



AFMC Production Acceptance Certification and Depot Maintenance Quality Software Project

K. Edward Lynch
Advanced Programming Institute, Inc.

Amid the thunder, flash, and crash of the monster, multimillion- and billion-dollar software projects wherever we look, an occasional small, productive step forward is taken. This article is about a success story on many levels. Although this project was relatively small by most software project standards, this article discusses the successes surrounding the software system, developed in-house by an Air Force Materiel Command (AFMC) organization, supporting a 24-hour-a-day, 365 days-a-year, real-life, in-the-trenches mission in the AFMC depot maintenance arena.

Among the many challenges faced by first-level depot maintenance supervisors in an Air Logistics Center (ALC) is to maintain the individual records (task and skill certification and completion of training) for the mechanics or technicians under their supervision in accordance with AFMC Instruction (AFMCI) 21-108, Production Acceptance Certification (PAC). This is a daunting task given the thousands of tasks associated with performing the overhaul and maintenance of U.S. Air Force and foreign air forces' equipment of every type. It is even a more daunting and crucial task in this era of rapid changes and diminishing resources. Mechanics' certification of skill to perform a task and the sign-off on their work have far-reaching and significant consequences. Many lives may be held in the balance by equipment that is installed or repaired by these workers.

Why Another Software System?

In 1993 there began an effort to produce a standard, command-wide software system to manage the data concerning the tasks and training of skilled mechanics and technicians. At this point, it had become clear that software systems developed at several ALCs used older software technologies limited in their ability to deal with changes on the horizon. Newer software and hardware technology began to impact operations at every level as operational workload was redistributed, and the availability of

technical skills was subject to a flux not experienced in decades.

Factors that Contributed to the Project's Success

The success of this software project is due to the participation of many users who contributed throughout the process. Although the project did not formally adopt a rigorous software lifecycle development method, in practical ways, the key elements for developing any good software were used during the process:

- Develop the requirements as quickly and clearly as possible.
- Use good tools that can keep up with the changes in technology.
- Involve the users and testers all the way through the process.
- Review! Review! Review!

More than anything else, it is clear that people determine the success of the software project—it is the combined effort of user and developer. Good software technology is important in any software development endeavor, but success is ultimately the product of effective communication and interaction among the people who do the functional and technical work. The lure of technology and the dizzying pace at which it changes dazzles most of us in the development trenches; often, keeping up becomes an end in itself. But great technology and brilliant developers can produce fabulous software that is never used because it does not conform to the real needs of the user.

However, the user can tolerate less than the most current technology and less than breathtaking design if the new software delivers the means to do more work better, faster, and cheaper. When the software forces the user to rearrange a work practice or takes a long time to learn, it will be resisted or, even worse, sabotaged, especially if the user feels that requirements based on the business objectives were not properly addressed in the software development. When the software does deliver those requirements, the user welcomes the better tool as long as the payback from converting to the new tool is greater than the effort.

Involve the User in Requirements Development

The first of many successes of this development project was produced by the initial project manager and development team leader. It is tough enough to get requirements from users who are all in the same vicinity, but getting requirements from users in widely separated geographical areas who have vastly different workloads and practices was one of many challenges the development team overcame. They went to the users to find out what they wanted the software to do, and the team went back to the users for periodic reviews over the life of the project.

As a participant in many of the later reviews, I was delighted that users and coordinators from the ALCs came to provide real participation and contributions to the end product. At no time were any of these users bashful about

expressing their needs. As a result of these reviews, many functional capabilities were incorporated to facilitate the movement of data between supervisors and to handle the association of training requirements with specific tasks. Additional capabilities were developed to allow supervisors to set up "templates" of training and tasks that are used to quickly incorporate new employee data in the system. In many areas, this system supplements training tracking and training management capabilities found in other systems: Some users anticipate using this system for training management when support for other systems is terminated.

Involve the User in Frequent, Regular Reviews of the Project Process

A series of reviews, held at six- to eight-month intervals at the ALCs and at Headquarters AFMC, Wright-Patterson Air Force Base, were accomplished using the demonstration of a progressively developed prototype to stimulate the evaluation and let participants see for themselves what progress had been made on the project.

Plan to Revise the Requirements During the Early Stages

The initial requirements specification, no matter how thorough, rarely includes everything. The project's users became smarter in that process and came up with better ideas for doing the work. Sometimes the developer suggested alternatives that users found acceptable. During the review process, all capabilities that did not meet the users' expectations were documented, and the requirements and design documents were revised to better reflect the users' expectations. All remaining items scheduled for future development were reviewed and priorities re-established as requirements and design issues were clarified. These items became the agenda for the next review.

Run-time copies of the software were made available to users who wanted to independently do testing and review. The emphasis was to get the product into the hands of the users as

soon as possible to enhance the prospects of receiving good feedback early in the process. The tendency for requirements creep dropped off after the third or fourth review session. This user feedback process continued to function effectively despite three changes of the AFMC headquarters office of primary responsibility, three changes of the primary contractor, one change of the General Services Administration (GSA) administrator, three changes of the ALC project manager, three changes of the development team leader, and several changes of development team programmers. During all these changes, GSA administrators still regularly reviewed project finances and progress with the contractor administrators and the AFMC project managers.

Use Good Development Tools

The second success was the decision to use a new (at the time) software development tool¹ that enabled the developers to produce a progressive Windows prototype in a Windows environment, which also enabled a stand-alone version to become a client-server product with minimal conversion effort. This tool provided many of the object-oriented features that would assure a long and healthy life for this software. Early choices of help authoring,² database design,³ and distribution and install software⁴ also proved to have the power to support the project over the long haul and through several changes of software technology and requirements. These tools enabled a small number of developers to accomplish the work in a reasonable length of time.

MIL-STD-498 was another tool that was key to the success of this project. It provided an extensive structure of guidelines for system requirement, design, user reference, and technical reference documentation, which allowed the developers to produce coherent and relevant documentation for the system. Having a wide spectrum of project components from which to choose assures that all relevant project aspects were considered for this project. It provided a ready sanity check to assure that all relevant project compo-

nents were considered as the project progressed. It also provided assurance to the users that all relevant components were considered, along with a physical document they could evaluate for verification of specific components.

Involve the User in Testing as Early as Possible

The third success was the ongoing and early involvement of key users and trainers at each ALC. They were the field testers who provided the development team with crucial feedback at all stages of the project. User tutorial and guide documents were developed at the same time features were established and tested, enabling quick training on the current build of the software.

Train and Support the User as Early as Possible

The fourth success was an early "train the trainers" program conducted at each ALC to maintain user support until deployment was accomplished in mid-1996. Working through a network of designated key contacts at each ALC,⁵ issues were reported to the development team for timely resolution. During the development phase, fixes were sometimes provided within hours. Post-deployment fixes are issued at quarterly intervals. Support personnel at each ALC download the software and fixes from an FTP site established at the development ALC site.

The project was structured into four software module phases: the first was used to develop the base module to manage the Production Acceptance Certification as a stand-alone system; the second, to deploy the base system as a client-server system; the third, to develop an interface to another command training system; the fourth, to develop a module to collect data for process improvement evaluation, also required by AFMCI 21-108.

System Features

The functions and capabilities of the main module support the procedures to maintain AFMC Form 75, Job Knowledge-Training Certification Standard. All capability is designed to be accom-

plished by the first-level supervisor. Additional functionality was added to support a number of administrator capabilities that are especially important in the client-server environment.

- Personnel data.
- Task assignment data.
- Training data.
- Supplemental training data.
- De-certification of tasks.
- Supervisor history.
- Task data, which assigns required courses to specific tasks.
- Course data management.
- Standard reports, transportable among site and supervisors.
- Task-training template.
- Ability to convert data from four pre-existing software systems.
- On-line user guide and help.
- Backup/restore utility: a single-click backup and restore process for stand-alone PC users developed using third-party software⁶ that integrated smoothly into the application system.
- Work center administrator: a separate set of utilities that perform across all work centers and convert data from four other programs.
- System security access management features.
- High- and low-level reports.

New and Planned Features

Recently, several new capabilities have been added to enhance the client-server version, and we are in the beginning stages of upgrading and optimizing to 32-bit technology and more current object-oriented coding technology. A recent addition allows supervisors to use a personal identification number to sign off on annual reviews electronically, eliminating volumes of paper files. Talks have also been developing in AFMC for an interface to a major production planning and control system that requires current certification data before work can be assigned to available mechanics. A broader course management module has been added to simultaneously support completion updates across multiple work centers. Many edit functions can now only be accessed by delegated administrators.

The third phase of the project, the interface with a command-wide training management system, is awaiting the completion of the most recent rewrite of the system, from a 1960s software technology to PowerBuilder 5.0.

The fourth phase of this project, a process improvement data collection module, is also a requirement of AFMCI 21-108 and supports the recording of data about depot maintenance processes that enables the user to evaluate process effectiveness and analyze areas where process improvement can be pursued.

Conclusion

As of mid-1997, the software had been deployed to all the depot maintenance ALCs and is widely used in the stand-alone mode. A division at one ALC has deployed the client-server version; other ALC divisions are preparing to follow suit. Over 1,000 supervisors are administering the records of several thousand employees at this time. Supervisors now maintain a large number of records for employees more accurately with a minimum of time and effort, and records can be moved from one supervisor to another with minimum effort.

This deployed software product may not be glamorous, but it

- has broad scope and impact at the working level.
- addresses day-by-day needs of first-level supervisors to facilitate the accomplishment of essential work.
- was developed with plenty of user participation and contribution.
- uses resources sparingly but effectively.
- provides a common, command-wide, standardized tool to maintain this data.

Whatever other legacy this project effort produces, all who have been involved can take pride in these accomplishments alone.

It is amazing that developing the technology to produce the blueprint for a software project has been so long in happening; nevertheless, there have recently been some encouraging technical advancements in the acquisition of system requirements, the establishment

of design specifications, and in modeling software systems. This is an exciting time to be in the software industry. Let the revolution begin. ♦

About the Author

K. Edward Lynch is a senior consultant with Advanced Programming Institute. He has over 20 years experience in all phases of software development for financial and business systems in mainframe and PC environments for American Airlines, Lockheed Missiles and Space Co., U.S. Army Logistics, Failure Analysis Group, California state and county organizations, Computer Science Corp., Ogden Government Services Corp., and Anteon Corp. He specializes in improving methods related to the critical earliest phases of software development, including management sponsorship, user participation, risk assessment and mitigation, and testing criteria.

Advanced Programming Institute, Inc.
SM-ALC/FMDD
Building 200, Room 230
3237 Peacekeeper Way
McClellan AFB, CA 95652
Voice: 916-643-6493 ext. 281
DSN 633-6493 ext. 281
Fax: 916-643-2217 DSN 633-2217

Notes

1. PowerBuilder from Powersoft, Inc. with a Watcom 4.0 database for stand-alone deployments, Sybase SQLAnywhere 5.0 for the client-server.
2. RoboHelp by Blue Sky Software.
3. ERWin/ERX for PowerBuilder by Logic Works, Inc.
4. InstallShield by Stirling Software.
5. Many individuals at the U.S. Air Force Materiel Command (Wright-Patterson Air Force Base) and at the ALCs located at Davis-Monthan, Tinker, Hill, Kelly, Warner-Robins, and McClellan Air Force Bases. The services of Computer Science Corp., Ogden Government Services Corp., Anteon Corp., and Advanced Programming Institute were provided through a Requirements Contract with the U.S. General Services Administration, San Francisco, under the guidance of Doris Lynch, and earlier, Paul Gurian. A more extensive list of key participants and current contact information can be found at <http://emerald.mcclellan.af.mil:2010>
6. DynaZip-16 - Inner Media, Inc.