DRUG DEVELOPMENT AND INDUSTRIAL PHARMACY, 14(18), 2717-2723 (1988)

THE VALIDATION LIFE CYCLE APPLIED TO SOFTWARE DEVELOPMENT: NEW CONCEPT OR SIMPLY GOOD BUSINESS

by

Herman Bozenhardt
Artificial Intelligence Technologies, Inc.
Hawthorne, New York

ABSTRACT

The singular, most important aspect which transforms a control system into a functioning part of a process is the software applications. The conception, design and implementation of these softwares, in general, are inconsistently applied discipline. Although the need for a methodical process is clearly evident, every project manager's and project engineer's approach to the problem is quite different. Ultimately, the time and effort is never circumvented. The concept of "pay me now or pay me later" is very true in this particular area:

The development of software for control applications is new to some industries which are beginning to automate heavily. However, the concept of "validation" (a significant word in the pharmaceutical industry's vocabulary) is common in the nuclear power industry, petroleum refining and utility systems. Also, unrelated industries such as banking have validation procedures for software. Although the final control elements and processes vary from money to oil to power, the "life cycle approach" of design, implementation, test and verify are consistent. The methodical processes and good engineering practice are the common links.

The purpose of this paper is to examine the methods, procedures, and concepts used in several industrial environments, and compare them to the current evolution of control software development methods of the batch processing industry. Specifically, we will discuss the life cycle concepts now as applies to the pharmaceutical industry and the validation concepts as imposed by the FDA.

The synergy of good development techniques and validation is far more than a coincidence. It is good business. Furthermore, the incentives in manpower, energy and product costs show it is a very profitable experience.

This paper is geared for project engineers, project managers, control engineers and programmers. The specific slant will be a "how to" guide for providing a solid validatable system for any project but specifically a pharmaceutical one. The design will be weave the life cycle concept into good design practices.

INTRODUCTION

Software development for process control systems have been going on since the first monolithic computer systems in the early 1960's. The computer control system field have had their system development by process oriented personnel and not necessarily computer professionals, specifically because an intimate knowledge of the process is required to develop the logic and procedures. The program or configuration of the system is a one-for-one emulation of the recipe or operating procedure. It is this fact that the SOP's of a pharmaceutical product must be translated into a control system which caused the realization that system quality is critical. Building software for an industry which has been primarily unautomated was a cultural shock initially. However, building a structured quality code initially also provided a hurdle. The pharmaceutical industry had not had the level of experience, (time and total manyears) to totally appreciate and understand why structure was necessary. Many industries had twenty to thirty years of making mistakes and realizing the impact of poor documentation and not testing. Although some in the pharmaceutical believe validation of systems in a bitter pill to swallow, it is not, and other industries require the same level of effort because they have had the time to come full cycle to realize the benefits.

SYSTEMS "DEVELOPMENT"

The implementation of a control system requires the configuration and often programming of commands to generate the manufacturing procedure. The configuration/programming must be accomplished for different tasks and often with different "tool sets" within the computer. The assumption is made that a modern distributed control system is being used for manufacturing automation. A direct application of this discussion to a PLC environment is not valid.

Development, as such, is done by using a vendor's control system hardware with his proprietary operating system. The operating system gives him specific tools to configure control, write reports and to integrate manufacturing. Therefore, the development is done in a menu driven or friendly macro language (often an English emulation).

The scope of the configuration encompasses five levels of configuration.

LEVEL 1

The unit operations level comprises the functions of a single vessel (as in the classical chemical engineering term). The unit vessel (or equipment) executes its process by means of continuous regulatory control (PID loops on level, pressure, temperature, and flow) and discrete regulatory controls (open/close valve, timer on/off, pump start/stop). The regulatory activity must provide control of items specified in a recipe (tight temperature control, exact composition control, etc.) It must also handle the nonlinear control such as pH control and Btu control of an exothermic kinetic reaction. On the "discrete" side, specific timed cycles, accurate weighing, and totalization of feedstock are process necessities. Safety, such as the interlocking security of discrete devices, valve opening, and high-limit switch verification, is of paramount concern and must be a real time element of control at this level.

This configuration involves a detailed filling out of computer image forms ("fill-in-the-blanks") which represent the basic acquisition of data, processing of the data for regulatory action and the timely output of the signal. To allow the batch procedural logic to execute a programming or flow chart language is used to describe the procedures. This program is detailed and is the prime focus for the development effort.

This level is the most essential level for complete organizational understanding. The process SOP must be duplicated exactly into software. The unautomated process (usually totally manual operation) must be flow charted out, planned out and reproduced with the configuration and programming tools provided in the system. The configuration or implementation of the application in its application's source code must be readable by the operations, engineering and Q.C. personnel. Although this may seem a complex task, it must be done because the configuration is in fact is your SOP. If you were to develop a new process and write an SOP, all these factions would be initially involved with its development and be able to read the English text of it. The same analogy applies here, where the critical parts of the organization participate, approve and have ownership of effort.

The result of this effort is completeness and correctness of thought for implementation which provides minimized corrections, shorted start-up time, minimized trouble shooting, and elimination of rejects and off-spec batches.

In order to implement such a system, the configuration language must be an easy, well understood tool similar to the English language. Ultimately, the validation audit of the FDA will require concurrent of approval and understanding of the code. Therefore, the driving forces from the FDA, organization groups, and for long-term system maintenance will provide the incentives to prepare the documents and implement the configuration in a methodical process (discussed later). Also, this will be a major factor in determining the type of system being implemented.

LEVEL 2

The train level, the next higher level of operations, essentially involves product recipe coordination. In most batch product manufacturing, unit level conditions change radically for different recipe steps (reaction, batch distillation, crystallization, etc.). This change must be in reconfiguration of the vessel or a vessel-to-vessel transfer. The change of the unit operation, whether physical or not, requires change of valve line-up, change from heating to cooling media, and other fundamental operations. The termination of cycle of one operation and changeover is critical from a quality as well as a capacity viewpoint. The product history from one operation to the next must be unified, monitored, and treated as one. Variations in product (via on-line analyzers) as feed to the next step must be corrected and managed under the recipe. Therefore, the train level is responsible for the entire product manufacture.

The train level is implemented by the configuration logic language and menu driven "recipe" facilities provided by the vendor's console system. This level contains and controls the parameters which are material quantities. This is the emulation of the dosage form or manufacturing recipe. This step's accuracy and security is critical.

LEVEL 3

At the process management level, the assignment of recipes to equipment is coordinated based on the company's marketing needs and physical availability of equipment. This level must

always assign the recipe/products to the equipment configuration that offers the absolute lowest possible cost of all permutations and combinations of recipes and equipment. When there is any change in the campaign strategy, the software assigns the recipes to the next lowest cost equipment unit available.

This level recognizes that each recipe requires a specific alignment of equipment, a specific use of common utilities and resources, and a specific use (CIP/SIP) of product manifolds. This requires facility management tools which are either custom built in a high level programming language, the flow chart logic language or via a tabular "fill-in-the-equipment matrix". this level must provide a precise development because product contamination, product isolation and proper equipment use is critical. Consistency with the maintenance planning database is important.

LEVEL 4

The plant management level is the plant database which unifies the process and product quality data. This facility produces the records and reports which portray the product quality. The configuration of this uses spreadsheets, database tools, data capture routines and high level programming languages. The data selection, calculation, and data compression are all critical for reporting and archiving, which product decisions are made from.

Development of data collection methods, security procedures, data storage architecture and reporting formats must be as good as and hopefully, better than the preautomated methods. The concepts of minimized human interaction with the data (i.e. data copying, transcribing and paper moving) must be eliminated to the maximum level possible. The primary concerns are humans introduce unpredictable data errors. Computers can make no cognitive errors or mistakes. Security levels (passwords, herarchy, etc.) must be structured, enforceable, and most of all practical.

LEVEL 5

The last level is the corporate level which allows movement of the data from the base process control system to external systems which use the product data and process data. The commercial control systems have communication packages which require both configuration and programming to match the file structure and protocol of the external system. Quality and design are essential to get the correct data from system to system at the proper time. The Management Information System groups must be integral with this effort.

INDUSTRIAL PERSPECTIVE

Now we have discussed the scope of what the development involves. Functionally we can develop parallel examples in the other industrial business segments.

As soon as computers were a practical reality, the banking industry adopted computer technology for record keeping. Although the applications are vastly different, the banking industry is a regulated industry, where errors and mistakes can cause lawsuits and corporate damage. Repeatable performance is the only acceptable performance. The structure of systems is a multi-layered architecture;

+ terminal server system/menu data entry (I/O)
+ database manager
+ report writers
+ audit trail/password security
+ communications

These systems are validated and were validated many years ago, before any process industry became fluent in use of computers. The programmers use very structured code standards and their design and code reviews go through many steps before they take them to trial or simulated runs.

In the Petroleum and Petrochemical industry, process control technology began in the early 1960's with varying degrees of success. The knowledge derived from these early experiences had a severe impact on the industry when distributed control was introduced in 1975.

The result was programs of detailed functional tests, utility exercising and simulation. These were rigorous test done on the vendor's site which typically lasted for several months. The applications programming was done in a predetermined, structured format, under a set of specific guidelines. These projects of testing, documenting and structure again were learned responses from the problems resulting from unstable vendor software, and poor programming design, review and implementation methods. Similar experiences were also seen in the chemical industry.

The last area involves another regulated industry - the Nuclear power industry. The nuclear industry, depending on the vintage of the plant has extensive validation of their computer control systems. The system used must undergo hardware qualification and testing, software testing and integrated system testing.

In summary, the industrial world, whether regulated or not, has learned the value of "validation" concepts and quality software. The value is evident by the mistakes and costs of the past or the rapid "trouble free" start-ups of today.

THE LIFE CYCLE APPROACH TO SOFTWARE AND SYSTEM

The life cycle concepts can be applied to any process or system that must be maintained and operated. The life cycle concept provides a living path to prove the repeatable performance and long term maintainability of the same. The pharmaceutical industry depends on these concepts to insure its product quality.

To repeatably produce a pharmaceutical product, the implementation of the SOP into an automated system has a specific chronological path of implementation with a specific set of guidelines applied at the appropriate steps of the process. The following discussion covers the methodology and philosophy.

PROCESS DESCRIPTION

When implementing a system for a GMP environment, the first step is to clearly document a process description of exactly how the manufacturing process works. This should be a combination of dosage form monograph data plus the SOP. This establishes a firm basis for what we are doing and how the system will continue to manufacture. Part of the base case should have some specific indication of how process control and data management is currently done.

The next step covers the software creation and documentation path.

SOFTWARE REQUIREMENT

In all control systems, the software functions are the focus of the project. The software contains the intelligence to manipulate the process and make decisions. Software requirements must be separated into specific functions and capabilities and must be documented. Therefore, the capability of running a process from whatever the previous method must be emulated by the computer. Operator activities that now will be handled by the machine must be specified.

Numerous operations must be logically divided into requirements of sequential control such as PID control, laboratory data entry, and communications. Such operations include:

- recording critical variables at specified frequency during every batch operation
- receiving lab data and taking appropriate processing action
- sequentially executing a batch compounding and transferring the material when released by QC

Within each category, the limits of performance and expectations must be delineated.

DESIGN SPECIFICATION

Once a thorough understanding of the process, data handling, and data storage requirements are established, a definitive specification is written. This specification must identify how the system should work, taking into account the software capabilities defined in the requirements and the process equipment. At this point, the organization should speak through its team members to insure that both process and GMP needs are met. From this document, the software and hardware should be quoted, purchased, and built.

SOURCE CODE

After the system is programmed based on the design specification, the project manager must insure that the programmable code which transformed generic hardware into a control system is retained and documented for maintenance purposes and validation. The source code which makes the generic hardware function (i.e., operating system) does not have to be retained; however, documentation on how the user configures and utilizes it is needed. The user must also have access to the operating system in case of a serious problem or inspection.

SYSTEM DOCUMENTATION

The system documentation should be a complete set of instructions covering how the physical system hardware works and is linked together. It must include maintenance routines, diagnostic routines, and systems operation. System documentation is primarily for the user's maintenance and is the physical analog of the source code.

USER DOCUMENTATION

The user documentation must be a comprehensively prepared document that describes all the functions available to an operator, lab technician, or any other "hands-on" user of the manufacturing system. It should also include the security system, password levels, and provision for self-diagnostics.

SYSTEM VALIDATION

Finally, with a completed self-contained set of documentation that allows the user to be self-reliant, the system must be validated to prove its functionality upon receipt or startup. The validation verifies that the system is in perfect physical operating condition, performs the process operation, and meets the design specification.

VALIDATION OF THE SYSTEM

The true test of a systems completeness and fitness is the "challenge" or validation test.

The validation protocol is a set of several procedures as follows:

The Acceptance Test. It proves all the operation functions of the equipment and demonstrates all aspects and logic paths of the design specification. Generally, this procedure requires hookup to a simulator system to prove valve movement and to simulate faults and failures. Functional system tests are combined with the exercise of every major logic path and unit operations procedure.

The Reproducibility Criteria. This procedure has a twofold purpose. First, it is designed to revalidate the system at installation. The first validation procedure is carried out in an off-site environment, so verifying the hardware and software after shipment is necessary. A procedure to check and calibrate all the input/output wiring into the new system in order to validate the field terminations must be implemented. In addition, equipment operability following a user's manual must be checked. Finally, a "water batch" or placebo run, which simulates an actual run but without the real raw materials, should be conducted. An operation check and evaluation assessment report should be developed to correlate the results with the past protocol and the design specification. In addition, this stage tests the data collection and data management with the report generation facility.

The second purpose of the reproducibility criteria is to periodically execute a placebo run. These periodic checks will verify that the system is functioning properly and that no unauthorized changes have been made.

Process Evaluation. This procedure is a process technical activity, which like the other procedures, must be completed prior to startup. This is necessary so that the first product batch can be evaluated in terms of quality (sterilization, potency, etc.). Periodic process evaluation must be made during normal operations.

Correction/Verification Control. From the acceptance tests, reproducibility tests, and process evaluation, a critical link of performance is drawn through the installation and operation of the system. Inevitably, errors will surface. A correction procedure must be written with process and engineering reviews before corrections are actually made. The intent of the correction procedure is to document any error and its source, to assure that it is a true error and not a misjudgment, and to trace corrections if a mistake is made in the correction process. This procedure need not be lengthy or complex, however documentation is critical.

Other key considerations in the validation of a computer system from a project standpoint include the following points.

The change control procedure differs from correction/verification control. When a review of process operation changes by process becomes necessary, engineering and quality control must be able to approve the change and how it will be done. Change to the validation protocol and subsequent testing must be made to conform to the reproducibility criteria and process evaluation.

A security system and pertinent criteria must be established at two levels:

At the administrative level, the security system must address such questions as: Who changes the system software and hardware? With what level of authority? Verification testing and documentation of modification must be a set procedure.

At the operations level, operations control, data entry, and keyboard accesses must be established with passwords or keylock, or both.

Finally, specific periodic review frequency--by number of batches and time--must be set for reproducibility criteria, process evaluation, review of the system's changes, and security system and passwords. This can be a mechanism for setting dates for review of the system and discuss future improvements.

PRACTICES AND METHODS FOR QUALITY SOFTWARE

With the basis of what we are configuring and programming defined and with an understanding of the life cycle, now we must develop or consider software standards. Software standards are the company guidelines which can be used at different levels of the life cycle to formalize methods and insure consistent quality of engineering. Although software standards may sound foreign to those not directly aligned in the software business, it is another form of an

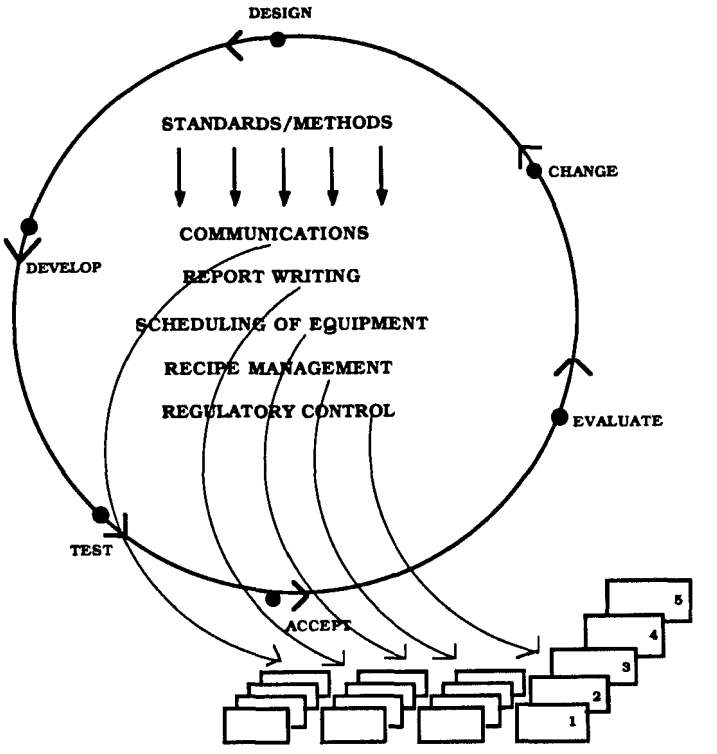

FIGURE 1 **DEFINED TASKS**

engineering standard. Just as a civil engineer has design practices and methods for building a structure, software engineering must have definitive bounds and methods for development. This provides easy maintenance, software readability, ease of trouble shooting and removes dependence on anyone individual. The following are a summary of the different disciplines:

- Design Reviews - A specific procedure for reviews of the;
  - + process description
  - + software requirements
  - + design specification
  - + source code

All four areas must be reviewed by QC, process operations and engineering supervision. The process description should be an easily approved and, if not immediately. This process will filter out any misunderstanding on just how the process works. Since an engineering individual probably will be doing the development, this provides a specific basis for understanding the process. The software requirements and design specification provides the opportunity for the organization to review and "buy into" how the system is to work. This provides constructive criticism, correction and builds a basis for what will be expected at the acceptance test. The final area is the source code design review which reviews the functional structure of the coding (high level review). This includes review of the logic flow diagrams, and the potential different philosophies of operations:

a) vessel by vessel control
or
b) unit operations (charge, cook, mix, etc.) control
or
c) a single hard coded program for all operations

As a general rule, the modularization of software (b above) into small reviewable and manageable units is the best method.

CODE STANDARDS

To provide an easy transition from software personnel and for a long term maintainability, code standards must be used in development of systems. In the absence of any standards, personal habits and educational background will determine how the program works and what the results are. The ANSI/IEEE organizations have provided several useful documents for code standards:

729-1983 Glossary of Software Engineering Terminology
730-1984 Software Quality Assurance Plans
828-1983 Software Configuration Management
829-1983 Software Test Documentation
830-1984 Guide to Software Requirement Specification
983-1986 Guide for Software Quality Assurance Planning
1012-1986 Software Verification Plans

The standards provides a basis for coding regardless of the system or language and how to organize the programming and develop an efficient product.

CODE REVIEWS

The code reviews for a pharmaceutical applications are critical. Since we accept the premise that the SOP and the system logic/recipe are interchangeable, then the applications code review especially must be reviewed, understood and approved by Manufacturing, Engineering and QC. The approval must emulate the approval of a new SOP.

The complication often occurs when using an obscure configuration language (e.g. assembler language or ladder logic). The level of difficulty is immaterial to the need of the task at hand. Regardless of the effort and pain, code reviews at the applications level are necessary. The operations and QC people must be completely aware of the procedure implementation. Ultimately, the drive in the market place is to implement systems with more user friendly English language configuration. This is necessary so code reviews can be made meaningful to all those participating in the process. Also, the use of PLC's is dramatically decreasing with major pharmaceutical companies developing a policy which states PLC's are unacceptable for product manufacture control.

SYNERGY OF METHODS

The "Life Cycle" approach to system implementation is the key to providing a validatable process. The life cycle approach cannot be easily digested without two factors:

- scope, which we have divided our tasks into five workable levels
- tools and methods which are the internal standards and organizational methods which guide how we code and who reviews it.

The synergy provides a list of specific tasks to be done in a methodical order by specific people to be reviewed by others. Specificity defines performance and responsibility will define internal organization harmony. These are the key elements to success (Figure 1) as individual tasks are addressed and completed.

SUMMARY

The pharmaceutical industry has experienced a rapid evolution by the imposition of system validation. The life cycle approach provides the development path while the efforts of the past (IEEE/ANSI standards) provide the tools. Validation provided the pharmaceutical industry the motivation to develop software and implement systems correctly. The long term effects are financially rewarding. Validation will prevent the industry from having a long history of system failures and software nightmares.