5.5 Software Subcontractor Management

The management of Software Subcontractor Teams is almost always a significant challenge.

On large contracts, there are usually numerous subcontractors contributing software products, and they are typically geographically dispersed (sometimes offshore). A *Lessons Learned Database* should be maintained on past experience with each teammate to tailor preparation of their Statement of Word (SOW) so as to avoid problems you, or your organization, encountered in previous software subcontracting management efforts. Subcontract management is discussed in four subsections:

- *Subcontractor Management Team (SCMT)*: Why the SCMT must be established (5.5.1).
- *Subcontractors Compliance with the SDP*: The Chief Software Engineer should be responsible for monitoring software subcontractor compliance with the program-level SDP including compliance of subcontractor site-specific SDPs with the program-level SDP (5.5.2).
- *Subcontractor Performance Reviews:* Subcontractor progress performance should be regularly reviewed and assessed by the SCMT and the CSwE (5.5.3).
- *Subcontractors Statement of Work*: The SOW must be focused on a clear, unambiguous definition of all Developer responsibilities including product deliveries, software metrics reporting, interfaces with the SCMT, and authorization for CSwE oversight (5.5.4).

5.5.1 Subcontractor Management Team

On large projects, a *SCMT* must be established. It should be led by your Subcontract Program Manager who has overall responsibility for monitoring technical, schedule and cost performance of the software subcontractors. On smaller projects, the SPM could take the dual role of Project Subcontractor Manager. Also, a *Software Subcontract Management Guidebook* should be prepared. The Software Quality Engineer (SQE) and the Chief Software Engineer must support subcontract management by technically monitoring the software portion of the subcontract. Table 5.8 is an example of typical SCMT membership and their responsibilities.

5.5.2 Subcontractor Compliance with the SDP

The CSwE should be responsible for technical monitoring of software subcontractors through attendance at the software subcontractor's formal reviews and status reviews, as applicable, and by regular metrics reviews. The CSwE must evaluate the performance of the software subcontractors and prepare subcontract evaluation reports as required. The CSwE can delegate these responsibilities.

Program-Level SDP. *The program-level SDP must apply to all software subcontractors.* The CSwE has the authority to enforce the processes described in the program-level SDP. Software subcontractors must follow the processes, procedures, and documentation defined in program-level SDP unless a formal waiver has been approved.

Site-specific SDPs. If needed, site-specific SDPs can be written and maintained by subcontractors at their sites; they provide additional standards and procedures specific to each site. *Site-specific SDPs expand upon, but must not conflict with, the processes and procedures defined in the program-level SDP unless a waiver has been approved.* The CSwE and SQE must perform software subcontractor product and process audits to determine compliance with program-level SDP and with the contract.

5.5.3 Subcontractor Performance Reviews

The progress and performance of the subcontractors should be regularly reviewed and assessed by the SCMT. These reviews address the total performance of each team toward meeting its objectives. The SEPG oversees the software development process and provides approval of all subcontractor specific appendices to the program-level SDP to ensure that software development methods, standards, practices and procedures are consistent with the contract.

5.5.4 Subcontractor Statement of Work

The subcontractor's Statement of Work must be focused on a clear definition of responsibilities, software metrics reporting requirements, interfaces with the SCMT for oversight, and identification of your CSwE as the single point of contact for software. Each team member performing software development, algorithm analysis, simulation development, or data set development should have its own SOW. The SOW should delineate development software products, scope, required reviews, schedule milestones, status reporting, performance evaluation criteria and acceptance criteria.

5.6 Software Verification and Validation

Software Verification and Validation (V&V) are two related but separate procedures. Because they are so related, there can be confusion as to how they differ.

- Verification: Ensures that each developed software function works correctly. Verification techniques include Managerial Reviews (see 5.3), Peer Reviews (see 6.5) and code walk-throughs (see 11.4.4).

Table 5.8 Subcontractor Management Team Members and Responsibilities

Member	SCMT Responsibilities
Subcontract Program Manager	■ Management of software subcontract technical, cost, and schedule performance ■ Ensures all software technical, cost, and schedule requirements are satisfied ■ Facilitates subcontractor's ability to plan and perform software more efficiently ■ Works program-level issues and manages software award fee program ■ Ensures subcontractor compliance with program plans and procedures
Subcontract Administrator	■ Single point of contact for contractual matters and administers the software subcontract ■ Negotiates and awards software subcontracts and approves vouchers/invoices ■ Ensures proper flow down of software technical requirements and software subcontract terms and conditions ■ Maintains configuration control of contractual documentation sent to the subcontractor ■ Receives, logs, and distributes incoming correspondence from subcontractors
Responsible Engineer or CSwE	■ Develops software specifications and associated technical documentation ■ Ensures subcontractor understanding of the software technical requirements ■ Coordinates approval of subcontract software deliverables and documentation ■ Conducts a technical evaluation of subcontractor's software proposals ■ Develops and/or approves software test plans, procedures and acceptance plans ■ Participates in or witnesses subcontractor software acceptance testing ■ Provides independent evaluation of subcontractor's technical progress and performance
Business and Financial Operations	■ Analyzes cost/schedule performance data including key indices and variance analyses ■ Helps subcontractor develop cost account plans and incremental planning packages ■ Integrates subcontractor budgets, costs, IMP and IMS into the centralized database system
Mission Assurance (SQA)	■ Monitors data, configuration, and quality processes used by the subcontractor ■ Ensures the software subcontract quality implementation program is consistent with the program's Software Quality Program Plan ■ Ensures that the configurations of all deliverable Software Items are identified with a clear audit trail ■ Chairs the Subcontractor Functional and Physical Configuration Audits (FCA/PCA)

- Validation: Ensures that the final product incorporates all of the system requirements. System testing also validates the inclusion of all system requirements. Validation techniques include White Box and Black Box testing:
  - *White Box Testing*: An *internal* perspective of the code functionality often assisted by automated testing tools.
  - *Black Box Testing*: An *external* perspective with little concern or knowledge of the internal structure of the software. Inputs are provided, and the outputs are compared to the expected outputs.

Figure 5.6 is a simple way to understand the difference between them. Verification is shown as making sure that each step in the software development process is responsive to its previous step. Validation is the procedure to make sure the final product is responsive to the original system requirements. Incremental validations could take place during the activities preceding System Test.

A common phrase used to explain the difference between validation and verification is: validation determines if you built the *right product*; verification determines if you *built it right*. The confusion between them is compounded depending on whether you are talking about V&V of the entire system or V&V of the software work products. In the latter case, for example, even though a software document may be complete, and it followed the standard and format for that document (you built it right), the contents of the document may not address or satisfy the customer's needs (you built the wrong product).

5.6.1 Software Verification and Validation with Associate Developers

On large contracts, the lead prime contractor organization may enter into what may be called "Associate Contractor Agreements" with other prime contractors whose performance will impact your program and project. Such arrangements facilitate *joint participation and collaboration* in meeting program requirements. The objective of these agreements is to create *ground rules* and an environment for *safeguarding* each other's technical and/or proprietary information and

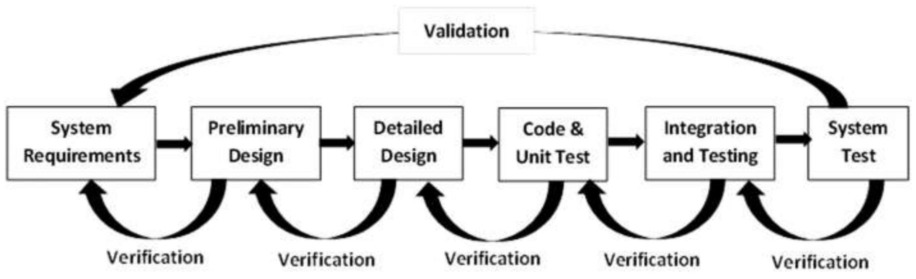

Figure 5.6 Software verification versus software validation.

resolving issues, to the maximum extent possible, without customer involvement or intervention.

As the need for these agreements become known, the lead prime contractor must advise the other prime contractors, in addition to other organizations that may be defined in the contract, where such agreements are necessary and then proceed with establishing those relationships. From a software perspective, Associate Developers are one type of software stakeholders.

5.6.2 Independent Software Verification and Validation Coordination

If required on the contract, Independent Verification and Validation (IV&V) contractors will interface with subsystem software development as a member of their respective Integrated Product Teams (SwIPTs). In addition to supporting software-related SwIPT tasks, the IV&V contractors may also perform audits of Software Development Files and software processes.

If audits are performed, the IV&V contractor must coordinate in advance with the SwIPT Lead and identify required Software Developer and Test Support. Additional details for interfacing with external software IV&V subcontractors may be defined in the SDP Annexes.

SwIPT personnel, including Software Developers and Test Engineers, should interface with the IV&V representatives to allow identification and resolution of software issues and problems at the lowest practical level and as early in the development process as possible. The IV&V representatives must interface with the SwIPTs and Developers, both formally and informally, through the following mechanisms:

- At scheduled software development status meetings, reviews, TIMs and SwIPT meetings
- Through the test process, either when the IV&V representative is a test witness or when the IV&V representative conducts independent testing
- Through periodic inspections and audits of Software Development Files, other software products, as well as the software development process

Table 5.9 is an example of the type of software products that must be provided to the IV&V agents and how problems are reported. Problems identified by the IV&V agents must be resolved through the Corrective Action Process (see Section 6.4).

As members of appropriate SwIPTs, the IV&V representatives should offer advice, assistance, and subject matter expertise in the development of program documentation, but must not co-author such documentation. This allows the IV&V representatives to preserve the degree of objectivity required to effectively discharge the IV&V role of verifying and validating both adherence to the software development process, and the adequacy, sufficiency, and performance of software products.

Table 5.9 Software IV&V Evaluations

Tasks	Software Products Reviewed	Problem Reporting Mechanism
Technical Interchange Meetings	Development phase documents	Action items
Formal software reviews	Development phase documents	Action items
Audits	SDFs, SDP	Audit report
Independent testing	Software test documents	SCR/SDRs

SDF = Software Development File (or Folder); SDP = Software Development Plan; SCR = Software Change Request; SDR = Software Discrepancy Report.

5.7 System Retirement or Disposal

Project Managers need to address the strategy and requirements for system retirement or disposal and ensure that sufficient information exists so that it can be carried out in a way that is in accordance with all legal and regulatory requirements relating to environmental, safety, security and occupational health issues. It is also an excellent idea to be proactive in planning for the *disposition* of government property that was used during the development of your project.

System disposal is one of those contract tasks that will often get little or no attention until contract completion, but it should be prepared for in advance to help the process run smoothly. For the retirement of government contracts, work with your local *Defense Contract Management Agency* (DCMA) for the procedure and options to return, purchase, reuse, sell, or scrap the government-owned property.

Planning for the disposal of the system while you are working so hard to develop it may seem to be counterproductive. However, system disposal can be a big headache if not properly planned for in advance. See Section 16.6 for more details on system retirement and disposal.

5.8 What is a Successful Project?

Demonstrating your product complies with the test cases, and successfully completing testing at various levels, is an obvious measure of success. Customer acceptance of your software product or system is a clear successful achievement. However, the definition of a successful product can be elusive.

The customer could be elated if the delivered system performed even better than expected even though the project was over budget and delivered late. The customer could be very unhappy if the delivered system did not provide the level of benefits planned for and expected even though you delivered the required system on time and under budget. Also, if there was disharmony between your Developers and the customer during development, the customer might be very unhappy with your team regardless of the quality, timeliness and functionality of the delivered system.

However, the definition of "done" should not be elusive; if it is, you are responsible for not making sure that every member of your staff precisely understands what is being built and what is meant by "*done*." You must be precise as to what reviews, testing, documentation, performance, scripting, etc. will be required for the code. The definition of *done* may change during development. The project kickoff is a good time to get an agreement on *when* a task, feature, Software Unit, or Sprint is finished.

If you don't define "done" up front, development costs and schedule could spiral out of control.