Validating Computer Systems, Part 3

GCP Software Verification

Teri Stokes

Writing software for GCP use is serious business. Applications must be tested and retested to ensure that they actually do what they are designed to do.

Software suppliers have a tough job. Writing software for use in a GCP work process adds an extra complication: preparing for many audits of the software engineering process itself as well as the software application produced. Part 1 of this series (August 2000) discussed a user acceptance package for computerized system validation (CSV) to prove that the "right" software was built for the user requirements in the GCP work process. Part 2 (September 2000) addressed the IT/IS department's CSV package to document that the software was installed on the "right" platform of components and infrastructure. Part 3 explores the software supplier's CSV package that documents the way the software was "built right" to meet its design specifications.

The life of a software application, as shown in Figure 1, has nine steps, beginning with a system idea and needs analysis and continuing through to retirement. The software development life cycle (SDLC) portion is shown in detail in steps 3 through 5 (design, build, test). These steps are the focus of the supplier's computer system validation (CSV) package.

Figure 1 also illustrates the areas of close coordination with the user group for refining the user requirements specification (URS, step 2), being audited in the commissioning step (step 6), and providing ongoing support and enhancements under a service level agreement (SLA).

Showing these steps in a flat, two-dimensional picture with arrows going in one direction is misleading, because there is usually a lot of cycling back and forth and refinement of specifications between steps 2 and 3, when users are shown demonstration or prototype versions of an application. At some point, however, the URS must be frozen so that the functional requirements can be finalized and the software design description made for the programmers to use as a blueprint for building the software application.

Requirements. The Questions on the Path to a URS box lists questions for defining requirements on any size of software application. The key factor here is that the user's work process requirements are written down and agreed to by all parties *before* the start of building the software. Extra time taken to refine the URS with the supplier can pay huge dividends, because the rest of the development process evolves as a response to the URS. Any error at the start builds error into the rest of the process and the final product. The cost to fix errors in specifications grows exponentially through each succeeding step of the development cycle. Unclear

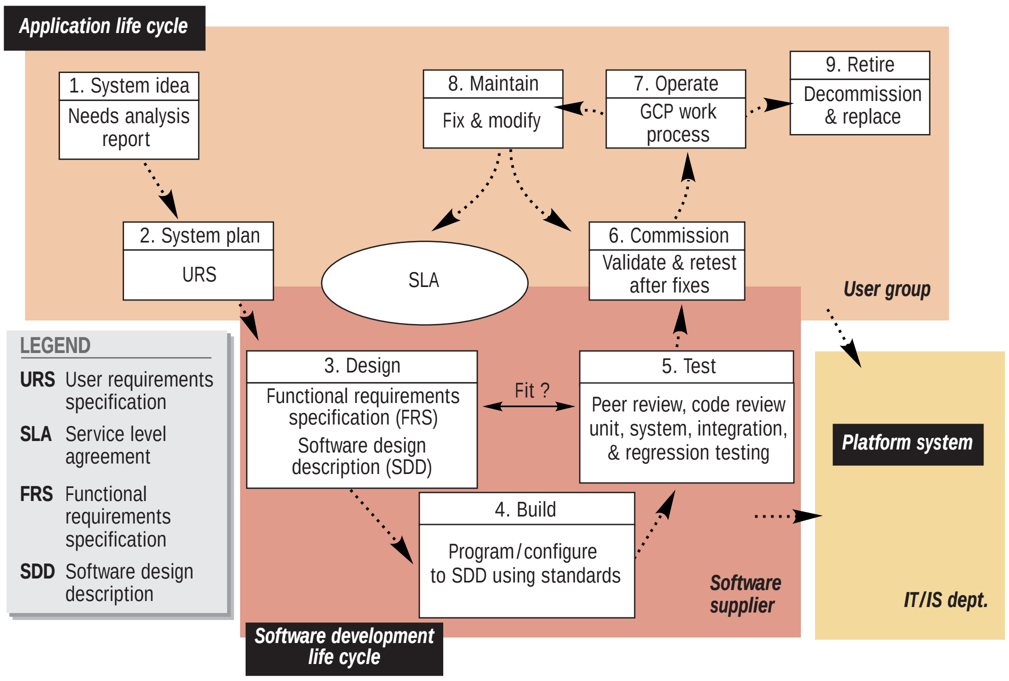
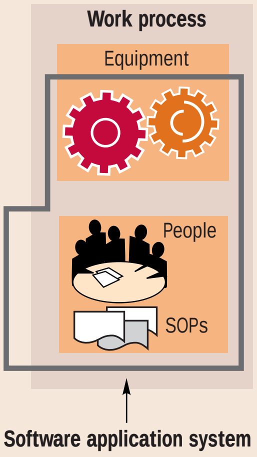

Application life cycle
1. System idea
Needs analysis report
2. System plan
URS
SLA
8. Maintain
Fix & modify
7. Operate
GCP work process
9. Retire
Decommission & replace
6. Commission
Validate & retest after fixes
User group
LEGEND
URS User requirements specification
SLA Service level agreement
FRS Functional requirements specification
SDD Software design description
3. Design
Functional requirements specification (FRS)
Software design description (SDD)
Fit ?
5. Test
Peer review, code review unit, system, integration, & regression testing
4. Build
Program/configure to SDD using standards
Software supplier
Platform system
IT/IS dept.
Software development life cycle

Figure 1. The software development life cycle in detail, and how it fits with the rest of the application development cycle.

Questions on the Path to a URS

Extra time here pays off in a smoother design process.

- ☐ Who are the sponsors and intended users?
- ☐ What job is the system expected to do in the work process? Which tasks?
- ☐ Where is the documentation to describe in detail the user's GCP work process?
- ☐ When is the software to be released, updated, fixed, tested, and maintained?
- ☐ Why does the work process need this software? Is it safety- or efficacy-related?
- ☐ How are users to be trained and problems resolved during ongoing operations?



requirements result in unreliable software.

Designing. The supplier's response to the URS may be in one or two technical documents, but will include both functional specifications and design description.

- Functional requirements specification (FRS)—document to describe the functions a system or component must perform.[1]
- Software design description (SDD)—document to describe system or component architecture, control logic, data structures, input/output formats, interface descriptions, and algorithms.[1]

The Software Specifications and Testing box shows that all requirements must be testable. A good requirements document lends itself to direct translation into test scripts. If a user, function, or design requirement cannot be stated in a clear and precise way so that it can be objectively tested, then it is a "wish" and not a "requirement."

Building. Well-defined requirements also smooth the path for the team building the software. The software application can be built in two ways.

- Programming. The supplier can use a programming language or object tool to write source code for a unique software application such as SAS (SAS Institute, Cary, NC) or Excel (Microsoft, Redmond, WA).
- Configuration. The supplier can configure an application by setting preprogrammed variations in a configurable system and writing small routines to adapt it to the user's specific needs. Examples of such applications are a macro for randomizing treatments and a spreadsheet for scheduling patient visits.

Testing. As different levels of software code are built, tested, adjusted, and retested, there is frequently a great deal of recycling between development steps 4 and 5 (Figure 1). Testing is conducted in several ways. Peer review is informal testing; a programmer gives the code to another software professional to try out and asks for comments and suggestions. Formal types of testing are defined below as per IEEE Standard 610.12-1990.[1]

- Code review. A meeting at which software code is presented to project personnel, managers, users, customers, or other interested parties for comment or approval.
- Unit testing. Testing of individual hardware or software units or groups of related units.
- System testing. Testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.
- Integration testing. Testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them.
- Regression testing. Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements.

Although this formal development and testing process is clearly needed for large systems—such as a sponsor's clinical data management system and high-end statistical system—many people question its usefulness for spreadsheets, macros, and small applications used at investigator sites. In fact, the same nine steps shown in Figure 1 need to be traversed for the life of any software used for GCP purposes, but the size and scope of each step is scaled to the size and scope of the software application. The CSV package documents are also scaled to fit the size of the software. A small software systems development form can be designed to cover steps 2 through 6 using two sides of one sheet of paper.[2]

Software supplier CSV package

Producing a good CSV package in software development is good business and exercises the standard practices for good software engineering. A well-defined and well-documented software development life cycle provides the software supplier with management control of the engineering process. Clear requirements and documented coding standards lead to efficient and effective building of software and the ability to bring new programmers into a project with minimal ramp-up time and reduced risk of errors.

Documented formal testing of a product under development provides traceability for changes and a measure of the reliability of the software and its capacity to protect the integrity of data it handles. It lays a strong foundation for problem resolution when the application is out on the market or future enhancements are considered. Test summary reports give developers and testers the opportunity to present objective evidence for how good they are at creating robust, reliable software. Good online and off-line user documentation facilitates user training, increases customer satisfaction with the application, and

Software Specifications and Testing

- ☐ User requirements, functional specifications, and design specifications must represent actual requirements of the work process and its platform system.
- ☐ All requirements must be testable.
- ☐ Testing must exercise all GCP aspects of specifications, including:
  - Functions
  - Limits
  - Invalid entries
  - Error messages
  - Access control
  - Online help messages.

Software Verification Plan[a]

Purpose and scope
- ☐ Inclusions, exclusions, and limitations.

Reference documents
- ☐ SOPs, manuals, and policies referenced by the plan.

Definitions
- ☐ Terms required to understand the verification plan.

Verification overview
- ☐ Organization and master schedule for the software verification effort.
- ☐ Resources summary and responsibilities for verification tasks (usually a three-column table listing verification tasks, role[s] responsible, and due date).
- ☐ Tools, techniques, and methodologies used in the verification effort.

Life cycle validation tasks at each phase until retirement
- ☐ Management of verification process. Control procedures of quality management system requirements for SDLC.
- ☐ Concept phase. Market analysis of user needs.
- ☐ Requirements phase. Refinement of URS for specific client and preparation of functional requirements specification.
- ☐ Design phase. Development of and approval for system design description. Perform design traceability analysis, design evaluation, and design interface analysis. Develop test plans.
- ☐ Implementation phase. Perform source code traceability analysis, evaluation, interface analysis, and documentation evaluation.
- ☐ Test phase. Execute test scripts for test plans and write test summary reports.
- ☐ Installation and checkout phase. Audit software installation package and check replication process for accuracy. Write CSV package summary report.
- ☐ Operation and maintenance phase. Establish help desk and any other support/maintenance activities. Develop bug fix, release, and upgrade plan for software product.
- ☐ Retirement phase. Plan for product retirement in original design. Establish the off-site archive with access and retrieval SOP.

Software verification reporting
- ☐ Required and optional records/reports to be written.

Verification administrative procedures
- ☐ Anomaly reporting and resolution. Problem/issues handling.
- ☐ Task repetition policy. Criteria for when to repeat testing or any other verification task when its input changes.
- ☐ Deviation policy. How to report actions taken that differ from the plan.
- ☐ Control procedures. How the software application and engineering system(s) are configured, protected, and stored—SOPs for backup/retrieval, disaster recovery, change control, and system testing.
- ☐ Standards, practices, and conventions for verification work—policies, procedures, and templates for logs, reports, and other items in the CSV package.

[a]*Adapted from IEEE Standard 1012-1986.*[3]

reduces the volume of help desk support calls.

Experienced software suppliers to the clinical trials marketplace understand that their clients are responsible to the authorities for the validation of software they use. To meet that responsibility, suppliers will be subjected to client QA audits to assess the way quality was built into the software during development. A well-organized supplier CSV package that emerges from the normal good engineering practices of the supplier's business will reduce audit time and audit follow-up work.

Software Package Summary Report[a]

- ☐ **Plan identifier.** ID number indicating system associated with the plan
- ☐ **Summary of all verification SDLC tasks** and their status.
- ☐ **Summary of all CSV package items** and their status.
- ☐ **Summary of unexpected problems/issues** and their resolution.
- ☐ **Summary of deviations** from the verification plan and rationale for deviations.
- ☐ **Assessment of overall software quality** based on package documentation, test summary report, and QA audit report.
- ☐ **Recommendations.** Release statement to management for the release status of the software application system.
- ☐ **Approval** signature(s) and date(s).
- ☐ **Appendix A.** Update report form for recording major system changes with related regression testing.
- ☐ **Appendix B.** Table of contents for CSV package.

[a]*Adapted from IEEE Standard 1012-1986.*[3]

The Software Supplier CSV box lists the items in the package. Whether each item is covered by a sentence, a paragraph, a page of text, or a manual of information will depend on the size and scope of the software product being developed and the size of the effort being described.

Verification plan

The Institute for Electrical and Electronics Engineers, Inc. (IEEE) publishes a standard for software verification and validation plans that can easily be adapted.[3] The Software Verification Plan box shows a sample of an outline for this purpose.

Every plan in the CSV package—verification plan, master test plan, subordinate test plans—must have a defined task list stating just what actions are to be taken to execute the plan's strategy. Every plan must also have its own summary report written to explain to management the outcome of the planned tasks. The package summary report includes the outcome of individual tasks as well as the highlights of summary reports from subordinate plans. After all the tasks in the supplier's verification plan have been completed, a package summary report is written (see the Software Package Summary Report box for an outline).

CSV package team

The supplier's CSV package is produced by an identified process carried out by a team of responsible people. The roles and responsibilities are essentially the same as for the user acceptance and IT/IS CSV packages. As shown in Figure 3, the product decision group initiating product development designates a CSV package sponsor. The package sponsor in turn assigns a package team from among the project group members, and the

Software Supplier CSV Package

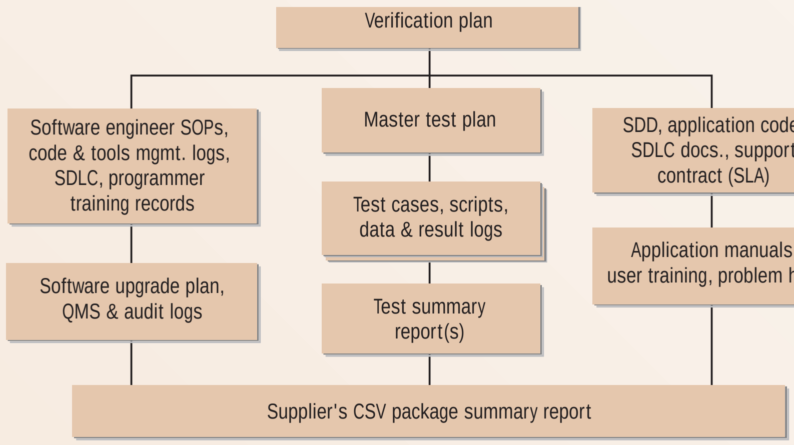

Verification plan
Software engineer SOPs, code & tools mgmt. logs, SDLC, programmer training records
Master test plan
SDD, application code, SDLC docs., support contract (SLA)
Test cases, scripts, data & result logs
Application manuals, user training, problem help
Software upgrade plan, QMS & audit logs
Test summary report(s)
Supplier's CSV package summary report

Figure 2. The software supplier's CSV package.

- ☐ **Verification plan.** Document describing the purpose, scope, approach, resources, and schedule of intended verification activities for a specific software development project (see Software Verification Plan box).
- ☐ **Software engineering SOPs.** Standard procedures for naming conventions, application interfaces, database calling conventions, documenting source code, testing practices, writing system manuals, release notes, online help formats, debugging process, change control.
- ☐ **Code and tools management log.** The first section in this tools management logbook binder contains a written description in text and in diagrams of the software engineering system configuration. It also includes a list of all major components of the development environment—server identifiers, associated disks, versions of database, language, and software tools, code management and testing tools, and overall network topology.
- ☐ **SDLC.** A written and approved description in text and diagrams to describe the software development life cycle with documented input and output requirements per phase and the roles responsible for review and approval to allow software to progress from one phase to the next.
- ☐ **Programmer training records.** CVs and training records to show that everyone on the software development team has experience and training appropriate to his/her role.
- ☐ **Software upgrade plan.** Documented business process for developing and releasing enhancements to the software in a controlled manner after first release to the market. Includes archive process for past versions with retention times suitable for regulatory requirements.
- ☐ **QMS.** Documented quality management system with management commitment to a corporate quality policy and QA organization to conduct independent internal audits of the SDLC activities for compliance with engineering SOPs and corporate quality policy and practices.
- ☐ **Audit logs.** Configuration management logbook binder section to record date, time, and participants in any audits or inspections of the supplier by internal QA, external clients, or regulatory authorities. Audit reports per se are kept company-confidential by QA department.
- ☐ **Master test plan.** Document that describes the technical and management approach to be followed at project level for testing a software product. Typical contents identify the items to be tested, tasks to be performed, responsibilities, schedules, and required resources for the testing activity. Document also describes the number and type of specific test plans to be developed for a given software product—unit test plan, system test plan, integration test plan, regression test plan.
- ☐ **Test case.** Document specifying the details of the testing approach for a platform feature or combination of features and identifying associated test scripts—system power up/down, system backup/recover, server connectivity to desktops/printers/remote sites across a network.
- ☐ **Test script.** Document specifying inputs, predicted results, and a set of execution conditions for an individual test. Includes one or more step procedures that describe keystrokes or other tester actions and provide log space for recording system response to test activity.
- ☐ **Test summary report(s).** Document(s) summarizing testing activities and results per test plan under the master test plan. Include(s) an evaluation of software quality based on testing results.
- ☐ **SDD.** Software design description—a blueprint or written model of the software system created to facilitate analysis, planning, implementation, and decision-making in building a software system.
- ☐ **Application code.** Computer instructions and data definitions in a form suitably designed to fulfill the specific needs of a user.
- ☐ **SDLC documentation.** All QMS documentation associated with the SDLC process—signed phase review documentation, testing documentation, code annotations.
- ☐ **Support contract (SLA).** Service level agreement(s) with clients purchasing a regulated application from the software supplier (see Figure 4).
- ☐ **Application manuals.** Instruction manuals to train IT/IS and/or system administrators in how to install and manage the software application system and instruction manuals to train users in how to operate the software in their GCP work process.
- ☐ **User training.** Online or hard copy training courses and materials for the client's users and their system administrators working with the software application in the GCP work process.
- ☐ **Problem help.** Documented process with defined escalation procedure for help desk technical support to resolve user problems with the software application.
- ☐ **Supplier's CSV package summary report.** Document summarizing all CSV package activities initiated under the verification plan and the results of those activities. It also contains an evaluation of the software application system's readiness to perform as intended and be operated in compliance with regulatory requirements. (See Software Package Summary Report box for recommended outline.)

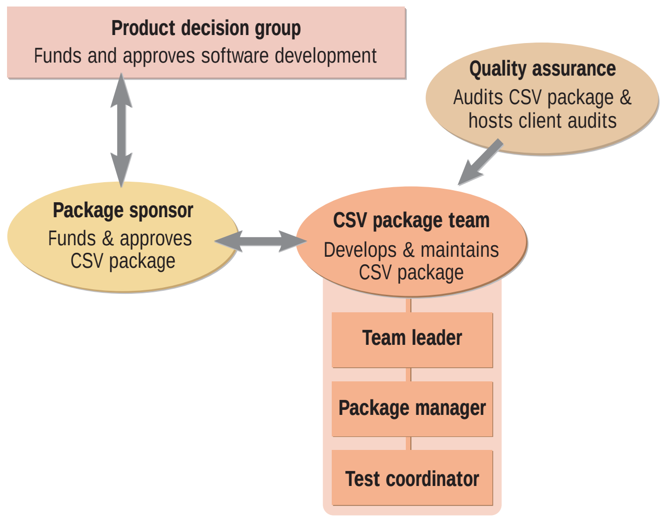

Product decision group
Funds and approves software development
Quality assurance
Audits CSV package & hosts client audits
Package sponsor
Funds & approves CSV package
CSV package team
Develops & maintains CSV package
Team leader
Package manager
Test coordinator

Figure 3. The software verification package team.

quality assurance department will audit the CSV package process and its documentation for compliance with the firm's quality management system and software quality standards.

The package sponsor is usually the senior manager responsible for the product's development effort. The team leader is assigned by the package sponsor and is usually the individual who carries the engineering responsibility for building a quality product. The team leader selects the rest of the team. The CSV package roles and responsibilities are shown in the Software Supplier CSV Team Roles box. These roles apply to the software supplier during development and also to the CSV package teams of the user group and their IT/IS department for validation of the software product in the work process.

It is extremely unwise to give in to the temptation to reduce the number of CSV package team members from three to one. With turnover in the workforce, it is only good business sense to have more than one person understand the software product and its CSV package well enough to defend it in audits and inspections. It is also good to assign roles to people whose jobs in the project are associated with creating key items in the CSV package—for example, project manager as team leader, quality control secretary as package manager, and software tester as test coordinator. In this way, package items can flow out of the direct work activities in the project rather than being seen as added overhead at the end. For very small projects, the team leader can choose to also carry one of the other two roles—but not all three.

Software Supplier CSV Team Roles

Package sponsor Provides personnel, budget, and equipment. Approves CSV package plan and package summary report. Assigns a team leader.

Team leader Identifies and leads a package team. Approves test plan and test summary report. Drives the package process and identifies ad hoc members as needed. Writes package summary report.

Package manager Drives item preparation, manages a package archive, and checks the quality of documents in production for their ability to pass audits.

Test coordinator Develops test plan and other test documentation. Identifies and trains testers and witnesses in formal testing practices. Manages formal testing process. Writes test summary report.

Ad hoc members Provide administrative support, specialty expertise, consulting, training, testing, or other support as needed. System size and scope determine the size of this component.

QA auditor Trains team on regulatory requirements for the system and audits the package for progress on plans and compliance with regulations.

Software development platform

Software applications are created using various configurations of hardware, operating systems, networks, databases, programming languages, design and coding tools, graphical user interfaces (GUIs), and testing tools. When developing critical software applications, it is important to document this platform and to check the application for any effects from changes made to the development platform.

Changing a database version, for example, may interfere with the performance of some database calls already programmed into the application; and these would have to be rewritten. Changing automated testing tool versions may result in certain prior test cases not executing as expected. Moving to a new GUI version may cancel the action of or hide the view of certain buttons previously programmed.

Replacing like-for-like components in a platform should not cause problems, but—as with generic replacement of branded drugs—some unexpected variations can emerge. Installation qualification testing should be performed on all components of the development platform. Checklists can be used to verify and document that all key aspects of the installation adhere to the specifications and recommendations of the component's manufacturer and that the component is appropriate for the development platform's configuration. The configuration itself should be documented and changes tracked in a configuration management logbook.

As the Documenting the SDLC Platform box illustrates, platform security, maintenance records, backup and recovery logs, and the disaster plan are to be documented in an organized way for software development platforms. It should always be possible to trace the logical and physical environment used for creating critical software.

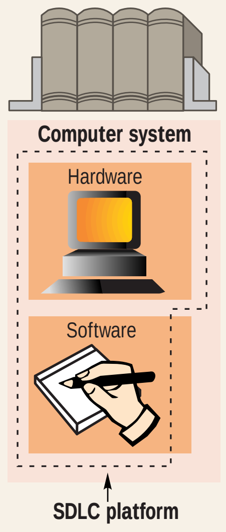
The escrow process is one major difference between user platform systems and software development platforms. The escrow process goes beyond the usual archive activity to specify that an official third party will have secured custody of a master copy of the firm's software product. In the event that the software supplier goes out of business, its customers have a defined legal right to contact the third party and access a copy of the source code for use in ongoing support of their operations.

Supplier and user SLA partnership

The service level agreement (SLA) for a critical software system used for GCP purposes is more than a commercial contract. It is a commitment from both sides for supporting the success of the application in its operational environment and compliance with regulatory standards. The more direct impact the software application has on the safety, efficacy, and quality of subject care or the integrity of safety and efficacy

Documenting the SDLC Platform

The software development platform must be traceable through careful documentation.



Configuration management logbook binder

- ☐ **Configuration.** System components and versions of hardware, software, languages, and tools.
- ☐ **Platform security.** Access rights, user privileges, software code manager, training.
- ☐ **Maintenance and support records.** Change records, service visit logs.
- ☐ **Backup and recovery log.** Disaster plan—daily, weekly, monthly, yearly.
- ☐ **Archive and escrow process.** Control of master copy of software and updates.

data in the trial, the more important is the need for an actively managed SLA. Examples of high-impact software in GCP areas would be software that calculates the amount of radiation treatment to be administered, or measures tumor reduction on digitized X rays, or tracks and reports serious adverse events.

An SLA establishes the mutually agreed-upon criteria and milestones for success of the software application during its operational phase. It discusses the ongoing needs for the user's work process and the supplier's ongoing needs for providing service and support. Both sides of any partnership have responsibilities. If the user group doesn't perform its half of the job (for example, allowing access and downtime for routine maintenance), the supplier's capacity for service and support is diminished and the application may be put at risk—if, for example, unresolved small issues lead to a major breakdown. Figure 4 illustrates the SLA process between software supplier and user team.

The number and type of topics to be addressed in an SLA will vary with the complexity of the application and the extent of its interaction with the work process. As the GCP work process changes, the software may have to be adjusted with interface modifications, functional enhancements, or performance improvement. Such fixes or patch releases then need to be retested under change control and reissued with relevant updates to the user training materials (release notes). While the user team retrains users, the supplier retrains the help desk personnel on query resolution for the new fixes and/or enhancements.

Documented control of software change on both sides of the SLA partnership is critical, and the tracking of problem resolution activities over time is important for analyzing trends and taking preventive action to ward off major system malfunctions. Regular reviews and reports of changes, problem trends, and continuing training efforts by the SLA partnership to the user team's management (business decision group) should keep the software application system in robust good health for GCP compliance.

Many of the SLA discussion points in Part 2 of this series ("GCP Validation of Platform and Infrastructure Systems," September 2000) can be adapted to the software supplier SLA situation. In addition, software suppliers are advised to closely study the sections on servicing and other relevant topics in the quality standard ANSI/ISO/ASQ Q9000-3-1997.[4] This is the latest version of the ISO 9003 standard that has been updated with annotations by the American Society for Quality (ASQ) and approved as an American National Standard. It discusses in detail what constitutes a strong quality assurance program for software development and support in a reputable software company for any marketplace.

It is important for all parties to remember that the goal of the SLA is the successful operation over time of the software application in the GCP work process. SLA success is measured by its

- controlled, reliable handling of GCP data in clinical research.
- protection of the system user and/or clinical subject from any hazard due to system operations.
- preservation of the integrity of GCP data throughout its collection, processing, storage, and retrieval by the software application in the work process.

Part 4 will conclude this series with a look at the role of quality assurance professionals in computer validation and the preparation and maintenance of all three

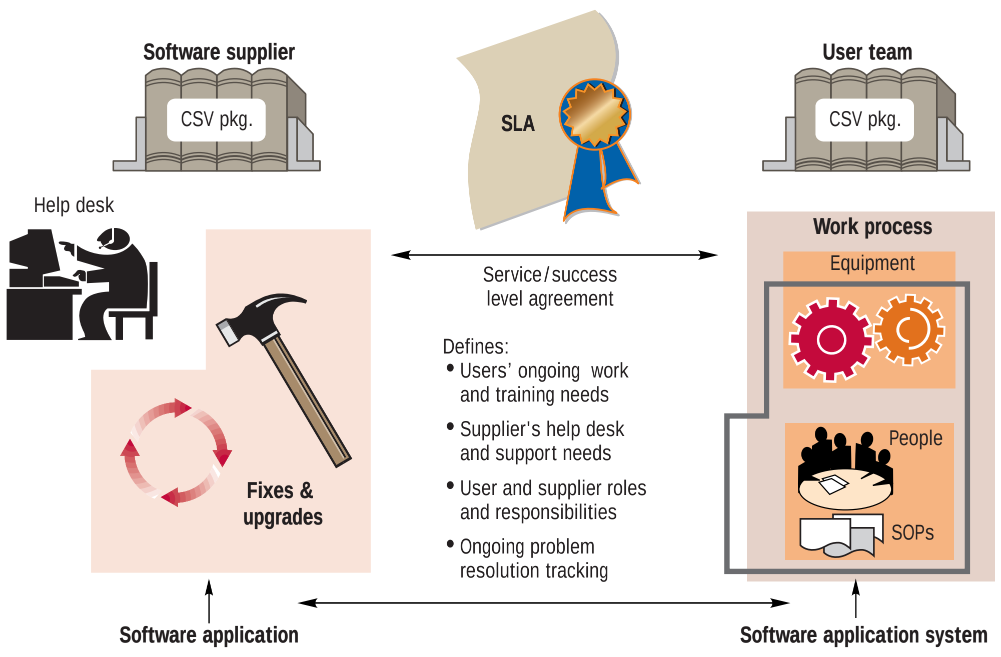
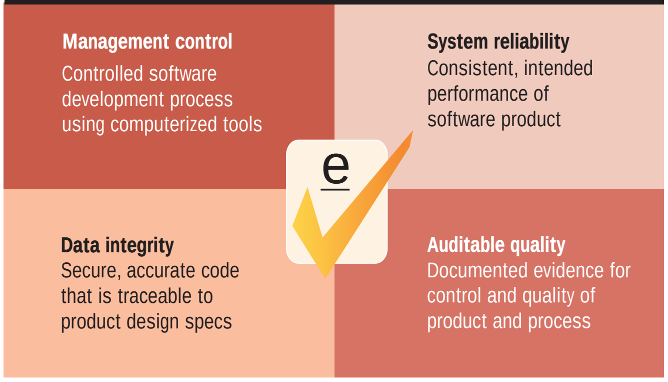

Software supplier
CSV pkg.
SLA
User team
CSV pkg.
Help desk
Service/success level agreement
Defines:
• Users' ongoing work and training needs
• Supplier's help desk and support needs
• User and supplier roles and responsibilities
• Ongoing problem resolution tracking
Work process
Equipment
People
SOPs
Fixes & upgrades
Software application
Software application system

Figure 4. The service level agreement (SLA) defines the ongoing partnership between the users and the software supplier over the life of the application.

Software supplier's verification goals


Management control
Controlled software development process using computerized tools
System reliability
Consistent, intended performance of software product
e
Data integrity
Secure, accurate code that is traceable to product design specs
Auditable quality
Documented evidence for control and quality of product and process

CSV packages described in the series. To be or not to be involved is the QA question. *How* to be involved will be the series' answer.

References

1. Institute for Electrical and Electronics Engineers, Inc., IEEE Standard Glossary of Software Engineering Terminology, Standard 610.12-1990 (IEEE, Piscataway, NJ, 1991).
2. Teri Stokes, "Computer Systems Validation, Part 4: Operating GCP Systems at Investigator Sites," Applied Clinical Trials, April 1997, 54–60 (available at http://pharmaportal.com/articles/stokes.cfm).
3. Institute for Electrical and Electronics Engineers, Inc., IEEE Standard for Software Verification and Validation Plans, Standard 1012-1986 (IEEE, Piscataway, NJ, 1993), pp. 12–19.
4. American Society for Quality, Quality Management and Quality Assurance Standards—Part 3: Guidelines for the Application of ANSI/ISO/ASQC Q9001-1994 to the Development, Supply, Onstallation and Maintenance of Computer Software (Quality Press, Milwaukee, WI, 1997; available for purchase at www.asq.org).

Teri Stokes, *PhD, is senior consultant and director, GXP International, 131 Sudbury Road, Concord, MA 01742, (978) 287-4393, fax (978) 369-5620.*