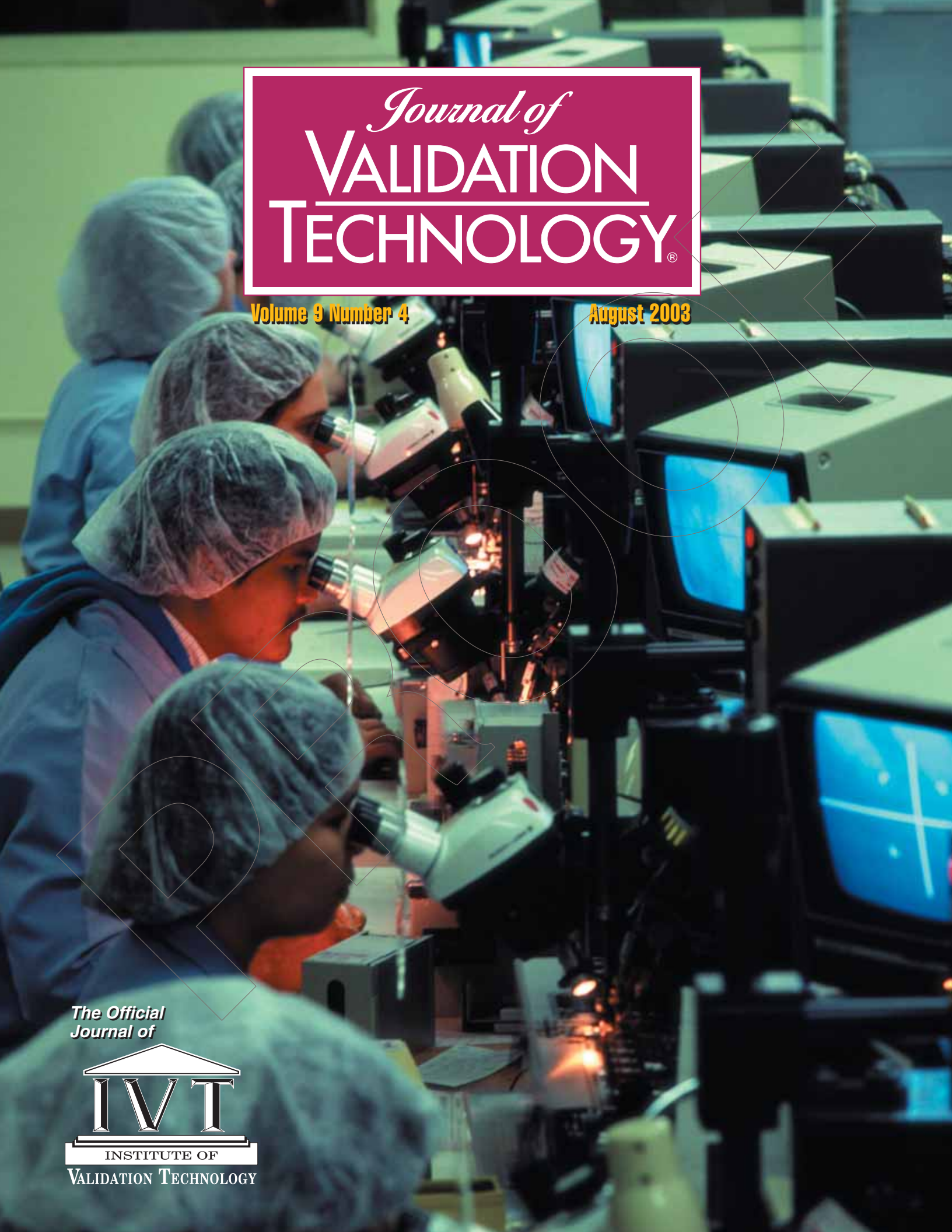


Journal of VALIDATION TECHNOLOGY®

Volume 9 Number 4 August 2003

The Official Journal of

IVT

INSTITUTE OF VALIDATION TECHNOLOGY

A Compliant Distributed Control System – A Framework to Manage Documentation Expectations

Doina Morusca
and
Mark Cupryk
Invensys Validation Technologies

Currently, the large manufacturers in the pharmaceutical and biotech industry are managing Distributed Control System (DCS) projects with different types of document deliverables and approaches to compliance. In some instances, these approaches even vary within the same organization. The objective of this paper is to provide a framework to help guide the reader to implement a compliant DCS. A general approach is provided with points to consider throughout the course of such a project. An emphasis has been placed on software testing and associated documentation tools. Checklists of typical projects have also been included to give the reader some tangible examples that are applicable to similar automation projects.

"If a VMP is prepared just for the FDA's sake, it will fail. A plan must provide both the compliance and project planning benefit."

To implement a compliant, robust, and effective DCS for a production facility is no simple task. From a business perspective, there are a variety of critical forces that may have different intensity and direction than those desired by the future owner of a DCS project. One driver could be that the organization needs the facility to be operational – yesterday. But the force may be further enhanced or weakened with the level of management commitment to support this project. Matters could be compounded with volatile environmental factors, such as the Food and Drug Administration (FDA) increasing system compliance awareness to ensure public safety and protection in the computer system arena. Competitors keep the pressure with their own success stories, and perhaps even by highlighting your own shortcomings. Your suppliers or vendors are all promoting their superiority in the business, and may leave you questioning their true objectives and goals. Finally, the most critical driver is the customer, and this is exhibited by a want of a product that is pure, safe, effective and, of course, cheaper than ever. These external forces, and how they could impact the project, need to be continually monitored, so that the proper actions can be applied to keep the project on track.

Once the external influences are identified and monitored, the focus can be narrowed to the project. Basically, the DCS is a large control system that enables decentralized control of plant process activity, whether in the laboratory, manufacturing, or packaging. Indeed, a powerful computer system, that in real-time, will continuously update and maintain

your plant data several times a second, this system has the capability to:

- Control and monitor numerous data points from 200 to 100,000+
- Execute special programmed logic
- Provide startup, shutdown, interlock, and maintenance control
- Alarm and manage events during the plant operations
- Trend, log, and report data
- Communicate with other controllers at the process level (e.g., Programmable Logic Controllers [PLC], Personal Computers [PCs] etc.) and at the business level (Enterprise Resource Planning [ERP], Manufacturing Execution System [MES], etc.).

At the macro level, the DCS can be broken down into the following three principal deliverables: hardware, software, and documentation.
Hardware can include:

- Workstations that perform computations, act as hosts for other stations, and perform the interface between human and machine
- Cabling, network, wiring, etc.
- Input/Output modules, which scan the instrument data
- Peripheral devices, such as media storage, keyboards, mouse, etc.
- Communication devices, such as printers, modems.
- Storage cabinets and junction boxes
- Process instruments, such as flowmeters, pH sensors, analyzers, valves, etc.

Software can include:

- Network and gateway management
- Configuration tools
- Operating system
- Human interface management
- Integrated control tools
- Customized packages, such as batch management, data historian

Documentation:

As mentioned, documentation in automation projects has yet to be optimized, both in terms of content and context of presentation. *Figure 1* depicts the major documents in a typical DCS project. One category is the foundation document, which represent the progress of the systems development, both in virtual and physical terms. Another category is all the planning deliverables associated with both short and long-term goals. These can range from tools to track how the risk will be mitigated on a long-term or strategic level, to how a particular test will be performed with the number of steps or actions. Finally, the last category is the proof of how well the plans were adhered to, and identification of where and why deviations took place.

Validation Master Plan

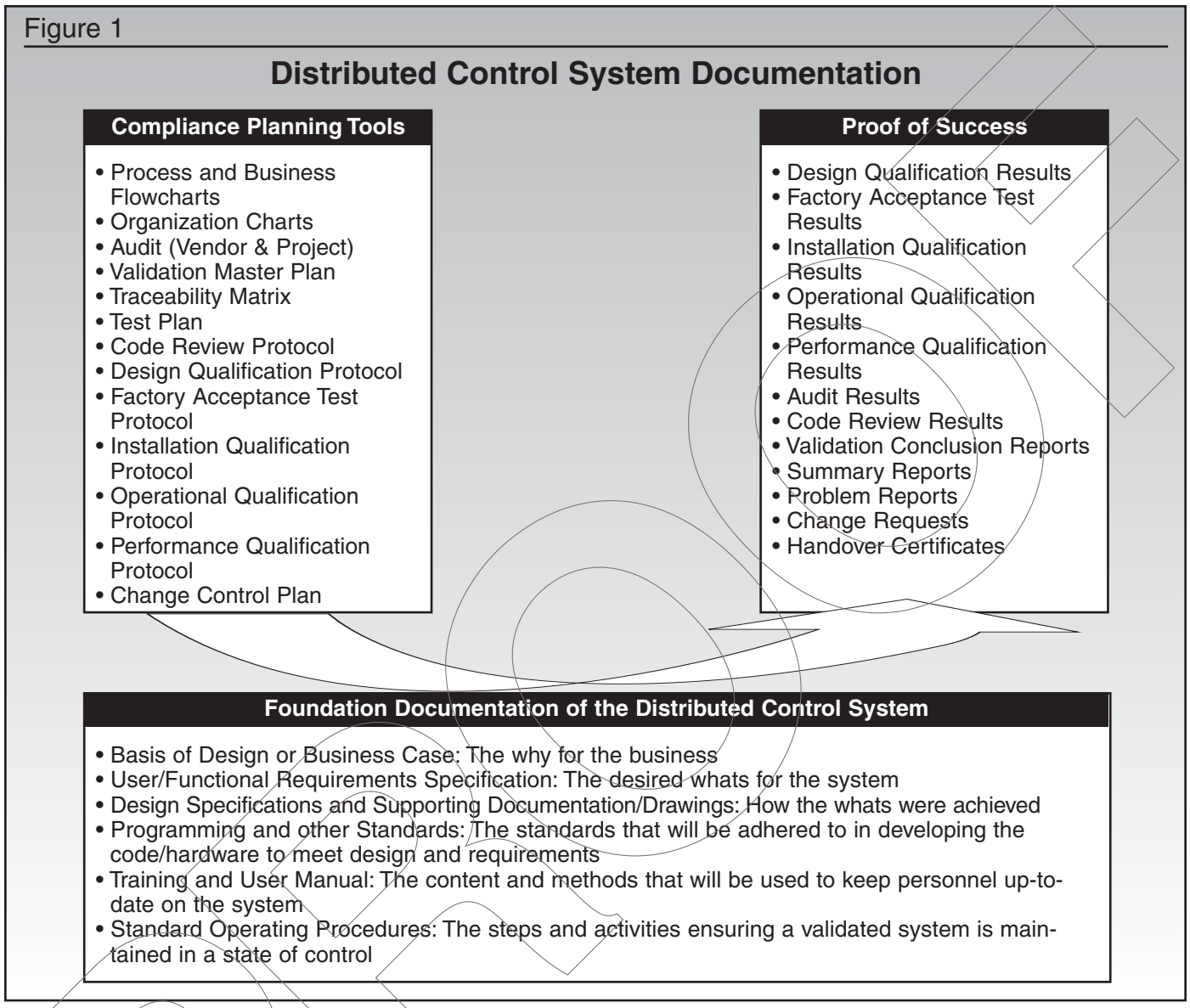
As in any project, too often the urge is to induce the design phase or even construction at breakneck speed. Typically, management confuses movement for actual progress, and, as a result, this misperception motivates a type of action-oriented behavior disregarding the value of a plan. Taking the time to better assess the future will ensure a clear goal and the required actions to make it happen.

The Validation Master Plan (VMP) is the most important project management tool for successful completion of the DCS project or any other type of pharmaceutical implementation, for that matter. Herein lies the project's genetic code that defines the project's activities, which will lead to the creation of all necessary deliverables for a compliant control system. When unraveled, the result should be a system with all the associated documents designed to produce a pure, safe, and effective product.

Success will be accomplished by having each detail clearly defined in the VMP. Leaving sections purposely general or ambiguous is certain to have tremendous negative impact down the line, because it will result in more confusion within the project team. Other examples of poor planning include no details as to what each tangible deliverable will be, and no timing when these will be produced. To make matters worse, sometimes even no resource is identified to do the work, review it, or approve it. The result is that personnel are not aware of their role, or their responsibility and what they need to be delivering; therefore, a plan with no substance is in place. If a VMP is prepared just for the FDA's sake, it will fail. A plan must provide both the compliance and project planning benefit.

The time invested upfront to have a few people

Figure 1

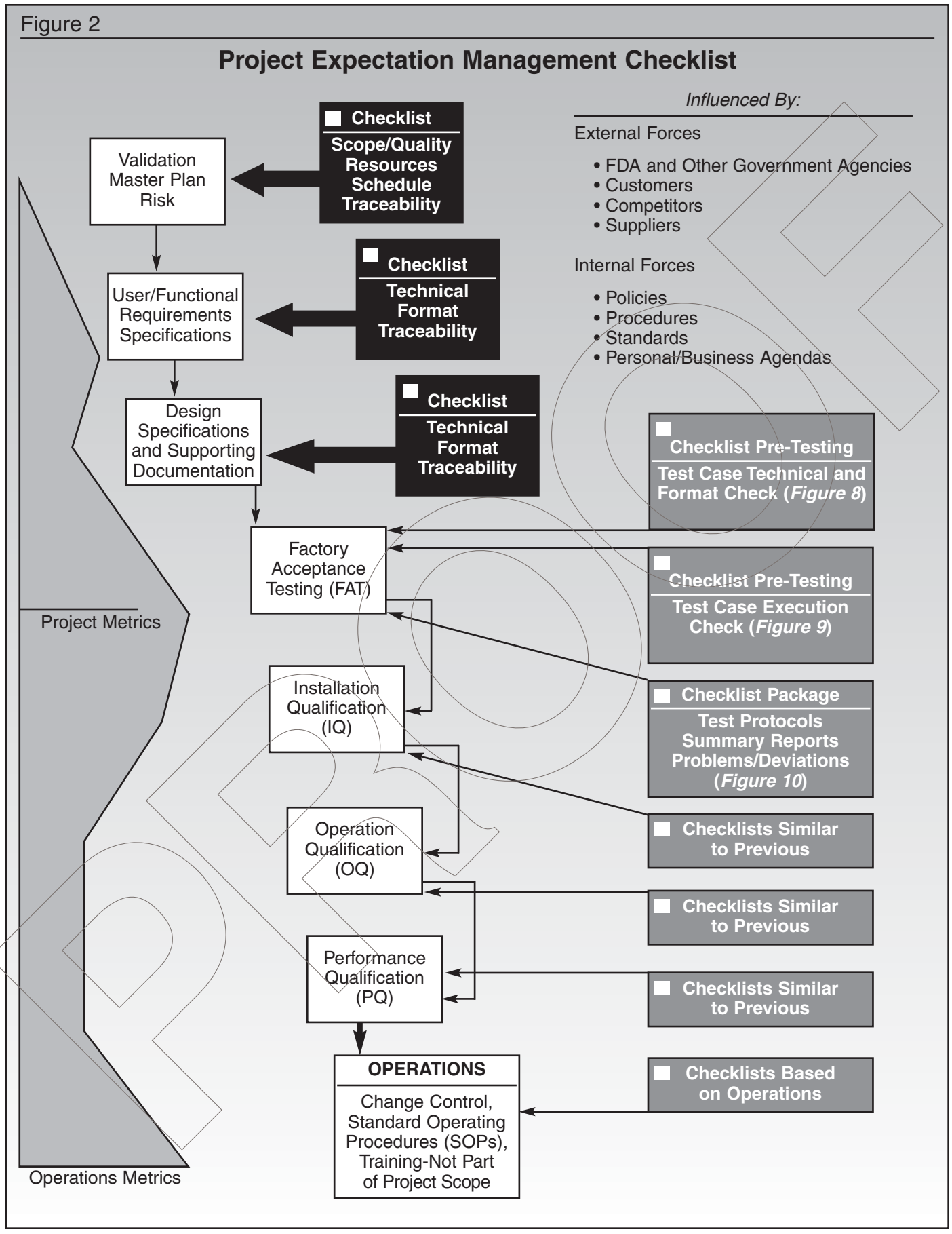


spend thinking about how to implement each detail of the DCS will provide high returns. The potential loss, if planning is not done, is to have a team of a hundred people not knowing how they fit into the big picture. Moreover, the expectations should also be defined at the technical level and documentation level. A master list of checklists to be prepared for each deliverable is recommended to serve as the baseline of what is important for each document at the technical and format level. Based on experience, *Figure 2* provides an illustration of some checklists required at different phases for the DCS project. The checklist will be influenced by the organization's policies and procedures, which will also be influenced by industry and the FDA. These lists will help to ensure that the whole team is aware of changes in expectations. Furthermore, if issues both technical and/or related to the documentation are found, they can be added to the checklist. If these issues are already present on the checklist, they may be indicative that additional training or detail for clarity is required.

It is important to plan to clearly define metrics to be collected, presented, and used for making changes throughout the project. Combining both estimated and historical data provides the team with the ability to track progress toward the goal, and make the required adjustments. The challenge is selecting the right data. Because of the uniqueness of each organization, the definition of a successful project is always different.

The documentation scope can be much better identified by preparing a table of the breakdown of the expected content for each deliverable. To draw an analogy, this is similar to an architecture block diagram

Figure 2


Project Expectation Management Checklist
Influenced By:
External Forces
• FDA and Other Government Agencies
• Customers
• Competitors
• Suppliers
Internal Forces
• Policies
• Procedures
• Standards
• Personal/Business Agendas
Validation Master Plan Risk
Checklist
Scope/Quality
Resources
Schedule
Traceability
User/Functional Requirements Specifications
Checklist
Technical
Format
Traceability
Design Specifications and Supporting Documentation
Checklist
Technical
Format
Traceability
Factory Acceptance Testing (FAT)
Installation Qualification (IQ)
Operation Qualification (OQ)
Performance Qualification (PQ)
OPERATIONS
Change Control, Standard Operating Procedures (SOPs), Training-Not Part of Project Scope
Checklist Pre-Testing
Test Case Technical and Format Check (Figure 8)
Checklist Pre-Testing
Test Case Execution Check (Figure 9)
Checklist Package
Test Protocols
Summary Reports
Problems/Deviations
(Figure 10)
Checklists Similar to Previous
Checklists Similar to Previous
Checklists Similar to Previous
Checklists Based on Operations
Project Metrics
Operations Metrics

identifying the number of instruments, junction boxes, marshalling panels, and interconnectivity of each. In the case of the documentation, a traceability matrix between the various sections of the key project document deliverables is a necessity. As for the hardware, we need a rough sketch of the architecture for assessing the major attributes of the project. Similarly, a breakdown of each typical document, such as requirement specifications, design specifications, Factory Acceptance Test (FAT) protocol, etc. with table of contents should be prepared, linked, and kept as a work in progress. Using the traceability matrix to frame the project deliverables and their major content areas will help to identify where items will be documented, and how these trace through the development cycle. The added benefit is that such a matrix can also be easily customized to track the project's progress, schedule, cost, and any other desired parameters. An example is shown in *Figure 3*, where the deliverables are sectioned out, the traceability between documents is established, and project attributes, such as completion and schedule, are tracked. Moreover, the completed documented deliverables provide a clear representation of the actual progress.

Typically, key project stakeholders in DCS projects involve the DCS developers, automation leader, process control leader, validation engineers, quality leaders, and operators. Requirements development should include all of these major players. Thinking out of the project constraint triangle of time, cost, and scope – needs to lead to thinking of the team. The selected individuals with the proper motivation and skill sets are going to make or break the DCS experience. They must be motivated to succeed, and well aware of how they can contribute to make the journey positive and memorable. Communication must be pre-defined and effective so that all are effectively engaged. Weekly meetings with key stakeholders must be organized for the duration of the project. A project management execution office is worth considering. This entity continuously energizes critical path activities. This will avoid getting entangled in non-critical activities. Planning to deal with different locations is crucial. In today's virtual project forum, there are still many barriers to high performing teams that arise due to geographic location, including cultural differences which can be present even in the same country. Face-to-face meetings continue to ensure better relationships, and a common objective is focused upon. With the high number of deliverables to prepare and circulate, workflows of the process and metrics need to be established to ensure that teamwork predominates and processes improve. These should be centralized and communicated regularly, so that everyone understands their role and actually sees their improvement and contribution to the overall project.

Risk planning must be done to determine the strategy and approach to be used for compliance. Compliance should not equate to identical effort across all areas. On the contrary, it is a well-thought out approach to ensure that the higher or lower-risk areas to product quality are mitigated with proportional efforts. High-risk areas should be identified in the requirements, and more extensive testing will be necessary.

A sound change control process is a must for success. This needs to be clear across all disciplines. Each change request should be assessed for impact across the entire project team's deliverables, and a methodology for communication of the change should be identified throughout the project. This will ensure that updates are made to all impacted deliverables. Needless to say, change control must get more rigorous as the project gets closer to completion, up until the DCS is formally delivered to production when the site change control SOP becomes enforced.

All training deliverables and Standard Operating Procedures (SOPs) must be identified. Furthermore, archiving and storage requirements for electronic records, paper, and software should be established.

One important step in the plan is to ensure that a detailed audit is performed with the DCS manufacturer and implementer prior to starting. The audit should verify both the manufacturing of the system and project procedures. Practices for both hardware and software to review should include the development process, maintenance, testing, manufacturing, documentation; training, subcontracting, and project management.

Requirements

A general requirement specification (the "what" the system is desired to do) document should be prepared for the software, as well as the hardware. Taking the time at this phase is always a challenge. Here, the push is "to convert the concept of what is needed to the tangible hardware and software," but the documentation needs to be treated with equal importance, since it retains history of the evolution of the physical reali-

Figure 3

Progress Traceability Matrix

Process Unit or Comp. Key Attributes	Requirement Spec.	Design Spec.	Factory Acceptance Test Cases	Factory Acceptance Test Cases	Installation Verification Tests	IQ Test Cases	IQ Test Cases	Commissioning Tests	OQ Test Cases	OQ Test Cases	PQ Test Cases	PQ Test Cases	Summary
	Preparation	Prep.	Prep.	Execution	Execution	Prep.	Execution	Execution	Prep.	Execution	Prep.	Execution	
Unit 1	1.1	1.1.1	1.1.1	1.1.1	1.1.1	1.1.1.	1.1.1.	1.1.1.	1.1.1.	1.1.1.	1.1.1	1.1.1	
Weight of Deliverable	25	20	10	5	5	10	5	5	10	5	NA	NA	NA
Progress	100	50	20	20	0	0	0	0	0	0	0	0	38
Current Variance on ETC (days)	0	5	5	5	5	5	5	5					NA
Current Variance on Effort (resource days)	120	-20	-40	-40	-40	-40	-40	-40	-40	-40	-40	-40	NA
Unit 1	1.2	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	1.2.1	
Unit 2	1.1												
Unit 3													
Unit 4													
Unit 5													
Integrate 1,4, 5													
Integrate 2,7, 9													
Integrate All													
General Hardware													
General Software													

Identifies each of the major sections of the documents, and how they will trace to each other

Weighing factor of each deliverable to better assess progress

Percent progress of document section

Bold borderlines identify the separate individual documents, along with each section that will be generated for review and approved

Variance on estimated and actual time for completion enables alarming for lateness or identifies slack

Effort variance between planned and actual will identify effort under and overestimates. Also will provide gauge to measure continuous improvement

ty. The sub processes for determining the requirements include identifying, building, organizing, and communicating with all key project stakeholders. Thinking of each of these steps will help finalize an agreed upon specification that will provide a baseline for future deliverables.

The general hardware requirements should consider all major components for operation, maintenance, and future expansion. *Figure 4* provides a list for typical content sections to be considered when identifying these requirements.

Similarly, the general software requirements should identify all items that are specific to the system, each process, and plant unit. For example, details, such as all critical alarms, are to be flashing red and valves green when opened for all process units depicted in the plant.

Figure 4

General Hardware Requirements Technical Checklist for Content

General Hardware Requirements to Consider

Processing Power

What you need:
- Central Processing Unit (CPU) (controlling center of entire data processing) attributes, such as the average time for reading data from computer storage; scanning of inputs per second
- Data transfer rate – amount of data that is moved from one place to another in a given time e.g., 10 megabits per second.
- Traffic load, idle time
- Memory (optimization of working versus storage)
- Number of Inputs/Outputs (I/O)
- Functions to be performed
- Battery
- Uninterruptible power supply (30, 60 minutes) etc.

Environment

What you need:
- Temperature range
- Pressure range
- Humidity range
- Power and grounding

Data Storage

What you need:
- Primary storage is data in Random Access Memory (RAM) and other "built-in" devices
- Secondary storage, such as data on hard disk, tape, and other external devices
- Capacity
- Access speed

Human Machine Interface

What you need:
- Type and number of workstations
- Physical security (keys, card, codes)
- General operation attributes (size, color, interconnections, etc.)
- Safety ratings
- Input devices, such as mouse, keyboard, microphone, scanners
- Output devices; such as printers, lights, etc.
- Time synchronization

Communication Interface

What you need:
- Type (serial/parallel, Local Area Network [LAN], Wide Area Network [WAN], fiber optic ethernet, etc.)
- Connections (RS 232, RS 443, coaxial)

Future

What you need:
- Standard plant policy (expansion 20%)
- Short-term potential expansion

Subsets of the software requirements or the "unit" requirements are those specifically defined for each individual process unit – for example, a unit requirement specification for a bioreactor or pure steam generator to detail the specific functionality desired for that unit. *Figure 5* provides categories for both general and unit specific software requirements that should be detailed when preparing these documents. Again for traceability, the unit specific requirement will be linked to the unit design specification(s), into the unit specific test case(s). Typically, a well-formed requirement should consist of a capability, condition(s), and constraint(s). For example, provide continuous stirring in fermentation vessel (capability) at an operating condition of 100 rpm (con-

Figure 5

General Software Requirements Technical Checklist for Content

General Software Requirements to Consider

Other Systems Interfaces

What you need:
- Data transfer rate and direction
- Handshaking
- Data points

Alarm Strategy

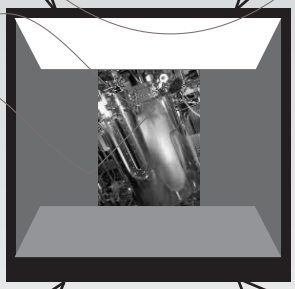
What you need:
- Priorities
- Limits
- Event planning
- Recording and printing requirements
- Disable and enable
- Error handling

Security

What you need:
- Password management
- Audit trails
- Backup and recovery
- Disaster plan

Future Capacity

What you need:
- Standard policy e.g., allowing extra 20% spare
- Known short-term expansions



Database

What you need:
- Number of points (Input/Output [I/O] list)
- Field devices description
- Instrument ranges and alarms limits
- Scan and log frequency
- Data source

Data Capture

What you need:
- Trend groups
- Data points
- Retention
- Manual intervention

Human Machine Interface

What you need:
- Graphics for plant, interlock, and alarms
- Function keys
- Refresh and response rates

Unit Specific Requirements

What you need:
- Graphics for boundaries of the unit
- Number and type of discrete and analog modules
- Calculation and timer modules
- Event and alarms modules
- Sequence and recipe details
- Process control strategy
- Integration with other units

dition) with a maximum of 120 rpm (constraint).

Typically, a set of requirements can be prepared, represented, documented, and communicated in the form of a drawing, database, spreadsheet, and/or document. In the current DCS market, each of these variations is present with its associated advantages and disadvantages. A key consideration is always identifying the linkage with the design and testing. This will ensure effective verification, minimize redundancy, and provide clarity to the overall team. Again, setting this up as priority one in the VMP is priceless. If this is not done, then when moving to the design phase, the design documentation should be prepared and integrated with direct links to each requirement.

Typical errors to be avoided include:

- Over specifying, for example, matching the requirements to a commercial system's design
- Over constraining, for example, adding constraints that are overly restrictive, but not required
- Qualitative, for example, stating open ended generalities without quantitative means to measure
- Including design and implementation information, for example, including design decisions, along with requirement statements.

Finally, metrics to consider at this stage can include:

- End user and project team report card
- Number of reviews and approvals
- Number of revisions and magnitude in pages and root cause for change
- Effort and calendar time to prepare

Design Specifications

Related Documentation and Drawings

Because there are endless ways of defining the requirements of an automated system, there are also a multitude of potential deliverables and approaches of documenting data to serve as the design information (the "how" the system will meet the "what"). In *Figure 6*, a list of typical design deliverables to be considered for the DCS project is presented in the frame of typical components (hardware, software, devices, equipment, personnel, and operating procedures) of a computerized system. How an organization integrates all of the highlighted items with check marks together is crucial for success of the DCS design, and must be determined upfront. Moreover, looking at the entire computerized system as one system will allow for better integration of all the associated documentation.

One of the primary reasons for better integration is the redundancy or lack of information that can result in an area, which can easily lead to inconsistencies during updates or corrections. For example, some firms include their control strategy on the Piping and Instrumentation Diagrams (P&IDs), as well as loop drawings and the design document. This means that the same information lies in three sources of documentation. There is clear opportunity for error, furthermore, if the primary source of data is not identified, the question then becomes, which document is to be considered "correct?" The optimization of this data is a great goal for streamlining this phase of the automation project, resulting in clear testing goals and limiting confusion.

Again, similar metrics to those mentioned in the requirements can be used to evaluate progress and success.

Standards

Details of the programming and other standards, such as the Institute of Electrical and Electronics Engineers (IEEE), Instrument Society of America (ISA), etc. to be respected must be clear to the entire team. Regular weekly reminder sessions and training on specifics of these should be demonstrated and documented in order to ensure that the team has developed code and hardware that is consistent across all units. Procedures should be provided for copying code, modules, scripts, files, etc. Centralizing code and ensuring repeatability allows for a more strategic approach to be utilized for testing. Otherwise, by "reinventing" parts of code each time, more rigorous testing must be executed to ensure that each new wheel is verified.

The methodology for identification of process devices should be clearly established, since these will be the link to many of the design deliverables, and changes to this can be very costly. The degree of detail on P&IDs should be established at the beginning, and these key documents can serve as the center of the change control process.

Design Qualification

Design Qualification (DQ) of the DCS is still in the infancy stage in industry, but it is a recommended step to ensure that the requirements have been

Figure 6

Typical Design Specification and Supporting Deliverables Checklist

Computerized System (In Its Operating Environment)	
Computer System	
Hardware (Including Instruments) and Software	**Equipment**
✓ **Design Specification (DS):** The detailed design description of the hardware and software (general and unit specific) provides the "as-built" representation, and is usually done in text format, but could be optimized in the database. In the past, this was communicated in a diagram format, including the device, loop, phase and interlock descriptions, sequences of operations, etc.	**Equipment List:** List that identifies the detailed attributes of the item, such as manufacturer, model type, process fluid, material, etc. Communicated in text, spreadsheet and/or database format for vessels, tanks, pumps, agitators, etc.
✓ **Loop Diagrams:** The representation of the control aspects of each instrument, valve, motor, along with auxiliary utilities, such as air and/or power. This was communicated in diagram format, but recently list and template representations are acceptable.	**Equipment Layout:** Represents the physical location of each component within the organization's plant. Usually communicated in drawing format.
✓ **Network and Communications Interfaces:** The representation of the hardware components and their interfaces. Typically, communicated in diagram format.	**Equipment Wiring Diagram (Single Line Diagram/ Motor Wiring Diagram):** Represents the source(s) and distribution of power to each major electrical component – motors on pumps and agitators.
✓ **Instrument List:** List identifying the detailed attributes of the item, such as manufacturer, model type, process fluid, material, etc. Communicated in text, spreadsheet, and/or database format.	**Mechanical Drawing:** Identify vessels cuts, ports, ratings, and structural supports. Includes isometric and/or piping arrangement drawings. Communicated in drawing format.
✓ **Software and Hardware List:** List identifying executive, application, operating, system support, communication software, as well as tools including name, manufacturer, and version number. As for hardware, it will include items, such as terminals, keyboards, I/O modules, etc. with manufacturer and model number.	
✓ **Instrument Location:** Represents the physical location of each component within the organization's plant. Usually communicated in drawing format, but not so popular in the pharmaceutical industry, due to smaller size of process devices and equipment – they are easier to locate.	
✓ **Panel and Junction Box Interconnections:** The representation of the signal wiring from the field devices to the actual I/O modules. Often communicated in diagram format, but also popular in a spreadsheet.	
Design Deliverables That are Related to Hardware, Software and Equipment	
✓ **Piping and Instrumentation:** The most important representation of the project. Identifies process flow, piping size/material, insulation, heat tracing, process devices, controls, and how they interrelate with the equipment. Communicated in diagram form.	✓ **Electrical Distribution System:** Represents the source(s) and distribution of power to each major component, such as motor control center/breakers, transformers, local standard distribution panels and emergency power panels. It includes the battery backup, uninterruptible power supply, and power regulation information. Also covered in this category is grounding, shielding and filtration. Usually communicated in a diagram format.
✓ **Process Flow Information:** Represents the major process, mass, and energy flows of the plant. Communicated in diagram format.	✓ **Spare Parts List:** This list identifies all additional components, which should be kept on-site as a back up in case of a failure. This information is usually communicated in spreadsheet format.
Personnel	**Procedures**
Training Manual	All procedures associated with the operations and maintenance of the DCS.
Training Certificates	

clearly documented, and that all links to the aforementioned design deliverables (*Figure 6*) are understood and verified. This provides a solid foundation for development, and for the next qualification phases. Again, a strategy of building the format into the overall documentation facilitates clear and traceable deliverables, and will make the DQ much more efficient and effective. This will reduce the time and money that would have been required to be invested in the testing phase.

Factory Acceptance Testing

Software

Some companies decide to have the entire DCS hardware and software developed and tested completely at the vendor's site. Typically, the set-up could involve testing and development at the vendor's site with regular visits, or a permanent presence from the future owners. The other extreme is to test and even develop all of the system at the future facility.

Both options have their advantages. Testing at a future facility gives you more control and influence over the software development. It reduces the geographic spread of the entire DCS team. Therefore, any problems found during testing can be resolved much faster, since the development/test team is always close-by, hence, facilitating clearer communication. This would not be defined as a FAT, but more of Site Acceptance Testing (SAT). The financial cost of this alternative is usually higher. Qualifying the software at the vendor's site has the advantage of being based on approved documents, and less on customer interpretations. It is also usually less costly, and the vendor has all the necessary resources in-house, which facilitates quick progress. This is the way most companies are driving their projects, since contractually, there is usually a major milestone (i.e., associated with major payment) when the system meets expectations, and is shipped partially or completely to the site. Hence, we have gone with the approach of an FAT being a phase of the DCS project. The test cases for integration of the process, and the sequence of when they will be tested, need to be identified.

The VMP should have already outlined the testing at the strategic level. The test plan should be prepared to discuss all the detailed logistics, with respect to testing. It should identify the testing scope, test case general structure, location, resources, relationship with other testing efforts (interfaces), schedule, protocol details, workflow of documents and software, problem reporting, simulation tools, and hardware to be used.

In the test plan, the software verification activities, such as code, design specification, and requirements specification reviews should be identified and outlined, with respect to scope and timing. Since most software problems are traceable to errors made during design and development, the verification steps are crucial. The foundation document for this activity is the traceability matrix, which establishes the linkage in all the deliverables, and identifies any gaps.

The code review is a documented activity that demonstrates the adherence to the requirements, design, and selected programming standards. At a high-level, the objectives are:

- Ensuring that there is no redundant code
- Detecting coding errors
- Verifying header information, file naming, program description
- Verifying that clear traceability is built into the code to the design and requirement specifications

The code review can be done by the customer or vendor. It is required practice that another person, rather that the one developing the software, executes the verification. If the vendor executes the code review, it is recommended that an inspection of their written procedures and approval of their results be obtained. After completion of code review, the software is ready for formal testing. The software can be identified using application, version number and date, developer, supplier, and DCS.

In the test plan, the software testing steps are outlined. The unit specific test cases are linked to each individual unit requirement specification. Ideally, boundaries should be the same, so that all requirements and design specifications are tested in one protocol, and all the standalone functionality is verified. In the case of the DCS project, the unit testing is often linked to the process unit, for example, a fermentation vessel, with sub units being the various device types and control modules.

The integrated test cases, and the sequence of when these are to be executed, needs to be identified. These

could all be tested at the end of unit testing, or better yet, should be planned for in parallel of the unit testing. Executing these integration test cases as soon as possible will identify problems earlier, and will assist in minimizing the propagation of these potential problems into other areas of the integration code. With respect to the DCS project, these tests typically include testing the transferring operations between process units and shared utility units that interact with all the process units, such as clean steam and Clean-In-Place (CIP). The goal is to ensure that the connectivity within units is sound, hence, verifying the architecture.

Finally, the last area is the system testing, which verifies the general software requirements. As mentioned, these should be defined in a separate document, and can be verified independently.

If at the factory, it is decided that some of the testing is to be completed later on site, the criteria and rationale of moving testing to the site should also be clearly defined. Moreover, this decision should not be made with intangible criteria, but should be planned with quantitative, such as a number of major and/or minor corrective actions that can be addressed later at the site as a function of time or cost.

Regardless of the decision made with respect to testing location, a procedure and workflow are essential in order to have a controlled transfer of the software from one location to another. This will assist in defining the team's role, and help project stakeholders know exactly what needs to happen.

The procedure and workflow for software transfer should detail:

- Type of recording media
- Personnel responsible to perform the uploading and downloading of the software
- Documentation that needs to be completed before the system can be tested and installed on site
- Steps that need to be followed in order to install the system
- How and what graphics, scripts, etc. should be stored on the recording media
- Where the media should be stored
- Backup procedure
- Change control of the software (version, date, change history)

With no definition on how to transfer software, the following are some of the risks:

- Not all the files are transferred from development, to testing, and finally to site
- There are changes to the software during the transfer that have not been documented and controlled
- The uploading and downloading of the software is different from person-to-person

When testing the software at the vendor's site, allow for a qualified test environment that will simulate conditions to those of the future production environment. What should be considered for the test environment include:

- How well the test environment aligns itself with actual production environment requirements
- Capacity and schedule – how many Inputs/Outputs (I/Os) the team expects to test each day for the project duration
- Future needs – this will minimize the need to requalify the testing environment
- Method of testing – are the I/O modules, actual field devices, simulation tools required?
- Testing interfaces to other systems – PLC, DCS, etc.

The qualification of the test environment will follow similar principles to that of the production hardware and software. If simulation tools are needed, the selected tools should also be qualified. Later, the simulation tools can be beneficial in production for training, and verifying the impact of requested changes to the system. Again, the requirement of using such tools is that a solid procedure with the respective documented training is complete, and a record to demonstrate that the tools perform as intended.

It is recommended that a different person than the one that developed the DCS software perform the testing. This idea is to keep more objectivity during testing. Two popular options for independent testing include contracting another company for the effort, or using a totally different team within the same development organization. Both are acceptable.

The FAT protocol should be created for use and reference during the Installation Qualifications (IQs) and Operation Qualifications (OQs). This document is a detailed description on how, who, and what will be tested. Although the form and content of the test pro-

tocol will vary from company-to-company, it should contain, at a minimum, the following elements, including:

- Scope and boundaries
- References
- Prerequisites
- Acceptance criteria
- Description of test cases, including expected outcomes
- Problem reporting
- Result review
- Closed-out report

Each test case should be written with clear objectives, in which predefined inputs, along with expected results, are documented. The FDA recommends quantified results, rather than qualified. These tangible results are more precise and can be easily reviewed. The test cases should ensure testing of normal and abnormal operations, invalid and valid type data, special values, initialization parameters, and logic paths.

The methodology of capturing actual data must also be considered. How the data will be presented as information within the test protocol needs to be well-defined, so that there is consistency throughout the testing, even with the numerous test personnel verifying and testing in parallel. Everyone needs to know details, such as – Where on the document will print screens be signed and dated?, or another case with respect to the print screen, how will the specific actual result be identified, – encircled, initialed, and dated with or without an explanation? The difference in focusing on these elements results in a much a higher level of presentation and professionalism.

Finally, understanding the specific expectations of the quality group is critical for success. This greatly relates to the type of format desired, as well as technical approach to be used in test case steps. An example checklist for test cases prepared in text format as seen in *Figure 8 – Test Case Preparation Checklist*, can be of great value to ensure consistency and understanding by all authors, approvers, and reviewers.

Creating masters for each type of test case can greatly reduce the amount of time necessary to replicate test cases, and reduce the potential for errors. Especially when qualifying a large system with over 10,000 I/Os, the benefits down the line are tremendous, both in terms of time and ensuring consistency. One could argue the use of a procedure or forms of document automation as being almost necessary, especially for later use in the production environment. Another option is to have master test cases stored in a library, which could be prepared and used for typical functionality, such as analog and digital inputs, valve, and motor verification. During testing, the actual results are documented and compared with pre-defined expectations. Problems are recorded, and corrective actions are proposed and executed.

For a DCS, problems can be categorized as the following:

- Inconsistencies with requirements, design specifications, programming standards, and test cases
- Database
- Interlocks
- Graphics

A post-execution test case checklist, like the sample provided in *Figure 9*, for verifying or reviewing an executed test case should be put in place to provide the reviewer with key items that must be checked and recorded. Once the entire package with its respective summary report are completed, another tool can be used, as in *Figure 10,* to provide the test team with a clear "things to do" list, hence, diminishing the chance of forgetting something. Because testing is moving at such a rapid rate to get it all completed, such a checklist will ensure that a thorough final quality check is performed before the product is circulated for approval. Another benefit of checklists is that these serve as tools to communicate expectations to the entire team. If new problems or issues arise, these can be easily incorporated into existing checklists. As the team grows, these lists facilitate understanding of what exactly is required for the organization. Having weekly "lessons learned" sessions can be a helpful and "friendly" way in sharing problems that occurred along the way.

Finally, no process can be defined and monitored for improvement without metrics. Selecting simple, but significant metrics, that will help the team focus on critical project parameters, is paramount. Typically, key parameters include number of issues and problem reports, which directly impact the project. The number of problems will add effort to the timeline and risk to the quality of the test protocols to be approved. The

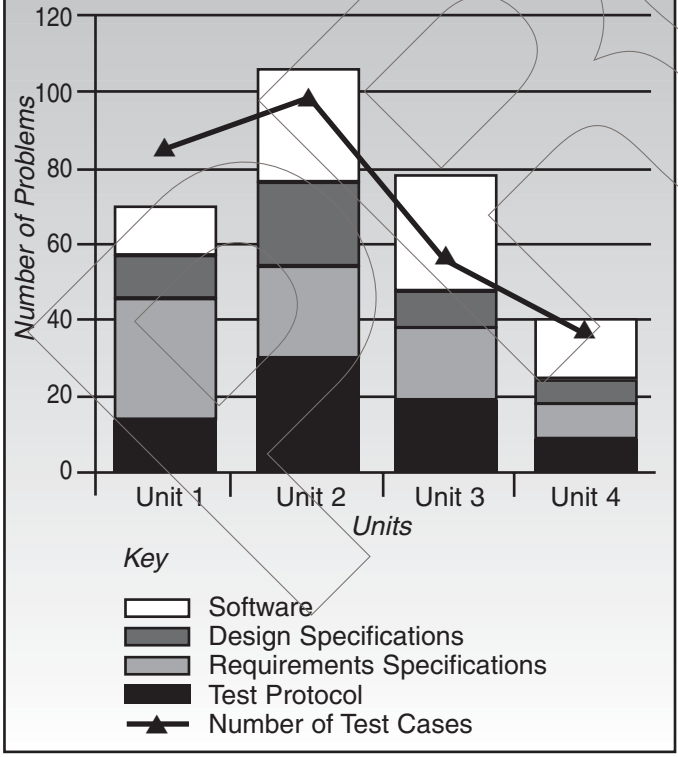
risk lies in that the corrective actions may not be exactly what both the approvers and the FDA expect.

Figure 7 provides a potential graph of problem/deviations on a unit testing effort, and a tentative categorization scheme for these. The complexity of the unit test is correlated to some extent to its size. However, the key metric to be used is the number of problems, because a root cause analysis can reveal significant areas of improvement. These problems should be communicated upstream to ensure that improvements are made to software that is currently in the pipeline. Typically, the second most important parameters are calendar and effort time, since most organizations want their systems in place as soon as possible. Regularly listing or illustrating the data at each step of the process flow provides a basis for areas of improvement.

In summary, metrics to consider at this stage and later qualification (IQ, OQ, Performance Qualification [PQ]) can include:

- Number of problem reports in each mentioned category
- Effort and calendar time to prepare, execute, and reexecute test protocols and test cases

Figure 7



- Number of reviews and approvals
- Number of revisions and magnitude in pages and root cause for change
- Number of code reviews and code changes
- Number of project changes

Hardware

Prior to shipment to the site, it is advantageous to verify that all the hardware functions are as specified. The DCS hardware should be physically connected, and using the approved hardware requirements and design specifications, the project team can systematically verify each documented requirement and design component as was presented in *Figures 4* and *6*. It is preferable that all problems and inconsistencies that arise be corrected at the factory. Like all heavier automation projects, it really is a case of "pay me now or pay me much more later." For both hardware and software testing, the goal should be to get as much as possible "correct" at the factory before shipping. Too many disasters occur due to a strong desire to get the merchandise at site in order to give the feel of progress to management. Resisting and communicating this will ensure that a robust system is delivered to the site.

A consideration for a successful FAT is to have a continuous customer presence. This should include both the technical and quality leads responsible to make it happen. With this in place, the project team will understand the quality expected, and the desired technical functionality in areas that may, for whatever reasons, have not been clear in the documentation. Moreover, clear rules of engagement should be defined, since there is also a risk of conflict between developer, tester, and customer. This will help provide agreement on how inconsistencies should be documented and resolved contractually.

Installation Qualification for Hardware and Software

At this phase, in the case of hardware IQ, almost a reexecution of the FAT occurs. This phase is really recording and ensuring that all the data to be considered "as-built" information is accurately representing the actual site installation. The same level of detail for the test protocol can be used as was done in the FAT. However, additional site pertinent tests will need to be

Figure 8

Test Case Preparation Checklist

Testcase #:	Reviewer:	
	Revision No.:	
1. Check the following information of the testcase		
Check purpose		
Check terminology – does the wording match with the standard list and all words capitalized?		
2. Testcase Technical Specific: complete the section that applies for the type of testcase		
3. General		
Check that there are no general statements used in expected results.		
Check that action statements are in the action cell only. Fonts are correct		
4. Steps Containing Printouts		
Check that there is no substep for the printout		
Check that all information verified on the printout is referenced in the same step.		
5. Check Page Setup Area		
Section 1 margins and layout are correct		
6. Check Header/Footer Formatting		
Check that the first page of Section 1 has no header or footer.		
Check that the rest of section has a header and footer		
7. Title Page		
Vertical positions are correct		
8. Revision History Page		
Format of page		
Title “Revision History” present on top		
Table column titles are correct		
Table column widths are correct		
Table borders are correct		
Latest revision date in table is the same as the last saved date in header		
Revision incremented and revision history entered		
Copyrights and trademarks block is present		
9. Testcase Description Page		
Format of title		
Standard text in acceptance criteria		
10. Test Table Formatting		
Table title is the same as the title from testcase description page		
Table column titles are correct		
Table column widths are correct		
Table column borders are correct		
Fonts are correct, step numbering is correct		
Substep lettering is correct		
Substeps lined up		
11. Comment and Signature Page		
Format of page (spacing and indention of tables)		
Table title is the same as title from testcase description page		
12. Language/Change Tracking		
Set the language		
Spellcheck after setting the language		
Check that diagrams are attached properly (file type, size, highlight, etc.)		
Ensure change tracking has been turned off, and all changes accepted		

Figure 9

Post-Execution Test Case Checklist

Test Case Number	
General	
• Last updated date is same as date of last revision	
• Print date is same or after last save date	
• The file name is the same on all pages	
• The author is the same in revision history, and header	
• Run # is filled in correctly	
• All executed steps are completely filled in. If a field cannot be filled in, then one horizontal line is placed through the blank and a comment is added to explain why.	
• All error entries are corrected, initialed, and dated	
Pass/Fail • Pass or fail is circled clearly • If pass, the actual results satisfy the acceptance criteria when compared with the expected results. • If fail, a Problem Report is listed	
Problem Reports: • All Problem Reports listed in test case are listed on the signature page • All Problem Reports listed in test case are listed on the corresponding Problem Report with the correct step numbers. • Problem description match between Problem Report and Test Case	
Reexecuted Test cases • All the steps that are listed for reexecution in the Problem Report are executed • All reexecuted test cases contain a comment explaining why they were reexecuted	
Unexecuted Steps • Unexecuted steps are crossed out • The executed steps reference a comment explaining why the other steps were not executed • If the rest of the test case cannot be executed, “End Step” is written on the last executed step and a comment is referenced to explain why Added or Deleted Steps • The steps that changed numbers are referenced correctly in the Problem Report and Revision History	
Comments Block • Each comment that is referenced in test case is numbered, initialed and dated • If no comments, “None” is written and initialed and dated	
Acceptance Block • If Passed with no typo Problem Report, yes is checked, no cell and Problem Report Number(s) cell are crossed out. The block is initialed and dated • If Failed, Yes is crossed out, no cell is checked, and all Problem Report(s) are listed in Problem Report Number(s) cell	
The testcase is signed and dated by the tester.	
• The date on the test case and Problem Report has to be the same • The date next to the error entry has to be the same as the executed step • Any notes cannot be dated later than the reviewer date without an initial and date from the reviewer	
Attachments Printouts • All items are circled, as directed by the step and lettered to reference the substep The following information is written on all attachments: • Test case name • Revision number • Run number • Step number • Page x of y • Date • Initial of tester	
The testcase is signed and dated by the reviewer.	

Figure 10

Execution Package Checklist

Package/Protocol name	
1. Package Original (on the cover there is a list with what is in the package)	
Test protocol	☐
Traceability matrix	☐
Testcase approval forms	☐
Test cases from 1 to X	☐
2. Package Execution (on the cover there is a list with what is in the package)	
Summary report filled in order: Summary report, final release, summary report for OQ, summary report for installation	☐
Test protocol filled in • Signature log	☐
Problem reports	☐
Check numbers – cross-reference with summary report	☐
Change request	☐
Test Cases	☐
Fax cover sheet	☐

performed, for example, the actual environment where the components are utilized will need to be documented to ensure compliance to requirements.

As for the software IQ, the installation items from the FAT, which are critical or new, are formally qualified at the site. In some other cases, the team will point to the test results of the FAT, which can be easily done if the rigorous rules of change control that were established earlier on were respected, and if the strategy of retesting was identified in the VMP.

Checklists for both technical and format requirements, similar to those in *Figures 8, 9,* and *10*, should be prepared to ensure that expectations of the IQ are clearly communicated.

The end result of the IQ phase is the formally installed software and hardware of the DCS.

Operational Qualification of the Software and Hardware

In the OQ, the focus shifts to a physical integration of mechanical (equipment, instrumentation, and valves), electrical, and the computer system to ensure that the software is operating as intended in its production environment. The final integration of the DCS is challenged with all other components, which should have completed their own FAT and IQ to ensure that this phase runs as smoothly as possible.

At this point, the testing of the DCS is driven by the operation of equipment. Production limits, equipment stress testing, critical path testing, and safety interlocks are some examples of user functionality. A milestone involving a handover of the qualified "computerized system" to the production team usually demarks the closing of this gate. Again, similar checklists to those in the FAT can be developed to ensure that the OQ requirements are understood and met.

Process Qualification

During process qualification, the "computerized system," that is hardware, software, and equipment, is ready to produce the desired product to get to market. Three validation batches are prepared, and the system has demonstrated its repeatability and robustness. The DCS should no longer be an issue, although a certain degree of adjustment may be required. This may lead to changes with potential impact on the mentioned deliverables that will need to be updated. With all documentation completed, the FDA is welcomed to perform the current Good Manufacturing Practice (cGMP) facility inspection of the facility, if required, as well as the Pre-Approval Inspection (PAI) in the case of a new product.

At the end, all deliverables should be filed in the documented location with the confidence that the correct things were done all along the way. The project team will feel the satisfaction of delivering a

compliant computerized system, and the production team will feel confident to provide and represent any relevant deliverable through any inspection or audit.

Conclusion

In any project, expectations are always a challenge to manage, especially in a large DCS project that involves numerous stakeholders. By implementing tools to assist the project team in the areas where there could be ambiguity in understanding what needs to be done, the risk of incomplete and inaccurate documentation will diminish. Checklists provide a baseline for expectations, facilitate communication, as well as training, and finally should target both technical and format of the documents, drawings, and other forms of content representation.

Traceability should be planned in the VMP to identify each major area of the project's deliverables. The upfront breakdown of all the major documentation will permit better alignment, enable better progress tracking, all the while ensuring that each area is thoroughly verified.

Planning for success upfront will eliminate expanding confusion down the development phase. The requirement specification is important, since it sets the foundation of the project, however, the planning of how and what should be provided in all deliverables ensures understanding and acceptance by all those involved in the project. A well-designed framework will ensure a structured and measurable approach to the DCS implementation. ❑

Article Acronym Listing

CIP:	Clean-In-Place
cGMP:	Current Good Manufacturing Practice
CPU:	Central Processing Unit
DCS:	Distributed Control System
DQ:	Design Qualification
ERP:	Enterprise Resource Planning
FAT:	Factory Acceptance Testing
FDA:	Food and Drug Administration
IEEE:	Institute of Electrical and Electronics Engineers
I/O:	Input/Output
IQ:	Installation Qualification
ISA:	Instrument Society of America
LAN:	Local Area Network
MES:	Manufacturing Execution System
OQ:	Operation Qualification
PAI:	Pre-Approval Inspection
PC:	Personal Computer
PLC:	Programmable Logic Controllers
PQ:	Performance Qualification
P&IDs:	Piping and Instrumentation Diagrams
RAM:	Random Access Memory
RS:	Requirement Specification
SAT:	Site Acceptance Testing
SOP:	Standard Operating Procedure
VMP:	Validation Master Plan
WAN:	Wide Area Network

About the Authors

Doina Morusca is a Project Manager for Invensys Validation Technologies. Doina has specialized in business and manufacturing systems, holding a Master's degree in Business Administration from Concordia University, as well as a Masters Degree in Education. She can be reached by phone at 508-549-6906, by fax at 508-549-4377, or by e-mail at doina.morusca@invensys.com.

Mark Cupryk is the Director of U.S. Operations for Invensys Validation Technologies. Mark holds a Chemical Engineering Degree from McGill University, and a Master's degree in Business Administration from Concordia University. He is also a certified Project Management Professional from the Pennsylvania Project Management Institute. He has worked in automation and validation for over 15 years. He can be reached by phone at 508-549-3761, by fax at 508-549-4377, or by e-mail at mark.cupryk@invensys.com.

Suggested Reading

- Angelucci A.A., Tomori J. "Automation Qualification – A Managed and Documented Approach." *Journal of Validation Technology*. Vol. 5, No. 4. (August). 1999. P. 342.
- Forstedt L. "Computer Validation as a Team Sport: Project Management Issues." *Journal of Validation Technology*. Vol. 8, No. 3. (May). 2002. P. 280.
- Amer G. "Checklist for Process Validation – Computer Document Collection Checklist." *Journal of Validation Technology*. Vol. 9, No. 1. November. (2002). P. 78-79.
- Chevlin D. "Verification and Validation for Embedded Software Systems for Medical Devices: An Introduction." *Journal of Validation Technology*. Vol. 8, No. 2. February. (2002).
- Wingate G. "Validating Automated Manufacturing and Laboratory Applications: Putting Principles in Practice." Interpharm. 1997. P. 365-383.
- Porter M.E. "Competitive Strategy – Techniques for Analyzing Industries and Competitors." The Free Press. 1980.
- GAMP 4. Guide for Validation of Automated Systems. December 2001. ISPE.
- FDA. *Final Guidance For Industry – General Principles of Software Validation; Final Guidance for Industry and FDA Staff*. Document Issued on January 11, 2002. U.S. Department of Health and Human Services, Food and Drug Administration, Center for Devices and Radiological Health, Center for Biologics Evaluation and Research.
- Thayer R.H., Dorfman M. *Software Requirements Engineering*. The Institute of Electrical and Electronics Engineers, Inc. (1997). P. 208-236.
- Craig R.D., Jaskiel S.P. *Systematic Software Testing*. (2002). Arttech House Publishers.
- Albert C.L., Coggan D.A. Fundamentals of Industrial Control – Practical Guides for Measurement and Control. Instrument Society of America. (1992). P. 477-535
- Grady R.B., Caswell D.L. *Software Metrics: Establishing A Company Wide Program*. (1987). Prentice-Hall Inc.

© Reprinted from JOURNAL OF VALIDATION TECHNOLOGIES, August 2003 AN ADVANSTAR PUBLICATION Printed in U.S.A.