### 

### 

### DRONEswarm™

### Software Development Plan (SDP)

### Final Draft

### ACME Software Company

### Brandon Lavinsky

### 6 December 2017

**Purpose**. The purpose of the DRONEswarm™ Software Development plan is to serve as a guide in the development of the DRONEswarm™.

### References

(CMMI) Software Engineering Institute, *Capability Maturity Model Integration, Version 1.3, CMMI for Development*, Report No. CMU/SEI-2010-TR-033, November 2010, Software Engineering Institute, Carnegie Mellon University. Capability Maturity Model® and CMMI® are registered in the U. S. Patent and Trademark Office by Carnegie Mellon University.

(SMS) Abelson, L. A., S. Eslinger, M. C. Gechman, C. H. Ledoux, M. V. Lieu, K. Korzac, *Software Measurement Standard for Mission Critical Systems*, Aerospace Report No. TOR-2009(8506)-6, 5 May, 2011, The Aerospace Corporation.

(SDSMCS) Adams, R. J., S. Eslinger, K. L. Owens, J. M. Tagami, and M. A. Zambrana,  *Software Development Standard for Mission Critical Systems (SDSMCS)*, Aerospace Report No. TR-RS-2015-00012, March 17, 2014, The Aerospace Corporation. This is the same as SMC Standard SMC-S-012, *Software Development Standard*.

1. Scope.
   1. Identification.

This Software Development Plan is the plans for the development of the DRONEswarm™  
system for the Cooperative Technology Consortium. The DRONEswarm™ system shall consist of three software segments the Ground Control Segment (GCS), Air Vehicle Segment (AVS), and the Swarm Simulation Segment (SSS) as well as a Ground to Air Segment Interface. It shall support the use of Commercial off the Shelf (COTS) laptops. This plan shall result in the development of the first release of DRONEswarm™ version 1.0. Table 1-1-1 below defines the work products for the DRONEswarm™ system. Table 1-1-2 lists the planed CSCI’s for the DRONEswarm™ system. Table 1-1-3 lists the CSCI’s their abbreviations, version and release numbers.

**Table 1-1-1.**

|  |  |  |
| --- | --- | --- |
| **Work Products** | **System(s)** | **Quantity** |
| Software Development Plan | DRONEswarm™ | 1 |
| Software Requirements Specification | AVS, GCS, SSS, AVS to GCS interface | 4 |
| Software Design Description | AVS, GCS, SSS, AVS to GCS interface | 4 |
| Software Test Plan | AVS, GCS, SSS, AVS to GCS interface | 4 |
| Software Test Report | AVS, GCS, SSS, AVS to GCS interface | 4 |
| Software User Manual | AVS, GCS, SSS, AVS to GCS interface | 4 |
| AVS to GCS Interface Control Document (ICD) | DRONEswarm™ | 1 |
| System Traceability Document | DRONEswarm™ | 1 |

**Table 1-1-2**

|  |  |  |
| --- | --- | --- |
| **CSCI** | **System(s)** | **Quantity** |
| Ground Control Segment Software | GCS | 1 |
| Air Vehicle Segment Software | AVS | 1 |
| Swarm Simulation Segment Software | SSS | 1 |
| Ground to Air Segment Interface | GCS/AVS | 1 |

**Table 1-1-3**

|  |  |  |  |
| --- | --- | --- | --- |
| **CSCI** | **Abbreviation** | **Version** | **Release** |
| Ground Control Segment Software | GCS | 1 | 1.1 |
| Air Vehicle Segment Software | AVS | 1 | 1.1 |
| Swarm Simulation Segment Software | SSS | 1 | 1.1 |
| Ground to Air Segment Interface | \*None | 1 | 1.1 |

\*This paragraph **shall** contain a full identification of the *system* and the *software* to which this *document* applies, including, as *applicable*, identification number(s), title(s), abbreviation(s), version number(s), and release number(s).\*

* 1. System overview.   
       
     The purpose of the system is to have three segments of software, the Ground Control Segment, Air vehicle Segment, the, and Swarm Simulation Segment. The Ground Control Segment will consist of mission specific software hosted on a COTS laptop computer. The GCS will confine the Air Vehicle System. The Air Vehicle System shall consist of a constellation of 850 or fewer multi-rotor unmanned air vehicles that perform autonomous control limited by the GCS. The AVS shall also be compatible with customer-furnished electronics. The Swarm Simulation Segment is an offline non-realtime software tool, hosted on COTS hardware, used for operator training and high-level system performance characterization.  
       
     The DRONEswarm™ system is a new development and no prior development, operation, or maintenance history exists. No sponsor, user(s), or support organizations have been established at this moment in time. Same goes for current or planned operating and user sites. The acquirer of the system is The Cooperative Technology Consortium (CTC) and the developer of the system is solely the ACME Software Company   
       
     \*This paragraph **shall** briefly state the purpose of the *system* and the *software*

to which this *document* applies. It **shall**: a) describe the general nature of the *system* and *software*; b)

summarize the history of system development, operation, and maintenance; c) identify the project sponsor, *acquirer*, *user*, *developer*, and support organizations; and d) identify current and planned operating and *user sites*.\*

* 1. Document overview.  
     The purpose of this document is to serve as a Software Development Plan for the development of DRONEswarm™ system by the ACME Software Company as well as the Cooperative Technology Consortium.  
        
     The document encompasses multiple sections.  
     Section 1 contains the overall scope  
     Section 2 contains a list of all referenced documents   
     Section 3 contains the overview of the required work  
     Section 4 contains the general requirements for development  
     Section 5 contains plans for performing detailed software development activities

Section 6 contains schedules and activity networks  
Section 7 contains project organization and resources   
  
This Software Development Plan shall only be distributed to the ACME Software Company as well as the Cooperative Technology Consortium. This plan is intended for only these two companies as well as the sponsor of the product (when established).   
  
  
 \*This paragraph **shall** summarize the purpose and contents of this *document*. This paragraph **shall** describe any *security* or privacy considerations associated with its use.\*

* 1. Relationship to other plans.   
     This SDP along with the Software Test Plan (STP) and Software Management Plan (SMP) will all serve as guidelines to the development of the DRONEswarm™ system. There will be 4 STPs in total one for each of the configuration items listed in Table 1-1-2. The STP will describe in full detail the testing procedures for the DRONEswarm™ system in depth. While the SDP touches slightly on testing the STP will further explain, describe, and guide in the testing phases of development. The Software Management Plan will describe the specific roles of each team member to the DRONEswarm™ system. This will include the responsibility, authority, and accountability of each team member to the DRONEswarm™ system. See table 7-1-1-1 for the list of team members. In addition the SMP defines the administrative as well as technical methods required to deliver the project requirements.  
       
     \*This paragraph **shall** describe the relationship, if any, of the SDP to other project management plans.\*

1. Referenced documents. \*This section **shall** list the number, title, revision, and date of all *documents* referenced in this plan. This section **shall** also identify the source for all *documents* not available through normal Government stocking activities.\*
2. Overview of required work. \*This section **shall** be divided into paragraphs as needed to *establish* the context for the planning described in later sections. It **shall** include, as *applicable*, an overview of:
3. *Requirements* and constraints on the *system* and *software* to be developed,
4. *Requirements* and constraints on project documentation;
5. Position of the project in the *system* lifecycle;
6. The selected project and acquisition strategy;
7. Any *requirements* or constraints on the selected project and acquisition strategy;
8. *Requirements* and constraints on project schedules and resources; and
9. Other requirements and constraints, such as on project *security*, privacy, methods, standards, interdependencies on hardware and *software development*.\*
10. General requirements. \*Provisions corresponding to nonrequired activities may be satisfied by the words “Not applicable.” If different *builds* or different *software* on the project require different planning, these differences **shall** be noted in the paragraphs. See Section 4 in the body of the Software Development Standard for Mission Critical Systems (SDSMCS) for the activities and topics to be addressed in this leading paragraph. This section **shall** be divided into the following paragraphs. In addition to the content specified below, each paragraph **shall** identify *applicable* risks and uncertainties and plans for dealing with them.\*
    1. Software development process.   
         
       The software development lifecycle model to be used is the incremental lifecycle model. Initial requirements elicitation will be performed before entering each phase of the multiple incremental builds.   
         
       The software development activities to be performed in each build include: planning, requirements, analysis, design, implementation, testing, deployment, and maintenance. A software test report shall be generated in each testing phase that occurs. Peer reviews will also be conducted in the deployment phase.  
         
       Separate builds for each segment (Air, Ground, and Simulation) will be performed. The final build will comprise of the integration of the Air and Ground segments that is validated by the Simulation Segment. The objective of each build is to complete the segments of the software and in the final build integrate them together to form the DRONEswarm™ system.  
         
         
       \*This paragraph **shall** describe the *software development process* to be used. The planning **shall** cover: a) identification of the *software development lifecycle model(s)* to be used; b) planned *builds*, if *applicable*; c) their build objectives; and d) the software development activities to be performed in each *build*. See Section 4.1 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph.\*
    2. General requirements for software development. \*See Section 4.2 and its subsections in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *software development*. This paragraph **shall** be divided into the following subparagraphs.\*
       1. Software development methods. \*This paragraph **shall** describe or reference the software development methods to be used. This paragraph **shall** include descriptions of the manual and automated tools and procedures to be used in support of these methods. Reference may be made to other paragraphs in this plan if the methods are better described in context with the activities to which they will be applied. See Section 4.2.1 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software development methods.\*
       2. Standards for products.  
          Table 4-2-2 lists the standards that coincide with the development of the DRONEswarm™ system. These standards are applicable to software requirements, architecture, design, code, test cases, test procedures, test results, test logs, and discrepancy and change reports. Standards can be found in their specified Appendices.  
           **Table 4-2-2. Standards List**

|  |  |  |
| --- | --- | --- |
| **Appendix** | **Section** | **Standard Name** |
| S | 1.1 | ACME JavaScript Coding Standards |
| S | 1.2 | ACME HTML Coding Standards |
| S | 1.3 | ACME CSS Coding Standards |
| S | 1.4 | ACME PHP Coding Standards |
| S | 2.1 | ACME Discrepancy Standards |
| S | 2.2 | ACME Change Reports Standards |
| S | 3.1 | ISO/IEC/IEEE 29119-1: Concepts & Definitions |
| S | 3.2 | ISO/IEC/IEEE 29119-2: Test Processes |
| S | 3.3 | ISO/IEC/IEEE 29119-3: Test Documentation |
| S | 3.4 | ISO/IEC/IEEE 29119-4: Test Techniques |
| S | 3.5 | ISO/IEC/IEEE 29119-5: Keyword Driven Testing |
| S | 4.1 | ACME Software Design Pattern Standards |
| S | 5.1 | ISO/IEC/IEEE 42010:2011: Software Architecture Standard |
| S | 6.1 | ANSI/IEEE 830-1984: Software Requirements Standard |

\*This paragraph **shall** describe or reference the standards to be followed for representing *requirements*, *architecture*, *design*, code, *test cases*, *test procedures*, test results, *test* logs, and *discrepancy and change reports*. The standards **shall** cover all *contractual requirements* concerning standards for *products*. Reference may be made to other paragraphs in this plan if the standards are better described in context with the activities to which they will be applied. See Section 4.2.2 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on standards for software *products*.\*

The contents of Paragraph 4.2.2 **shall** be placed into separate appendices of the SDP, not in Paragraph 4.2.2. Paragraph 4.2.2 **shall** reference these appendices.

4.2.2.1 Standards for code.

Standards for code include:

ACME JavaScript Coding Standards

ACME HTML Coding Standards

ACME CSS Coding Standards

ACME PHP Coding Standards

\*Standards for code **shall** be provided for each programming language to be used. The coding standards for each language **shall** include, as a minimum:

* + - 1. Standards for format (such as indentation, spacing, capitalization, and order of information);
      2. Standards for header comments, requiring, for example, name and identifier of the code; version identification; modification history; purpose; *requirements* and *design* decisions implemented; notes on the processing (such as algorithms used, assumptions, constraints, limitations, and side effects); and notes on the data (e.g., inputs, outputs, variables, data structures);
      3. Standards for other comments, such as required number and content expectations);
      4. Naming conventions (e.g., for constants, types, variables, parameters, packages, procedures, classes, objects, methods, functions, files);
      5. Restrictions, if any, on the use of programming language constructs or features; and
      6. Restrictions, if any, on the complexity of code aggregates.\*
      7. Standards for DCRs.

Standards for DCRs include:   
  
ACME Discrepancy Standards  
ACME Change Reports Standards

\*Standards for *discrepancy and change reports* (*DCRs*) **shall** be provided. The *DCR* standards **shall** include, as a minimum:

1. A glossary and definitions of terms that can be used in *discrepancy and change reports* (*DCRs*), including all specialized terms used in *DCR* titles, descriptions, causes, and resolutions;
2. Alternative and additional definitions, if any, for *DCR* terms specified in Appendix C.2.1 of this standard;
3. A *DCR* acronym list that includes all acronyms that are used (or are permitted to be used) in *DCRs.* These acronyms might appear in (e.g., *DCR* titles, free text descriptions of *test incidents*, *discrepancies*, failures, causes, resolutions, and development, integration and qualification test activity names);

Note: This *DCR* acronym list is in addition to the acronym list for the entire SDP.

1. A list of activity names and their definitions used for *DCRs* besides those in Appendix C, Table C.2-2, of the standard; and
2. The names and sequence of the *DCR* steps that can be used.\*
   * + 1. Standards for test logs.

Standards for test logs include:   
  
ISO/IEC/IEEE 29119-1: Concepts & Definitions

ISO/IEC/IEEE 29119-2: Test Processes

ISO/IEC/IEEE 29119-3: Test Documentation

ISO/IEC/IEEE 29119-4: Test Techniques

ISO/IEC/IEEE 29119-5: Keyword Driven Testing

\*Standards for test logs **shall** be provided. The test log standards

**shall** include, as a minimum:

1. The test log fields and terms specified in Appendix F.2 of (SDSMCS); and
2. Alternative and additional definitions, if any, for test log terms specified in Appendix F.2.1 of (SDSMCS).\*
   * 1. Traceability.   
        In order to maximize bidirectional traceability multiple traceability matrices and graphs will be developed between system requirement and software requirements, between software requirements and design, between design and implementation, between system requirements and qualification test information, and between computer hardware resource utilization requirements and measured computer hardware resource utilization. This will create a very clear picture of the software’s traceability. Traceability matrices and graphs will be created using Microsoft Excel. The collection of matrices and graphs will be compiled into System Traceability Document for further analysis.  
          
          
        \*This paragraph **shall** describe the approach to be followed for establishing and maintaining *bidirectional traceability* between levels of *requirements*, between *requirements* and *design*, between *design* and the *software* that implements it, between *requirements* and *qualification test* information, and between *computer hardware* resource utilization *requirements* and measured computer hardware resource utilization. See Section 4.2.3 in the body of (SDSMCS) for the activities, topics, and other items to be addressed in this paragraph on *bidirectional traceability*.\*
     2. Reusable software products.\* See Section 4.2.4 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *reusable software products*. This paragraph **shall** be divided into the following subparagraphs.\*
        1. Incorporating reusable software products. \*This paragraph **shall** describe the approach to be followed for identifying, evaluating, and incorporating *reusable software products*, including the scope of the search for such *products* and the criteria to be used for their *evaluation*. Candidate or selected *reusable software products* known at the time this plan is prepared or updated **shall** be identified and described, together with benefits, drawbacks, alternatives considered, rationale for those selected, remaining viable alternatives, and restrictions, as *applicable*, associated with their use.\*
        2. Developing reusable software products. \*This paragraph **shall** describe the approach to be followed for identifying, evaluating, and reporting opportunities for developing *reusable software products*.\*
     3. Assurance of critical requirements. \*See Section 4.2.5 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on assurance of critical *requirements*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for handling *requirements* designated critical.
        1. *Safety*
        2. *Security*
        3. *Privacy protection*
        4. *Reliability, maintainability,* and *availability*
        5. *Dependability*
        6. Human system integration, including *human factors engineering* and
        7. Assurance of other mission-critical *requirements* as agreed to by the *acquirer* and

*Developer\**

* + 1. Computer hardware resource utilization. \*This paragraph **shall** describe the approach to be followed for allocating *computer hardware* resources and monitoring their utilization. See

Section 4.2.6 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on computer hardware resource utilization.\*

* + 1. Recording rationale. \*This paragraph **shall** describe the approach to be followed for recording rationale that will be useful to the support organization for key decisions made on the project. It **shall** interpret the term “key decisions” for the project. It **shall** state where the rationale are to be recorded. See Section 4.2.7 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on recording rationale.\*
    2. Access for acquirer review. \*This paragraph **shall** describe the approach to be followed for providing the *acquirer* and its authorized representatives access to *developer* and *software team member* facilities for review of *products* and activities. It **shall** cover all *contractual requirements* concerning *acquirer* team access for review. See Section 4.2.8 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on access for *acquirer* review.\*
    3. Contractual requirements. \*This paragraph **shall** describe the approach to be followed for meeting all the *contractual requirements* regarding *software development*, including *testing*, *transition*, *maintenance*, and operations. Reference may be made to other paragraphs in this plan if the approach to be followed for meeting *contractual requirements* is better described in context with the activities to which they will be applied. These *contractual requirements* can be found in, e.g., the *Statement of Work (SOW)*, *Contract Data Requirements List (CDRL)*, compliance documents and their tailoring, Integrated Master Plan (IMP), specifications, Section H of the Model *Contract* (Sections A-K of the RFP and Contract), and other *contractual documentation*.\*

1. Plans for performing detailed software development activities. \*The paragraphs below cover the plans for performing detailed software development activities. Provisions corresponding to nonrequired activities may be satisfied by the words “Not applicable.” If different *builds* or different *software* on the project require different planning, these differences **shall** be noted in the paragraphs. If different planning is required for new development, modification, *reusable software products*, *reengineering*, and *maintenance*, these differences **shall** be described in the paragraphs. The discussion of each activity **shall** include the approach, i.e., plans, *processes*, methods, procedures, tools, roles, and responsibilities, to be applied to: 1) the analysis or other technical tasks involved, 2) the recording of results, and 3) the preparation of associated deliverables, if *applicable*. For each activity, include a) entrance criteria, b) inputs, c) tasks to be accomplished, d) *products* to be produced, e) *verifications* to be used to ensure tasks are performed according to their defined *processes* and *products* meet their *requirements*, f) outputs, and g) exit criteria. The discussion **shall** also identify *applicable* risks and uncertainties and plans for dealing with them. Reference may be made to paragraph 4.2.1 if *applicable* methods are described there. This section **shall** be divided into the following paragraphs.\*
   1. Project planning and oversight. \*See Section 5.1 and its subparagraphs in the body of the Software Development Standard for Mission Critical Systems (SDSMCS) for the activities and topics to be addressed in this paragraph on project planning and oversight. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for project planning and oversight.
      1. Software development planning
      2. Software integration and test planning
         1. Software integration planning
         2. Software item qualification test planning
      3. System qualification test planning
      4. Planning for *software transition* to operations
      5. Planning for *software transition* to *maintenance*
      6. Following and updating plans\*
   2. Establishing a software development environment. \*The *developer* **shall** *record* the results of the *software engineering environment* adequacy analysis in the SDP. The *developer* **shall** *record* the results of the *software integration and qualification test environment* adequacy analysis in the SDP. See Section 5.2 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on establishing and maintaining software development environments. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for establishing, controlling, and maintaining a software development environment.
      1. *Software engineering environment*
         1. *Software engineering environment* description
         2. *Software engineering environment* adequacy analysis reports
      2. *Software integration and qualification test environment*
         1. *Software integration and qualification test environment* description
         2. *Software integration and qualification test environment* adequacy analysis reports
      3. *Software development library*
      4. *Software development files*
      5. *Nondeliverable software\**
   3. System requirements analysis. \*See Section 5.3 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on system requirements analysis. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for participating in system requirements analysis.
      1. Analysis of *user* input
      2. Operational concept
      3. System *requirements* definition\*
   4. System architecture and design. \*See Section 5.4 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *requirements* for system architectural *design*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for participating in system architectural *design*.
      1. System-wide architectural design decisions
      2. System architectural *design\**
   5. Software requirements analysis. \*This paragraph **shall** describe the approach to be followed for software requirements analysis. See Section 5.5 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software requirements analysis.\*
   6. Software architecture and design. \*See Section 5.6 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software *architecture* and *design*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for software *design*.
      1. Overall software *architecture*
      2. Software item *architecture*
      3. Software item detailed *design*
         1. Software unit detailed *design*
         2. Software interface *design*
         3. Database *design*, as *applicable*
         4. User interface *design*, as *applicable*
         5. Other applicable software *design* (e.g., model-based software, as *applicable)\**
   7. Software implementation and unit testing. \*See Section 5.7 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software implementation and unit *testing*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for software implementation and unit *testing*.
      1. Implementing *software*
      2. Preparing for unit *testing*
      3. Performing unit *testing*
      4. Analyzing and recording unit testing results
      5. Unit *regression testing*
      6. Revising and retesting units\*
   8. Unit integration and testing. \*See Section 5.8 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software unit integration and *testing*.\*

\*This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for unit integration and *testing*.

* + 1. Testing on the *target computer system*
    2. Preparing for unit integration and *testing*
    3. Performing unit integration and *testing*
    4. Analyzing and recording unit integration and *test* results
    5. Unit integration *regression testing*
    6. Revising and retesting unit integration\*
  1. Software item qualification testing. \*See Section 5.9 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *software item qualification testing*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for *software item qualification testing*.
     1. Independence in *software item qualification testing*
     2. Testing on the *target computer system*
     3. Preparing for *software item qualification testing*
     4. Dry run of *software item qualification testing*
     5. Performing *software item qualification testing*
     6. Analyzing and recording *software item qualification test* results
     7. Software item qualification *regression testing*
     8. Revising and retesting *software items\**
  2. Software-hardware item integration and testing. \*See Section 5.10 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *software- hardware item* integration and *testing*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for participating in *software*-*hardware item* integration and *testing*.
     1. Testing on the *target computer system*
     2. Preparing for *software*-*hardware item* integration and *testing*
     3. Performing *software*-*hardware item* integration and *testing*
     4. Analyzing and recording *software-hardware item* integration and *test* results
     5. Software-hardware item integration *regression testing*
     6. Revising and retesting *software-hardware item* integration\*
  3. System qualification testing. \*See Section 5.11 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on system *qualification testing*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for participating in system *qualification testing*.
     1. Independence in system *qualification testing*
     2. Testing on the *target computer system(s)*
     3. Preparing for system *qualification testing*
     4. Dry run of system *qualification testing*
     5. Performing system *qualification testing*
     6. Analyzing and recording system *qualification test* results
     7. System qualification *regression testing*
     8. Revising and retesting the *system\**
  4. Preparing for software transition to operations. \*See Section 5.12 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on preparing for *software transition* to operations. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for preparing for *software transition* to operations.
     1. Preparing the executable *software*
     2. Preparing version descriptions for *user sites*
     3. Preparing user manuals
        1. Software user manuals
        2. Computer operations manuals
     4. Installation at *user sites\**
  5. Preparing for software transition to maintenance. \*See Section 5.13 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on preparing for *software transition* to *maintenance*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for preparing for *software transition* to *maintenance*.
     1. Preparing the executable *software*
     2. Preparing source files
     3. Preparing version descriptions for the *maintenance* site(s)
     4. Preparing the “as built” software *architecture*, *design,* and related information
     5. Updating the *system*/*subsystem design* description
     6. Updating the software *requirements*
     7. Updating the *system requirements*
     8. Preparing *maintenance* manuals
        1. Computer programming manuals
        2. *Firmware* support manuals
     9. *Transition* to the designated *maintenance* site\*
  6. Software configuration management. \*See Section 5.14 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software configuration management. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for software configuration management.
     1. Configuration identification
     2. Configuration control
     3. Configuration status accounting
     4. Configuration audits
     5. Packaging, storage, handling, and delivery
     6. *Baselines\**
  7. Software peer reviews and product evaluations. \*See Section 5.15 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software peer reviews and product *evaluations*. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for software peer reviews and product *evaluations*.
     1. Software peer reviews
        1. Plan for software peer reviews:

Each peer review will be conducted by a team with particular roles. *\*Note some roles may consist of multiple team members not necessarily one.* These roles will consist of a leader, initializer, moderator, recorder, spokesperson, reviewer, and validator. The leader is in charge of a work product and initiates for a peer review to take place with a statement of his objective for the peer review. The initializer is in charge of gathering a team consisting of a moderator, recorder, spokesperson, reviewer, and validator for a leader’s peer review. The moderator works closely with the leader to plan and schedule the peer review. They are also in charge of leading the peer review acting as a facilitator by making sure everything is running smoothly and is staying on track. The recorder records issues, comments, defects and any other notable gestures made during the peer review. The spokesperson is the main communicator during the peer review; they present the product to the team in a way so that team members can thoroughly review the product at hand. The reviewer analyzes the work product at hand and determines how it works, find defects in it and make note of them. The validator cooperates with the leader to determine if the work product is suitable and ready for a peer review.  
  
**Plan:**   
Step 1: A leader submits a request for a peer review to the validator who either accepts or disapproves of the request.

Step 2: Upon request acceptance the leader seek out an initializer who generates a peer review team.  
  
Step 3: The leader and moderator meet to discuss the goal of the peer review and the moderator plans and schedules it, determining how much time the review will take.  
  
Step 4: The moderator issues a notice to peer review members in advance when the peer review meeting will take place.

* + - 1. Prepare for an individual peer review:

**Prepare:**   
Step 1: The leader and moderator meet with reviewers to discuss the vision of the peer review.

Step 2: The reviewer inspects the work product taking note of defects and its overall structure and functionality.   
  
Step 3: The reviewer gives the inspection data to the leader and moderator to aid in preparation for the peer review.

* + - 1. Conduct peer reviews:

**Conduct:**   
Step 1: The moderator opens the peer review meeting stating the overall goal of the review as well as state each team members role in the peer review effort.

Step 2: The moderator addresses the reviewers and ask if they have sufficient data to conduct the peer review. If there is not enough data available for review, the peer review will be rescheduled.  
  
Step 3: The spokesperson will introduce the work product to the peer review team, describing its components.  
  
Step 4: Reviewers will raise problems, defects, and general concerns of the work product to the rest of the review team.  
  
Step 5: The recorder records all defects, problems, and concerns during the course of the peer review.

Step 6: Allow the floor to be open to team member questions and feedback. *\*Questions, and answers from the leader should be recorded.*  
  
Step 7: The leader and reviewers decide the future vision of the work product, changes, additions, or if sufficient it will be accepted.  
  
Step 8: All team members will submit a review of the peer review meeting suggesting additional feedback or improvements of the process.

* + - 1. Analyze and report peer review data:

**Analysis and Report:**Step 1:The recorder generates a defect and issue report that is sent to the leader for revision and submission.  
Step 2: The leader reviews the future vision of the work product determined from the peer review and generates a formal document of future work product vision and submit to higher management.  
  
Step 3: The leader generates an overall report of the peer review stating what was accomplished and what was not and their final feedback on the peer review.  
  
Step 4: The leader generates a goal report to be sent to the moderate to aid in preparation for the next upcoming peer review if required.

* + 1. Product *evaluations*
       1. In-process and final product *evaluations*
       2. Product *evaluation* records
       3. Independence in product *evaluations\**
  1. Software quality assurance. \*See Section 5.16 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software quality assurance. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for software quality assurance.
     1. Software quality assurance *evaluations*
     2. Software quality assurance records
     3. Independence in software quality assurance
     4. Software quality assurance noncompliance issues\*
  2. Corrective action. \*See Section 5.17 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on corrective action. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for corrective action.
     1. *Discrepancy and change reports (DCRs)*

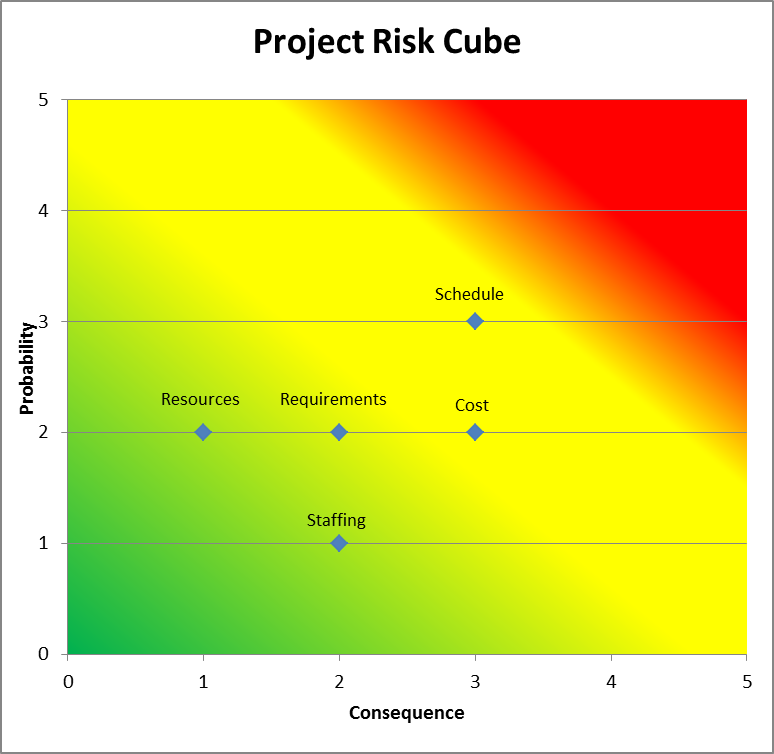
These *DCRs* **shall** include the items to be recorded specified in Appendix C, Table C.2-5 of (SDSMCS).

* + 1. Corrective action system\*
  1. Joint technical and management reviews. \*See Section 5.18 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on joint technical and management reviews. See Appendix E, Joint Technical and Management Reviews for additional *requirements* for joint technical and management reviews. This paragraph **shall** be divided into the following subparagraphs to describe the approach to be followed for joint technical and management reviews.
     1. Joint technical reviews
     2. Joint management reviews\*
  2. Software risk management.

The approach to managing risk throughout the lifecycle of the program will be to develop a risk management process that identifies, measures/tracks, and establishes the progress of risk mitigation throughout the program. Table 5-19-1-1 defines the initial risk for the program development. Identified risk will be mapped into a risk cube identifying the probability of occurrence and the resulting consequences. 0 is the lowest possible probability or consequences while 5 is the maximum for both. The green area represents minimal, yellow is moderate, and red is high risk. The initial risks have been mapped into a risk cube in diagram 5-19-1-2.

**Table 5-19-1-1.**

|  |  |  |
| --- | --- | --- |
| **Risk** | **Mitigation** | **Measurement** |
| Cost | Ensure a solid baseline cost estimate.  Create a sufficient management reserve.  Ensure scope is thoroughly understood. | Tracking of cost to the baseline cost estimate and management reserve. |
| Schedule | Ensure a solid baseline schedule that details incremental task with earned value applied.  Conduct an Initial Baseline Review (IBR).  Ensure a proper amount of critical milestones are reflected in the schedule. | Tracking of progress to the baseline schedule.  Tracking Earned Value.  Track completed Milestones. |
| Staffing | Create a detailed staffing plan that reflects the required staffing/ staffing skillsets throughout the development of the project.  Ensure that staffing is available via hiring or other methods throughout the project development.  Conduct thorough interviews with potential staff personal to ensure skill set requirements are met | Track the baseline staffing projection to actuals.  Track potential employees.  Track amount of interviews conducted. |
| Requirements | Conduct a Software Requirements Review.  Minimize gold plating.  Define the Requirements verification methodology early. | Track requirements verified  Track new requirements. |
| Resources | Make sure all recourses are identified for the development process early.  Create a Resource Reserve that can be allocated if more resources are required throughout the development process. | Track resource allocation.  Track the Resource Reserve. |



**Diagram 5-19-1-2.**

\*This paragraph **shall** describe the approach for performing *risk* management. See Section 5.19 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software *risk* management. \*

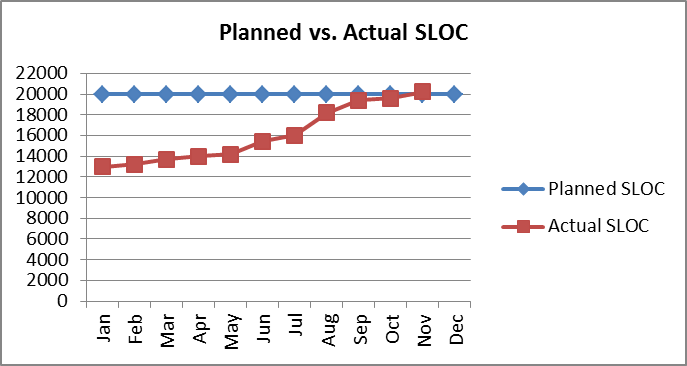
* 1. Software measurement.

The overall approach to be used for software measurement is to define and identify the critical items to be tracked throughout the program development. As shown below in table 5-20-1. the specific software measurements to be collected, analyzed, interpreted, applied, tracked, and the importance are listed.

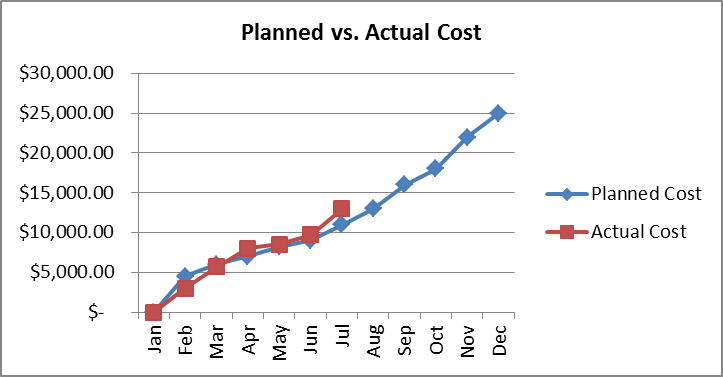
**Table 5-20-1.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Measurement** | **Report** | **Importance** |
| Project Size | Planned vs. Actual SLOC, Contract deliverables | Line Graph  & Table | Track if the project is expected to be the same size as planned or is growing. |
| Cost | Planned cost vs. Actual cost expended | Line Graph | Track whether or not the program is over or under budget |
| Staffing | Planned staff size vs. actual staff size | Line Graph | Track whether or not the project is staffing more, less or equal individuals than planned |
| Schedule | Planned schedule dates vs. Actual schedule dates | Gantt chart | Track whether or not the project is on or behind schedule |
| Quality | Amount of DRs vs. SLOCs  DRs closure rate | Line Graph | Track the overall quality of the project and closure rate of DRs |
| Requirements | Requirements verified vs. Schedule | Line Graph | Track the verification of requirements & any new requirements definition |
| Risk | Mitigated Risk vs. Schedule | Line Graph | Track the definition and mitigation of risk identified. |

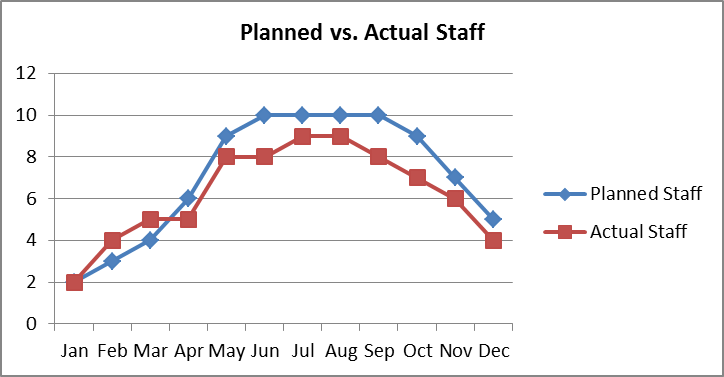
**Project Size Report Template**



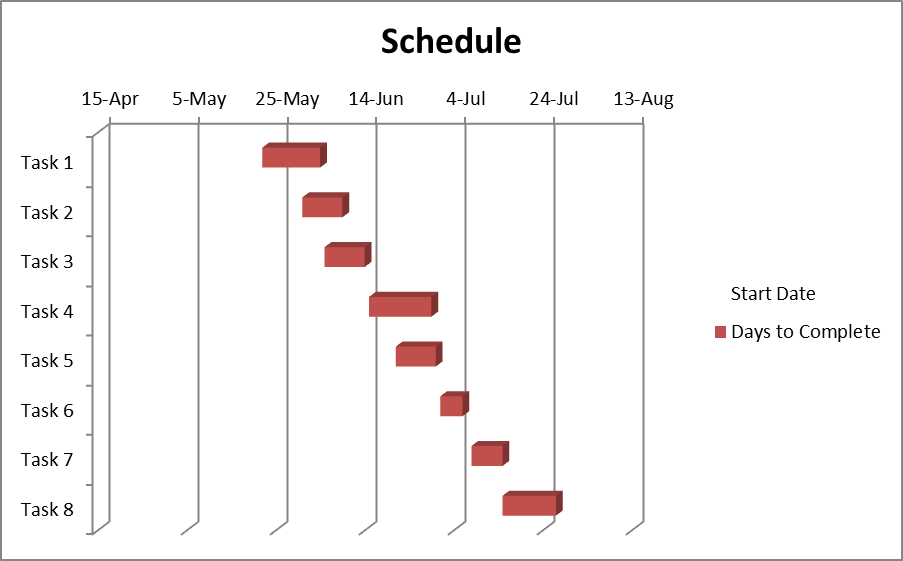
**Cost Report Template**



**Staffing Report Template**

****

**Schedule Report Template**



**\*The rest of the Report Templates can be found in Appendix P.**

\*This paragraph **shall** briefly summarize the approach to be used for software measurement throughout the system development lifecycle. This paragraph **shall** also itemize the specific software measurements to be collected, analyzed, interpreted, applied, and reported. In addition, this paragraph **shall** summarize the importance of each specific measurement used for decision making, corrective actions, and reporting to the *acquirer*. See Section 5.20 and its subparagraphs in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on software measurement. When a separate software measurement plan (SMP) is not required on *contract*, this paragraph **shall** include the content described in the SMP template provided in Appendix H.4. When a separate SMP is required on *contract*, this paragraph **shall** include a reference to the SMP.

* + 1. Software measurement planning
    2. Software measurement reporting
    3. Software measurement working group (SMWG)\*
  1. Security and privacy. \*This paragraph **shall** describe the approach for meeting the *security* and privacy *requirements*. See Section 5.21 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *security* and privacy.\*
  2. Software team member management. \*This paragraph **shall** list all software *developers* at any level (e.g., *prime contractor*, *software team members*). This paragraph **shall** identify all *software,* including custom, *COTS*, modified, and reused, developed by foreign contractors at any level (e.g., *prime contractor*, *software team members*) that will be delivered to the *acquirer*. This paragraph **shall** identify the foreign contractor’s company name and foreign location(s). A “foreign contractor” means any foreign corporation, business association, partnership, trust, society or any other entity or group that is not incorporated or organized to do business in the United States, as well as international organizations, foreign Governments, and any agency or subdivision of foreign Governments (e.g., diplomatic missions). This paragraph **shall** describe the approach for performing *software team member* management. This paragraph **shall** specify the mechanisms to be used to ensure that all *contractual requirements,* and all changes to *contractual requirements,* are flowed down to all levels of *software team members*. See Section 5.22 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on *software team member* management.\*
  3. Interface with software independent verification and validation (IV&V) agents. \*This paragraph **shall** describe the approach for interfacing with the software *IV&V* agents. See Section 5.23 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on interfacing with software independent verification and validation agents.\*
  4. Coordination with associate developers. \*This paragraph **shall** describe the approach for performing the coordination with *associate developers*, working groups, and *interface* groups. See Section 5.24 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on coordination with *associate developers*.\*
  5. Improvement of project processes. \*This paragraph **shall** describe the approach for performing the improvement of project *processes*. See Section 5.25 in the body of (SDSMCS) for the activities and topics to be addressed in this paragraph on improvement of project *processes*.\*

1. Schedules and activity network. \*This section **shall** be divided into the following paragraphs:\*
   1. Schedule. \*This paragraph **shall** present schedule(s) identifying the activities and showing initiation of each activity, availability of draft and final deliverables, other milestones, and completion of each activity. This paragraph **shall** provide the detailed schedule activities for each *software item,* and other *software*, for each build, and for the entire *software development* lifecycle. See paragraph 7.2.1.4 below for inclusion of the rationale for the software cost and schedule estimation, including software cost and schedule estimation techniques, the input to those techniques (e.g., software size and *software* cost driver parameters and scale factors), and any assumptions made.\*
   2. Activity network. \*An activity network depicting sequential relationships and dependencies among activities and identifying those activities that impose the greatest time restrictions on the project.\*
2. Project organization and resources. \*This section **shall** be divided into the following paragraphs to describe the project organization and resources.\*
   1. Project organization. \*This paragraph **shall** be divided into the following subparagraphs to describe the organizational structure to be used on the project, including the organizations involved, their relationships to one another, and the authority and responsibility of each organization for carrying out required activities.

Note: *COTS software suppliers* are not included in this paragraph and its subparagraphs.\*

* + 1. Software team members.

Table 7-1-1-1 lists each team member as well as important additional information regarding each team member. Diagram 7-1-1-2 displays the Program Organization Chart for the ACME Software Company.  
  
**Table 7-1-1-1.**

|  |  |
| --- | --- |
| Team Member | Information **(*\*See list 7-1-1 below)*** |
| Program Manager | a. ACME Headquarters b. ACME Software Company c. Program Management d. Orange, CA, USA e. Management f. Program Management |
| Business Manager | a. ACME Headquarters b. ACME Software Company c. Program Management d. Orange, CA, USA e. Management f. Program Management |
| Systems Engineer | a. ACME Headquarters b. ACME Software Company c. Program Management d. Orange, CA, USA e. System Development & Design f. |
| Software Engineer | a. ACME Headquarters b. ACME Software Company c. Program Management d. Orange, CA, USA e. Software Development & Design f. Software Development Group |
| Integration and Test Engineers | a. ACME Headquarters b. ACME Software Company c. Program Management d. Orange, CA, USA e. System & Software Integration & Testing f. Test Group |
| Quality Assurance | a. ACME Headquarters b. ACME Software Company c. Program Management d. Orange, CA, USA e. Management f. Test Group |

**List 7-1-1.**

a. Organization site name (e.g., XYZ Co. Div ABA, City, State);  
b. Parent organization name (e.g., company);  
c. Internal organization name, i.e., name of division or other level (e.g., ground software development);  
d. Organization site location, i.e., city, state, and country;  
e. Software-related activities that the software team member is expected to be performing at this site;   
f. Internal structure of each software team member, showing all software-related entities (e.g., software development groups, test groups, software process organizations, software quality assurance, software configuration management) and how they relate to other project and organizational entities (e.g., program management, systems engineering, system integration and test, hardware engineering, quality assurance, configuration management).

**Diagram 7-1-1-2.**

**Program Organization Chart**

Business Manager

Program Manager

Software Integration & Test

Quality Assurance

Software Engineer

Systems Engineer

\*This paragraph **shall** identify each geographic site of each *software team member* organization that is performing the project-related effort for any software-related activities. (See Section 3.1 of the body of (SDSMCS) for the definition of *software team member*.)

For each *software team member* organization site this paragraph **shall** include all of the following information:

* + - 1. Organization site name (e.g., XYZ Co. Div ABA, City, State);
      2. Parent organization name (e.g., company);
      3. Internal organization name, i.e., name of division or other level (e.g., ground software development);
      4. Organization site location, i.e., city, state, and country;
      5. Software-related activities that the *software team member* is expected to be performing at this site; and
      6. The internal structure of each *software team member*, showing all software-related entities (e.g., software development groups, test groups, software process organizations, software quality assurance, software configuration management) and how they relate to other project and organizational entities (e.g., program management, systems engineering, system integration and test, hardware engineering, quality assurance, configuration management).

Note: References to supplied organization charts could provide this information for item f.\*

* + 1. Full set of project software.

The full set of project software is represented in both Table 7-1-2-1 and Diagram 7-1-2-2. Table 7-1-2-1 lists each software item the system it belongs to, its category, and important details regarding the software item. Diagram 7-1-2-2 is a displays the System Specification Tree for the DRONEswarm™ system.

**Table 7-1-2-1.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Software Item** | **System** | **Category** | **Details (*\*See list 7-1-2 below)*** |
| Ground Control Segment Software | GCS | Software | (1) Software Engineer (2) ACME Headquarters  (3)All (4)New |
| Air Vehicle Segment Software | AVS | Software | (1) Software Engineer (2) ACME Headquarters  (3)All (4)New |
| Swarm Simulation Segment Software | SSS | Software | (1) Software Engineer (2) ACME Headquarters  (3)All (4)New |
| Ground to Air Segment Interface | GCS/AVS | Software | (1) Software Engineer & Integration/Test Engineer (2) ACME Headquarters  (3)All (4)New |

**List 7-1-2.**

(1)Responsible software team member;  
(2) Organization site name  
(3) Part(s) of the *software item* or other *software* for which the *software team member* is responsible  
(4) Source of *software,* i.e., new, reused as is, modified reuse, *COTS*(5) If the *software* is of mixed source, then list the percentages of each type of source

**Diagram 7-1-2-2.**

**System Specification Tree**

DRONEswarm™

Ground Control Segment

Air Vehicle Segment

Swarm Simulation Segment

Ground to Air

Interface

\*This paragraph **shall** identify the full set of *software items* and other *software* for all *categories of software* for this project. This paragraph **shall** include for each *software item* and other *software*:

* + - 1. Name of the *software item* or other *software*;
      2. *System, subsystem,* and any other *components* to which the *software* belongs;
      3. *Category of software*;
      4. For each *software team member* responsible for all or part of the *software item* or other

*software*, identify the:

* + - * 1. Responsible *software team member*;
        2. Organization site name, i.e., same as in 7.1.1.a;
        3. Part(s) of the *software item* or other *software* for which the *software team member* is responsible;
        4. Source of *software,* i.e., new, reused as is, modified reuse, *COTS*; and
        5. If the *software* is of mixed source, then list the percentages of each type of source.\*
    1. Software team receiver-giver relationships. \*This paragraph **shall** show the *contractual* and intracorporation receiver-giver relationships among the *software team members,* including the *prime contractor*. An internal *software team member* receiver is the *software team member* requiring and receiving a *software item,* or other *software,* from one of the other *software team member*s. An internal *software team member* giver is the *software team member* supplying or giving the required *software item* or other *software* to the *software team member* that required the *product*. This paragraph **shall** be organized by internal *software team member* receiver with the *prime contractor* relationships first. If a single *software team member* has multiple sites performing software-related efforts on this project, then each organization site **shall** be treated separately. For each internal receiver-giver pair, this paragraph **shall** identify each *software item,* and other *software,* produced by the internal giver for the internal receiver. This paragraph **shall** include for each receiver-giver *software team member* relationship:
       1. Internal *software team member* receiver organization site name,
       2. Internal *software team member* giver organization site name, and
       3. List of name(s) of each *software item* and other *software* to be produced by the giver for the receiver*.\**
  1. Project resources. \*This paragraph **shall** be divided into the following subparagraphs to describe the resources to be applied to the project.\*
     1. Personnel resources. \*This paragraph **shall** provide the following items for the entire *software development* lifecycle and for each *software item,* and other *software*:
        1. Staff hours by software item. The estimated staff-loading for the project i.e., number of total personnel hours by month throughout the system development lifecycle, broken out as follows:
           1. For each *software team member:*

For each *software item* and other *software*,

For each other piece of *software*,

For each *build*, and

For the entire *software development* lifecycle; and

* + - * 1. For all *software team members* for the entire *software development* effort.\*
      1. Staff hours by responsibility. \*The breakdown of the staff-loading for the project, i.e., number of total personnel hours by month throughout the system development life cycle, broken out by responsibility (for example, management, *software engineering*, *software testing*, *software* configuration management, product *evaluation*, *software* quality assurance):
         1. For each *software team member:*

For each *software item*,

For each other piece of *software,*

For each *build*, and

For the entire *software development* lifecycle; and

* + - * 1. For all *software team members* for the entire *software development* effort.\*
      1. Number of personnel by skill level. \*For each *software team member*, a breakdown of the number of personnel by skill level of those personnel performing each responsibility used in paragraph 7.2.1.2:
         1. For each *software team member:*

For each *software item*,

For each other piece of *software,*

For each *build*, and

For the entire *software development* lifecycle; and

* + - * 1. For all *software team members* for the entire *software development* effort.\*
      1. Rationale. \*The rationale for the schedule estimates in paragraph 6.1 and the effort and head count estimates in section 7.2, including *software* cost and schedule estimation techniques, the input to those techniques (e.g., *software* size and *software* cost driver parameters and scale factors), and any assumptions made.\*
      2. Training. \*Description of the training required for each *software team member* organization site. Also a description of the training required for each new staff member.\*
    1. Overview of developer facilities. \*For each organization site, this paragraph **shall** list the development and test facilities, secure areas, and other site features to be used, as *applicable* to the *software development*, including which work will be performed at each facility, area, or other site feature. This paragraph **shall** include a schedule detailing when these items will be needed, developed or acquired, and validated.\*
    2. Acquirer-furnished equipment and information*. \**This paragraph **shall** list *acquirer-*furnished equipment*, software*, services, *documentation*, data, and facilities, as *applicable*, required for the *software development* effort. A schedule detailing when these items will be needed **shall** also be included.\*
    3. Other required resources, \*This paragraph **shall** list other required resources, including a plan for obtaining the resources, dates needed, and availability of each resource item.\*

1. Notes. \*This section **shall** contain any general information that aids in understanding this *document* (e.g., background information, glossary, rationale). This section **shall** be divided into the following paragraphs.\*
   1. Abbreviations and acronyms. \*This paragraph **shall** include an alphabetical listing of all acronyms, abbreviations, and their meanings as used in this *document*.\*
   2. Glossary. \*This paragraph **shall** include a list of any terms and their definitions needed to understand this *document*. Terms often used differently between organizations (e.g., acquisition phase names, *build*, block, development phase names, effectivity, evolution, increment, and iteration) **shall** be defined to avoid confusion. If the terms used are exactly as defined in the Software Development Standard (SDSMCS), they need not be redefined here.\*
   3. General information. \*This paragraph **shall** contain any other general information that aids in understanding this *document* (e.g., background information, rationale).\*
2. Appendices. \*Appendices may be used to provide information published separately for convenience in *document maintenance* (e.g., charts, classified data). As *applicable*, each appendix **shall** be referenced in the main body of the *document* where the data would normally have been provided. Appendices may be bound as separate *documents* for ease in handling. Appendices **shall** be lettered alphabetically (Appendix A, B, etc.).\*

END of SDP Template