

3.2.2 Software Engineering Specific Skills

3.2.2.1: Skill Sets Developed: This set of skills for the Software Functional Band Team has been revised to reflect the specific skills that need to be highlighted as a software engineer within the STRICOM Engineering Directorate.

Software Engineering Specific Skills:	Section:
Operating Systems (OS)	3.2.2.3.1
Programming Languages (PL)	3.2.2.3.1
Software Architecture (SA)	3.2.2.3.2
Software Configuration Management and Quality Assurance (SCM/QA)	3.2.2.3.3
Software Cost Estimation (SCE)	3.2.2.3.4
Software Development and Support Process (SDSP)	3.2.2.3.5
Software Development Methods and Tools (SDMT)	3.2.2.3.6
Software Measurements and Analysis (SMA)	3.2.2.3.7
Software Requirements Analysis (SRA)	3.2.2.3.8
Software Resource Estimation (SRE)	3.2.2.3.9
Software Test Engineering Methods (STEM)	3.2.2.3.10
Trusted Software Development Methodology (TSDM)	3.2.2.3.11

3.2.2.2 Personnel Assigned as SME: The personnel assigned to the Software Functional Band Team are assigned SME skill areas in order to insure that training and elements of the skills are kept up to date in this document. The review of these skill areas will be accomplished by team and will be made available for editing of this document.

3.2.2.3 Skill Qualifications and Developmental Process: This section will delineate the skill qualification and development process for skill acquisition and maintenance from entry level through subject matter expert (SME). The process will show the education, tasks and OJT needed to maintain mastery of the skill.

3.2.2.3.1 Operating Systems and Programming Languages: Although the areas of skill are combined into one section it is believed that if an Engineer is to be successful in one that they must have significant knowledge of the other associated skill. Therefore the two skills are treated as one skill set. This section will address each skill separately for ease of clarification of needed training and OJT requirements.

Operating Systems (OS):

Description of Skill:

The engineer shall have a working knowledge of modern Operating System concepts, principles and implementations for the commonly utilized computer systems in the modeling and simulation domain. Skill emphasis is on having a broad understanding of the various aspects of Operating Systems as well as a user-level ability to perform fundamental computer system tasks.

Basic/Minimum Knowledge:

- What is a Portable Operating System Interface for Computer Environments (POSIX) compliant OS
- What is meant by the term: 16-bit, 32-bit, 64-bit OS
- What is meant by the term multiprogramming
- OS Development Scheduling
- Schemes for non-real-time Operating Systems
- Schemes for real-time Operating Systems/applications
- Thread states: running, blocked, & ready
- Preemptive versus non-preemptive

- Differences between Unix, Win95, & WinNT
- Relationship between threads and processes
- Thread/process priority
- Asymmetric vs. symmetric multiprocessing (SMP)
- To what level do Win95, Win98, WinNT, LINUX, and Unix types of Operating Systems support multiple processors
- Virtual memory
- Methods available for processes to share data (shared memory,
- Dynamic-link libraries (Win95/NT): define; describe pro's and con's
- How operating systems address error handling
- OS File systems operation
- Purpose and usage of OS
- Impact of block-size on storage capacity; performance
- Largest file size for various Operating Systems
- Access restriction features of various Operating Systems
- Swap space
- Screen Interfaces
- Command Line Interfaces
- Graphical User Interfaces (Windows, X-Windows/Motif)
- Daemons
- User-level and root/administrator level accounts and uses of each

Training or Work Assignments Contributing to Basic Knowledge:

- Various training classes on Operating Systems (Entry Level) to gain familiarity with each Operating System
- Training in theory/concepts of Operating Systems (Theory behind Operating Systems. Not tied to a specific Operating System)

Developmental Path:

Continued Training in various Operating Systems to gain specific/detailed knowledge of various Operating Systems. (This will also help in determining which Operating System may be the best choice for a specific application/project.)

Sustainment to Maintain Expertise:

- Training (Hands on preferred) to keep up with new versions/changes in various Operating Systems along with any new Operating Systems that may arise.
- Knowledge sustainment: At least 30-min per week of reading to keep up with new changes and continued learning of other Operating Systems.
- Ability sustainment: At least one 1-hour hands-on session per month to gain expertise/experience in using Operating Systems (not just theory).

Programming Languages (PL):

Description of Skill:

The engineer shall have a working knowledge of modern standardized Programming Languages. The engineer should know concepts, principles and implementations for the commonly utilized Programming Languages in the training, modeling and simulation domain. Skill emphasis is on having a broad understanding of the various aspects of Programming Languages and the differences/advantages between them. As the military increases their reliance on computing resources to dominate the battlefield of the future, engineers engaged in the development of all types of military equipment must increase their knowledge of software programming languages and the impact these languages can have on the success of their project.

Basic/Minimum Knowledge:

- Understand the need for a diverse set of software programming languages to solve computational problems.
- Understand the use of a software development metrics program and technical documentation (i.e. trade studies) to support software programming language selection.
- Understand the impact of software programming languages on a project's level of effort, costs, staffing and training requirements.
- Understand the advantages and disadvantages (strengths and weaknesses) of different software programming languages.
- Knowledge of COTS (commercial-off-the-shelf) software, re-use driven approaches, certifications (ISO 9001 and SEI capability maturity model), and reengineering.
- Must possess the skills, knowledge and abilities to advise, assist, inform and educate the other members of a project team in all matters related to programming languages.
- Ability to communicate clearly, in non-technical terms, in oral and written form, on matters related to programming language selection, impacts, issues and recommended solutions.
- Ability to review and understand software code.
- Ability to review and understand technical papers and documentation (i.e. trade studies, CDRLs) related to programming languages.
- Ability to assess the impact of software programming languages on a project's level of effort costs, staffing, and training requirements.
- Ability to discuss the advantages and disadvantages (strengths and weaknesses) of different software programming languages.
- Knowledge of resources and available assets related to programming languages.

Training or Work Assignments Contributing to Basic Knowledge:

TASKS to gain basic knowledge:

- Demonstrate the ability to read and understand software code written in two software programming languages.
- Prepare a brief synopsis of what a software program (or part of a program) does during its execution.
- Ability to identify key areas and constructs within the software code.
- Explain how choices of programming language tools affect the level of effort and cost of software projects.

Developmental Path:

- A one or two semester sequence in fundamental programming concepts including common data processing algorithms, constructs and data structures found in most programming languages.
- A one or two semester sequence in a particular standardized programming language focusing on implementing the common data processing elements above utilizing the syntax and semantics of the language. This course should include the use of software engineering tools.
- A one semester sequence in a second programming language focusing on the syntax and semantics of the language. This course should include the use of software engineering tools.
- A one semester sequence in an advanced programming language topic i.e. language design, compiler construction, tool development.

Training or Work Assignments Comprising Developmental Path:

- Initial assignment to a project under the mentoring of a senior software engineer for a period of 3-12 months.
- Progressive assignment to projects with increasingly complex software.
- Assignment to projects utilizing any of the programming languages studied or utilizing a combination of the languages studied.
- Assignment as a mentor to a junior software engineer on a software intensive project.

Sustainment to Maintain Expertise:

- Self-education utilizing CBT, web-based tutorials, and self-study.
- Attendance at a conference discussing programming languages once every 5 years.
- Periodic monitoring of electronic newsgroups and mailing lists dedicated to programming language issues.
- Training in programming languages that the employee does not have prior expertise in.
- Training in development of applications using a standardized programming language (hands-ON training/experience.)

3.2.2.3.2 Software Architecture (SA):

Description of Skill: There is no universally accepted standard for software architecture. It is still a developing field in software engineering. A good description of software architecture comes from Bass, Clements, and Kazman. Software Architecture in Practice, Addison-Wesley 1997: which states:

- The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.
- By "externally visible" properties, we are referring to those assumptions other components can make of a component, such as its provided services, performance characteristics, fault handling, shared resource usage, and so on. The intent of this definition is that a software architecture must abstract away some information from the system (otherwise there is no point looking at the architecture, we are simply viewing the entire system) and yet provide enough information to be a basis for analysis, decision making, and hence risk reduction.

The primary skill is the ability to analyze software architectures to determine the ability to satisfy functional and interface requirements, and to assess the capabilities and tradeoffs for the achievement of qualities such as performance, availability, and survivability. It is cost effective to try to determine, before a system is built, whether it will satisfy its desired qualities. STRICOM software engineers should be able to

identify or develop risk mitigation methods that can be done early in the software development life cycle when it is relatively inexpensive to change architectural decisions. It is also important to be able to analyze the architecture of the countless legacy systems in existence. Unfortunately, the architectures of legacy systems are frequently undocumented or existing documentation is inaccurate due to the unavoidable architectural drift and erosion making analysis impossible. Software Engineers should have the necessary skill to reconstruct architectures from source code and to check the conformance of as-built systems to their documented architectures.

Software Engineers should possess the skills needed for participating in the development and validation of the technology and techniques necessary for analyzing software architectures, specifically: attribute-specific models, representation approaches, analysis methods, reconstruction and conformance tools and techniques. Software Engineers should be capable of using tools and methods for representing varying views of software architectures.

In summary, Software Engineers who are skilled in SA should be able to:

- Establish, implement, and transition validated techniques for analyzing the effect of software architectural decisions on selected product quality attributes.
- Establish, implement, and transition validated techniques for reconstructing the architecture of legacy systems and for determining the conformance of as-built systems to defined architectures.
- Establish, implement, and transition validated techniques for representing software architectures
- Promote understanding of software architecture and architecture analysis.

Basic/Minimum Knowledge:

- Basic knowledge of what software architecture encompasses.
- Knowledge that any design involves trade-offs. Modifiability affects performance, security affects modifiability, scalability affects reliability, and everything affects cost.
- Prescriptive method implicitly or explicitly assumes that some of these qualities are more important than others do and guides users toward the maximization of that goal.
- Elementary knowledge of techniques used for designing, building, and evaluating software architectures.
- Software architecture is the development product that gives the highest return on investment with respect to quality, schedule, and cost.

Training or Work Assignments Contributing to Basic Knowledge:

- The employee should have the opportunity to work on a software intensive project to gain experience working with a real world software architect and dealing with all of the issues that accompany managing the development effort.
- There are numerous Software Architecture Publications that possess information that will give the employee basic knowledge of software architecture and the importance a well designed architecture plays in a development project. (The SEI website has a list of publications dealing with software architectures.)
- SEI has a good series of slides that describe software architecture. It is called: "What is Software Architecture? And Why Do I Care?". It can be downloaded (Postscript file, can be viewed with Ghostscript) at: ftp://ftp.sei.cmu.edu/pub/sati/Papers_and_Abstacts/what_is_sw_arch.slides.ps
- Attending SAM 101 and SAM 201 Defense Acquisition University Courses will give the employee basic knowledge of software architectures along with numerous other software related topics.

Developmental Path:

- Skills needed to be proficient in the area of Software Architectures:
 - All of the above mentioned skills and abilities.
 - Knowledge that reusable components are best achieved within an architectural context. But components are not the only artifacts that can be reused. Reuse of architecture leads to the creation of families of similar systems, which in turn leads to new organizational structures.
- Comprehensive knowledge of sets of techniques for designing, building, and evaluating software architectures.
- Understand techniques for quality requirements in the context of an architecture and for building architectures that meet these quality requirements.

- Knowledge of architecture description languages as a means of describing and validating software architectures and techniques for analyzing and evaluating an architecture's fitness for its purpose.
- Knowledge of a number of different architectural tools (layering, multiple views, patterns, blackboards, and so forth) and techniques (analysis methods, integration strategies, engineering principles).

Training or Work Assignments Comprising Developmental Path:

- The employee should have experience as the software engineer on a large software intensive system to gain experience with managing a program dealing with a large software architecture.
- The employee can gain more extensive knowledge of software architectures by reading many, more advanced publications dealing with software architectures. SEI has a good website with a list of many of these publications. It can be found at: <http://www.sei.cmu.edu/architecture/projects.html>
- Attending the Advanced Software Acquisition Management (SAM 301) Defense Acquisition University Course will give the employee information on software architectures along with program management form software prospective.
- Attend Events (Conferences, Workshops, etc.) dealing with Software Architectures to discuss software architecture issues with colleagues. This will give the employee up to date information on software architecture knowledge and practices along with lessons learned from real world projects. A web address showing various Events concerning Software Architectures is at: <http://www.sei.cmu.edu/architecture/events.html>

Sustainment to Maintain Expertise:

The field of Software Architecture is still developing and therefore is very dynamic. To maintain a level of expertise, the employee must keep up with the latest developments in tools and techniques dealing with software architectures. This can be accomplished by keeping up to date with new publications dealing with software architectures. Also, attending numerous events dealing with new concepts in the area of software architectures will give the employee an up to date view of what is happening in the field of software architectures.

3.2.2.3.3 Software Configuration Management and Quality Assurance (SCM/QA):

Software Quality Assurance (SQA):

Description of Skill: SQA is a special discipline of the overall QA effort. SQA is a planned and systematic pattern of all actions necessary to provide confidence that software is adequately developed, tested and supported throughout its life cycle by:

- Establish checkpoints and procedures to validate, and verify software activities of others
- Create products and services that conform to the established software quality requirements and common recognized practices

SQA activities involve administrative and technical work concerned with monitoring, controlling and maintaining the quality and reliability of hardware, software, integration, services or processes. A person that normally assigned to this task ensure that program software quality aspects are adequately considered in pre-award (i.e., software development plan), design reviews (i.e., code walkthrough), configuration audits (i.e., physical and functional), quality management system audits (i.e., metrics reporting), production readiness reviews (i.e., checkout), etc

Basic/Minimum Knowledge:

- Has strong math background and analytical ability.
- Has keen observation and trouble-shooting skills.
- Ability to review software code.
- Proficient in the use of software development tools to gain visibility into the software development process.
- Capable of establish monitoring and control activities (i.e., metrics) and reporting findings through an independent chain of command.
- Understand the various types of specification requirements and design parameters involved in software development, and testing. Understand related quality assurance policy, procedures and responsibilities and their implementation. Resolve issues without provoking resentment relating to software

processing and maintaining a set of standardization documents to document software development progress.

Training or Work Assignments Contributing to Basic Knowledge:

- Courses in programming language(s), a degree in computer engineering or advanced mathematical /statistical field. Two introductory courses taught by the Joint Logistics Commanders, Joint Group on Systems Engineering includes PSM and Risk Management. Other courses will be added as they are developed.
- Attend workshops, seminars, or conferences **focusing on SQA**. These include the SISO conference, The ITSEC conference and other vendor and corporate conferences as available.
- Courses in maximize Rate-of – Returns by implementing SQA.
- DAU courses for entry-level applications include PQM 101,103, and 104.

Development Path:

Activities need to establish the mid-level development activities are:

- Establish or assist in the establishment of the SQA function at STRICOM.
- Establish or implement SQA functions as part of the overall software development process at STRICOM.

Training or Work Assignments Comprising Development Path:

- Job assignment at Contractor’s plant working with the DCMC personnel.
- Job assignment at other MACOM where SQA organization is still part of the infrastructure.
- Mid level DAU courses include: PQM 201,202,203

Sustainment to Maintain Expertise:

- Present findings on the benefits of SQA.
- Serve as the SME on the subject of SQA.
- DAU course PQM 301 and certification in the PQM arena

Software Configuration Management (SCM):

Description of Skill: SCM is the application of technical, administrative and surveillance effort to gather and maintain the functional and physical baseline of a software item, (Configuration Control), and to record and report change processing and implementation status (Configuration Status Accounting). It also includes supporting periodic quality assurance/technical reviews and audits (Configuration Audits).

Basic/Minimum Knowledge:

- Understand basic SCM practices and be able to implement establish SCM policies.
- Change control methods/procedures such as configuration control boards (CCBs) and the documentation necessary to support.
- Be organized and be proficient in using the CM tools.

Training or work Assignments Contributing to Basic Knowledge:

- Attendance at seminars/workshops on the latest CM methods and tools.
- Proficient in Personal Computer, Database, Data-Mining and Data Management.
- DAU courses to include: LOG 101

Developmental Path:

- Establish or maintain SCM database for rapid/seamless input and retrieval of data.
- Establish and implement STRICOM-wise SCM process and procedure.
- DAU courses to include LOG 201, 203, and 204

Sustainment to Maintain Expertise:

- Provide periodical refinement of the SCM methods, process, procedure and products to meet/exceed the demands of STRICOM users.
- Joint Logistics Command courses on CM and Practical Software Management.

3.2.2.3.4 Software Cost Estimation (SCE):

Description of Skill: Despite the terminology, software cost does not refer directly to the dollar figure associated with software development. Software Cost Estimation (SCE) consists of the following three elements:

- Manpower loading is the number of engineering and management effort personnel allocated to the project as a function of time.
- Effort is defined as the engineering and management effort required to complete a project, usually measured in units such as person-months. The types and the levels of skills for the resources will come into play here.
- Duration is the amount of time (usually measured in months) required to complete the project.

A SCE process is the set of techniques and procedures that an organization uses to arrive at a software cost estimate. Generally there is a set of inputs to the process (e.g., system requirements) and an output of effort, manpower loading, and/or duration. The knowledge of how much each of the above elements influences the SCE process output will influence the final cost estimate figure. Deciding which factors to include and combining them to arrive at the estimate make up the software cost estimation process.

Basic/Minimum Knowledge:

- Understand the advantages and disadvantages of different software development cost and schedule estimation methods.
- Use of a software development metrics program to support estimation
- Understand the importance of new costing concepts and paradigms
- Knowledge of COTS (commercial-off-the-shelf) software, re-use driven approaches, certifications (ISO 9001 and SEI capability maturity model), and reengineering

Training or Work Assignments Contributing to Basic Knowledge:

The work assignments related to this skill are: assignment to a development al project as an assistant software engineer for the gathering of requirements and cost coordination in contract negotiation, assignment of to a fielded system for CCB support and estimation of maintenance and sustainment costs for a major training system that is currently used in the field.

Training should include courses at the local University level and industry provided training that is general enough in nature to cove the major elements of SCE.

Developmental Path: Through participation in such activities as a Requirements Analysis phase, the engineer will be able to understand the scope of the program from its infancy. Then he/she will use this information along with the other factors listed above (manpower, effort and duration) to calculate the effort required to complete the program using some sort of computer model such as the Constructive Cost Model (COCOMO) . This information will aid the engineer during activities such as a Source Selection Board to make an informed decision on all the bidders proposals.

A formal approach would include taking the following DAU courses:

- BCF 101 – Basics of Cost Analysis - Fundamentals of Cost Analysis enables DOD personnel new to the cost estimating field to prepare materiel system life cycle cost estimates. The course covers DOD policies governing these estimates and the techniques used in their preparation. Topics include a statistics review, regression analysis, learning curves, risk analysis, software cost estimating, exploratory data analysis, inflation adjustments, cost as an independent variable (CAIV), analysis of alternatives (AOA), contract cost structure, earned value, cost estimation for budget preparation, and economic analysis. Students apply the techniques they learn in a series of case studies.
- BCF 206 – Cost Risk Analysis- Cost Risk Analysis prepares cost analysts to model the cost risk associated with a defense acquisition program. Topics covered include basic probability

concepts, subjective probability assessment, goodness-of-fit testing, basic simulation concepts, and spreadsheet-based simulation. Practical exercises, a small-group workshop, and a capstone article review reinforce techniques taught.

- BCF 208 – Software Cost Estimating - Software Cost Estimating is primarily for practitioners of software cost estimating. The course is designed for cost analysts and others whose duties should include estimating the cost of software development efforts or reviewing such estimates. Topics in the course include software life cycle management, architecture, interoperability, software development paradigms, software design approaches, metrics, capability evaluations, risk analysis, software reuse, open systems, function points, and software cost estimating models. Two software cost estimating case studies allow students to apply the course material.

Training or Work Assignments Comprising Developmental Path:

The software engineer should lead increasingly complex software intensive engineering efforts for programs and should develop SCE packages based on knowledge from the formal training packages. The SCE should know how to use and present concepts resident in CASE tool packages which measure and analyze SCE concepts and to work with contract personnel to insure the best development methods are being used for cost-benefit requirements. The SCE engineer should use or create briefing/review material which describes methodologies and provides engineering input data for program directors decision making process.

Sustainment to Maintain Expertise: The SCE engineer must learn and operate the latest versions of software estimating tools like COCOMO. Attend conferences/symposiums, which provide the engineer with knowledge of the latest information available in SCE analysis. Maintain proficient in SCE methodologies by membership in professional organizations such as INCOSE and IEEE and to read technical journals and publications concerning SCE and participate in technical conferences and presenting papers at professional organizations meetings for review by peers.

3.2.2.3.5 Software Development and Support Process (SDSP):

Description of Skill: Provides a focused approach for development of a software product. Performs the role of integrating the technical disciplines to achieve the customer's objectives. Provides process definition and improvement in all areas of software development, including: Requirements Management, Project Planning and Tracking, Project Estimation, Software Quality Assurance, Functional and Design Specification, Configuration Management, Testing, Change Management, Verification and Validation Efforts, and Post Deployment Support

Basic/Minimum Knowledge: Basic knowledge includes the following capabilities:

- Ability to listen and learn (not be a bag of hot air)
- Ability to give credit where credit is due and not always seek one's own credit
- Ability to do hands on work
- Ability to work in a team
- Patience in dealing with other organizations to foster cooperation
- Capability to Analyze Candidate Software Solutions
- Ensure Software Quality
- Familiarity with CASE Tools and Programming
- Domain Expertise
- Coordinate with Contractors and Customer
- Derive and Allocate Software Requirements
- Manage Software Configurations
- Participate in Definition of Organization's Software Engineering Process
- Evolve Software Architecture
- Manage Risk Integrate Disciplines

- Monitor and Control Software Technical Effort
- Manage Software Product Line Evolution
- Integrate Software
- Plan Software Technical Effort
- Manage Software Engineering Support Environment
- Understand Customer Needs and Expectations
- Provide Ongoing Knowledge and Skills in Simulation Domain
- Verify and Validate System
- Coordinate and Participate in Software Testing

Training or Work Assignments Contributing to Basic Knowledge:

Training should include participation in professional organizations such as SISO, IEEE, I/ITSEC and other organizations that deal in the SDSP. SEI and other industrial organizations and associations provide a great deal of SDSP related conferences and symposia for development of these skills. Training provided by contractors that are prime on major command projects is a very good source of instructional material. University courses from computer science departments on initial development and logistics support are valuable. Courses in Software Logistics Support are being offered in the industrial engineering departments.

Assignment as a Systems/Software Engineer here at STRICOM. Types of assignments are typified by the projects listed below:

- AC-130U Navigator/Fire Control Officer (NAV/FCO) Test Bed
- Close Combat Tactical Trainer (CCTT)
- Federation Test System (FTS)
- Modular Semi-Automated Forces (ModSAF)
- Semi-Automated Forces (SAF)

Developmental Path:

The development path should include duties from initial development through lifecycle support of an item. The developmental path is broken up into 3 sections:

Entry level: At this level the individual should be a part of a developmental item team. The person should be responsible for configuration Control of documentation and ECP items. The engineer should be provided the opportunity to review for completeness the code and the supporting documentation and be provided oversight and guidance in the conclusions offered. The engineer should be given opportunities to brief contractor staff on decisions and actions to be taken. The engineer should be allowed to attend professional meetings and present briefings on project topics.

Mid Level: The engineer should be given the responsibility for the overall vision of the project and provide the PD with the technical aspects of of the project from cradle to grave. They should prepare estimates for cost (manpower) and schedule. They should be responsible for accurate and timely documentation support and should be able to answer issues of both higher command and support contractor support.

SME should be made responsible for overall technical aspects of item lifecycle from development throughout usage. The engineer should be fully aware of all documentation, briefings, issues, ECP and support issues tied with the project. The SME is responsible to the PM for complete knowledge of all technical issues and solutions.

Training Path:

The training of the engineer is an on going process. The process is divided into the three levels as depicted above.

The entry-level engineer should take at least one of the following three DAU certification paths: SPRDE (SYS 101), Testing (TST 101) or Logistics (LOG 101). In addition the engineer should

take University and Industry instruction in Software Logistics Management, Software Cost Estimation, Development issues and lifecycle support issues.

The mid-level engineer should take the next series of DAU courses for certification. In addition the mid-level engineer should participate in professional organizations which support this effort and also study industrial and government standards on documentation, logistics control and software development CASE tools and evaluations.

The SME must finish at least one certification, and most likely 2. Should have a minimum of a graduate degree in a Software intensive engineering field. The engineer should belong to at least one professional organization e.g. IEEE, SNE, SISO, I/ITSEC, MORES etc. The engineer must maintain professional competence through seminars, briefings and symposia which delineate changes and enhancements to the field.

Sustainment to Maintain Expertise:

Sustainment of experience is through professional organizations, symposia, conferences and technical meetings. Post Graduate courses, which are directly tied to this area, are encouraged. Briefings and presentations to technical judged symposia and conferences to include process definition organizations such as SPC and SEI. The SME must also be involved in Joint operations projects that involve FMS and joint services agreements.

3.2.2.3.6 Software Development Methods and Tools (SDMT):

Description of Skill: - SDMT covers the traditional waterfall, spiral, and Objected-Oriented Technology modeling, Non-Developmental Items (NDI) to the complete use of COTS to develop a system. The life cycle activities associated with SDMT are similar to system engineering: Requirement definition, design, test, validation and sustainment. There are many different tools and languages supporting different types of process (e.g., waterfall and Object Oriented (OO)) definition (e.g., requirement generation), modeling (e.g., functional & Behavioral), and simulation (e.g., virtual, constructive) and they change as the technology changes.

Basic/Minimum Knowledge: This is a field that one must understand the past SDMT and be able to comprehend the new trends. One should have degree/or working knowledge on Software Development, Acquisition, and software engineering. One must also frequently attend trade-shows, and product demo.

Training or Work Assignments Contributing to Basic Knowledge: DSMC/DAU has several courses lead to certification in Software Acquisition Management. Software Technology Support Center at Hill Air Force Base has an extensive library/topics on SDMAT. There are a lot of lessons learned (undocumented) about the goods and the bad of software acquisition at STRICOM and DOD. There are product demo(s) and trade-shows one can attend all year around. However currently there is no gage or metrics or outlets to measure ones understanding of SDMAT. There needs to be feedback mechanism setup and a well-defined depository to increase the level SDMAT awareness. This will be the task of the SDMT skill lead in the future.

Developmental Path:

Entry Level: Be able to identify software development methods to include:

- Choice between evolutionary, incremental, or waterfall development methods
- Choice of SW requirements analysis and design methods (e.g., Data flow oriented, Data structure oriented, Object oriented)
- Software peer review methods
- Coding, unit test, and integration methods

Mid-Level: Be able to identify the types of tools used for each development period of the software life cycle:

- Requirements, Analysis, Specification, and Design Tools (e.g., Test Case Analysis and Generation, Graphical Modeling Tools);

- Programming and Debugging Tools and Components (e.g., Ada Compilers and Programming, C/C++ Compilers and Programming Components, CORBA-Related Products, Embedded Systems Development, Java Programming);
- Testing and Test Management Tools
- Database Design and Development Tools
- World Wide Web Development Tools (e.g., Java Programming Environments Web Page and Site Editors) **Note—C4I Systems currently utilized web pages.
- Project, Process, and Product Management Tools (e.g., Configuration Management and Version Control Problem Tracking Systems)
- Rapid Application Development Tools
- Middleware and Connectivity Products (e.g., CORBA-related Products)
- Other Tool Categories (e.g., GUI Builders, Program Analysis and Metrics, Frameworks and Integrated Environments, Simulation, License Management)

Subject Matter Expert: Be able to identify the type of tools used based upon the system modeling approach required (e.g., state machine, virtual – constant delta time intervals, constructive -- time stepped, irregular intervals)

Training or Work Assignments Comprising Developmental Path:

- Attend in-house sponsored courses (e.g., Introduction to Object-Oriented Technologies for Engineers by Software Productivity Consortium)
- DAU courses to include:
 - Entry Level: SYS 201, SAM101.
 - Mid Level: SYS 301, SYS 211, SAM 201.
 - SME Level: SAM 301
- Attend university courses
- Attend short contractor sponsored courses
- Assignment to a virtual simulation project
- Assignment to a constructive simulation project
- Assignment to a C4I project
- Assignment to research efforts

Sustainment to Maintain Expertise:

- Ongoing personal web research and reading (e.g., Index of Software Development Methods and Tools <http://www.methods-tools.com/html/tools.html>)
- Attend professional meetings and conferences (e.g., Software Technology Conference)

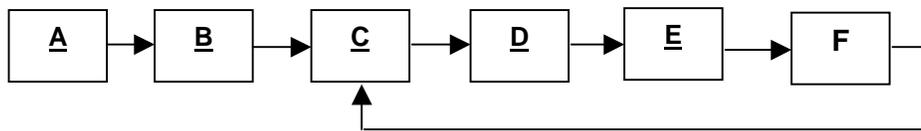
***3.2.2.3.7 Software Measurements and Analysis (SMA):

Description of Skill:

This skill focuses on data-driven decision making. This data-driven decision making is based on a measurement program that provides information that improves decision-making in time to affect the outcome of the process and/or project. A measurement program can: (1) Provide early insight into program risks and potential problems, (2) Provide quantitative support for management decision making, (3) Help forecast trends, (4) Provide visual indicators of progress, (5) Correlate diverse data and trends, and (6) Track effectiveness of corrective actions. A measurement program can not: (1) Fix a problem or eliminate risk, (2) Identify the solution to a problem, (3) Guarantee product quality, or that the product meets mission goals.

Basic/Minimum Knowledge:

This skill requires a framework to be able to establish a measurement program for a given project or an organization. The following is a description of a framework (or process) for adopting software measurement in an organization/project.



A – ESTABLISH A SUPPORTIVE CULTURE. The purpose of this task is to ensure there is management sponsorship for the adoption of software measurement technologies. Following are tasks to establish a supportive culture:

- Understand the organization’s mission and goals
- Establish (or develop) sponsorship for adopting software measurement
- Establish measurement roles and accountability at all levels of the organization

The more significant elements of a supportive culture include:

- Proactive leadership
- Supportive management style
- Open organizational communications
- Quality-oriented work environment

B – DETERMINE CURRENT MEASUREMENT CAPABILITY AND USE. The purpose of this task is to determine how well an organization is using measures and how ready it is for additional technology adoption. The steps for determine the current measurement capabilities include:

- Examine (or evaluate) current use of software measures.
- Examine underlying organizational readiness to implement software measurement.
- Determine current information needs, including project progress, product quality, and process effectiveness.

There are two factors that make the difference in this area to support and effective measurement program. The following two factors are additive to the factors noted above.

- Educated management expectations.
- Effective organizational infrastructure.

C – DEVELOP ORGANIZATION’S MEASUREMENT PLAN. The purpose of this task is to develop a plan to adopt software measurement. This plan is like any project plan because the adoption of technology by an organization must be managed like any other project the organization would undertake. In other words, develop a project plan that will:

- Establish goals and measurable objectives
- Identify risks and mitigation activities
- Identify measurement adoption tasks

- Identify the organization's needs for education and training
- Establish resources and budget requirements to meet objectives
- Define milestones and schedules for implementation
- Gain support for and approval of the plan across the organization

The goals of the project planning are to:

- Establish specific objectives – goals, quotas, or target.
- Define high-level requirements of a statement of work
- Define a realizable, measurable approach to meeting those objectives – actions and strategies to follow
- Risk management – identification of risks, probabilities of occurrence, and mitigation steps
- Establish cost and schedule baselines – time, people, other resources
- Set and manage expectations

D – DEFINE AND USE SOFTWARE METRICS. The purpose of this task is to execute the software measurement adoption plan. The goal is to see the results of measurement in action within the organization.

- Execute the plan
 - Define needed information and data that fulfill the information needs
 - Define software measures to provide needed information
 - Ensure the measures support organizational objectives
 - Implement the defined measures in pilot projects
 - Evaluate progress against plan
- Provide education and training, keyed to your organization, to support software measurement

E – EVALUATE RESULTS. The purpose of this task is to determine how well the initial measurement objectives have been met and to establish a motivation to refine and improve the measurement program over time. The steps are to:

- Determine whether the organization reached the goals and objectives stated in the plan
- Determine whether some goals are unfulfilled
- Establish how the current software measures support objective “data-driven decision making.” For example,
 - Has a project performance baseline been established?
 - Are historical data kept to help predict the organization's performance on new projects?
- Establish new measurement goals the organization needs to fulfill. For example,
 - What decisions are still being made without data to support them?
 - What questions need to be answered using objective means?

F – DETERMINE ON-GOING NEEDS. The purpose of this task is to provide the foundation to evolve measurement implementation within the organization. Following are questions to be answered to gain insight into the ongoing measurement needs for the organization:

- Is there support to continue work on the measurement goals left unfulfilled?
- Do the current measures show how well the organization is meeting its objectives?
- Do the current measures provide the information needed to support management decisions?
- Do new goals support the organization's mission?
- Does ongoing measurement implementation and refinement have a sponsor?
- Who is accountable for implementing and using software measurement to fulfill the new goals?

Training or Work Assignments Contributing to Basic Knowledge:

This skill should have familiarity with the following DOD software measurement initiatives:

- Practical Software Measurement (PSM), Joint Logistics Command, Software Productivity Solutions, Inc.
- U.S. Army Software Metrics / Software Test & Evaluation Panel (STEP) Metrics, DA PAM 73-7, Chapter 10, 15 Jul 96 (pre-publication).

- CECOM Streamlined Integrated Software Metrics Approach (SISMA), U.S. Army CECOM Software Metrics Working Group, Software Productivity Solutions, Inc.
- SEI Software Engineering Measurement & Analysis, Software Engineering Institute (SEI)

Developmental Path:

Training or Work Assignments Comprising Developmental Path:

Sustainment to Maintain Expertise:

3.2.2.3.8 Software Requirements Analysis (SRA):

Description of Skill: Software Requirements Analysis (SRA) is the translation of user needs into a complete set of quantifiable, measurable, and testable software requirements; i.e., what the system software must do considering the environment in which it is to operate. Thorough requirements analysis includes a comprehensive cost/benefit analysis, an estimate of the resources required to develop, operate, and maintain the software, and regulatory or policy controls that affect software development and operation. Requirements baselined during the SRA phase are the basis for subsequent testing activities, which determine whether a requirement has been correctly interpreted and implemented. The Software Requirements should be testable. Ideally, if the contract and schedule allow, some overlap between requirements and design phases should be allowed.

The product of this phase is a set of control and data flows, supplementary text, and graphic materials that fully describe the functions the software must perform, including specific algorithms and step-by-step processing. Traceability between SRA products and specific user requirements (typically in the System Specification) should be documented. One product is the Software Requirements Specification, which contains the Software Requirements. The traceability between system requirements and software requirements is essential.

The analysis and definition of software requirements is the most important, yet difficult phase of any software development. If done improperly, the impact can be devastating, and resulting system deficiencies may be difficult, if not impossible, to correct. If done correctly, this phase will save lots of time in design.

Basic/Minimum Knowledge:

- Familiarity with various structured analysis techniques including:
 - Functional decomposition
 - Hierarchy diagrams
 - Object-oriented analysis
 - Data flow analysis
 - State transition charts
- Familiarity with the following software development methods which provide views of the system from different perspectives:
 - Object-Oriented
 - Process-Oriented
 - Behavior-Oriented
- Familiarity with Domain Engineering and Software Reuse concepts
- Ability to evaluate and use CASE Tools to support SRA
- Strong communication skills are needed to ensure understanding of user needs
- Familiarity with one or more high level programming language
- Knowledge of System Engineering concepts commensurate with the GS-854-13 level
- Knowledge of the importance of interface definition between software components. For example, what data does one software component expect to get from another? If one component changes, who needs to know?

Training or Work Assignments Contributing to Basic Knowledge:

- System Engineering Training (that emphasizes Software Intensive systems)
- Basic Army 101 courses (for example C4I principles) to better understand user requirements

Developmental Path:

- BS Degree in Computer, Electrical, or Software Engineering
- DAU Courses, Software Acquisition Management, SAM 101, 201, and 301
- DAU Courses, SYS 201 and SYS 301
- MS Degree in Software or Computer Engineering
- Rotational or Training With Industry Assignment on a development project during SRA phase

Training or Work Assignments Comprising Developmental Path:

- Assignment to multiple software intensive projects in the requirements analysis phase
- Complete Software Engineering related courses

Sustainment to Maintain Expertise:

- Completing Software Engineering related University Courses
- Attending Industry Conferences and Symposiums
- Periodically attending training on new Tools and Methodologies
- Participation in the SRA phase of a STRICOM project

3.2.2.3.9 Software Resource Estimation (SRE):

Description of Skill: The skill of Software (SW) Resource Estimation (SRE) is one of many keys to successful project management. SRE encompasses the need to accurately estimate the expected effort needed, to complete a project's software development effort IAW proposed completion dates. The esoteric nature of SRE stems from the fact that most (if not all) cost-estimated techniques are based upon past experiences. These experiences include those of the estimator as well as past performance of a contractor captured by collection of quantitative data from their past projects. Analysis of this collected quantitative data entails being able to estimate and analyze software metrics of resources. These metrics can include estimates of the software development efforts - size, expected manpower needs; cost; and progress in meeting programmatic milestone/schedule dates.

Basic/Minimum Knowledge: An understanding of structured development and an understanding of object-oriented development methodologies. An understanding of developmental concerns and cost drivers to include:

- Requirements creep coupled with unrealistic expectations;
- Incorporation of rapid prototyping/feature development methodologies and concerns
- Positive effect of OOD methodologies;
- Noncoding tasks such as CM/QA, Test Interface control, and documentation taking time and money to complete;
- Need to rethink the design due to changes in operation requirements;
- Over optimistic schedules for alpha and beta testing;
- Need to use project management software for estimation of task completion times for each task; identifying critical paths; and establishing final dates.

A development cycle must specify, design, prototype, review, implement, and test - should be established. Basic knowledge required should include the ability to identify which metrics are being used on a program; consider their influence and impact; determine whether they help predict time or quality of development. Should be able to identify irrelevant metrics while adopting appropriate ones. Should be able to look at system requirements, and contractor proposal or metrics report, and estimate the software resources required for the project, including software size, manpower, cost, and progress.

Cost and Schedule Estimation Techniques that are required include:

- Time for analysis, design, implementation, and testing;

- Hierarchy metrics, including nesting level, number of abstract classes, "fanout
- Methods metrics, including size (in lines of code) and number of parameters;
- Coupling and cohesion metrics;
- Reuse metrics.

Advanced Knowledge:

The following is a list of a number of cost estimates commonly included in discussions of general software engineering cost models.

- Algorithmic Models, including algorithms for producing a software cost estimate e.g. COCOMO;
- Rules of Thumb, including guidelines that have evolved within the software engineering community over time
- Expert Judgement, including consultation with one or more experts;
- Estimation by Analogy, including comparisons with completed projects;
- Design to Cost, including matching the product to the effort (cost) available;
- Price-to-Win Estimating;
- Top-Down Estimating;
- Bottom-Up Estimating.

Training or Work Assignments Contributing to Basic Knowledge:

The entry-level person must have a BS Degree in Computer Science, Electrical/Electronic, or Software Engineering. The entry level person must pursue the following training to progress to the mid level status:

- DAU courses in Software Acquisition Management to include SAM 101 and SYS 201
- Graduate courses in Industrial Engineering (interactive Simulation), computer science or engineering, training with industry, system engineering training with emphasis on software intensive systems and/or use of cost estimation and other CASE tools for systems' development.

Training and Work assignments for Mid level Developmental Path:

- Masters. Degree in Industrial Engineering (Interactive Simulation) or Computer Science/Engineering. Rotational or Training with Industry Assignment on a development project during SRA phase.
- DAU Courses, Software Acquisition Management, SAM 201.
- DAU Courses, SYS 301.
- Cost Estimation training and further development in CASE tool usage.
- System Engineering training that emphasizes software intensive systems.
- Basic Army Information Assurance (IA) and Automated/Computer Systems courses.
- Be assigned on a proposal evaluation team, to be exposed to how the contractor estimated its required resources.
- IBR (Integrated Baseline Review) training, to learn earned value cost and schedule metrics.
- Study previous SW metrics methods. Look at such things as how SLOC was estimated against requirements, and cost against SLOC). Become proficient software size and manpower estimates through study of like systems' requirements.
- Be assigned as the metrics POC on a simulation project.

Sustainment Training and work assignments to Maintain Subject Matter Expertise:

To maintain expertise, I recommend periodic training and assignments as described above. In addition:

- DAU Courses, Software Acquisition Management, SAM 301
- Completing Software Engineering related University Courses
- Attending Industry Conferences and Symposiums
- Attending training on new Tools and Methodologies
- Participation in the SRA phases of STRICOM projects
- Attend project's metrics training. This training gives insight into the contractor's justification and method for SW resource estimation. Participate in the approval process for creation of realistic metrics.
- Participate in symposia and conferences in the creation and acceptance of MOE for systems' effectiveness and acceptance.

3.2.2.3.10 Software Test Engineering Methods (STEM):

Description of Skill: Knowledge of software requirements traceability, from requirements decomposition, to software documentation, through software verification. Knowledge to include development and requirements of test portion of software development files. Basic knowledge of software development methodologies, software configuration management, and software quality assurance, and their relationship to software testing. Basic knowledge of the Test Incident Report generation, monitoring, and closure process. Knowledge of unit/component/configuration item requirements definition, testing, integration and testing processes and techniques. Means to generate the appropriate test cases, requirements, procedures and expected outcomes. Knowledge of boundary testing and various stress testing techniques (erroneous input, etc). Knowledge of usage of test drivers and stubs within unit/component/CI testing, and their configuration management requirements. Knowledge of automated software-testing tools. In depth knowledge of various testing techniques to include path testing (and path generation/complexity

measurements), mutation testing, symbolic execution, state transition testing, transaction flow testing, etc. Knowledge of Integration testing techniques (top-down, bottom-up, big bang, sandwich). Understanding of software metrics and their relationships to software testing (complexity, depth of testing, McCabes, Halstead's, etc).

Basic/Minimum Knowledge:

- Alpha and Beta testing, and their relationship to fielding and the means to track problems through these forms of testing.
- The difference between testing and debugging.
- The differences between host and target software testing and the limitations/concerns of each test type.
- Cold start procedures, value, intent and process, and its relationship to software CI testing.
- Software reliability testing, tools and techniques.
- How to perform software “spare” resource (sizing, timing, etc) testing.
- The various software development techniques (structure design versus object oriented versus real-time systems) and their impacts on testing methodologies.
- The differences between developed software, COTS software, and 4GL software testing and requirements verification.

Training or Work Assignments Contributing to Basic Knowledge:

DAU courses:

Basic Software Acquisition Management (SAM 101)
Introduction to Test and Evaluation (TST 101)

Developmental Path:

Participate in development of software test plans, onsite acceptance testing, and Test Incident Report meetings to get a feel of the STEM process and procedures.

Training or Work Assignments Comprising Developmental Path:

DAU courses:

- Intermediate Software Acquisition Management (SAM 201)
- Intermediate Test and Evaluation (TST 202)

Sustainment to Maintain Expertise:

Attend conferences (ex. Software Technology Conference, ITSEC, CALS) to keep up to date on the latest practices and programs used in software test and evaluation.

3.2.2.3.11 Trusted Software Development Methodology (TSDM):

Description of Skill: The skill of developing software IAW a Trusted Software Development Methodology (TSDM) requires knowledge of TSDM trust principles as outlined in the July 2, 1993 TSM Report. {Note: This document contains a rationale for each trust principle; a set of compliance requirements for the trust principle as well as identification of applicable trust classes. In addition, the document identifies a list of associated DoD requirements that describe activities similar to those addressed in the trust principle and provides a list of useful references for the trust principle. }

Basic/Minimum Knowledge: The basic knowledge required includes an understanding of security to include factors such as threat, vulnerability, safeguards, and configuration management (CM). The candidate should exhibit familiarity with hardware, software, and firmware that have been shown to be robust and secure enough to support TSDM such as appropriate relational database management systems and operating systems (OS) e.g. Oracle meeting NSA Orange book Levels and Sun Solaris Trusted OS. An understanding of encryption/decryption devices is expected as well as knowledge of appropriate networking hardware items such as routers and switches that have been approved for operation in a collateral environment (supporting operation at multi-echelons of security).

As a minimum, the candidate should understand each of 25 TSDM principles that can be grouped into the following four areas:

- Management Policy (trust principles 1-6);
- Environment Controls (trust principles 7-10);
- Environment Management (trust principles 11-14);
- Software Engineering (trust principles 15-25).

Should understand what discriminates between the five TSDM Levels:

- T1 (minimal trust);
- T2 (moderate trust);
- T3 (preferred);
- T4 (malicious attack);
- T5 (ideal).

Because TSDM is a process to measure software/information assurance (IA), a candidate should be cognizant of the implementation/integration of each of the TSDM 25 principles into a project's general software development process. This means an understanding of the importance that a program's Software Development Plan (SDP) plays in ensuring acceptable TSDM practices are being carried out, is necessary. This is because, the SDP will capture TSDM compliance methods as well as preliminary software engineering team risk analysis results.

Additionally, it is important to understand how TSDM compliance will be identified and tracked using software engineering analysis standards such as the Software Engineering Institutes (SEI) Capability Maturity Model (CMM). Knowledge is required to be able to estimate of the cost associated with providing TSDM training on a program to the software engineering team members. Finally, knowledge of software reuse and metrics collection is necessary.

Training or Work Assignments Contributing to Basic Knowledge:

Developmental Path:

- BS Degree in Computer, Electrical/Electronic, or Software Engineering.
- DAU Courses, Software Acquisition Management, SAM 101, 201, and 301 (entry, Mid, Expert levels respectively).
- DAU Courses, SYS 201 and SYS 301 (mid and Expert levels).
- MS Degree in Industrial Engineering (Interactive Simulation) or Computer Engineering.
- Rotational or Training with Industry Assignment on a development project during SRA phase.
- Cost Estimation training utilizing state of the art CASE tools.
- System Engineering training that emphasizes software intensive systems.
- Basic Army Information Assurance (IA) and Automated/Computer Systems courses.
- Assigned to a proposal evaluation team, to be exposed to how the contractor estimated its required resources.
- IBR (Integrated Baseline Review) training, with emphasis on earned value cost and schedule metrics, like ACWP and BCWP.
- Study previous SW metrics methods. For example, WARSIM is using knowledge obtained from CCTT SW metrics. Look at such things as how they estimated SLOC against requirements, and then cost against SLOC. Review SW size and manpower estimates, both projected and actual.
- Be assigned as the metrics SMS on a simulation project.

Training or Work Assignments Comprising Developmental Path:

- Assignment to multiple software intensive projects in the requirements analysis phase
- Complete Software Engineering related courses

Sustainment to Maintain Expertise:

To maintain expertise, I recommend periodic training and assignments as described above. In addition:

- Complete Software Engineering related University Courses
- Attend Industry Conferences and Symposiums
- Attend training on new Tools and Methodologies
- Participation in the SRA phase of a STRICOM project
- Attend project's metrics training. Which depicts the contractor's justification and method for SW resource estimation.