Bridging the IT/shop floor divide

Plant engineers and computer geeks can work together, if…

BY JIM DANTIN

Consider a factory floor data handling project where there is a clear definition of the problem, detailed functional specifications, and ongoing involvement of all shop floor users.

Now, contrast that to a system specified by management and designed by information technology (IT) developers, without either group having a clear understanding of shop floor users' needs or expectations.

The preferred choice seems obvious, especially as books and articles document the high failure rate of IT projects. However, the second scenario still happens all too often.

The following four-step process reduces project risk to owners and developers, adds value to the project, and provides a high probability of success:

1) Prepare a document that presents a clear definition of the problem so everyone focuses on the same issues.
2) Develop a detailed functional description and scope of work that encourages comparable and competitive bids and clearly identifies project expectations, deliverables, and metrics for success.
3) Communicate clearly and get users involved during project execution to ensure the developers are producing the expected solution and identifying design issues before they become problems.
4) Maintain team relationships via ongoing support activities to validate the project's success and encourage user acceptance.

SAME OLD, SAME OLD

Envision this typical situation. Shop floor personnel need a program that gathers data, calculates efficiencies, and displays information. They want it simple. They want it cheap. They want it now.

George in Area 2 developed a spreadsheet that does everything, except it works only in his production cell. John in Area 3 has seen the application and wants something just like it only a little better.

Tom, the PLC programmer, says he can put all of Final Assembly's production and quality data on the plant network if only somebody would write a little program to capture and display it.

IT says it can develop an enterprisewide solution for all of these needs, but it will take two to five years and cost $3.5 million.

The capital budget just came out; every department took a 25% cut.

Sound familiar?

Throughout all businesses, whether discrete manufacturing, process, distribution, or utility, the challenge is the same. How do we gather data, process it, store it, and deliver critical information to decision makers? How do we identify and implement projects in an environment where IT and manufacturing departments are frequently at odds over technology, techniques, and system responsibility?

CULTURE CLASH

When the simple request from the shop floor meets the development realities of IT, the results often aren't pretty. The shop focuses on the short term—today's production schedule, quality targets, and parts availability. IT relates to standards, security, consistency, scalability, and long-term implementation and support.

The shop floor staffers typically view the computer as an evil device that just complicates their lives. IT cannot understand why software systems already deployed are not used.

This culture clash can exist wherever IT meets the shop floor. Business's spectacular inability to solve this conflict has resulted in squandered development costs and resource time and, most importantly, affected enthusiasm and morale.

Individuals resort to an anarchistic environment where they each develop just a simple spreadsheet or database to solve their own needs. Everything comes down to the pure basics, solutions are unsophisticated, no one normalizes databases, and data structures and naming conventions are unlike anything others in the company. However, they are used.

If these simple solutions work, why should management care?

Visit any company where this environment exists. Ask the individuals how information flows throughout the company. The answer is: It doesn't. Instead of flowing, it pools. Each area has its own pool of information that it protects, massages, and carefully doles out to support its own needs and agendas. Data is not centrally stored because someone might see it and come to the wrong conclusions. Control of the data provides the opportunity for damage control when things go wrong.

ENTERPRISE ISSUES

The culture clash between IT and the shop floor is further complicated when IT faces enterprisewide planning and implementation issues. The shop floor not only resents IT's intrusion into its "real" world but also resists solutions that seem to favor the culture of other plants that may be located in other cities, states, or countries.

Consider an enterprise with facilities in the U.S., Canada, and Mexico. U.S. plants are a mix of strong union in the industrialized North and weak or nonunion in the rural South. Newer plants focus on a progressive team structure, while older plants live with a legacy of restrictive work rules and an antagonistic worker vs. management environment. The company has expanded to Mexico to benefit from low hourly labor costs, but it did not fully consider the impact of cultural differences.

Can a corporate IT or engineering group successfully implement a shop floor system in this diverse enterprise? The steps leading to a successful solution do not change, but they do become more difficult. The leaders of this project must be capable of addressing the concerns of all groups and must be prepared to deploy solutions that are configurable to meet local needs.

Enterprise solutions become expensive because of this flexibility and scope of features. Nevertheless, without the ability to deploy a solution that meets each plant's local needs and culture, the application will fail.

Before we find ourselves in the middle of another doomed project, let's lay out a plan that will reduce our risks and maximize our chances for success.

STEP 1: PROBLEM DEFINITION

A careful definition of the problem becomes the foundation for all future work. To paraphrase Lewis Carroll's *Alice's Adventures in Wonderland*, "Without a plan, all roads will get you there." If you do not know what problem you are solving, how can you say you even have a solution?

The definition should be developed, and agreed to, by everyone involved. After everyone agrees, they should then formalize the definition into a document, then circulate and approve it. All team members should sign the document as their commitment to its accuracy and success. Other stakeholders should also commit with written approval.

In most environments, problem definition will be the most politically charged and divisive part of the project. By working through this process, you should be able to ensure that everyone's understanding of the problems and expectations for solutions match.

The definition of the problem should do the following:

- Describe the existing conditions or difficulties. What is wrong, what is needed, what doesn't work, and why? Will existing systems or procedures need replacement?
- Identify people and departments impacted. Who is the process owner? Who are the customers? Are they internal or external? Are multiple locations involved? Are different languages involved?
- Identify goals or metrics for success. What benefits will result from solving the problem? In broad terms, what should the solution provide? What are the expectations?

An essential requirement is this document must address the problem, not technology. There should be no discussion of SQL databases, Ethernet, or programming languages.

The report should also document the intent of the project. What is the goal: reducing paperwork, improving quality, increasing communications, or what? Does the project add value or reduce cost? What is the project worth to both the individual and the company?

If shop floor employees want a system that improves the accuracy of daily production reports, but the IT department delivers a system that focuses on reporting production efficiency, will the shop floor be satisfied? If report accuracy doesn't improve, and management starts commenting about poor efficiency, shop floor employees will view the system as the source of their problems.

If everyone agrees on the problem, the next step is to identify a solution. Without agreement, you cannot proceed. Go back and try again, or you're on the road to failure. That one dissenting voice will reappear as a roadblock when you least expect it.

STEP 2: SCOPE, FUNCTIONAL DESCRIPTION

Before you write the first line of code or purchase the smallest piece of hardware, you should produce a document that details what the system will (and will not) do. What will be included? What functions must it provide?

This document will become the reference for all development work. It can be an attachment to a Request for Quotation and will validate that system deliverables meet requirements.

The scope includes a more detailed problem description, defined in Step 1. Document as much as possible about the current process or problem. How do you create a report? Who gets copies? How are the numbers calculated? What are the rules and exceptions?

Everyone associated with the process should have input. Everyone should clearly identify his "pain." If an essential individual cannot read about his problem in this report, he will feel left out and will be reluctant to support it later.

Do not overlook simple issues by believing they are common knowledge or insignificant. If Joe in Shipping needs the font size on the screen to be large enough to read from 15 feet away, document that fact.

After you document the current process, proceed to describe what the new system will do. Remember, we're talking about "what," not "how." We don't want to deal with the technology yet. Describe what the function does, not how it should do it.

Identify all rules. Is the system required to be available 24 x 7? Are reports required on demand or at the end of a shift? Must the system run on existing hardware and databases? Are there standards to consider? What's the implementation timeframe?

Describe all data. What data do the operators enter? What data comes from the machine control system? Which screen gets what data? What data is stored and for how long? What data moves among other systems?

This functional specifications document becomes the road map for developing the problem's solution. If the developers provide a system that functions as described in the document, they will have met the users' expectations, and all will be satisfied.

Publishing the final document in this step is a major milestone, so get everyone to sign it and then give certificates of appreciation.

STEP 3: PROJECT EXECUTION

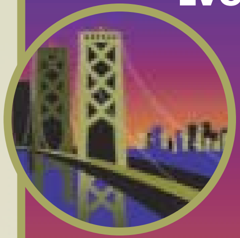
With the detailed project scope and functional description now in hand, it's time you address the "how." This task belongs to the developers, not the shop floor team. This team should be concerned with results. If the functional specifications are complete, and the system meets all of those specifications, there should be no disputes.

If multiple developers are bidding for the project , the functional specifications become a significant part of the package. They ensure all developers are bidding on the same system and clearly define the deliverables. Bids will be comparable and competitive. The detailed functional specifications document will reduce the risk to both the developers and the owners —and that will reduce the project cost.

The shop floor team should help evaluate the proposed systems and help select developers. Ongoing communication between users and developers will be crucial to success.

REALITY CHECK TIME

Despite all the hard work done in Step 2, some things may have slipped through the cracks. As developers work through the system, they will hopefully discover any miscues, or maybe a team member will remember it somewhere along the line. This is not a problem. It's just change order time. The change order should be a graceful way of correcting mistakes, not a punitive instrument.



Everyone associated with the process should have input. Everyone should clearly identify his "pain."

Depending on the environment that existed in Step 2, a change order may require review by a large number of people. Who has the authority to change the specifications with a change order? Think this through. If a change order results in a substantive system modification, all impacted users should be involved in the approval. This will be time consuming but necessary for the system's acceptance.

One simple way to smooth the change order process is to circulate a proposed change for comment. If no one objects within a stated time (a day? a week?), the change is approved by the process owner. This is where identifying a process owner, way back in Step 1, becomes important: one individual who is the point of contact between the developers and the users.

At appropriate points throughout the development process, define milestones such as presentation of prototype displays, reports, or simulations. These are time consuming and potentially costly, but remember the "no surprises" goal. If the users see the system develop, they won't be surprised at the end.

No one can afford the disaster of a "That's not what I wanted" response from users when the system goes live.

Review meetings are an important part of user training. If end users see how the system works and get a chance to gain familiarity during the development process, training becomes simpler. Typically, some users will also execute creative attempts to break the system. It is far better that any weak points be discovered early in development and not at 3 a.m. on a Sunday after the system has been live a week.

The development stage ends with a formal project acceptance by the users, which is easier said than done if there are unanswered issues remaining from Step 1 or 2. If, however, the functional description is complete and approved, it can be a checklist for project acceptance.

STEP 4: SUPPORT

Ongoing support for users is necessary to ensure a project's long-term success. Most projects will require revisions or improvements; the key is to maintain communications among project owners, users, and developers.

At the end of a poorly executed project, developers and owners are looking for a way to escape from future dealings. Anger, resentment, contractual disputes, and unhappy users are the unwanted results. The end of a successful project is a time of enthusiasm and excitement. It is a time to generate ideas for new or expanded features. There will probably be bugs or problems, but the already established communications enable constructive fixes.

Keep users involved, answer their needs, and the system will deliver the benefits everyone expected.

In conclusion, systems for the shop floor are a challenge to develop, not because of the technical difficulties but because of people. Technical innovations, elegant designs, and corporate standardization are key to a successful system, but they are insignificant compared with user satisfaction. By concentrating efforts on the early stages of problem definition and functionality, you can ensure successful development and user acceptance. IC

Behind the byline
Jim Dantin is a senior member of ISA and the operations manager of Pfeiffer Engineering Co. in Louisville, Ky. He has solved information systems problems in manufacturing, utilities, and process businesses. His address is jad@PfeifferEng.com.