COMOPTEVFORINST 5235.1A
Code 0T3
16 August 1996

COMOPTEVFOR INSTRUCTION 5235.1A

Subj: SOFTWARE DEVELOPMENT APPRAISAL METHODOLOGY

Ref: (a)COMOPTEVFOR Instruction 3960.1H

Encl: (1) Software Development Process Review (SDPR) Checklist
(2) Software Development Process Review (SDPR) Questionnaire
(3) Software Development Process Cross-Reference

1. Purpose. This instruction provides the Operational Test Director (OTD) with specific guidance in conducting Operational Test and Evaluation Force (OPTEVFOR) software development process appraisals.

2. Objective. The objectives of software development process appraisals described herein are to:

a. Provide the OTD a better understanding of the developer's software development process and techniques.

b. Identify possible problem areas early on which could lead to potential operational risks.

c. Develop a level of confidence in the software system and its capabilities prior to formal OT&E.

d. Improve the effectiveness of OT&E.

e. Ensure that immature software will not enter operational test and will not be deployed.

3. Cancellation. COMOPTEVFORINST 5235.1.

4. Background. Software has become a major part of today's systems. With an accelerated increase in the use of new software technologies, the ability to successfully evaluate the operational effectiveness and suitability of software has become increasingly more difficult. One significant contributor to Navy program acquisition success rates has been earlier operational tester involvement. To this end, OPTEVFOR will continue to establish and maintain liaison with program managers in order to ensure open and effective communications. OPTEVFOR's ground-breaking initiative of early presence and participation in the software development process has necessitated specific guidance on appraising software development techniques. New methods are required to help the OTD effectively identify strengths and weaknesses of a software development process from the operational testing perspective. This early and proactive approach will significantly improve OT&E effectiveness, as well as overall program success. Various approaches supporting early OPTEVFOR presence in the software development process are described in paragraph 7.

5. Overview. A good software development process is a key element of successful quality software products. This document presents a methodology for how an OTD can implement an effective appraisal process for software development programs. The OTD will be able to provide the Program Manager with the information required to make informed decisions which impact the program objectives.

This document contains three enclosures:

- Enclosure (1) is a set of guidelines that will assist the OTD with initiating the appraisal process.
- Enclosure (2) is a broad-based, representative questionnaire designed to help the OTD quickly assess software development plan issues, strengths, and weaknesses.
- Enclosure (3) is a cross-reference of some important software development process terminology that the OTD can periodically refer back to while appraising the system.

6. Scope. The material contained herein applies to software-dependent programs that are evaluated by OPTEVFOR, including joint programs where OPTEVFOR is the lead Operational Test Agency (OTA). An appraisal should also be considered as part of the accreditation process for models and simulations due to their software-intensive nature.

7. Approaches. OPTEVFOR early involvement includes establishing and maintaining liaison with program managers to ensure open and effective communications. Early OPTEVFOR presence in a software development process can occur in a number of forums, including integrated product teams (IPT), software requirement and specification review meetings, preliminary software design reviews, SDPR, and process Quick-Looks. The latter two are the principal methods by which the software development processes. OTD can appraise.

a. SDPR . The SDPR serves as an informal platform that enables the OTD and other participants to understand the software development process and identify potential program risks early on. Early involvement by the OTD in the system's software development process and clear understanding of the system's capabilities will help the OTD to design more efficient operational tests focusing on those identified risks. Additionally, any operationally related risks/concerns emanating from the review can be subsequently tracked for corrective actions. The guidance for implementing an SDPR is described in enclosure (1). The SDPR questionnaire, contained in enclosure (2), can be used during a 1-to-2 day period or in conjunction with related software development meetings listed in the paragraph above. All SDPRs will be coordinated through the Assistant Chief of Staff of the appropriate OPTEVFOR warfare division.

b. Quick-Look. A more structured appraisal that goes into greater detail than an SDPR is a Quick-Look. A Quick-Look is typically employed as a comprehensive program review after significant risks have been identified. The information obtained from the Quick-Look review are used to determine the software capability to support the fleet. The basic approach of a Quick-Look follows the framework of the Software Engineering Institute’s (SEI) Capability Maturity Model (CMM). The method follows a tailorable version of the CMM-Based Appraisal for Inter-

nal Process Improvement (CBAIPI) which assesses an organization's software development capability. This approach has also been adapted for use by the Office of the Secretary of Defense (Acquisition and Technology). Before a Quick-Look can occur, there must be mutual agreement between OPTEVFOR and the program office on the scope of the appraisal. The OTD does not normally initiate a Quick-Look. Quick-Looks are conducted by a team of approximately eight members and is normally led by an SEI CMM-IPI certified assessor. Quick-Look team membership includes those key members identified by an SDPR, representatives from OPTEVFOR (e.g., OTD and Code 0T3 office) and/or an independent software engineering support representative. The Quick-Look appraisal adopted by OPTEVFOR requires 4 to 5 days and primarily focuses on documentation review and interview sessions. The certified lead assessor and their team are guided by a CBA-IPI handbook which provides an objective approach to appraising the documentation as well as the interviews. A consensus from the team members on valid findings are then categorized by key process areas. The details involving team composition, agenda, and necessary team training all require advanced planning. The Assistant Technical Director for Software (Code 0T3) will maintain liaison with government agencies that have certified lead assessors, and serve as the coordinator for all Quick-Looks.

8. Reporting. The SDPR observations and Quick-Look findings typically consist of highly proprietary information that should be handled with strict confidentiality among those directly involved in the appraisal process. It is important for the OTD to realize that SDPR observations represent a snapshot of the developer's processes and, consequently, do not warrant any attachment of grades (e.g., color coding, PASS/FAIL criteria) to the findings. The manner and detail in which appraisal findings are reported remain at the discretion of the OPTEVFOR Assistant Chief of Staff for the respective warfare division. OPTEVFOR (Code 0T3) can provide the OTD with suitable reporting formats. The findings for any appraisal technique will not be attached to the OTD's final report, but rather should be used as part of the overall information and data collected to determine a system's operational effectiveness and suitability.

9. Lessons Learned. In order to maintain an updated data base of appraisal lessons-learned, the OTD will provide a copy of SDPR/ Quick-Look findings to OPTEVFOR (Code 0T3). The purpose of the data base will be to provide the OTD with generic agendas, typical questions to ask (categorized by software process areas), and OTD comments on how to improve the appraisal methodology.

10. Training. OPTEVFOR personnel will continually be offered a number of training opportunities relating to the software development process through a series of on-site training programs. OPTEVFOR (Code 0T3) will act as a principle training coordinator for the subject matter listed below and will endeavor to provide the OTD with the necessary training to effectively participate in software appraisals. However, the Assistant Chief of Staff for each OPTEVFOR warfare division is ultimately responsible for ensuring his/her OTDs are adequately prepared before undertaking any software development process appraisal initiative.

a. Software Development Process training includes, but is not limited to:

(1) OPTEVFOR OTD Course

(2) Introduction to the SEI's CMM

(3) SDPR Questionnaire

(4) Terminology and practices

(5) SDPR/Quick-Look preparation

(6) SDPR/Quick-Look lessons-learned

(7) Software metrics

(8) Review of software-related directives and MILSTD documents (e.g., DOD and SECNAV directives, MIL-STD 498, 882, ISO 9000 series, etc.)

b. A variety of informative lectures that focus on software-related issues will be presented by OPTEVFOR (Code 0T3) and by guest lecturers. Topics may include, but are not limited to:

(1) Software Program Managers Network (SPMN) Software Methods for Managers (to include risk planning considerations)

(2) SPMN Software Survival

(3) SPMN Software Test and Integration

(4) SPMN Best Practices

(5) Emerging Software Technology and Issues (e.g., Year 2000 compliance)

11. Reference Material. OPTEVFOR (Code 0T3) maintains a reference library of software publications and instructional videotapes. OTDs are encouraged to use these resources and the expertise of OPTEVFOR (Code 0T3) to answer any questions they may have regarding software development practices.

12. Responsibilities. The OPTEVFOR Technical Director (Code 00T) will provide the OTD with the necessary staff, support, and training upon request to effectively prepare and participate in the various software appraisal activities described herein. Code 0T3 is responsible for reviewing and implementing changes, as necessary, to this instruction on an annual basis.

13. Summary. Early involvement by OPTEVFOR in the software development process can significantly improve OT&E by identifying and reducing potential operational risks. Once these risks are identified, the process of risk reduction through a collaborative risk mitigation plan can begin. The OTD can then develop a level of confidence in the system and its capabilities, which will translate into a more effective test plan. More efficient test planning focuses mainly on potential operational risks, thereby saving valuable test resources. Overall, mutual trust between

OPTEVFOR and the program offices will promote a stronger acquisition team. The ultimate pay-off of early OPTEVFOR presence is systems that are delivered to the fleet on schedule and that perform to mission requirements the first time. To this end, we are dedicated to understanding and enhancing software development processes whenever possible.

14. Point of Contact. For questions about software development appraisals, contact Ms. J. Huynh, Assistant Technical Director for Software Development, who can be reached at COMM (757) 444-5546 or DSN 564-5546, ext. 3281.

S. H. BAKER

Distribution: (COMOPTEVFORINST 5216.2M)
Lists I, III, IV & V
CNO (DOT&E, N8, N86, N7, N88, NO91, N912)
COMNAVSEASYSCOM (SEA-91T)
COMNAVAIRSYSCOM (AIR 1.6)
COMSPAWARSYSCOM (SPAWAR-07-02)
AFOTEC
OPTEC
MCOTEA

SOFTWARE DEVELOPMENT PROCESS REVIEW (SDPR) CHECKLIST

1. An SDPR is a relatively simple and nonintrusive method of quickly identifying weaknesses in a software development process. An OTD can isolate many inherent weaknesses in a matter of hours by using the SDPR Questionnaire [(Enclosure (2)] as his/her primary review tool. Initial findings can be recorded and compared with subsequent SDPR efforts. Since SDPR preparation involves coordination between a number of key personnel and activities, the OTD is strongly encouraged to follow the guidelines contained in this checklist as closely as possible. Additionally, the use of an OTD Journal to record SDPR findings is recommended. The journal can serve as a quick and accurate reference during test planning and OTD personnel turnovers.

2. This checklist provides the OTD with basic procedures which, when followed, will help to ensure that the SDPR proceeds as smoothly as possible. OPTEVFOR Code 0T3 and/or the divisional software support representative can provide assistance with completion of any portion of this checklist. Recommended OTD actions include:

a. Determine precisely what it is that you expect to accomplish by conducting an SDPR. Without a clear goal in mind, SDPR effectiveness could be reduced.

b. Discuss your SDPR intentions with your divisional chain of command, including your ACOS and respective divisional software support representative.

c. Discuss your desire to conduct an SDPR with the program office. Mutual agreement must exist prior to initiating an SDPR, since the software development and software support activity will be participating.

d. Determine when an SDPR can be scheduled. Ideally, the SDPR should occur during phase I/II of the acquisition process. However, each program is unique, and OTD turnover may also preclude scheduling a review in the early stages of developmental testing (DT). (Note: In the event a program does not have a dedicated DT period (i.e., special contract), then the OTD should make arrangements through the program office to plan an SDPR as early as possible)

e. Create a draft SDPR agenda and distribute to the program office for review. Ensure that your agenda includes time to observe some of the processes in action (e.g., lab tours). An effective SDPR agenda includes the OTD meeting with representatives from the program office, software developer, and respective Software Support Activity (SSA).

f. Distribute the final agenda prior to commencing the SDPR. Ensure sufficient time is allotted for recipients to review the agenda prior to the visit.

g. Review the SDPR Questionnaire in Enclosure 2 to ensure you understand to whom each question will be directed, what the questions are asking, and what responses to expect for each question.

h. Maintain the responses to the questionnaire in an OTD Journal, per the Operational Test Director's Guide (reference (a)).

i. Ensure SDPR findings are accurately recorded in an acceptable format that will facilitate easy retrieval of data at a later date for comparison with previous or subsequent SDPR efforts.

j. Debrief your SDPR findings to the developer and program office as a matter of courtesy and in the interest of promoting the software improvement process.

k. Debrief your respective divisional software support representative to discuss your findings and any follow-on actions necessary.

SOFTWARE DEVELOPMENT PROCESS REVIEW (SDPR) QUESTIONNAIRE

1. Demonstrate how requirements are traced from the Statement of Work (SOW) to individual modules.

2. Demonstrate how requirements changes are controlled.

3. Does a documented configuration management process exist for this project? Demonstrate the change control process.

4. Is there a Risk Management Plan in place? Demonstrate how risks are identified and resolved? Has a Risk Analysis been completed?

5. Does this project employ a formal metrics tracking plan? Provide some examples of program-specific metrics.

6. Can you demonstrate how the metrics program is tied to the risk management plan?

7. Provide an illustration of your scheduled review process and explain how action items from reviews are tracked and closed.

8. Describe your software testing plan. Does the plan include provisions for utilizing hardware-in-the-loop (HIL) facilities? Explain how all key external system interfaces will be tested.

9. Describe how built-in-test software will be tested prior to release.

10. Provide an example of how software safety and security issues are identified and resolved.

11. Describe the source of your software. Is it COTS, GOTS, NDI, reuse, new or modified?

Enclosure (2)

SOFTWARE DEVELOPMENT PROCESS CROSS-REFERENCES

Built-in-Test (BIT) Testing. BIT circuitry is designed to generate signals that give some indication of main system circuitry, hardware, or software performance. Particular attention is paid to equipment inputs and outputs. Failure Modes, Effects, and Criticality Analysis (FMECA) shows what happens when a component fails and shows whether or not BIT circuitry would detect the failure. Failures can also be inserted into engineering prototype modules to determine if the BIT circuitry provides the desired signals. BIT is embedded in the hardware design of modules and the software/firmware design of various code packages. It is programmed in whatever language is being used for the other software/firmware that is running on the processor on which the BIT code will run. BIT software is tested in a manner consistent with the test plan for system software during the development phase, including BIT software requirements verification. Once the prototype system is integrated, the true test of BIT takes place with the use of prefaulted modules. A well-designed BIT test uses prefaulted modules, tests system design and verifies that the end requirements are met.

Configuration Management (CM). Provides management (both government and contractor) an identification, control, and accounting system for changes to the software baseline. CM is central to controlling the development process. Changes to formal system baselines can directly impact both cost and schedule. With formal control, any changes to the base-lined system must be approved by the authority responsible for system integrity as defined in that baseline.

DoD Software Acquisition Best Practice Initiatives. This initiative was established to bring about substantial improvements in productivity, quality, timeliness, and user satisfaction by implementing best practices as a new foundation for DoD software management. These practices are focused upon effective management processes and techniques for finding defects as they occur, eliminating excessive and unnecessary costs, increasing productivity, and other beneficial effects.

Metrics. Metrics are measurements of various program attributes that provide a way for a manager to understand the status of a program and whether that program is in trouble. Metrics are categorized as either management, quality, or process metrics. *"**Management metrics** support forecasts of future progress, early trouble detection, and realism in plan adjustments. **Quality metrics** measure product attributes affecting performance, user satisfaction, supportability, and ease of change. **Process metrics** measure organizations, tools, techniques, and procedures used to develop and deliver software products."*[1]

Enclosure (3)

[1] Guidelines for Successful Acquisition and Management of software Intensive System, vol. 1, Feb. 95; Dept. of the Air Force.

Peer reviews. Peer reviews are an industry-proven, verified and documented, successful method for removing defects and reducing development costs. When developers know their work will be critically examined by the government and/or their peers, they are motivated to work more carefully, either to avoid embarrassing mistakes or through pride in exhibiting a quality product. *"They can eliminate approximately 80 percent of all software defects. When combined with normal testing, they can reduce the number of latent defects in fielded software by a factor of 10."*[2] Peer reviews verify the presence or absence of the quality attributes that satisfy the requirements.

Requirements Change Control. Ideally, requirements are fully identified before the Statement of Work is written. In practice, this is almost never the case. In large, complex software-intensive systems, requirements continually evolve throughout the system's life cycle. Requirements must constantly be managed through a formal change control program because they can significantly impact total system development cost and schedule.

Risk Management. Formal process of actively assessing, controlling, and reducing software risk on a routine basis. Risk management actions include:

1. Identify. Search for and locate risks before they become problems that adversely affect the program.

2. Analyze. Process risk data into decision-making and information.

3. Plan. Translate risk information into decisions and actions (both present and future) and implement those actions.

4. Track. Monitor the risk indicators and actions taken against risks.

5. Control. Correct for deviations from planned risk actions.

6. Communicate. Provide the visibility and feedback data internal and external to the program on activities and current and emerging risks.

Requirements Traceability. Process of translating user specified requirements into derived (or implicit) requirements necessary for the solution to be turned into code. This process guarantees the final system meets original user requirements. This tracking process prevents requirements from being "lost" during the development process, with a resultant loss in system performance or function. Missing requirements may not become apparent until system integration testing, where the cost to correct this problem is exponentially high.

Software Development Plan (SDP). An intrinsic part of the software development process is the SDP, which records all necessary planning and engineering information for the production of the software. Separate plans, referenced in the SDP, are also developed for software quality and configuration management.

[2] ibid.

SEI CMM. A framework developed by the SEI that describes the key elements of an effective software development process. Primarily, the CMM describes an evolutionary improvement path for that process while emphasizing process improvement within an organization. Process improvement is achieved through a focused and sustained effort towards building a repeatable process infrastructure of effective software engineering and management practices. The CMM has five maturity levels (numbered one to five) that define an ordinal scale for measuring the maturity of an organization's software process and for evaluating its software process capability. Level one denotes a software process that is both ad hoc and chaotic. Few processes are defined, and success depends on individual effort. Level five indicates that there exists continuous process improvement, enabled by quantitative feedback from the process and from innovative ideas and technologies.

Software Program Managers Network (SPMN). The SPMN, sponsored by The Office of the Assistant Secretary of the Navy (ASN) for Research and Development (RDA), was formed to facilitate sharing of successful software program management techniques as opposed to articulating official guidance. Its focus is on the exchange of useful lessons-learned, insights, and tips from real people with real experience and success. Videotapes, handbooks, and specialized seminars developed by the SPMN are available to anyone involved with software acquisition and testing and can be obtained by coordinating through OPTEVFOR (Code 00T3).

Software Safety. The objective of a software safety program is to ensure that the software does not perform unintended functions that pose hazards to people, equipment, the environment, or the mission. It has been recognized that the only means to limit the occurrence of software faults is to define requirements to build safety features into the software. The focus is on the planned interactions between hardware and software. DoD safety standards include MIL-STD882B/C (System Safety Program Requirements) and MIL-STD-1547 (System Safety Program for Space and Missile Systems).

Software Security. Security is a crucial aspect of software development that is often overlooked. Consequently, many weapons, satellite communications, logistics, air traffic control, and global financial systems are considered "soft" because they are vulnerable to software hackers. Failure to plan for software security can prove catastrophic. As in risk abatement, not planning for security up-front and having to address these issues after development is underway (or the system is already deployed) can severely impact the cost and schedule of software development.