

Published in the *IEEE Engineering Management Society Newsletter*, Q3 2006.

Small Company, Bulletproof Software

How a Small Control Systems Team Delivered Big

by Sue Dorward

I recently had the opportunity to interview the former CEO¹ of a small energy sector company that specializes in building control systems for gas turbines and turbocompressors. Their international client base uses these systems in gas pipelines, oil and gas production and transmission, electrical power plants, chemical processing plants, and the like. A serious malfunction of software or hardware could lead to destruction in seconds. Consequences could include loss of life and downtime costs up to millions of dollars per day. Thus, hardware redundancy and flawless software are essential.

This company, which has delivered several hundred such systems, has maintained a perfect record of no in-operation software problems for the past 15 years. An in-house team of merely six engineers developed all the critical control software, using a development process also developed in-house. Here are some lessons that the company learned along the way.

DON'T OUTSOURCE CRITICAL CODE

The CEO explained that by the mid 1980's, software-based integrated control systems were rapidly going to obsolete his company's systems, which were based on assemblies of hardware-based analog blocks. To avert this, they engaged a highly regarded consulting firm to implement their application technology in software. The CEO described their experience. "This didn't work very well. First, the result retained all of the limitations of our analog systems because we didn't understand the functional enhancements possible with software, and the consulting firm didn't have enough understanding of the particular control requirements to know what enhancements to suggest. Second, their development process divided the project among several programmers, each responsible for functional definition and coding, and their code was later integrated.

"Consequently, they quickly delivered software that met 95% of our needs, but each improvement beyond that increased our costs exponentially. There were

¹ who wishes to remain anonymous

endless integration issues, actual software began to drift from the documentation, and code changes had unanticipated consequences. When we had spent 10 times the budget, we realized the system would never meet 100% of our needs, especially in reliability and documentation integrity, and that was not acceptable."

They could either start over by comprehensively educating the consulting firm's staff in all aspects of the control requirements for turbomachinery, including techniques that gave them a competitive advantage, **or they had to learn how to develop software in-house.** They chose the latter. The CEO proudly explained, "Our first step was to create a development methodology likely to produce beautiful software, which we did."

SEPARATE DEFINITION FROM CODING

As a reaction to their outsourcing experience where the roles of designer and implementer were combined, the CEO banned the title of Programmer from his company. "We borrowed the building construction model. We had an architect and a builder: two distinct roles that we called detailed functional Spec Writer and Coder." The Spec Writer was a master of the real-world application of the system, in this case often a mechanical engineer. **The Spec Writer never wore the Coder's hat on any given project, and vice versa.** However, a qualified employee could be the Spec Writer on one project and the Coder on another.

ENFORCE A RIGOROUS, WELL-DEFINED DEVELOPMENT PROCESS

The Spec Writer was responsible for creating a detailed functional definition of the system, what they called "the Book", on which both the implementation and the user manual would be based. The Book had to be complete before coding could begin. Other engineers (not Coders) expert in the application area thoroughly reviewed and signed off on the Book prior to implementation. "We had to make sure that we were doing the right thing, in addition to doing the thing right," the CEO explained.

The Coder could only implement what was in the Book. If the Coder found any problems or had ideas for changes in the Book, he had to go back to the Spec Writer, who had to work through the issue and update the Book before the Coder could implement the change. This ensured that the Book accurately reflected the implementation and that the Spec Writer oversaw the entire design.

BUILD A ROBUST, REUSABLE LIBRARY

"When building a large office building, you don't want the architect and builder to make a unique design for every window and door." They developed a code base that was primarily a library of highly reliable reusable modules (object

classes), many duplicating the old analog function modules with which they were already familiar. In essence, they built an object library before object-oriented design was popular.

They spread the development cost over many systems, so they were able to justify the expense of carefully designing each module to be highly versatile and efficient. They had a standard process of thoroughly documenting and rigorously unit testing each module, to ensure the required functionality and reliability. If the Coder found that there was no module for a particular function, **the Coder was required to follow the process for adding a new library module rather than write custom code.** "If we had outsourced this, we could never have enforced this level of discipline and I would have been worried about technology leakage on top of that," reflected the CEO.

With the rigorously tested library in place, the process for development of a new control system was simpler. The Spec Writer could largely create The Book by linking graphic representations of available function blocks. Coding consisted largely of connecting and configuring the blocks exactly as shown in the Book. The CEO explained that "the definition and development of software-based systems became very similar to the previous definition and assembly of systems based on analog hardware blocks. What's more, when a system was completed, it was usually 100% correct or could easily be brought to 100%."

The company found that the best way to develop new products was to first develop a master system (and associated library modules) with all available features and options. To meet the requirements of a particular sale, they removed unneeded features prior to compiling the final code. This ensured a solid design and avoided enhancement-related issues, but had the bizarre consequence that the most complete and highest cost systems required the least effort to deliver!

COMPLETE, ACCURATE DOCUMENTATION IS CRITICAL

The Book, code, and library each contained considerable documentation. They developed a standard, detailed documentation format that they followed for each project. "In addition, The Spec Writer and the Coder communicated primarily through the Book, which forced the Book to be 100% accurate," the CEO explained. Since the Owner's Operator Manual was derived from the Book, the client could be confident that the manual accurately represented the functionality controlling the machinery. The Book remained platform independent, and so could be used as the basis for implementation on any technology platform. Only the code was platform-specific.

Library and application source code were required to be at least 80% comments, a standard set and enforced by the R&D manager. Coders were encouraged to adopt a uniform coding and documentation style, so that the code base was consistent and easier for everyone to work with and understand.

BUILD A SMALL, COMMITTED TEAM

A team of just six employees, including management, built this company's real-time control systems. The CEO explained, "I found that a team's progress and success was driven by the one brightest person, as opposed to the sum of all of the team members' talents. **A small team of very capable, well-paid people performed better than a larger group of merely average people.** "

So how did a small New Orleans-based company attract and retain the skilled employees that they needed to succeed? They built a relationship with a local university's engineering school, by giving guest lectures and even teaching courses in real-time control systems. They would hire top students, and then put them through a six-month hands-on training rotation.

The company experienced very low staff turnover. "Being based in New Orleans, there weren't many other opportunities in town for our Development employees. We found that if they were technically challenged, involved hands-on and well paid, most stayed."

CAN YOU ACHIEVE THIS KIND OF SUCCESS?

As you can see, there was no magic, no special sauce here. A small company in a city not noted for technology was able to produce robust and capable software-based control systems with a very small team by creating and committing to an effective development process. They were able to develop this process by having a clear vision, performing a careful analysis of a failed development process, and building on already-existing strengths. The result was a highly successful company with a reputation for top-notch products. Could this recipe for success work for your company?

Sue Dorward is a tech management coach who specializes in coaching high-potential employees. She is based in New Jersey and can be reached at sue@sudocoaching.com.