INTRODUCTION

The Software Product Development Report (DD Form 2630-Revised) is used to describe a single software development or upgrade effort. This can be a single software contract, the software component of a larger contract, or an internal ("organic") DoD effort. Only new software developments or efforts to upgrade existing software systems should use the DD2630-R, not purely software maintenance or operation efforts. For convenience here, the term contract is used to mean the authorizing vehicle or agreement that describes the software development project whether or not this is in the form of a formal contract.

This document explains the content of the DD2630-R by describing each data item. This document does not explain the exact process by which each reporting project must complete and submit the form. In general, the form was designed to be able to record initial expectations about a project as well as actual results at the end of a project. For this reason, every reporting project should submit an initial instance of the form before development begins (e.g., at the initial CARD), a second instance within 60 days after contract award, and a final instance at contract completion describing the as-delivered software product. Details about the submission process are outside the scope of this document.

It is assumed that forms will be submitted as computer files. This allows tailoring of the names and numbers of data items. It is also assumed that the data definitions included in these instructions will be tailored as needed and submitted with the data form files as a Software Metrics Data Dictionary. The sign-off area on page two includes space to identify the file name and revision for the associated Software Metrics Data Dictionary.

The form is divided into two pages. Page one has three parts (numbered 1 through 3). Page two has three additional parts (numbered 4 through 6) plus a sign-off area at the end. The questions for each part are described below.

INSTRUCTIONS FOR PAGE ONE

Page one is entitled Report Context, Project Description, and Size and has three parts that correspond to these elements.

Part 1. Report Context

The first half of Part 1 (items 1 through 4) should be completed for all three submissions of the DD2630-R but the second half (items 5 through 10) should only be completed for the two submissions after the development organization has been identified (contract award and final).

1. System/Element Name (version/release)

Enter the name used to refer to the software product being developed, including any applicable version, release, build, or other identifier. The reporting process (the specification of which is external to this document) may require a separate series of reports for each delivered release of the software.

2. Report As Of

This is the date as of which all other answers are meaningful for this submission of the form. If a subsequent report supersedes a previous report, for example to correct an error, this date would be the retroactive date of the superseded report rather than the current date.

3. Authorizing Vehicle (MOU, contract/amendment, etc.)

Cite the contract number (if applicable) and amendment number (if applicable), or memorandum of understanding, or other documentation that authorizes the development of the subject software.

4. Reporting Event

Indicate the event that drives this submission of the DD2630-R. Possible choices are, "CARD," "Contract Award," or "Final."

5. Development Organization

For report submissions after contract award, enter the name of the company or organization that is the responsible developer of the software product being developed. (For the initial submission before development begins, leave these last six items of this section blank.) Data may be aggregated across all software development subcontractors and support contractors on a single form or one form may be submitted for the work of each development organization (or some organizations may be aggregated and others may appear on separate forms). As with any other tailoring of this form, agreement on the level of aggregation must be reached between the developer and program office. Use the associated Software Metrics Data Dictionary section to explain any use of multiple forms to represent a single submission.

6. Certified CMM Level (or equivalent)

Enter the Software Engineering Institute (SEI) Capability Maturity Model (CMM) number of the level (1 through 5) at which the primary development organization has been formally certified. If no formal certification has been conducted, enter 0 (zero). If multiple forms are used to represent a single submission, enter the appropriate level for the relevant organization on each form (or zero). If a single submission is used to represent the work of multiple organizations, enter the level of the organization that will be expending the most amount of effort on the development project (not necessarily the prime contractor) and note this in the Software Metrics Data Dictionary. If the government has accepted an alternate assessment mechanism, enter the results here and explain the meaning of the assessment in the Software Metrics Data Dictionary.

7. Certification Date

If the answer to item 7 is non-zero, enter the date when the formal assessment associated with the indicated level was conducted.

8. Lead Evaluator

If the answer to item 7 is non-zero, enter the name of the person that lead the formal SEI CMM assessment and determined the maturity level indicated.

9. Affiliation

Enter the affiliation of the Lead Certifying Analyst in the previous item.

10. Precedents

List up to five analogous systems that have been developed by the same software organization or development team.

Part 2. Product Description

1. Primary Application Type

Please describe the primary application type being developed using one or more abbreviations from this list, if possible. The primary application type describes the domain of the largest part of the software product. The primary type may be the only application type listed, but any number of application types may be listed. (Space for four is provided on the form but computer file submissions of data may include any number.) If none of the examples in the following list are appropriate, please enter a phrase in the style of those in this list to describe the application type.

Embedded applications:

AV	Avionics
AUD	Audio signal processing and enhancement
CC	Command and Control
CCI	Command, Control and Information
C3I	Command, Control, Communications and Information
C4I	Command, Control, Communications, Computers and Information
DSP	Digital Signal Processing
GDE	Guidance and control
IMG	Image processing and enhancement
OFP	Operational Flight Program
SIM	Simulation
TEL	Telemetry
TRG	Target seeking
TRE	Embedded trainer software
WE	Embedded Weapon

Networked applications and other computer systems:

DS	Decision Support
IS	Information System
MIS	Management Information System
OS	Operating System
TRO	Online training or education software

2. Percent of Product

Enter the approximate percentage of the product size that is of the indicated primary application type, up to 100%.

3. Development Process

Identify the name of the development process being followed for the primary application of the system. Use common industry terms to describe it, such as waterfall, spiral, or RAD, rather than a proprietary name that is internal to the development organization. Do not indicate a software architecture method (such as object-oriented development) or a development tool (such as PowerBuilder).

4. Upgrade or New

Indicate whether the primary development is new software or an upgrade. A software system is considered new either if no existing system currently performs its function or if the development completely replaces an existing system. A software system that replaces part of an existing system (such as the replacement of a database) should be considered an upgrade. An existing software system that is being ported to a new platform or being reengineered to execute as a web or distributed application (for example) would be considered an upgrade unless it is also being completely redeveloped from scratch (new requirements, architecture, design, process, code, etc.).

5. Secondary Application Type

If there is a major secondary application type, indicate it here using the same list as shown for question 1, above.

6 - 8. Secondary Application Type Details

Describe and indicate the percentage, development method, and novelty for the secondary application type being developed, if applicable.

9 - 12. Third Application Type and Details

If the project includes a third application type, describe it and indicate the percentage, development method, and novelty for the third application type being developed.

13-16. Fourth Application Type Details

If the project includes a fourth application type, describe it and indicate the percentage, development method, and novelty for the fourth application type being developed.

If a project includes more than four application types, add new lines when the submission is in the form of a computer file. If the submission is by hard copy, submit additional sheets to describe the additional application types in terms of the same data items (type, percentage, development method, and novelty).

17. Primary Language

Enter the computer language in which most of the development will be conducted. This can be a compiled language, such as FORTRAN, Ada, or C, or it can be an interpreted language, such as Forte. Use the amount of effort spent in development to determine the primary language rather than the amount of function delivered. For example, if a system is being developed with a COTS product that supplies most of the end function and shifts the bulk of the work to creating C language interfaces to data stores, then C would be the primary language and not the COTS tool scripting language. Explain any interpretation of this item in the associated Software Metrics Data Dictionary.

18. Percent of Product Size

Enter the approximate amount of the final development effort that will be involved with producing code in the Primary Language. This may differ somewhat from the percent of the final physical product that will be written in this language since a large portion of the delivered product might use generated code or COTS products that are not directly developed.

19. Secondary Language

Enter the secondary language used in the development (if any), using the same definitions given under the Primary Language.

20. Percent of Product Size

Enter the approximate amount of the final development effort that will be involved with producing code in the Secondary Language. This may differ somewhat from the percent of the final physical product that will be written in this language since a large portion of the delivered product might use generated code or COTS products that are not directly developed.

21. List COTS/GOTS Applications

List the names of the applications or products that will participate in the final delivered product, whether they are commercial off-the-shelf (COTS) or Government off-the-shelf (GOTS) products. If a proprietary application or product will be included which is not generally available commercially, list it here and include an explanatory remark in the associated Software Metrics Data Dictionary.

Part 3. Product Size Reporting

Part 3 asks for quantitative information about the size of the software development. If this is not the final submission of the DD2630-R then estimates-at-complete should be shown in the right-hand column. If this is the final submission after software delivery then provide actual values for the entire development project.

1. Number of Requirements, not including External Interface Requirements

Indicate the number of requirements satisfied by the developed software product. In the initial reports, provide estimates of the total number of requirements to be implemented by the software being developed. In the final submission of the DD2630-R,

provide the actual number of requirements implemented by the developed software using, if possible, the same counting method as was used in the previous reports. Do not count requirements concerning external interfaces not under project control. Explain any details about requirements counting methods used in each submission in the Software Metrics Data Dictionary.

2. Number of External Interface Requirements

Indicate the number of *external* interface requirements not under project control that the developed system must satisfy. External interfaces include interfaces to computer systems, databases, files, or hardware devices with which the developed system must interact but which are defined externally to the subject system. In the initial reports, provide estimates of the total number of interface requirements to be handled by the software to be developed. If the developed system interfaces with an external system in multiple ways (such as for reading data and also for writing data) then each unique requirement for interaction should be counted as an interface requirement. In the final submission of the DD2630-R, provide the actual number of interface requirements handled by the developed software using, if possible, the same counting method as was used in the initial reports. Explain any details about requirements counting methods used in each submission in the Software Metrics Data Dictionary.

3. Code Size Measures

Use item 3 to indicate the code size measure used in items 4 through 11. A measure other than those listed may be indicated if none of those shown are applicable. The preferred size measures are total physical source lines of code or carriage returns (to be indicated below by “S”), noncommented and nonblank source lines of code (to be indicated by “Snc”), or number of logical source statements (to be indicated by “LS”). If another size measure is being used, provide an abbreviation for it and briefly explain it. For example, unadjusted function points, adjusted function points, object points, feature points, classes, algorithms, or other functional measures could be indicated. Use the Software Metrics Data Dictionary if more than a few-word explanation is necessary.

The size measure chosen should allow independent verification of the project size by examining the software products produced by the development. For this reason, one of the source code counting methods is preferred as a size measure, where “code” can refer to any hand-edited product such as lines of a computer language or lines in tables used to configure a reusable product. Many models normalize to SLOC, which is a convenient common denominator for describing product size, even if the initial planning is done using another measure, such as function points, objects, classes, screens, algorithms, etc. However, developed code size may be expressed in other terms if SLOC is a meaningless measure of the output for the majority of the programmer effort (such as when developing a web page using an iconographic publishing tool interface).

The size measure used should be in accordance with the approved Software Metrics Plan, which is developed by the Cost WIPT.

The next eight items are intended to capture the size of the system under development by partitioning (exhaustive with no overlaps) the code into eight categories. (Any tailoring of this form should maintain a partitioning categorization.) The configuration control system is assumed to be the repository for completed code. Only

the most recent version of each code unit should be counted. For each of the next eight lines, indicate the size measure used by inserting an abbreviation from item 3, including any user-provided abbreviations, in the blank provided.

4. New Code for COTS/GOTS Integration and under Configuration Control

Most software projects utilize a combination of new, reused, and generated code to accomplish the required function. Any code that was developed specifically for this project, or was reused or generated by tools but then extensively modified (more than 25% of the lines), should be considered new code.

New code can be integration and instantiation code (sometimes called the “glue” code) required for one or more COTS or GOTS products to perform the required function. This type of new code should be reported under item 4. Enter the balance of all other new code under item 5.

5. All Other New Code under Configuration Control

Enter the total size of all newly developed code (including code that was generated or reused but was modified more than 25%) except integration or instantiation code reported under item 4. Code generator inputs prepared by hand, such as tables or scripts, should be counted as new code. The generated code should be counted under items 6 or 7, as appropriate.

6. Modified Generated Code under Configuration Control

Source code that was generated by tools, and was then reused with minor modifications (less than 25% modified) by this project is to be reported under item 6.

7. Unmodified Generated Code under Configuration Control

Source code that was generated by tools, and not subsequently modified, is to be reported under item 7.

8. Modified Internally Reused Code under Configuration Control

Source code that is obtained from within the organization, and subsequently modified (even slightly), should be reported under item 8. Any units of code that are more than 25% modified should not be reported here but should be reported as new code.

9. Unmodified Internally Reused Code under Configuration Control

Source code that is obtained from within the organization, and not subsequently modified (even slightly), should be reported under item 9.

10. Modified External Reused Code under Configuration Control

Source code that is obtained from outside the organization (i.e., the development team has no access to the developers of the reused code) and is subsequently modified should be reported under item 10. Any units of code that are more than 25% modified should not be reported here but should be reported as new code under items 4 or 5.

11. Unmodified External Reused Code under Configuration Control

Source code that is obtained from outside the organization, and not subsequently modified (even slightly), should be reported under item 11. Any units of code that are more than 25% modified should not be reported here but should be reported as new code under items 4 or 5.

INSTRUCTIONS FOR PAGE TWO

The page two of the DD Form 2630-R is entitled Project Resources, Schedule, Staffing, and Quality, and is a continuation of the three parts of the form included on page one. It has three parts as described below.

Part 4. Detailed Schedule Reporting

Project development is typically broken down into phases or activities. This form can be tailored to include the names of the phases or activities that are appropriate for the subject development. Items 1 through 6 under Part 4 are taken from the activity definitions used in ISO12207 and are intended to be generic to any software development (though they may not be strictly associated with development phases by the same names). These activities may be performed simultaneously, sequentially, or both.

1 - 6. Software Development Activities

The first six rows show the names of the default activities into which software development activity can be partitioned. For planning purposes, most software schedules assume the listed activities will peak during certain intervals and will drop to a minimum during other intervals. To the extent that the activities listed are associated with periods of peak activity, provide estimates of (or actual values for, in the case of the final delivery report) the duration of that interval in the Start Month and End Month cells (beginning with month 1 at contract award). If activities continue at a constant levels of effort and overlap each other, the end date of one activity could occur much later than the start date of a subsequent activity in this list. In general, however, the listed activities are likely to peak during different intervals of time, and it is those peak intervals that are to be defined by the indicated start and end month values. The two initial report submissions should show estimates and the final submission should show actual start and end month values.

Report in the third cell of each row the total number of direct labor hours expected to be expended (or actually expended in the case of the final delivery report) in this activity. Hours expended on an activity should not be limited to the indicated month numbers that define the peak period of the activity but should include all hours spent on each activity.

7. Other Direct Software Engineering Development Effort

Item 7 is intended to cover all other directly charged activities not specified in the first six items, above. In the text space provided, summarize the kinds of activities included, such as project management, IV&V, configuration management, quality control, problem resolution, library management, process improvement, measurement, training, documentation, data conversion, or supporting a customer-run acceptance test. Also include software delivery, installation, deployment and/or implementation, to the

extent these activities are included in the development contract. If allocated direct charges are incurred on a project, they should be included in this item. Only the total estimated or actual direct hours are to be reported for Item 7. Do not include indirect effort, such as contracts, finance, or accounting effort that is not directly charged or allocated to the subject software project. Use the Software Metrics Data Dictionary to explain (non-quantitatively) if indirect software-related effort contributes to the project (e.g., training, process improvement, methodology research).

Part 5. Staffing Profile

1. Peak number of software development personnel

Provide the estimated or actual peak FTE (full-time equivalent) count of persons providing direct labor to any software development aspect of the project during any one month. This count should cover efforts directly contributing to any of the tasks included in Part 4, items 1 through 6 and should be based on 40-hour work weeks and 52-week (2080-hour) work years. (Overhead charges account for holiday, vacation, and other leave external to the computation of FTE.) If a different method is used to compute FTE, indicate this in the Software Metrics Data Dictionary.

2. Month number of midpoint of peak

For the initial reports at the time of the CARD and contract award, enter the month number, starting with month 1 at the beginning of the project, of the expected midpoint of the peak staffing indicated in the previous item. For the final report, enter the actual month number of the midpoint of the peak staffing period.

3. Personnel experience

Partition the total number of software development personnel who will contribute to the project into three experience levels: highly experienced in project domain (three or more years of experience), nominally experienced in project domain (one to three years of experience), and entry level (zero to one year of experience). Normalize by staff-years (such that, for example, a single highly experienced person who works on the project for two years constitutes the same percentage of the total as two entry level people who each contribute a year of effort). Enter the percentage for each category in the appropriate spaces. Classify by the experience level of a person as he or she joins the project so that experience on this project is not counted towards the ratings.

Part 6. Product Quality Reporting

1. Required Mean Time to Defect (MTTD) at delivery

Part 6 begins by asking for the required quality of the delivered product in terms of its mean time to defect (MTTD). A value is only to be provided at the time of the CARD submission of the DD Form 2630-R, and only if this is a meaningful method of communicating the quality of the expected system. If applicable, the answer should be given in hours, counting only the time the software is expected to be operational. In other

words, if the software is expected to be operational for twelve hours a day, five days a week, then the expectation that the software will run without encountering a defect for a week would translate to a 60-hour MTTD (twelve times five). On the other hand, if the software is expected to be operational for 24 hours a day, seven days a week, a week of defect-free operation would translate into a 168-hour MTTD. (Do not multiply by the number of possible simultaneous executions of the software. For example, in the previous case, if five processors were executing the software continuously for a week, the answer would remain 168 hours.) The indicated MTTD value describes the expected behavior of the software product immediately after delivery, and should be assumed to be the average value over the first month, unless a longer period is indicated in the Software Metrics Data Dictionary. The estimate should allow for encountering any defect, at any severity level, not already known to be present (i.e., ignore defects that have been accepted as part of the delivered software but do count as a defect a problem that was known but was thought to have been fixed).

If a quantitative MTTD is not appropriate for the system under development, provide an alternate method of comparing the required reliability of this system with the nominal reliability for systems of this type. For example, if the system is an operational flight program (see Part 2, item 1), higher than nominal reliability might be expected if the OFP is for a stealth aircraft that cannot use radar or other means to serve as a backup to the pilot. On the other hand, if the OFP were to control a pilotless vehicle, such as a surveillance or drone aircraft, the required reliability might be lower than nominal among other OFP systems. A tailoring of this item could allow the response to be in terms relative to other similar systems, for example a scale such as “much higher,” “somewhat higher,” “nominal,” “lower,” or “much lower” might be appropriate. As with any tailoring, the explanation of the data must be included in the Software Metrics Data Dictionary.

2 - 7. Cumulative Critical, Serious and Total Defects Discovered and Resolved

Part 6 then asks for actual counts of defects discovered and resolved throughout the software development up to the point of software delivery. These items are only to be completed at the Final Delivery data submission. Critical defects are classified as priority 1 (the highest priority) and affect safety or prevent meeting a critical mission requirement. Serious defects are classified as priority 2 and adversely affect mission accomplishment and have no known workaround. The count of total defects includes critical and serious defects plus all other defect categories, including minor and cosmetic defects. (An example of five defect categories can be found in the superseded MIL-STD-498. Developers should use existing definitions as long as the reported categories of high priority defects substantially correspond to the definitions suggested here. Developers should also use existing procedures for distinguishing defects from standard development, such as problems found after an inspection, after a configuration control baseline, or after advancement to the next state of a development process.)

Items 2 through 7 include two columns for reporting the cumulative critical defect counts at the end of software qualification test and at the end of the complete operational test and evaluation period (just prior to delivery). If other names are used to define two points at the end of the project for which defect totals can be provided, the form should be tailored to use those terms and the Software Metrics Data Dictionary should explain

their meaning. Item 2 asks for the counts of critical defects at each of these two points in the project. Item 3 the number serious defects (see the foregoing discussion for definitions of defect severities). Item 4 asks for the number of all defects in all categories, including minor and cosmetic defects (but not including problem reports that are actually suggestions for additional features or functionality).

Items 5 through 7 are analogous to lines 2 through 4 but are used to report the number of resolved defects in each category. The three columns are for reporting the actual numbers of resolved defects in each severity after SQT and after OT&E, as before. (The number resolved in each case must be lower than or equal to the discovered number reported under items 2 through 4. Also, a critical defect can be downgraded to serious or lower if the requirement that exhibits it is downgraded from critical to nominal. In this case, the critical defect would become resolved and a new defect would be entered in a lesser category.) If a project chooses to report unresolved defects instead of resolved defects (the complementary number) for items 5 through 7, explain this in the Software Metrics Data Dictionary.

Filename and Revision Date of Applicable Software Metrics Data Dictionary

The definitions of any tailored item or any other clarifying definitions of metrics reported on a submitted DD Form 2630-R should be contained within a Software Metrics Data Definition document. Submitters are encouraged to submit both the DD Form 2630-R and the Software Metrics Data Definition as electronic files. The name of the file containing the data definitions should be indicated here.

2 - 4. Cumulative Critical, Serious and Total Defects Discovered

The form concludes with a sign-off line for the name, phone, and e-mail of the contact person to handle any inquiries about the data submitted, plus the date of completion (as distinct from the date in part 1).