Bottom Up Methods of Estimating Software SEPM and Non-DCTI Cost

2017 ICEAA Professional Development & Training Workshop

James R. Black

June 2017

Bio

- Presenter: James “Jay” Black has 13 years of cost estimating experience
  - Currently works as a software project manager for the Administration for Children and Families within the U.S. Department of Health and Human Services
  - In this role, he supports the Grant Solutions software suite used to administer 1200 grant programs in eight Federal departments
  - Jay has a Masters in Systems Engineering from Johns Hopkins University and holds a current CCE/A certification
- Jay worked for the Navy cost community (NAVAIR 4.2, NCCA, & NAVSEA 05C) from 2003 through 2015
- For questions, comments, and/or feedback please email me at:
  - James.robert.black@gmail.com
  - James.black@acf.hhs.gov

If any material in this brief is of use to you now or later, please email me, I’d love to know

References & Abstract

- References/Acknowledgements:
  - Mike Popp's observations on IEEE Standard 11207
  - Tim Lawless observations on Systems Engineering/Program Management
  - Cost Estimating Body of Knowledge (CEBoK) Module 1 Cost Estimating Basics
- Presentation abstract:
  - Systems Engineering/Program Management (SE/PM) and additional non-Design, Code, Test, and Integration (Non-DCTI) activities performed during software development efforts are often significant and drive estimates of total project costs
  - Yet, cost estimates often omit the detailed research and analysis needed to adequately model SE/PM & Non-DCTI costs
  - This brief will present bottom up methods useful for understanding and estimating these costs and share analysis of recent SE/PM & Non-DCTI data

This presentation is the result of practicing detailed actual cost data collection for a variety of platform and system estimates/analyses

Bottom Line Up Front (BLUF)

- The WBS is the foundation and a common WBS speeds the uptake of information for the cost estimate's audience
- Be diligent in identifying how effort is allocated to DCTI, Non-DCTI and/or SEPM: take care not to double count or omit effort
- IEEE 12207 is useful when differentiating between SEPM & DCTI
- Because SEPM is a predominately fixed & recurring cost, estimating it best served by a bottom up methodology
- Factor driven estimating methodologies are useful in the absence of time/data and can be useful crosschecks

WBS = Work Breakdown Structure
DCTI = Design, Code, Test, & Integration
SEPM = Systems Engineering & Program Management
IEEE = Institute of Electrical and Electronics Engineers

WBS is the Foundation

Per Cost Estimating Body of Knowledge (CEBoK) Module 1 Cost Est. Basics:

- A Work Breakdown Structure (WBS) establishes a **common frame of reference** for relating job tasks to each other and relating project costs at the summary level of detail
  - It provides a consistent and visible framework for specifying the objectives, labor, materials, and contracts of the system/program.
  - The structure is used to define the total program/system by providing detailed definitions of individual elements required (via a WBS Dictionary)
- A WBS should be tailored for each system or program for the purpose of **capturing all the idiosyncrasies** endemic to each system/program
- Estimate uses a WBS that is at a level of detail appropriate to ensure that **cost elements are neither omitted nor double-counted**
- Estimate is presented in a WBS fully **traceable to the system specification**
- WBS structure is **aligned to organizational structure performing the work**; WBS element tasks are traceable to data, which is traceable back to the respective source documents

The WBS is the foundation and a common WBS speeds the uptake of information for the cost estimate's audience

WBS Example - MIL-STD-881C

- For Department of Defense projects, WBS templates are provided in: MIL-STD-881C Work Breakdown Structures for Defense Material Items
- An example WBS from MIL-STD-881C for an Automated Information System follows:

1.0 Automated Information System (AIS)

1.1 AIS Prime Mission Product (PMP) Release/Increment X

- **1.1.1 Custom Application Software 1…n**
- **1.1.2 Enterprise Service Element 1…n**
- **1.1.3 Enterprise Information System 1…n**
- **1.1.4 External System Interface Development 1…n**
- **1.1.5 AIS Platform Hardware**
- **1.1.6 System Level Integration***

- **Design, Code, Test, & Integration (DCTI)**
- **Element-level software development**
- **Non-recurring, variable costs: driven by a software sizing measure, e.g. Source Lines of Code (SLOC)**

1.2 System Engineering

1.3 Program Management

- **"SE/PM" or "SEPM"**
- **System-level activities related to software, hardware, and integration**
- **Recurring, fixed costs: driven by a Level-of-Effort headcount mostly independent of software size**

1.4 Change Management

1.5 System Test and Evaluation

1.6 Training

1.7 Data

1.8 Peculiar Support Equipment

1.9 Common Support Equipment

1.10 Operational/Site Activation

1.11 Industrial Facilities

1.12 Initial Spares and Repair Parts

So, what does it mean when estimators refer to "Non-DCTI" ?

* Note: 1.2 & 1.3 also include system level integration efforts

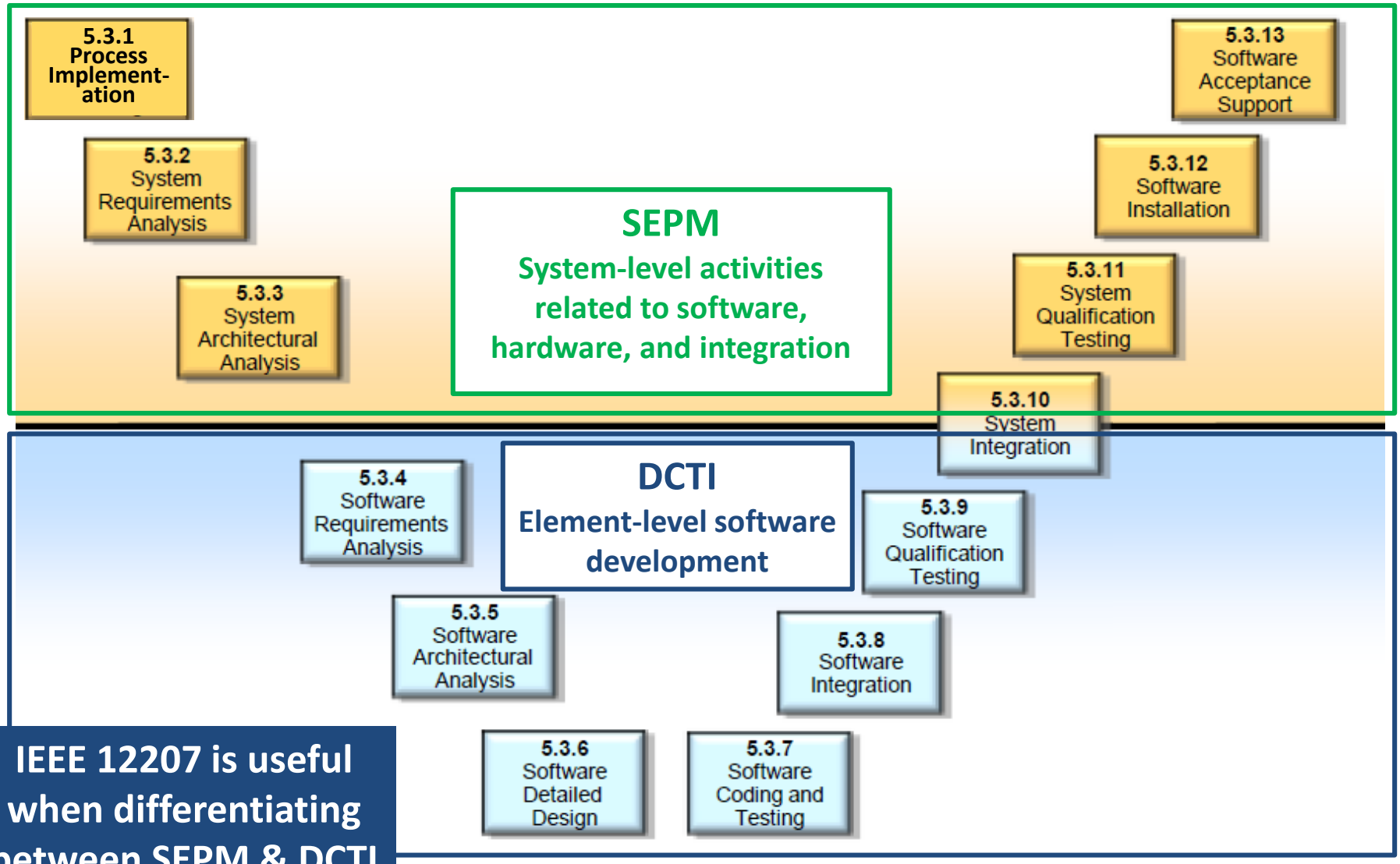
Software Cost Estimating Key Words

- Design, Code, Test, & Integration (DCTI)
- Systems Engineering & Program Management (SEPM)
- "Non-DCTI" - In the presenter's experience, Non-DCTI is the SEPM related to software development
  - Exists as a recurring development activity
  - Often begins before the start of non-recurring development activities (i.e. DCTI)
  - Continues throughout the execution of DCTI activities
  - Keep in mind: since Non-DCTI is not immediately identifiable in MIL-STD-881C, it can mean different things to different people

Be diligent in identifying how effort is allocated to DCTI, Non-DCTI and/or SEPM: take care not to double count or omit effort

IEEE Standard 12207

"Systems and Software Engineering - Software Life Cycle Processes"



IEEE 12207 is useful when differentiating between SEPM & DCTI

IEEE = Institute of Electrical and Electronics Engineers

SEPM Estimating

- When estimating SEPM:
  - For continued development on an existing system (e.g. future spiral or increment): collect SEPM FTE actuals from prior spirals/increments (e.g. from CCDRs 1921 & 1921-1)
  - For a new system: collect SEPM FTE actuals for analogous systems (e.g. from CCDRs 1921 & 1921-1)
  - Normalize collected data:
    - Identify how representative prior FTEs are of future effort required
    - Parse total FTEs to identify software efforts
- To estimate SEPM costs:
  - First, use a bottom up estimating methodology: FTEs/year * Years of development * Labor Rate
  - Then, use a crosscheck: DCTI cost or hours * Non-DCTI factor

Primary estimating approach: bottom up methodology
Crosscheck: factor driven approach

FTE = Full-Time Equivalent
CCDR = Contractor Cost Data Reporting

SEPM as a Fixed Cost

- Why not use a parametric technique to estimate SEPM?
  - I.e.: why not use Excel's "Analysis ToolPak" or ACEIT Costat to develop a correlation between SEPM Hours (or cost) and an objective measure (ESLOC, DSLOC, weight, volume, etc.)?
- SEPM is a predominately fixed and recurring cost:
  - SEPM does not scale directly with the project's objective measures
  - E.g.: for every additional ESLOC of a project, there is not a proportional increase in the number of SEPM FTEs
  - SEPM often varies significantly depending on the project/contractor/vendor
- However, SEPM ***does scale*** with the total number of concurrent baselines under development:
  - E.g.: if the system under development will be integrated onto ***2 different*** host aircraft types; then, the total SEPM costs will be greater than if the system was integrated onto 1 host aircraft type
  - I.e.: costs for integrating a system onto 2 platforms > 1 platform

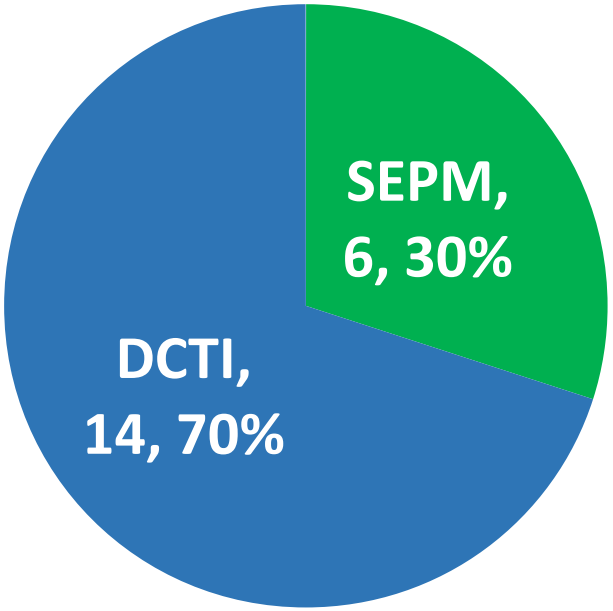
Because SEPM is a predominately fixed & recurring cost, estimating it best served by a bottom up methodology

ESLOC = Equivalent Source Lines of Code

Collecting SEPM Actuals by Labor Category

- When collecting SEPM FTE actuals from a contractor/vendor:
  - Separately identify FTEs by constituent labor categories
  - Parse FTEs by effort split between SEPM and DCTI
  - E.g. a small software development effort could be:

	FTEs	
Labor Category	SEPM	DCTI
Project Manager	1	
System Architect	0.5	
Business Architect	0.5	
Software Development Lead	0.25	1
Business Specialist	1	
Security Specialist	0.25	
Business Analyst	1	1
Database Administrator		1.5
Software Developer		8
Quality Assurance Lead	0.5	0.5
Test Engineer		2
Technical Writer	1	
Total	6	14



For future crosschecks:
here, Non-DCTI factor =
6 / 14 = ~40%

Software Development Cost Rosetta Stone

Software Development Cost = Variable Costs + Fixed Costs

Software Development Cost Element Structure:	Design, Code, Test, & Integration (DCTI)	Non-DCTI (a.k.a. software SEPM)
Analog in IEEE/EIA 12207:	**Element-level Activities:** Software Requirements Analysis, Software Architecture Analysis, Detailed Design, Code & Unit Test, Software Integration, Software Qualification Testing, & Part of System Integration	**System-level Activities:** Process Implementation, System Requirements Analysis, System Architecture Analysis, System Integration, System Qualification Testing, Software Installation, & Software Acceptance Support
Analog in MIL-STD 881C:	Prime Mission Product (PMP)	Inc. in Systems Engineering Program Management (SEPM)
Analog in CSDRs:	Scope Captured in SRDR (2630 forms)	Scope Captured in CCDR (1921 & 1921-1 forms)
Analog in Production Estimating:	Touch Labor | Support-to-Touch	SEPM

Slide POC: james.robert.black@gmail.com

Recommendations When Estimating Software Costs

- Get as much ***visibility*** as possible when collecting SEPM FTE actuals from contractors/vendors:
  - Use IEEE 12207 to forge a common understanding of SEPM and DCTI
  - Separately identify recurring and non-recurring FTEs
  - Separately identify FTEs performing software, hardware, and integration related activities
  - Identify FTEs by constituent labor categories (i.e. how many project managers, systems architects, business analysts, domain specialists or subject matter experts, etc.)
  - If there are multiple concurrent spirals/increments or baselines, identify how the total SEPM is split between them
- ***Partition*** a software cost estimate into two components:
  - Variable, non-recurring costs - DCTI, i.e. costs dependent on software size
  - Fixed, recurring costs – Software SEPM or Non-DCTI , i.e. costs independent of software size

Visibility & Partitioning are useful when comparing cost estimates and briefing results to senior leaders

Summary

- The WBS is the foundation and a common WBS speeds the uptake of information for the cost estimate's audience
- Be diligent in identifying how effort is allocated to DCTI, Non-DCTI and/or SEPM: take care not to double count or omit effort
- IEEE 12207 is useful when differentiating between SEPM & DCTI
- Because SEPM is a predominately fixed & recurring cost, estimating it best served by a bottom up methodology
- Factor driven estimating methodologies are useful in the absence of time/data and can be useful crosschecks

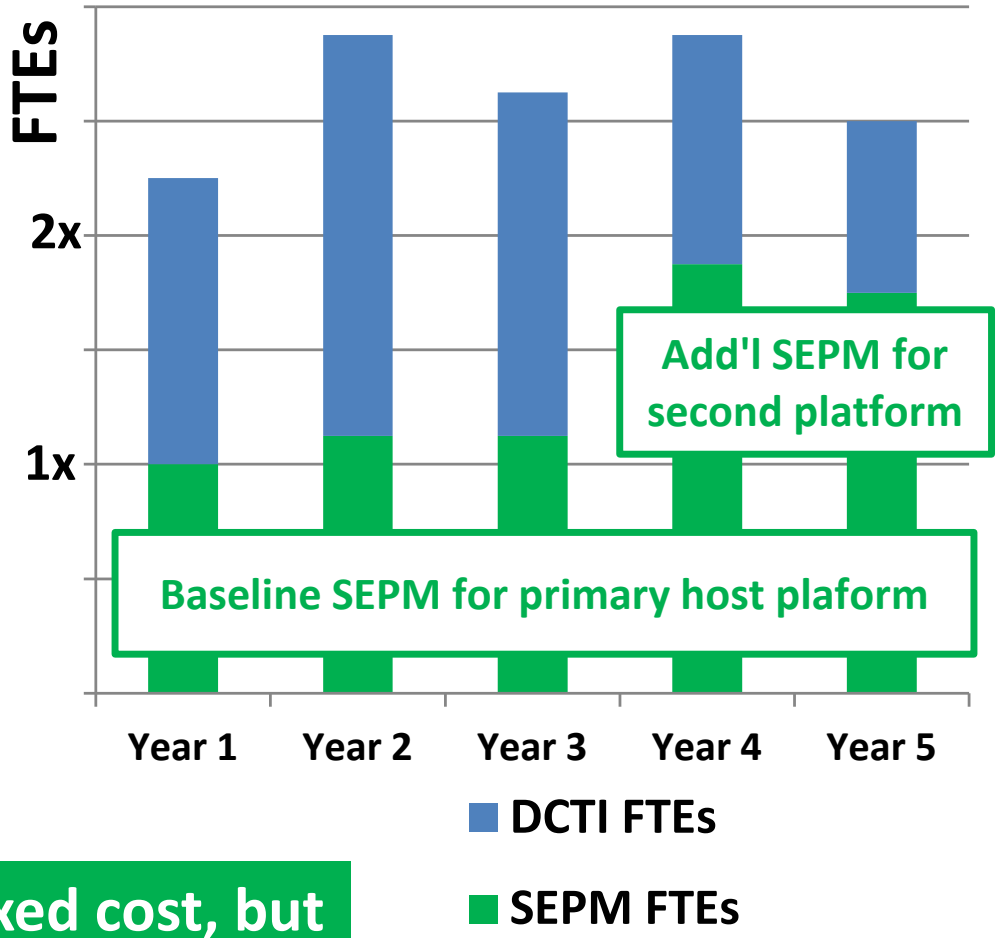
- Q&A

- Backup

SEPM Scaling with # of Platforms

- SEPM ***does scale*** with the total number of concurrent baselines under development:
  - E.g.: if the system under development will be integrated onto ***2 different*** host platform types; then, the total SEPM costs will be greater than if the system was integrated onto 1 host platform type
  - I.e.: costs for integrating a system onto 2 platforms > 1 platform

Example where a second host platform is added in Year 4 of development:



SEPM is a predominately fixed cost, but does scale with the # of platform types

IEEE Standard 12207

"Systems and Software Engineering - Software Life Cycle Processes"

SEPM
System-level activities related to software, hardware, and integration

DCTI
Element-level software development

SEPM
System-level activities related to software, hardware, and integration

5.3 The Development Process			
Activity	**Tasks (paraphrased)**	**12207.1 Information Item guidelines**	
5.3.1 Process Implementation	.1 Define software life cycle model	--	--
	.2 Document and control outputs	--	--
	.3 Select and use standards, tools, languages	--	--
	.4 Document development plans	Plan, Desc	6.5 DPP, 6.17 SDSD
	.5 Deliver all needed products	--	--
5.3.2 System requirements analysis	.1 Specify system requirements	Specification	6.26 SRS
	.2 Evaluate requirements against criteria	Spec, Record	6.26 SRS, 6.6 SRER
5.3.3 System architectural design	.1 Establish top-level architecture	Description	6.25 SARAD
	.2 Evaluate architecture against criteria	Desc, Record	6.25 SARAD, 6.6 SAER
5.3.4 Software requirements analysis	.1 Document software requirements	Desc	6.22 SRD, 6.30 UDD
	.2 Evaluate requirements against criteria	Desc, Record	6.22 SRD, 6.6 SRER
	.3 Conduct joint reviews iaw 6.6	--	--
5.3.5 Software architectural design	.1 Transform requirements into architecture	Description	6.12 SAD
	.2 Document top-level design for interfaces	Description	6.19 SIDD
	.3 Document top-level design for database	Description	6.4 DBDD
	.4 Document preliminary user documentation	Description	6.30 UDD
	.5 Document preliminary test requirements	Plan	6.27 T/VP
	.6 Evaluate architecture against criteria	Desc, Record	6.12 SAD, 6.6 SAER
	.7 Conduct joint reviews iaw 6.6	--	--
5.3.6 Software detailed design	.1 Document design for each component	Description	6.16 SDD
	.2 Document design for interfaces	Description	6.19 SIDD
	.3 Document design for database	Description	6.4 DBDD
	.4 Update user documentation	Description	6.30 UDD
	.5 Document unit test requirements	Plan	6.27T/VP
	.6 Update integration test requirements	Plan	6.27T/VP
	.7 Evaluate detailed design against criteria	Rec, Desc	6.6 DDER, 6.16 SDD
	.8 Conduct joint reviews iaw 6.6	--	--
5.3.7 Software coding and testing	.1 Document each unit, database and tests	Desc, Rec, Proc	6.4 DBDD, 6.24 SCR, 6.28 T/VPr
	.2 Conduct and document unit testing	Report	6.29 T/VRR
	.3 Update user documentation	Description	6.30 UDD
	.4 Update integration test requirements	Plan	6.27T/VP
	.5 Evaluate code and test results	Rec, Plan	6.7 EOCR,6.6 SCTRE, 6.24SCR, 6.27T/VP
5.3.8 Software integration	.1 Document integration plans	Plan, Proc	6.18 SIP, 6.28 T/VPr
	.2 Conduct and document integration tests	Report	6.29 T/VRR
	.3 Update user documentation	Description	6.30 UDD
	.4 Document qualification tests	Proc	6.28 T/VPr
	.5 Evaluate plans and tests against criteria	Proc, Desc	6.28 T/VPr, 6.30 UDD
	.6 conduct joint reviews iaw 6.6	Record, Plan	6.6 SIER, 6.18 SIP
5.3.9 Software qualification testing	.1 Conduct and document qualification testing	Report	6.29 T/VRR
	.2 Update user documentation	Description	6.30 UDD
	.3 Evaluate tests against criteria	Record	6.6 SIER
	.4 Support audits iaw 6.7	--	--
	.5 Prepare product for next phase	Record	6.24 SCR
5.3.10 System integration	.1 Integrate software with hardware & others	Report	6.29 T/VRR
	.2 Document integration tests	Procedure	6.28 T/VPr
	.3 Evaluate integrated system against criteria	Record	6.6 SQTER
5.3.11 System qualification testing	.1 Conduct and document qualification tests	Report	6.29 T/VRR
	.2 Evaluate system against criteria	Record	6.6 SER
	.3 Support audits iaw 6.7	--	--
	.4 Prepare product for installation	Record	6.24 SCR
5.3.12 Software installation	.1 Plan installation in target environment	--	--
	.2 Install software iaw plan	--	--
5.3.13 Software acceptance support	.1 Support acquirer's acceptance tests	Report	6.29 T/VRR
	.2 Deliver product per contract	Record	6.24 SCR
	.3 Provide training per contract	--	--