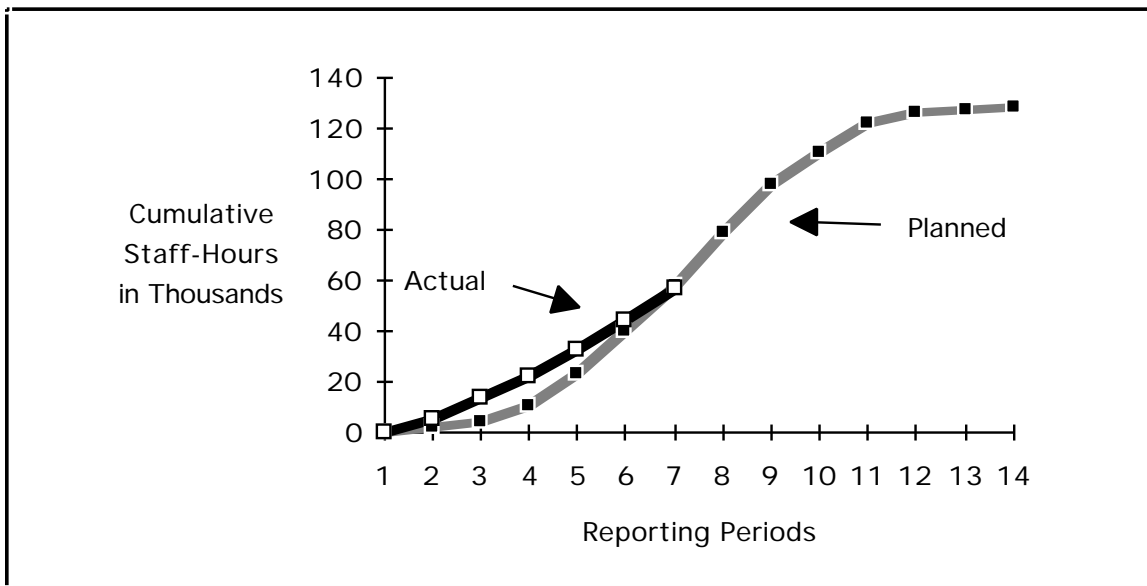
Figure C-7 Example of a Planned vs. Actual Cumulative Effort Profile

By just examining Figure C-7 you could conclude that the over-expenditure in the early months of this effort has been corrected.

Since additional information is available, you can construct actual effort profile plots for each CSCI, comparing the planned expenditure of effort with the actual expenditure.
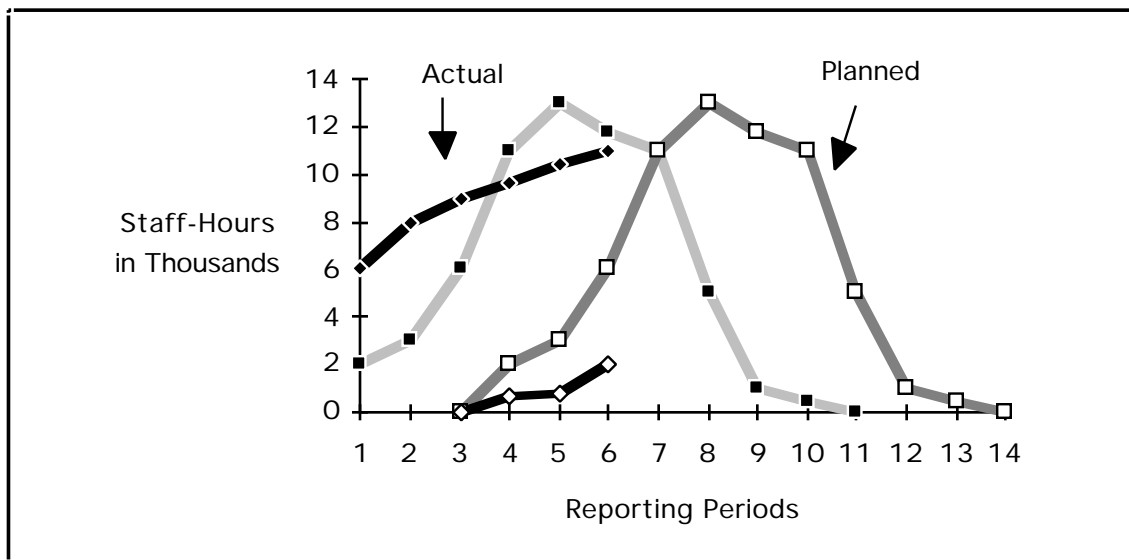
Figure C-8 Example of a Planned vs. Actual Expenditure for Each CSCI

Figure C-8 provides more details indicating that the development for both CSCIs may be in trouble–one CSCI by expending resources above the planned amount and the other CSCI by significantly underexpending. Any significant deviations between planned and actual expenditure can be used as an indicator for further investigation for possible causes.

C.2. Productivity Trend Example

A software development manager might desire a specific subset of the CSCI-level staff-hour information to be able to derive metrics for the development portion of the software life cycle. In Figure C-9 we show the additional entries to the checklist in Figure C-4 to specify the collection of staff-hours for detailed design, code, and development testing. Figure C-9 shows only the new entries. Again the checklist is used to communicate unambiguously what has been included in and what has been excluded from the measurement of staff-hours.

	Totals include	Totals exclude	Report totals
Product-Level Functions			
CSCI-Level Functions (Major Functional Element)			
Software requirements analysis	✔		
Design			
Preliminary design	✔		✔
Detailed design	✔		✔
Code & development testing			

Figure C-9 Example of a Productivity Staff-Hour Definition Checklist

You can calculate productivity rate by dividing the source line count (size measurement) for a given CSCI and build by the sum of these individual staff-hour subtotals for the same CSCI and build. Such a productivity rate can be used in estimating development costs and schedules for subsequent builds.

You can calculate a productivity rate for an entire system of multiple CSCIs as well. Here the sum of the source line counts for all CSCIs for a given build is divided by the sum of the individual staff-hour subtotals listed above for all CSCIs for the same build. When you calculate this metric for each of the builds of a system, trends can be determined and used in the planning of future software development projects.   An x-y line graph, such as Figure C-10 shows, can be drawn from this data. A negative trend in a productivity rate can alert the software development team to determine the cause of the lowered productivity and, if necessary, revise their software development process to counter the trend.
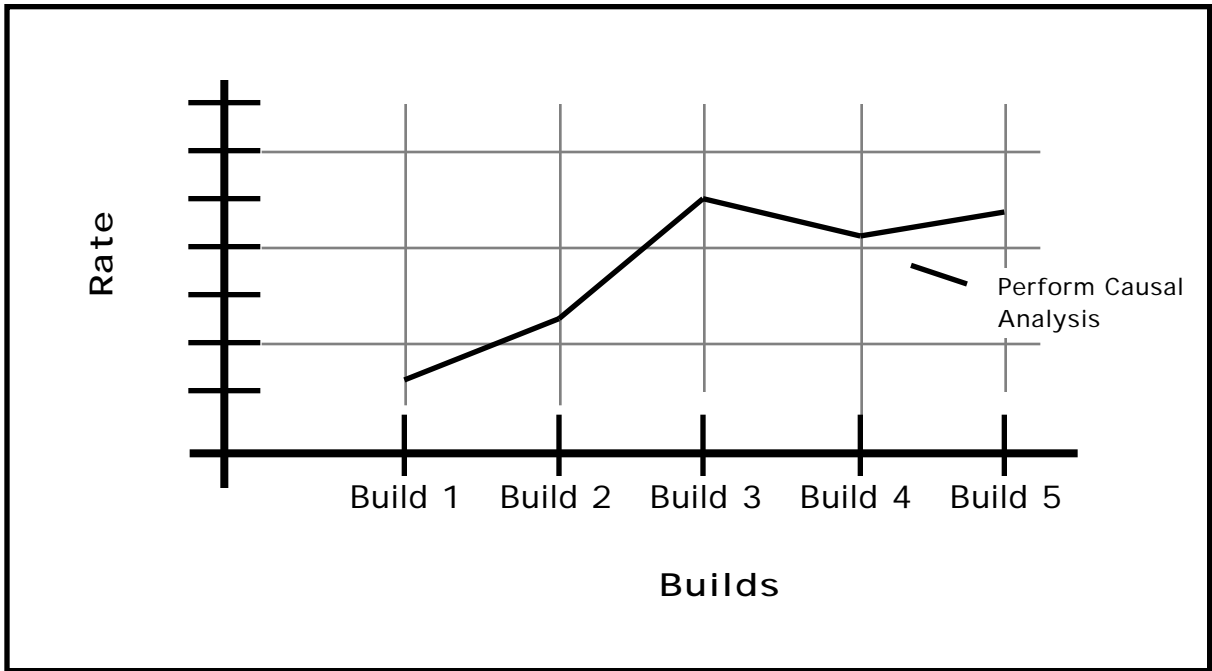
Figure C-10   Example of a Productivity Trend

Appendix D: Tailoring Schedule Checklist for Progress or Status Information

D.1. MIL-STD-2167A

Figure D-1 shows a tailoring of the generic progress checklist shown in Figure 6-10, geared specifically to the terminology and work units described in MIL-STD-2167A. The notations in the parentheses refer to the paragraph numbers of the relevant Data Item Description   (DID).

Schedule Definition Checklist (cont.)

Part B: Progress/Status Information

Project will record planned progress: Yes _____   No _____
     If Yes, reporting frequency: Weekly _____   Monthly_____   Other: _____
Project will record actual progress: Yes _____   No _____
     If Yes, reporting frequency: Weekly _____   Monthly_____   Other: _____

Deliverable Product Milestones	Work Units to Be Tracked	Work Unit Completion Criterion*
System/Segment Specification	System capabilities specified (3.2.1.X.Y.Z)	
System/Segment Design Document	System requirements allocated (4.2.X)	
	System internal interfaces specified (4.4.X)	
Interface Requirements Specification	Interface requirements specified (3.X.Y)	
Interface Design Document	Interface requirements designed (3.X.Y)	
Software Requirements Specification(s)	Engineering requirements specified (3.X[.Y])	
Software Preliminary Design Document(s)	CSCs designed (3.2.X[.Y])	
Software (Detailed) Design Document(s)	CSUs designed (4.X.Y.2)	
	CSCI data elements/files defined (5, 6)	
Software Test Description(s) (cases)	FQTs described (4.X.Y.1-5)	
Software Test Description(s) (procedures)	FQT procedures defined (4.X.Y.6)	
Source Code	CSUs code-inspected	
	CSUs unit-tested	
Software Test Report	FQT test case results described (4.X.Y.Z)	

*Key to indicate "Work Unit Completion Criterion":
     1 - None specified
     2 - Peer reviewed
     3 - Engineering review held
     4 - QA sign-off
     5 - Manager or supervisor sign-off
     6 - Inspected
     7 - Configuration controlled
     8 - Entry in employee status report
     9 - No known deficiencies
   10 - Reviewed by customer
   11 - All relevant action items closed

Figure D-1 Schedule Definition Checklist, Progress/Status Information (MIL-STD-2167A)

D.2. ARMY STEP Set of Measures

Figure D-2 shows a tailoring of the generic progress checklist shown in Figure 6-10, geared specifically to the progress measures and completion criteria called for in the Army's STEP set of measures [Betz 91].

Schedule Definition Checklist (cont.)

Part B: Progress/Status Information

Project will record planned progress: Yes __✔__   No _____
   If Yes, reporting frequency: Weekly _____   Monthly __✔__   Other: _____
Project will record actual progress: Yes __✔__   No _____
   If Yes, reporting frequency: Weekly _____   Monthly __✔__   Other: _____

Activities	Work Units Tracked	Work Unit Completion Criterion*
CSCI requirements analysi	Requirements documented or specified	
CSCI preliminary design	Requirements allocated to CSCs	
	CSCs designed	
CSCI detailed design	CSUs designed	10
CSU coding and unit testing	Lines coded	
	Lines unit tested	
	Number CSUs coded	
	Number CSUs unit tested	12
	Number lines unit tested	
CSCI integration	Number of CSUs integrated	13
	Number of lines integrated	
CSCI testing	Number of tests passed	

*Key to indicate "Work Unit Completion Criterion":
   1 - None specified
   2 - Peer reviewed
   3 - Engineering review held
   4 - QA sign-off
   5 - Manager or supervisor sign-off
   6 - Inspected
   7 - Configuration controlled
   8 - Entry in employee status report
   9 - No known deficiencies
10 - Reviewed by customer
11 - All relevant action items closed
12 - All test cases completed with no defects
13 - CSUs actually and logically connected with all required modules

Figure D-2 Schedule Definition Checklist, Progress/Status Information (STEP)

D.3. Air Force Pamphlet 800-48

Figure D-3 shows a tailoring of the generic progress checklist shown in Figure 6-10, geared specifically to the progress measures and completion criteria called for in the progress measures described in Air Force Pamphlet 800-48 [AFSC 90].

Schedule Definition Checklist (cont.)

Part B: Progress/Status Information

Project will record planned progress: Yes ____✔____   No _____
    If Yes, reporting frequency: Weekly _____   Monthly ____✔____   Other: _____
Project will record actual progress: Yes ____✔____   No _____
    If Yes, reporting frequency: Weekly _____   Monthly ____✔____   Other: _____

Activities	Work Units Tracked	Work Unit Completion Criterion*
CSCI requirements analysis	Requirements documented or specified	
CSCI preliminary design	Requirements allocated to CSCs	
	CSCs designed	
CSCI detailed design	CSUs designed	9
CSU coding and unit testing	Lines coded	
	Lines unit tested	
	Number CSUs coded	
	Number CSUs unit tested	9
	Number lines unit tested	
CSCI integration	Number of CSUs integrated	11
	Number of lines integrated	
CSCI testing	Number of tests passed	11

*Key to indicate "Work Unit Completion Criterion":
    1 - None specified
    2 - Peer reviewed
    3 - Engineering review held
    4 - QA sign-off
    5 - Manager or supervisor sign-off
    6 - Inspected
    7 - Configuration controlled
    8 - Entry in employee status report
    9 - No known deficiencies
    10 - Reviewed by customer
    11 - All relevant action items closed

Figure D-3 Schedule Definition Checklist, Progress/Status Information
(AF Pamphlet 800-48)

D.4. MITRE

Figure D-4 shows a tailoring of the generic progress checklist shown in Figure 6-10, geared specifically to the progress measures and completion criteria called for in the MITRE set [Schultz 1988].

Schedule Definition Checklist (cont.)

Part B: Progress/Status Information

Project will record planned progress: Yes ____✔____   No _____
   If Yes, reporting frequency: Weekly _____   Monthly ____✔____   Other: ____
Project will record actual progress: Yes ____✔____   No _____
   If Yes, reporting frequency: Weekly _____   Monthly ____✔____   Other: ____

Activities	Work Units Tracked	Work Unit Completion Criterion*
CSCI requirements analysi	Requirements documented or specified	1
CSCI preliminary design	Requirements allocated to CSCs	1
	CSCs designed	1
CSCI detailed design	CSUs designed	3, 13
CSU coding and unit testing	Lines coded	
	Lines unit tested	
	Number CSUs coded	
	Number CSUs unit tested	12, 7
	Number lines unit tested	
CSCI integration	Number of CSUs integrated	1
	Number of lines integrated	
CSCI testing	Number of tests passed	1

*Key to indicate "Work Unit Completion Criterion":
  1 - None specified
  2 - Peer reviewed
  3 - Engineering review held
  4 - QA sign-off
  5 - Manager or supervisor sign-off
  6 - Inspected
  7 - Configuration controlled
  8 - Entry in employee status report
  9 - No known deficiencies
10 - Reviewed by customer
11 - All relevant action items closed
12 - Passed CSU test
13 - Design packages closed

Figure D-4 Schedule Definition Checklist, Progress/Status Information (MITRE)

Appendix E: Checklists and Forms for Reproduction

The following figures are repeated in this appendix in reproducible form. We have removed figure numbers, page numbers, and document footers so that you can copy and use the pages for your own purposes.

Report forms on pages 43-45 are tailored to a specific example. These are included as an example of tailoring.

Staff-Hour Definition Checklist

Definition Name: _____

_____

_____

Date: _____

Originator: _____

Type of Labor	Totals include	Totals exclude	Report totals
Direct			
Indirect			
Hour Information			
Regular time			
Salaried			
Hourly			
Overtime			
Salaried			
Compensated (paid)			
Uncompensated (unpaid)			
Hourly			
Compensated (paid)			
Uncompensated (unpaid)			
Employment Class			
Reporting organization			
Full time			
Part time			
Contract			
Temporary employees			
Subcontractor working on task with reporting organization			
Subcontractor working on subcontracted task			
Consultants			
Labor Class			
Software management			
Level 1			
Level 2			
Level 3			
Higher			
Technical analysts & designers			
System engineer			
Software engineer/analyst			
Programmer			
Test personnel			
CSCI-to-CSCI integration			
IV&V			
Test & evaluation group (HW-SW)			
Software quality assurance			
Software configuration management			
Program librarian			
Database administrator			
Documentation/publications			
Training personnel			
Support staff			

	Totals include	Totals exclude	Report totals
Activity			
Development			
Primary development activity			
Development support activities			
Concept demo/prototypes			
Tools development, acquisition, installation, & suppc			
Non-delivered software & test drivers			
Maintenance			
Repair			
Enhancements/major updates			
Product-Level Functions			
CSCI-Level Functions (Major Functional Element)			
Software requirements analysis			
Design			
Preliminary design			
Detailed design			
Code & development testing			
Code & unit testing			
Function (CSC) integration and testing			
CSCI integration & testing			
IV&V			
Management			
Software quality assurance			
Configuration management			
Documentation			
Rework			
Software requirements			
Software implementation			
Re-design			
Re-coding			
Re-testing			
Documentation			
Build-Level Functions (Customer Release)			
(Software effort only)			
CSCI-to-CSCI integration & checkout			
Hardware/software integration and test			
Management			
Software quality assurance			
Configuration management			
Documentation			
IV&V			

_____

_____

Product-Level Functions continued	Totals include	Totals exclude	Report totals
System-Level Functions			
(Software effort only)			
System requirements & design			
System requirements analysis			
System design			
Software requirements analysis			
Integration, test, & evaluation			
System integration & testing			
Testing & evaluation			
Production and deployment			
Management			
Software quality assurance			
Configuration management			
Data			
Training			
Training of development employees			
Customer training			
Support			

Supplemental Information Form

Staff-Hours Measurement

Definition Name: _____

Project Name: _____

Hour Information

Indicate the length of the following:

Hours

Standard work day

Standard work week

Standard labor month

Labor Class Information

Describe the typical responsibilities and duties for the labor categories indicated.

Labor Class   **Description**

Software Management

      Level 1

      Level 2

      Level 3

      Level 4

Technical analysts and designers

Programmer

Test personnel

Others

Product-Level Functions

Describe at what level(s) (major functional element, customer release, and/or system) staff hours are counted for the functions indicated.

Function   **Level**

Management

Software quality assurance

Configuration management

Documentation

Other

Direct Staff-Hours Report
CSCI (Major Functional Element) Development

System Name: _____   Build ID: _____
CSCI Identification: _____   Version : _____

Direct Staff-Hours

	Total	Compensated	Uncompensated
Regular Time =	[____]	[____]	
Overtime =	[____]	[____]	[____]
Total =	[____]	[____]	[____]

Work Performed Time Frame

Beginning Date: _____   Ending Date: _____

CSCI (Major Functional Elements) Level Function

	Included	Excluded	Don't Know	Staff-Hours (If requested)
Software requirements analysis	☐	☐	☐	_____
Design	☐	☐	☐	_____
Preliminary design	☐	☐	☐	_____
Detailed design	☐	☐	☐	_____
Code & development testing	☐	☐	☐	_____
Code & unit testing	☐	☐	☐	_____
Function (CSC) int. & testing	☐	☐	☐	_____
CSCI integration & testing	☐	☐	☐	_____
IV&V	☐	☐	☐	_____
Management	☐	☐	☐	_____
Software quality assurance	☐	☐	☐	_____
Configuration management	☐	☐	☐	_____
Documentation	☐	☐	☐	_____
Rework	☐	☐	☐	_____
Software requirements	☐	☐	☐	_____
Software implementation	☐	☐	☐	_____
Re-design	☐	☐	☐	_____
Re-coding	☐	☐	☐	_____
Re-testing	☐	☐	☐	_____
Documentation	☐	☐	☐	_____

Direct Staff-Hours Report
Build (Customer Release) Development

System Name _____   Build ID: _____

Direct Staff-Hours

	Total	Compensated	Uncompensated
Regular Time =	☐	☐	
Overtime =	☐	☐	☐
Total =	☐	☐	☐

Work Performed Time Frame

Beginning Date _____   Ending Date _____

CSCIs Associated with This Build

CSCI ID	Version
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Build (Customer Release) Level Functions

	Included	Excluded	Don't Know	Staff Hours (If requested)
CSCI-to-CSCI integration & checkout	☐	☐	☐	_____
Hardware/software int. and test	☐	☐	☐	_____
Management	☐	☐	☐	_____
Software quality assurance	☐	☐	☐	_____
Configuration management	☐	☐	☐	_____
Documentation	☐	☐	☐	_____
IV&V	☐	☐	☐	_____

Direct Staff-Hours Report
System Development

System Name   _____   Build ID: _____

Direct Staff-Hours

	Total	Compensated	Uncompensated
Regular Time =			
Overtime =			
Total =			

Work Performed Time Frame

Beginning Date   _____   Ending Date   _____

Builds Associated with this System

Builds

_____

_____

_____

_____

Direct Staff Hours Report
System Development

System Identification: _____   Version: _____

System Level-Functions	Included	Excluded	Don't Know	Staff Hours (If requested)
System requirements and design	☐	☐	☐	_____
System requirement analysis	☐	☐	☐	_____
System design	☐	☐	☐	_____
Software requirements analysis	☐	☐	☐	_____
Integration, test, and evaluation	☐	☐	☐	_____
System integration & testing	☐	☐	☐	_____
Testing and evaluation	☐	☐	☐	_____
Production and deployment	☐	☐	☐	_____
Management	☐	☐	☐	_____
Software quality assurance	☐	☐	☐	_____
Data	☐	☐	☐	_____
Training	☐	☐	☐	_____
Training of development employees	☐	☐	☐	_____
Customer training	☐	☐	☐	_____
Support	☐	☐	☐	_____

Date: _____

Originator: _____

Project will record planned dates   Yes _____   No _____

   If Yes, reporting frequency   Weekly _____   Monthly_____   Other: _____

Project will record actual dates   Yes _____   No _____

   If Yes, reporting frequency   Weekly _____   Monthly_____   Other: _____

Number of builds   ____

	Include	Exclude	Repeat each build	Relevant dates reported*
System requirements review				
System design review				
Software specification review				
Preliminary design review				
Critical design review				
Code complete				
Unit test complete				
CSC integration and test complete				
Test readiness review				
CSCI functional & physical configuration audit				
Preliminary qualification test				
Formal qualification test				
Delivery & installation				
Other system-level: Delivery to prime contractor				

*Key to indicate "relevant dates reported" for reviews and audit

   1 - Internal review complete

   2 - Formal review with customer complete

   3 - Sign-off by customer

   4 - All high-priority action items closed

   5 - All action items closed

   6 - Product of activity/phase placed under configuration management

   7 - Inspection of product signed off by QA

   8 - QA sign-off

   9 - Management sign-off

  10 - _____

  11 - _____

Part A: Date Information (cont.)

Deliverable Products	Include	Exclude	Repeat each build	Relevant dates reported*
System-Level				
Preliminary system specification				
System/segment specification				
System/segment design document				
Preliminary interface requirements spec.				
Interface requirements specification				
Preliminary interface design document				
Interface design document				
Software development plan				
Software test plan				
Software product specification(s)				
Software user's manual				
Software programmer's manual				
Firmware support manual				
Computer resources integrated support doc.				
Computer system operator's manual				
CSCI-Level				
Preliminary software requirements spec(s)				
Software requirements specification(s)				
Software preliminary design document(s)				
Software (detailed) design document(s)				
Software test description(s) (cases)				
Software test description(s) (procedures)				
Software test report(s)				
Source code				
Software development files				
Version description document(s)				

*Key to indicate "relevant dates reported" for deliverable products
    1 - Product under configuration control
    2 - Internal delivery
    3 - Delivery to customer
    4 - Customer comments received
    5 - Changes incorporated
    6 - Sign-off by customer
    7 - _____
    8 - _____

Part B: Progress/Status Information

Project will record planned progress Yes _____   No _____
        If Yes, reporting frequency   Weekly _____   Monthly_____   Other: _____
Project will record actual progress   Yes _____   No _____
        If Yes, reporting frequency   Weekly _____   Monthly_____   Other: _____

Activities	Work Units	Tracked	Completion Criterion*
CSCI requirements analysis	Requirements documented or specified		
CSCI preliminary design	Requirements allocated to CSC		
	CSCs designed		
CSCI detailed design	CSUs designed		
CSU coding and unit testing	Lines coded		
	Lines unit tested		
	Number CSUs coded		
	Number CSUs unit tested		
	Number lines unit tested		
CSCI integration	Number of CSUs integrated		
	Number of lines integrated		
CSCI testing	Number of tests passed		

*Key to indicate "Work Unit Completion Criterion"
        1 - None specified
        2 - Peer review held
        3 - Engineering review held
        4 - QA sign-off
        5 - Manager or supervisor sign-off
        6 - Inspected
        7 - Configuration controlled
        8 - Entry in employee status report
        9 - No known deficiencies
        10 - Reviewed by customer
        11 - All relevant action items closed
        12 - _____
        13 - _____

Schedule Reporting Form
Date Information

System-Level Information

Date: _____

Originator: _____

Project: _____

Period ending: _____

Milestones, Reviews, and Audits*

	Planned	Changed	Actual
Contract award/project start			
Preliminary qualification test			
3 - Sign-off by customer			
Formal qualification test			
3 - Sign-off by customer			
Delivery to prime contractor			
3 - Sign-off by customer			

*Only those completion criteria specified on the checklist appear below each deliverable.
  Enter a check mark in the "Changed" column if Planned date has changed since last reporting.

Schedule Reporting Form
Date Information

System-Level Information

Date: _____

Originator: _____

Project: _____

Period ending _____

Milestones, Reviews, and Audits*

	Planned	Changed	Actual
Contract award/project star			
Preliminary qualification tes			
3 - Sign-off by custome			
Formal qualification tes			
3 - Sign-off by custome			
Delivery to prime contracto			
3 - Sign-off by custome			

*Only those completion criteria specified on the checklist appear below each delivera
 Enter a check mark in the "Changed" column if Planned date has changed since last repo

Schedule Reporting Form

Date Information

System-Level Informatio

Date: _____

Originator: _____

Project: _____

Period ending _____

Deliverable Products*

Deliverable Products	Planned	Changed	Actual
Preliminary interface design documer			
3 - Delivery to custome			
Interface design documer			
1 - Product under configuration contr			
3 - Delivery to custome			
5 - Changes incorporate			
6 - Sign-off by custome			
Software development pla			
3 - Delivery to custome			
5 - Changes incorporate			
6 - Sign-off by custome			
Software test plar			
3 - Delivery to custome			
5 - Changes incorporate			
6 - Sign-off by custome			
Computer system operator's manu:			
1 - Product under configuration contr			
6 - Sign-off by custome			

*Only those completion criteria specified on the checklist appear below each delivera
 Enter a check mark in the "Changed" column if Planned date has changed since last repor

Schedule Reporting Form
Date Information
System-Level Information

Date: _____

Originator: _____

Project: _____

Period ending: _____

Deliverable Products*

Deliverable Products*	Planned	Changed	Actual
Preliminary interface design document			
3 - Delivery to customer			
Interface design document			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software development plan			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test plan			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Computer system operator's manual			
1 - Product under configuration control			
6 - Sign-off by customer			

*Only those completion criteria specified on the checklist appear below each deliverable.
 Enter a check mark in the "Changed" column if Planned date has changed since last reporting.

Schedule Reporting Form

Date Information

CSCI-Level Information

Date: _____

Originator: _____

Project: _____

Period ending _____

CSCI: _____

Build: _____

Deliverable Products*

	Planned	Changed	Actual
Preliminary software requirements specification			
3 - Delivery to customer			
Software requirements specification			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software preliminary design document			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software (detailed) design document			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test description (cases)			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test description (procedures)			
1 - Product under configuration control			
3 - Delivery to customer			
5 - Changes incorporated			
6 - Sign-off by customer			
Software test report			
3 - Delivery to customer			
7 - IV&V sign-off			
Source code			
1 - Product under configuration control			
2 - Internal delivery			
3 - Delivery to customer			
6 - Sign-off by customer			
7 - IV&V sign-off			

*Only those completion criteria specified on the checklist appear below each deliverable

 Enter a check mark in the "Changed" column if Planned date has changed since last report

Progress Report Form
Periodic Summary by Work Uni

Date: _____
Originator: _____
Project: _____
CSCI: _____
Build: _____

Frequency of reporting _____
Work unit kind _____
Estimated total number of unit _____

Period	Ending date*	Planned completed unit	Actual completed
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
10			
11			
12			
13			
14			
15			
16			
17			
18			
19			
20			
21			
22			
23			
24			

-
-
-

*Ending dates could be preprinte