From: www.cio.com

Fixing the Software Requirements Mess

– Christopher Lindquist, CIO

November 15, 2005

Hugh Cumming had his work cut out for him. The gap between what his not-yet-implemented call center management application at a large European company could do and the requirements list created by 40 eager business-side stakeholders now filled 3,000 pages and threatened to delay an already overdue call center consolidation effort another four to five years. "My first instinct was that the project had absolutely no chance of success," says Cumming, currently CIO for ADP Employer Services Canada.

Requirements, as every CIO knows, are a problem, but CIOs may not be aware of just how catastrophic the problem has become. Analysts report that as many as 71 percent of software projects that fail do so because of poor requirements management, making it the single biggest reason for project failure—bigger than bad technology, missed deadlines or change management fiascoes. Though CIOs are rarely directly responsible for requirements management, they are accountable for poor outcomes, which, when requirements go bad, can include: project delays, software that doesn't do what it's supposed to and, worst of all, software that may not work correctly when rolled out, putting the business—and the CIO's job—at risk.

Mishandled requirements can torpedo a project at any time, from inception to delivery. Start down the wrong road and you arrive at the wrong destination. And even if you're heading in the right direction, making fumbling changes midstream can be almost as deadly. Not integrating requirements with your test process can have you racing back late in the game to correct problems that might have been solved early on (and more cheaply).

It's up to the CIO to establish an overall requirements process that works and to support it with the political skills necessary to get buy-in from both the business and development sides. The CIO must also have the organizational backbone necessary to shove wayward requirements processes back into line.

None of this is easy. Business users often don't know exactly what they want, can't prioritize what they do want, request things IT simply can't deliver (because of complexity or cost), or can't describe their desires in terms that translate accurately into code. On the IT side, analysts, architects and coders regularly try too hard to please and don't set realistic expectations for projects; they don't use every

means possible to guarantee that what they're building is what the user really needs, and sometimes they even fail to make sure that they're talking to all the right stakeholders.

In short, the traditional practice of requirements is broken. But some IT folks are doing everything they can to fix the situation. To a man, they say process is key. Exactly what process? They all have their own ideas. One executive decided to simply enforce rules that should have been enforced all along. Another rewrote the rules from the ground up. And a pair threw out the old rule books completely, one taking a business-process-focused approach and the other choosing to build applications with quick iterations rather than long requirements documents. But they all agree that you should choose a formal requirements-gathering process and stick to it.

Writing requirements is hard. It will always be hard. But with a handful of smart decisions you can create a requirements process that will produce positive results—and maybe keep your next project from becoming another statistic.

Forty's a Crowd

Cumming's solution to his requirements nightmare was radical surgery. First—with backing from ADP's chief executive—he stripped down the scope of the consolidation project, lopping off existing processes that worked as-is and didn't need to be rolled into the new application. He also pared the group of 40 stakeholders to five active participants. He allowed the others to stay involved, but only in the more passive role of reviewing the implementation plan and feature specifications, without actually adding feature requests of their own. He then repeatedly went back to the remaining five stakeholders and asked them if specific requirements were really must-haves or simply nice-to-haves. After less than two months of pressing the issue, his new requirements list was less than 10 percent of the original. And after the project went into production, it needed to accommodate only one major change before being rolled out to 12 global locations.

Cumming says the problem in this case—and in many cases—is that IT often does not take a leadership position in the requirements process, instead taking the attitude that "the business is requesting it, so it must be the best thing to do." But that kind of thinking can lead to requirements lists that are unmanageable and unforgiving. Instead, he says, IT people need to develop a valuable skill: saying no with a smile. "Really what you're saying is, 'Not yet,'" Cumming says.

To paraphrase Daniele Vare, managing requirements—like diplomacy—is the art of letting everybody have your way.

When Cumming reduced his army of 40 stakeholders to five, he admits that there were some "interesting conversations" about who would stay in primary roles, noting that people were worried they were going to lose features they felt were important to their business units. To ease their fears, Cumming and the core stakeholders created a "high-level vision" (a summary of the most important functions) for the project and spent time demonstrating how the final project lined up with that vision. He also showed all the stakeholders how they would get at least some value from the project—even if they weren't going to get every single detail they wanted.

The more passive stakeholders were also encouraged to become more active as the call center system began rolling out. When the system moved into their departments, these stakeholders became directly responsible for sponsoring any necessary application changes within those departments. This task was assisted by the intense interest that senior management had in getting the project into production. Cumming felt he needed to know who really wielded influence in the company (versus what appeared in the org chart), plus he wanted to identify stakeholders with sufficient technical expertise to add value to the requirements-gathering process.

"The list of people who would have the most to contribute to a requirement list always ends up being small in my experience," Cumming says.

The Rules of the Roles

Tired of his company's hodgepodge of requirements practices, Jesse Hanspal, director of development

technology services at Bank of Montreal Financial Group, decided to create his own process by combining pieces of existing requirements techniques and adding a quality assurance process as well. Hanspal says that after five years of effort, the bank has defined the requirements process at a level of abstraction high enough that it can be applied to any project or problem. After much consideration, the bank decided that it needed a process built around responsibility and job roles in order to guarantee that all necessary stakeholders had a say.

"It's important to get all the stakeholders around the table and get the requirements from the horse's mouth," Hanspal says. And, he adds, by defining stakeholders according to their roles, you get a more accurate cross-section. For instance, he says, for a given project, you need representation of the end user role, of course, but also of the application administrator role, not to mention roles related to security and regulatory compliance.

Hanspal notes that in the past IT spent 10 percent to 20 percent of its time and energy on defining requirements. "What we've learned is that once you have defined a process, then you go and get an ISO 9000 certification for that," he says. Having the certification lets people know what is required of them. It also gives the bank a chance to evaluate effectiveness and improve the process. And Hanspal says the new process has produced results. For instance, the number of software defects related to requirements has dropped by some 50 percent since implementing the new controls.

Bank of Montreal also wanted to make sure that its analysts had the skills necessary to execute the new process. Unfortunately, while it had been easy to find external certification for project management (the Project Management Institute) and functional testing (the Quality Assurance Institute), no similar body existed for business analysis. So the bank created its own. The International Institute of Business Analysis now boasts some 800 international members, Hanspal says, and any company can send analysts for training in the Bank of Montreal approach.

Going Agile

Given the trouble companies continue to have with requirements, some practitioners argue that the process needs to be more flexible.

Gregor Bailar, CIO at Capital One, is a convert to one of these antiestablishment philosophies: agile development. Agile development advocates argue that old-style requirements processes are too rigid, put walls between users and developers, and, given the ever-changing nature of software and business, are fated to fail. Instead, they say, developers and users should sit down together and start coding almost immediately, stopping frequently to evaluate progress and make necessary changes based on user input without feeling the need to follow a monolithic requirements document.

"What we needed wasn't more process but to get to the value [in a project]," Bailar says.

After piloting the concept in early 2004, Bailar began forming the ultra-lean, connected teams that are the basis of the agile method. Agile teams at Capital One generally consist of three businesspeople, two operations people, and five to seven IT folks, including a business information officer (in effect, a translator who works between the business and IT sides), a project manager and at least one of the 80 developers that Bailar sent to formal agile training classes. Along the way, some architects and security experts will add their skills as necessary. Each team gets its own agile coach (one of 20 Bailar hired) to keep an eye on the proceedings and offer advice and support. Teams meet in dedicated, open rooms, and initial requirements are limited to a goal for the project, a handful of cards with specific needs, and some models or prototypes for reference. Teams work together in close quarters throughout the project, and development stops regularly—perhaps three or four times in a typical nine-week development cycle—to assess progress and determine if changes are needed. Larger projects are built by breaking projects down into small pieces and assigning each subsection to an agile team (the method is sometimes called "swarming" in agile circles), with the overall progress controlled by a project manager.

To test the results of the system, Bailar took several in-development projects and switched them midstream from older waterfall-style development to "scrum," an agile technique that prescribes small, flexible groups that include developers and users and divides development into a sequence of 30-day "sprints." These sprints begin with a planning meeting and end with the group reviewing test results before the start of the next sprint.

He then tracked their progress against the historically expected progress of the older method, and he was happy with what he saw. Agile reduced development time by an average of 30 percent to 40 percent (sometimes nearer to 50 percent) while simultaneously improving the quality of the deliverables. He's sold, though he acknowledges that agile has its limitations.

"There are lots of things we don't use agile for," Bailar says, noting that the method excels where requirements are ambiguous and priorities are unclear, or for situations where you have the triple constraint of "faster, cheaper, better" but can't afford to drop one of the three. For extremely large projects or those with very distinct and ordered requirements, Bailar says more traditional approaches are probably a better fit.

Every Line of Code Connected to a Business Process

Robert Sherman might not see it that way, however. Sherman, the strategic methods leader for IT at Procter & Gamble Pharmaceuticals, isn't a huge fan of traditional approaches to requirements. He considers requirements only one of the first threads in a tapestry that includes everything from business processes to a finished software application. And unless IT managers begin to realize the importance of this interconnectedness, he warns, countless projects will continue to crash and burn.

Like ADP's Cumming, Sherman had a requirements epiphany in the late 1990s. At the time, he was involved in an effort to standardize all of P&G's 150 factories on a single factory-floor information management system. He and nine other experts at the company compared a 70-page specification written by the supplier to a 200-page requirements document written by P&G. Experts and vendor alike agreed that the document contained everything necessary for a successful project. It was concise. It was complete. It also "went to hell in a handbasket," Sherman says.

Poking through the rubble, Sherman at first couldn't understand what had gone wrong. Why had the seemingly ideal specification failed to produce a suitable application? He hired a contractor who spent two months tracing every requirement to every relevant sentence in the specification. P&G found that 30 percent of the deliverables were not adequately addressed. And from what Sherman could tell, the misfires were a result of being too dependent on team members' recollection of document comparison. The supplier had looked for common patterns that it could duplicate in order to reduce coding complexity. To Sherman and the others reviewing the specification, it looked—on the surface—as if those patterns matched exactly to the requirements. But they hadn't. The problem, Sherman eventually decided, was that everyone involved had simply run up against the limits of their ability to comprehend extremely complex situations. The management tools they were using were also unable to make proper connections between deliverables and the actual business processes they would support—connections that would have highlighted the subtle distinctions that turned success into failure.

Frustrated with the inability of requirements management software vendors to address the overriding disconnectedness he felt was at the core of many development problems (not just this one), Sherman decided to build a system using his own schema and a collection of tools that now includes Visio and Telelogic's Doors. The premise, he says, was simple: granular traceability. His vision was to be able to take a piece of code and quickly trace it back through the development process, back to requirements and then—rather than stopping there—map it all the way back to every affected business process to better gauge the application's impact on the business and to find hidden stakeholders.

Getting to this point has taken five years and has required the IT team to gain an encyclopedic understanding of business processes, but the results have been worthwhile. Using complex pharmaceutical project lifecycle management tools as a benchmark, Sherman says he produced the application at one-quarter the cost and with fewer than 10 percent of the expected defects compared with outside development estimates. Sherman also helped another group in P&G apply the technique to a teetering ERP implementation that seemed destined for failure thanks to a never-ending requirements process. By shifting midstream to the new schema, however, the project got back on track and now looks like it will be a success, he says.

The P&G schema has produced numerous side benefits as well, Sherman adds. For instance, an approach called "initiative scenarios" helps IT teams identify potential enterprise-level stakeholders, with the aim of converting them to sponsors before a project even gets under way. Since every requirement links back to a business process, business-side stakeholders, developers, architects and analysts can

trace their way through the organization, identifying groups that feel an impact from the new application, even if they weren't the actual end users. As an example, Sherman points to an Electronic Lab Notebooking (ELN) application (a digital data collection tool for researchers) that P&G had been having trouble getting rolled out. Previous attempts at justifying and delivering the ELN limited the requirements analysis to the lab bench and the scientist who used the notebook. But Sherman was able to demonstrate a domino effect that showed how notebook data would affect acquisitions, divestitures, patent filing and more. As a result, the IT group was able to seek additional sponsors inside the organization, and the project is heading toward 4,000 users.

"If you've done the appropriate joins [to these work processes] and you understand the linkages back to the roles, you can get the clearance ahead of time—or kill the project if you don't have the buy-in," Sherman says.

The schema also has a dramatic impact on compliance. A too-restrictive view of regulatory issues can cripple projects in the pharmaceutical industry. A too-loose interpretation, meanwhile, could open the company to legal action. But previous requirements methodologies at the company relied more on gut instinct to determine the proper balance. Now, Sherman says, compliance experts can trace their legal requirements all the way from business process to final code to determine if regulations come into play. And the schema's chartlike format and standardized sentence structure make it possible for just about anyone to trace a path, which helps users and developers prioritize requirements based on which ones will have the greatest impact on a business process.

Even so, Sherman says, successfully using the schema requires a couple of rules. Projects must be broken down into pieces. More complex applications can be built by combining these pieces, but Sherman believes that going beyond a certain level of documentation leads only to more documentation—and greater complexity—instead of execution.

Required Thinking

As these cases show, requirements processes must change, and CIOs need to drive the charge. Fixing your broken process probably won't be easy or quick, so start now.

"Today, survival depends on game changing—certainly for IT it does," P&G's Sherman says. To change the development game, "IT is going to have to understand the intersections between requirements and business processes." Failure to achieve that understanding could have dire consequences, he warns.

"If you're not rewriting the rules of the game," Sherman says, "then you deserve to have your job offshored."