# Lessons Learned in a Non-Regulated Software Validation Project

BY BRIAN SHOEMAKER, PH.D.

## INTRODUCTION

Not all software validation projects are created equal.

Though this statement is obvious to any team leader or consultant who has executed software validation in an FDA-regulated company, it can be difficult to grasp for managers more concerned with production schedules and challenges than with validation, and especially for managers in companies not directly regulated by the FDA. Risk can be higher or lower; team understanding of the need to validate can be greater or less, and technical specifics of the software in question may present a variety of challenges.

Because of these differences, every validation project presents a unique set of lessons. In the case described here, the company (call them "UnderOver Widgets") is not directly regulated by the FDA, but manufactures and supplies specialty products to a number of medical device companies. Though UnderOver's customers are entirely medical device companies, their technology falls within a larger industry not subject to FDA regulations. Driving the validation project was the medical device customers' demand that UnderOver become certified to ISO 13485. UnderOver – the "daughter" company of a group in their manufacturing field and geographical area – used software extensively in their manufacturing processes and quality systems. Many of the applications had been inherited or adapted when the company was spun out from its parent; but validation (a requirement specifically called out in ISO 13485) was to them a new concept.

This "fresh start" situation proved to be extremely fertile ground for a mutual learning experience. Key lessons, for the validation consultant as well as the UnderOver team, fell in three major areas: technical, organizational, and human-relations.

## TECHNICAL LESSONS: VITAL, BUT ONLY ONE COMPONENT

In software or IT projects, the devil is always in the details – and technical details provided some interesting lessons on this project.

### *Do Not Hesitate To Adapt Time-Honored Processes to Fit the Situation*

Nearly all the applications covered in the UnderOver project had been in place for months to years; fifteen of the twenty-one (listed in *Figure 1*) had been developed in-house.

The project plan was to follow the standard "V" model (*Figure 2*) to the extent appropriate or possible. Where applications were clearly off-the-shelf (such as the gauge calibration tracking program), validation would consist of developing end-user requirements and tests to demonstrate those requirements.

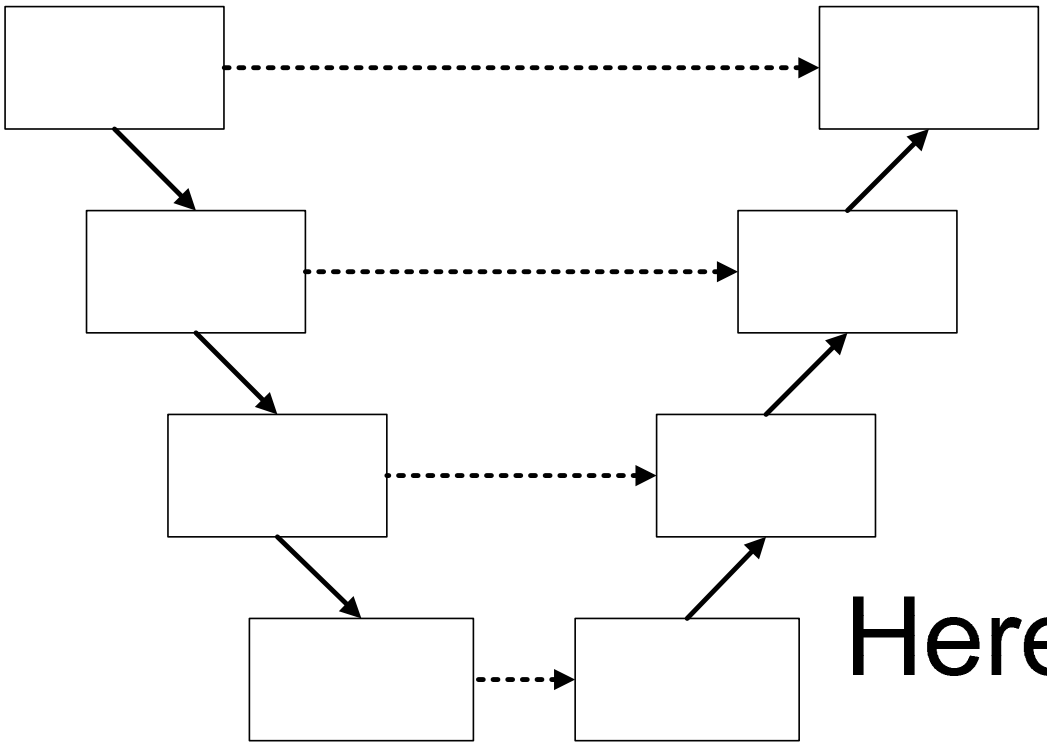
The in-house applications were another matter. No development documentation existed for any of UnderOver's custom software. Something would need to be created in the middle of the "V," between User Requirements and acceptance testing. Complete development documents would be unnecessary and impractical, since the applications were already in use. The decision was to document

***Figure 1***

List of Applications in the UnderOver Project

| Name | Function | Access | VBasic | Notes | OTS | Other |
|---|---|---|---|---|---|---|
| Devns | Product deviation approval | ✓ | | | | |
| ProdQC | Product QC test specifications and results | ✓ | | | | |
| Labels | Barcode label printing (production machines, scrap codes) | ✓ | | | | |
| CO_Sys | In-house change order application | | | ✓ | | |
| RD_Track | New product development tracking | | | ✓ | | |
| Com_ERP | Commercial ERP | | | | ✓ | |
| Prod1Spec | Product type 1 manufacturing sheet system | ✓ | | | | |
| TraceRetrieve | Product Lot traceability application | | ✓ | | | |
| DocIndex | Document master index | ✓ | | | | |
| Tester1 | Test instrument station software | | | | ✓ | |
| GaugeCal | Gauge calibration tracking | | | | ✓ | |
| Plan_Sampl | QC Sampling planner | ✓ | | | | |
| ProdSched | Production scheduling | ✓ | | | | |
| Tester2 | Product testing data acquisition | | | | ✓ | |
| Training | Employee training database | ✓ | | | | |
| ManuFlow | Barcode-enabled shop floor workflow | | | | | ✓ |
| ProdSys | Production setup sheets: revision/approval, lot-specific printing | | | ✓ | | |
| Patterns | Pattern design (output files placed directly on production eqpt) | | | | ✓ | |
| Prod2Spec | Product type 2 manufacturing sheet system | ✓ | | | | |
| TraceInfo | Enter key information for product-lot traceability | ✓ | | | | |
| PP_Sched | Schedule and track post-production processing of product lots | | ✓ | | | |

***Figure 2***
The Standard V Model




the "as built" design of these applications, to serve as a baseline for subsequent change control.

Because thirteen of the custom applications were either Microsoft Access® or Lotus Notes® databases, documenting their design required more than an annotated code listing. Fortunately, several tools will generate complete reports of all tables, forms, queries, reports, modules (program code), and macros (if any) in an Access application; a similar utility exists within the Lotus Notes development environment. These outputs could be automatically generated and printed to PDF, and archived to capture the complete design of the database applications.

It also proved necessary to adapt the concept of "installation qualification" (IQ), to provide useful information for this project. The applications were already in place, so a detailed procedure to confirm that they were being installed correctly would have no meaning. However, documenting the specifics of how and where the applications were installed would have considerable value for future software maintenance. Instead of performing IQ, a so-called Configuration Specification was created for each application, to document anything a programmer or information technology (IT) specialist would need to know in order to reinstall, maintain, or decommission the application. *Figure 3* lists examples of the types of information to include in these Configuration Specifications.

In both the "as built" design documentation, and the Configuration Specifications in place of installation qualification, this project "bent" the classical validation products – and in so doing, fulfilled the project purpose.

***Be Ready to Delve into Technical Specifics, even if these Should Be the Province of Developers and Architects***

Consider the shop-floor workflow application (dubbed "ManuFlow" for this discussion) pictured in consisted of an off-the-shelf "container" with drivers for barcode scanners and a desktop interface, all within

**User Requirements**
(technology independent;
end-user language)

**Functional**
**Specificatio**
(selected implement
developer langua

***Figure 3***

Information to include in Configuration Specifications

- Version of any underlying system (Access, Excel, Lotus Notes)
- Configuration options (where applicable – such items as default file-save directories, user security settings, or compatibility switches)
- Computer or network location where the application is installed
- Data files or database tables the application reads / writes
- External data required for security (if applicable)
- Database links, if any, needed for the application to function
- Resources required on the user's station (e.g. client-side program, browser plug-in, mapped drives, shortcuts)

which any given company had to build its own suite of workflow scripts. The "container" was a commercial off-the-shelf application, but all the functionality of the system resided in the custom-developed scripts. Complete User Requirements for these scripts were virtually impossible to build from user interviews, since (a) the manufacturing floor users were too close to the functioning system, and had difficulty expressing what the ManuFlow application should do; and (b) the scripts, which had been inherited from the parent company, underwent extensive revision (in part to clean out unused scripts and code) in the course of the project.

Determining both the user requirements and overall design of the workflow script suite became an exercise in reverse engineering. The IT director provided automatically-generated flowchart diagrams for all of the scripts. From these diagrams, the "connectivity" of the scripts could be determined (which ones comprised the main menu, which ones were called by the main selections, and so on down the hierarchy – see *Figure 4*), and the general actions occurring in each script could be puzzled out. The developer provided brief synopses of the scripts, but deducing the important logic tests and user inputs required studying the flowcharts in detail. This kind of down-in-the-code study is not typically expected of a validation consultant, who may or may not be familiar with the program language, but for this project it was vital.

***Help Solve Specific Technical Issues Where Necessary***

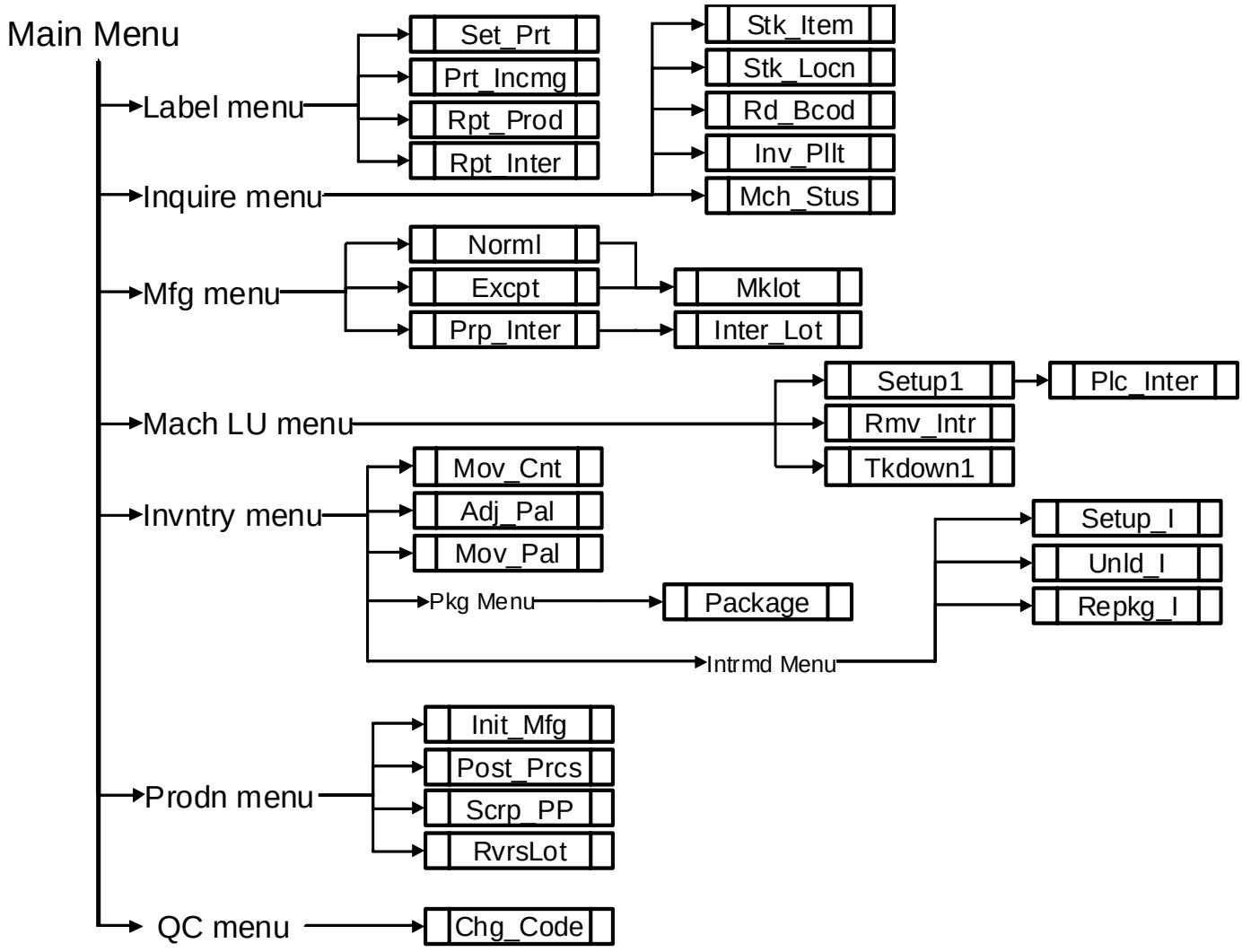
Several times in the course of the UnderOver project, it was necessary to help the project team see that a certain output was manageable, and not some insurmountable obstacle.

Creating design documentation was a prime example. The UnderOver team leader at first quailed at the task of documenting design of the Access databases. After researching Access documentation tools, it was possible to recommend several possibilities, and to list the essential information such a tool would need to provide. With these suggestions, what seemed unattainable became a fairly straightforward task.

Once the Access examples had been generated, the Lotus Notes developer could see the type of information that would be needed, and employed built-in developer tools to create equivalent outputs for the Notes applications.

***Figure 4***

Connectivity of the ManuFlow Scripts




***Learn the Client Systems, and Adapt to the Client's Tools***

All software in the project in some way affected the design, manufacturing, or quality control of UnderOver products. Systems involved many different Access databases; some were freestanding, but others interacted with the resource-planning system (labeled "Com_ERP" in *Figure 1*). Scheduling affected work order creation; product work orders would drive demand for intermediates made from the incoming material, and so on (*Figure 5*).

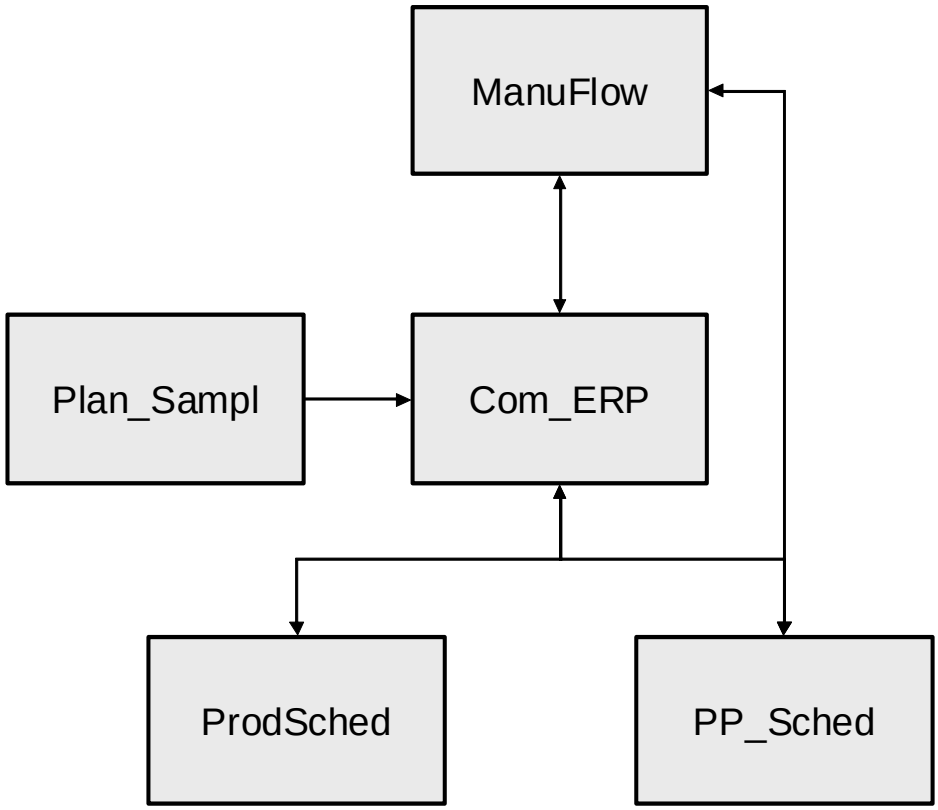
The Lotus Notes applications were similarly interdependent: at the end of a development project, RD_Track could create a change order in CO_Sys to release the new widget into production, and preliminary setup sheets generated in RD_Track could be ported directly into ProdSys, as shown in *Figure 6*.

***Do What is Necessary to Understand Fundamentals of the Client's Technology***

Inevitably, understanding the software required comprehending the terminology and the manufacturing processes. Learning about the manufacturer's processes

***Figure 5***

Interdependence of ERP / Access Databases




proved to be one of the project's more intriguing challenges. Incoming material arrived in units with one name, and had to be repackaged to an intermediate form for final production, using a setup with a specific name. In the manufacturing process, a specifically-named action indicated both the process of removing product from the machine, and creating a lot. *Figure 7* depicts the overall flow of UnderOver's production processes. Each form and each process had a unique name, specific to the industry.

Besides the technology of UnderOver's product, more mundane terms provided a chance for misunderstanding. Where many companies maintain standard operating procedure documents they call SOPs, UnderOver's quality system consisted of Quality System Procedures and Work Instructions – mention of "SOPs" proved confusing. Once this difference was discovered, project plans and weekly reports were modified to refer to quality documents using UnderOver's terms.

Though the technical lessons were helpful in moving the project forward, it was clear from the earliest requirements discussions that more than technical learning would be necessary for success.

## ORGANIZATIONAL LESSONS: HELPFUL FOR THE PROJECT AND BEYOND

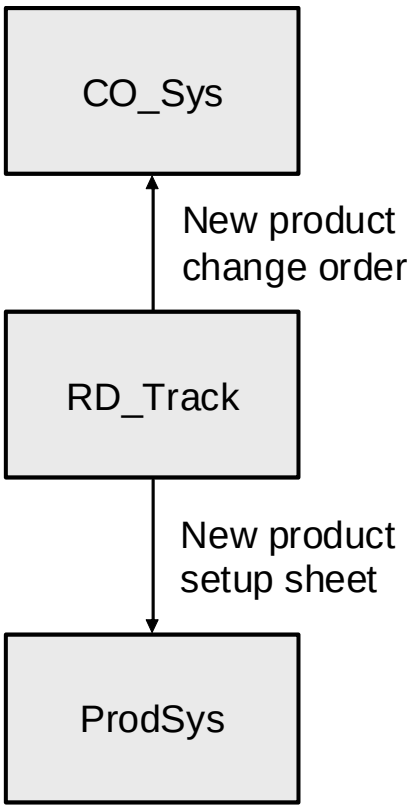
Several lessons from the UnderOver project, here called "organizational" lessons, fell more in the realm of planning, tracking, and communication.

### *From the Beginning, Be Aware of What is Driving the Validation Project in this Context*

Although UnderOver is not subject to FDA regulation, its medical device customers are. Everything in the project – documenting software requirements and design, establishing software change control processes, creating a

***Figure 6***

Interdependence of Lotus Notes Databases




problem-reporting mechanism – could be justified for classical process improvement, and in the long run, process improvement and its effect on efficiency and product quality were the "real" reasons for the project. However, behind all these good reasons was a single immediate motivation. The customers had demanded ISO 13485 certification, which spells out validation of software. UnderOver's business was at stake, and with it the team members' jobs.

***Develop a Clear Plan and Show Progress against it***

A written plan, where the team agreed on the applications to be included and the roles of the members, started off the project. All training sessions included a flowchart of the overall project, with the current point clearly indicated (*Figure 8*). At project's end, the final report referenced this initial plan, noting not only what had been completed but also the deviations (applications removed and added, approaches modified in the course of the validation).

Though it was helpful to provide high-level reports, after a few weeks, several team members asked to see a more complete overview of progress (a component of the next lesson).

***Communicate Often***

Keeping the team members informed was essential throughout the project. How far the work had come and how far it had yet to go, when the next site visit was scheduled, what issues needed to be addressed: all were featured in a brief weekly update. After completing several User Requirements documents, a table was added to the weekly report to list the applications being validated and the status of each document (requirements, configuration specification, design, test procedures). A glance at the table quickly told team members both the work which had been completed and the tasks still ahead.

Eventually, the UnderOver team leader asked that the company president be copied on these weekly reports. This visibility to top management as well as to the project team helped immensely in planning activities, keeping focus on the issues, and allowing team members to see and celebrate how far they had come.

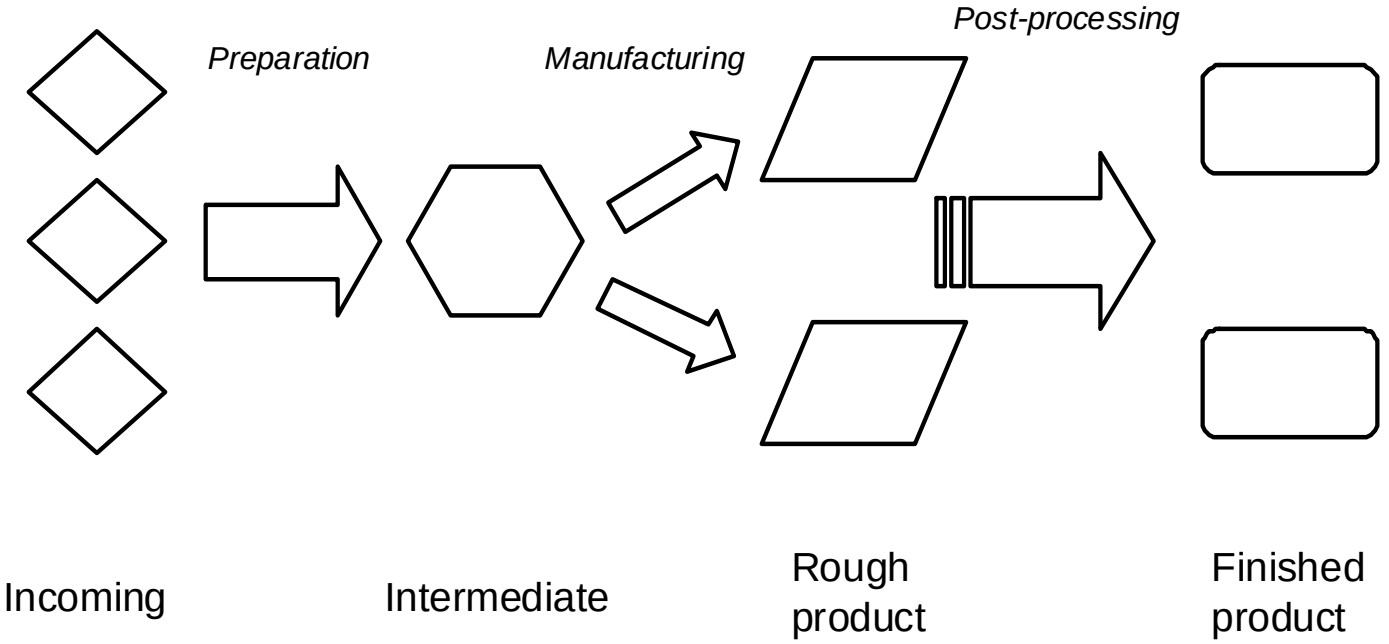
***Educate, Educate, Educate***

A Computer System Validation (CSV) training session started the project, and a refresher session followed at the eight week time point. As team members came into the project, they reviewed these presentations. Just prior to the team executing their software test procedures, still another training session provided them specifics of how to perform and document the testing, while reminding them of the project's basis and of the overall plan.

These training sessions were helpful, but far from sufficient. Team members, though dedicated, were simply not accustomed to the extent of documentation required by the standard and by their customers. Every conversation with one of the team members became an opportunity to explore questions and clarify the need for validation – and these one-on-one discussions proved essential in helping the team reach the right comfort level with the project.

***Figure 7***

Overview: Phases of UnderOver's Manufacturing




***Trust, but Verify***

This lesson is unique to the testing phase, but definitely not unique to UnderOver's project. No matter how carefully the instructions and the expected results are described in a test procedure, it is always possible to misinterpret them. In a number of cases, the tester believed that actual results disagreed with expected results and had to be marked as "fail"; some of these were true failures, some resulted from performing the test incorrectly, and some showed that the corresponding requirement was erroneous or had not been implemented. *Figure 9* lists several cases where an initial "fail" result was changed to "pass" (or "not applicable") on review. Of twenty-five cases marked "fail," only eight remained after review (of course, all were explained in the project final report).

Happily, the UnderOver tests did not include any cases where the tester counted a result as a "pass" but in fact the result should have been marked "fail." These have arisen in other projects, and are often very difficult to communicate to the tester or developer (to the point of causing disagreements within a project team).

## INTERPERSONAL LESSONS: DIFFICULT TO DEFINE, BUT VITAL TO SUCCESS

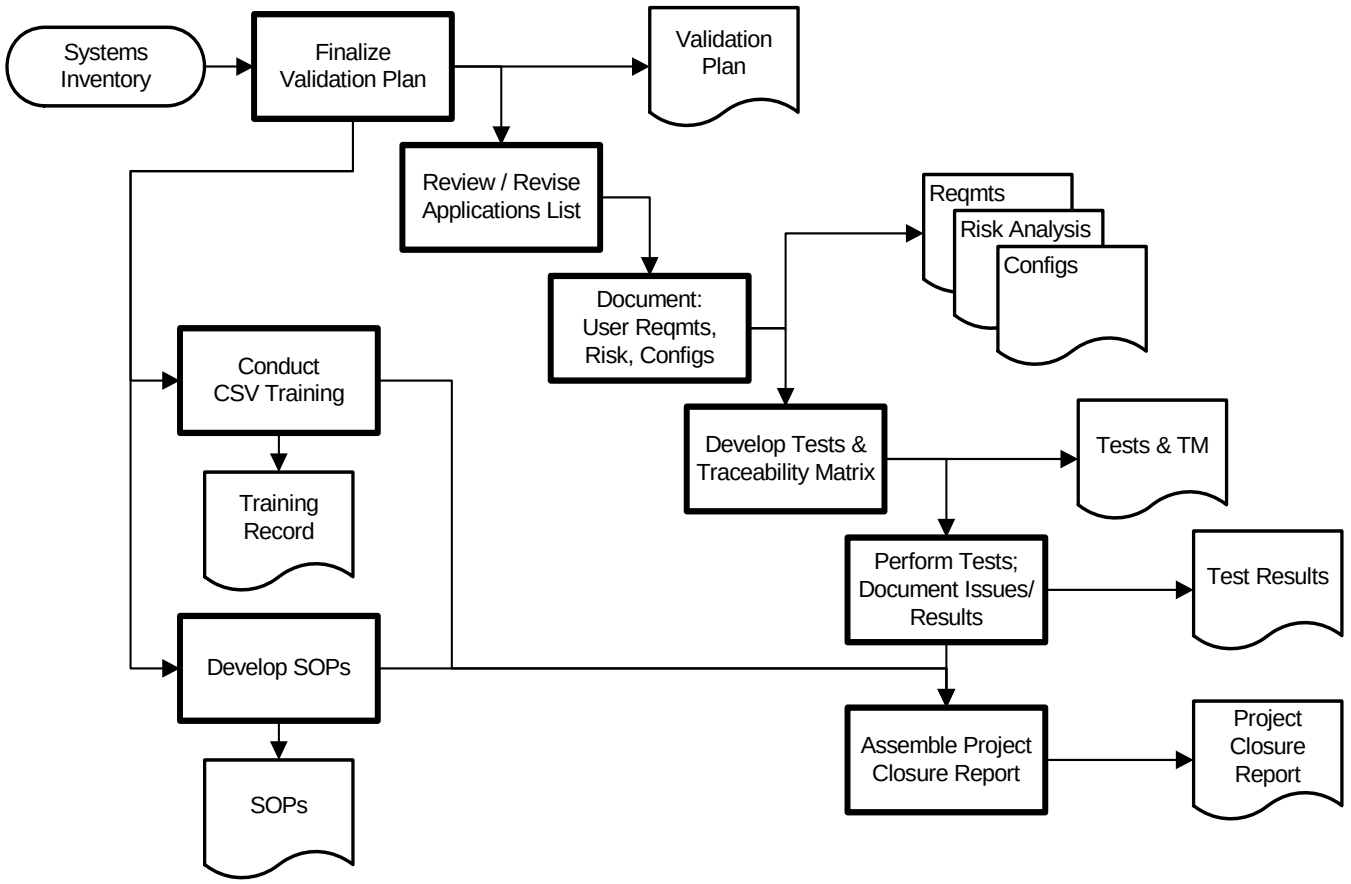
Computer software is working machinery created from pure "thought stuff" – both ethereal and practical, but completely objective. Human beings need to use that machinery, however – and working with a team of human beings on the task of showing that the thought stuff is correctly built, taught several powerful lessons in relating to human beings.

***Learn How to Listen***

This project team taught the validation consultant some crucial listening skills. During the very first site visit, as key users described the Prod2Spec application (see *Figure 1*), important statements for the User Requirements emerged either as logical conclusions or as implicit assumptions. Restating these apparent requirements ("So what you're saying is that the program needs to keep track of XYZ – is that correct?") allowed refining the points that would be documented. In only one case

## *Figure 8*

The UnderOver Project Plan, in Flowchart Format




could the person interviewed express an application's user requirements without assistance – and that person had developed the application.

Simply listening also contributed to the testing phase. More than one team member asked for help while executing a test procedure; these requests identified a number of script errors or true application failures. In one case, the frustrated tester working on the ProdSched test pointed out a confusing section, talked through a series of steps, opened the dialogs described in the test, and answered her own question without the test coordinator ever saying a word! Her comment: "I guess all I really needed was to have you listen to me."

### *Each Individual on a Small Team Has a Different Communication Style*

Some team members could write a requirements document with very little input. Others could describe their application verbally and provide nearly enough information to deduce the requirements. Still others had to show the user interface, live or as screen captures, in order to explain what they expected the program to do for them.

Similarly, some could read the numbered User Requirements and judge whether these were correct; others needed to talk through the document with reference to the program itself, to provide feedback. Each application, and each team member, required a different type of communication to make sure the documents were correct – and in the end, those few errors which remained either in the User Requirements or in the test documents, were the result of still less-than-perfect communication.

### *Show Patience as Timeframe and Priorities Change*

Throughout the project, the UnderOver team was clearly stretched. The ten participants were nearly the entire salaried staff at the company; all of them needed to address not only their everyday work but other priorities in addition to software validation. Customers visited the

***Figure 9***

List of Initial Test Failures and Reasons for Changing to Pass

| Applicn | Sec | Step | Reason | Comment | Revise |
|---|---|---|---|---|---|
| Labels | 4 | 7 | No way to select just eight codes; printing the codes gives everything on the list | Requirements error. | Pass |
| CO_Sys | 4 | 15 | Rejected change notice is not automatically closed. | Requirements error. | Pass |
| CO_Sys | 4 | 41 | After a change order is approved, a user cannot edit a task assigned to another user. | Requirements error: only the approver should be permitted to make such modifications once the change is approved. | Pass |
| ProdSys | 4 | 8 | Production type 1 setup sheet was created, but data were not pulled in from sample request. | Tester error. Using a different command successfully created the setup sheet with the sample request data. | Pass |
| DocIndex | 1 | 2 | Work Instruction for DocIndex needs to cover some topics in more detail. | Changed to Pass with comment that more exact detail was needed in the work instruction. | Pass |
| Tester2 | 3 | 4 | Application does not give option to print a single result or all results for a specific sample. | Requirements error. The “all results” option is not essential to the application. | Pass |
| Training | 4 | 12 | Cannot change employee ID. | Requirements error. Each employee has a unique ID, which cannot be changed. A new ID is a new employee. | Pass |
| TraceInfo | 4 | 5 | Application permits saving incoming material lot with empty vendor ID. | Script error: instead of getting an error message, program gives a blank screen and does not allow proceeding any further. | Pass |
| PP_Sched | 3 | 14 | Validity of stockroom/bin locations is not verified by the program. | Requirement not implemented in current release of the program. | N/A |
| PP_Sched | 5 | 9 | Test was intended to show that deleting a work order (not yet started) from PP_Sched also deletes the corresponding manufacturing order in ERP. | Requirement not implemented in current release of the program. | N/A |

plant, quality audits needed to be performed, production reviews were held on a regular schedule, and other areas of training had to be addressed. One team member's husband fought a losing battle with cancer through most of the project, and died just as the testing was to begin.

Against this backdrop, the initial project timeline proved unworkable. No amount of chastising via email would help team members provide timely feedback on requirements documents or test procedures, and site visits were only practical every few months. Indeed, roughly six months after project kickoff, all work on the project ceased for six weeks.

Friendly prodding sometimes yielded results, but when the validation project stalled, the only reasonable response was patience. The team's silence did not mean that the project had been abandoned – rather, that other issues had taken the foreground for a time. Patience and confidence paid off; when the project resumed, the team's focus was even sharper than before.

### *Lead the Team Where Necessary, but Let the Client Team Leader Address Internal Issues*

An internal manager led the UnderOver team, providing resources where needed and keeping the top management apprised of the progress. On technical and organizational issues – how to organize the requirements documents, whether a test procedure would be workable, outlining the needed procedural documents, updating the project schedule and keeping all informed of progress – the validation consultant worked directly with UnderOver team members, and in time with the contract software developers. In nearly all cases, this direct interaction worked exceedingly well.

No project is free from snags, however. One application proved even more difficult to characterize than the rest, perhaps because there had been no opportunity to meet the "keeper" of that program in person at the outset of the project. Early information was helpful – a flowchart, a number of screenshots, some amount of explanation – but filling in the holes became problematic. When obtaining information became difficult and communication strained, work on this application was set aside for several months. The validation consultant could only use collegial influence, and had no way to coerce this individual or the programmer, so the matter was referred to UnderOver's internal team leader.

Getting cooperation took time, but referring the issue to internal management was precisely the right choice. Beginning with a surprise telephone call one Friday afternoon, the floodgates opened: screen captures arrived, questions were answered, and this corner of the validation project was back in motion.

## THE ULTIMATE SUCCESS: LESSONS LEARNED ON BOTH SIDES

From start to finish, the UnderOver project took roughly nineteen calendar months (bearing in mind the hiatus mentioned above). During that time, several new employees came on board at UnderOver, several applications underwent major changes, and specific programs were added to and removed from the project.

Validating these twenty-one programs (*Figure 1*) helped the UnderOver team members see software not as a magic genie, always doing the master's bidding flawlessly, but as an engineered product, designed to serve a purpose but limited by the developer's fallible understanding. Getting down to basics – writing down what a program should do, then testing to be sure that the program works as intended – has encouraged these team members to watch for possible future errors. From there the software problem reporting work instruction gives them a mechanism for reporting those errors.

For the validation consultant, this project was at least as much of a learning experience. How to listen, how to persuade, how to determine different communications styles, and how to keep an entire team informed as a large project moved forward: all were skills the UnderOver interaction helped sharpen. ❑

*NOTE: Access, Excel, and Visio are trademarked products of Microsoft Corp. Lotus Notes (also called "Notes") is a trademarked product of Lotus Development Corporation.*

## ABOUT THE AUTHOR

Brian Shoemaker, Ph.D., is owner and principal consultant of ShoeBar Associates, which offers consulting services and training in computer system validation, software quality assurance methodology, and electronic records and signatures. He has been responsible for validation of software in a variety of FDA-regulated settings, from the embedded applications driving immunodiagnostics instruments to custom applications for clinical-trial data management. He has also designed and instituted quality systems for software development.

Dr. Shoemaker served CSS Informatics (previously PPD Informatics) as Quality Assurance Manager and later as validation consultant. His work revolved around clinical data management systems, clinical safety data systems, and software heavily used in the clinical-trials market.

Previously, Dr. Shoemaker was QA/Validation Manager at Doxis, Inc., a software company that provided flexible, 21 CFR Part 11 compliant, document-based data capture tools for operations such as manufacturing, packaging, or inspection in the regulated industry. As Systems Engineering Manager at Behring Diagnostics (previously PB Diagnostics; a manufacturer of clinical immunoassay systems), he was responsible for embedded-software validation, instrument design assessment, and interfacing with assay development, instrument manufacturing, field service, and quality-assurance groups. His awareness of software quality and validation issues began with his development of instrument interface and data analysis applications in support of his R&D work at Technicon Instruments, and earlier at Miles Inc.

Brian earned his Ph.D. in chemistry from the University of Illinois; he holds the ASQ Software Quality Engineer certification and can be reached at bshoemaker@shoebarassoc.com.

## Article Acronym Listing

| | |
|---|---|
| CSV: | Computer System Validation |
| ERP: | Enterprise Resource Planning |
| FDA: | Food and Drug Administration (U.S.) |
| IQ: | Installation Qualification |
| ISO: | International Organization for Standardization |
| IT: | Information Technology |
| MS: | Microsoft |
| OTS: | Off-The-Shelf |
| SOP: | Standard Operating Procedure |