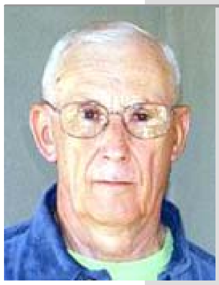

Performance Improvement

Geary Rummler & Alan Ramias
Consultants
Performance Design Labs (PDL)

GRummler@ThePDLab.com
ARamias@ThePDLab.com

The IT-Business Gap: Another Root Cause

In our last two columns, we ranted at length about the all too prevalent scenario of IT projects that have failed spectacularly, costing the companies that funded them millions of dollars (in some cases, hundreds of millions) with no usable outcomes. We also pointed out that one major cause of this phenomenon is the tendency to drive IT projects from a functional silo approach. In this column we add another cause we think is behind this dismal history.

The following are quotes from individuals in different companies that suffered large-scale IT project disasters:

- "Darryl is my right-hand man. I never make a decision without him."
- "I think this happened because the CIO told us everything was okay, until it was too late."
- "When we looked at what they built, it had none of the characteristics we had asked for. They had no explanation – take it or leave it. Arrogant little buggers, aren't they?"

In the first case, the speaker was a highly qualified mid-level technical manager handpicked by the top executive team to guide the overhaul of the company's entire legacy architecture. Darryl was the vendor, and Darryl was an expert in providing soothing explanations, dazzling PowerPoint updates, and glib reassurances, and he made all the real decisions – until the whole enterprise unraveled, resulting in a $45 million write-off and no usable code.

The second situation was a group of senior executives who stayed on the sidelines and didn't get involved, trusting the details to the CIO and not realizing how bad the solution was until it was piloted, and by then, the only recourse was to stop the project and fire the guilty.

The third quote is from a senior executive who participated in what he thought was a useful series of sessions to describe the business's desires and requirements, only to realize that the builder simply went away and built what he had built before, and the whole requirements-gathering activity was make-believe. But it wasn't until they saw the solution that the executives realized they weren't getting what they had asked and paid for.

All of these situations have something in common – a lack of adequate management oversight of IT projects. Unlike other areas of the company, IT is often treated as a realm of arcane knowledge and specialized skills where one cannot go if one lacks the requisite technical insight. This aura has been fueled over the years by what we might call the Geek Mystique.

Bill Gates is the accidental role model for this mystique. Gates became the richest man in history while speaking in a language few of us comprehended but creating products we all have come to need despite their notorious user-unfriendliness. Who are we lesser mortals to question the all-knowing, intimidating gods who create and maintain our technical systems? And if we question them too closely, can they not shut us down and otherwise punish us in ways we can only suffer but be unable to stop?

The fear factor in messing with IT is all too real in the executive suite. But this has to stop. IT must become managed and accountable just like any other function. And senior executives must pay as much attention to the details of what goes on inside IT as they do regarding what goes on inside Sales or Manufacturing. (Or as Geary might say, on a particularly cynical day, they should pay the same inattention to IT as they do to other functions.)

So what can you, if you are a senior executive, do to bring IT to heel? Some practices that would help:

- Link all IT projects to *business* requirements you actually understand. At the Performance Design Lab, we do this for clients by having them define what we call their value creation architecture (see attachment 1). We start with a view of the business as a system, define the value creation system that produces goods and services to the marketplace, and then define the processes and jobs that perform the work. Deriving technical requirements from such an architecture helps ensure, but doesn't guarantee, that IT developers are thinking about business needs when they build software. *(We realize this is a complex diagram, which we will explain in more detail in a future column, but if you have an immediate interest, please drop us a line.)*
- Insist on IT proposals and reports being in ordinary business language. Make IT talk in non-technical terms. Instead of being buffaloed by mysterious lingo, require them to make their case in plain English and provide the details of their work in words you understand. What are those words? Whatever helps you to understand in business terms exactly what you are getting and exactly what work it will take for a given project. The same goes for progress reports.
- Do not accept promises of miracles to come. Building software systems is costly and hard. Remember that. Force your people to be realistic about what a system can do, whether it is even possible to build what they would love to build. Be the world's worst skeptic if you have to, or risk being the world's biggest sucker.
- And while you're at it, stop expecting miracles. Executives ask for failure when they simply expect technologists (or experts in any part of their organization) to simply give them what they ask and brook no excuses. Listen carefully to the doubters.
- Those so-called excuses, if properly unraveled and understood, are often the earliest warning that your requests are unrealistic or the technology is being overpromised and the pain underestimated.
- Your best guide for what should be produced is what the given solution is supposed to do to improve work, from the viewpoint of the workers. If nobody has bothered to explore that, you're already in trouble.
- Get the business analysts and software designers to become experts first about the performers and their work processes, and, second, experts about technology. It is a key flaw of requirements-gathering during software development projects to let the technologists leave the room when *they* are satisfied. A requirements session should end only when the business people are satisfied that they have made their needs clearly understood and achieved an agreement.
- Technology is often described as an "enabler" of human performance, but that's a bit vague and even misleading. In highly automated processes, technology is doing far more than enabling humans, it is performing tasks – compiling and moving data from place to place, making calculations, drawing up relevant information, triggering other systems. To better understand technology yourself, think of it as a performer. What tasks is it meant to perform, what goals to reach, what standards to meet? Stay away from the language of bits and bytes, and stick with the language you already know. What does this technology do to aid the work or accomplish the work?

- Get down there where the work is done, and look. Become engaged in the early testing of functionality. Volunteer to be a guinea pig for new technology. Very often, if you can't understand it, neither will your troops.
- Hold the developers closely accountable. Look past the glowing PowerPoint reports from vendors and mid-level managers and find out what the developers are learning as they try to build the things you want. They'll tell you. There may be some fear at first, but when they realize you sincerely want to know the truth about the status of a project, they'll tell you all you want to know. Do not let them go off somewhere (a particular danger when the product is being designed by outsiders) and return only at major milestones. Track them down and stay on their case.
- Then move fast to redirect when projects are going askew. If you can't get reasonable answers to reasonable questions, kill it immediately. It's not only a signal to everyone that you're serious about accountability, you might save millions of dollars that in today's world can get burned away in a matter of a few months on failing software projects.

Managing IT projects should not be different from managing anything else. Complex it may be, but such work is as manageable as any other kind. Keep your eye on the work to be done, and always look at the technology in the context of that work. The average executive or manager is not going to be able to evaluate the intricacies of the software, but they should be able to monitor whether the work can be done or not, given the software in question.

Demystifying IT is not only possible, it may be one of the most important actions a senior executive team could take in the 21st century, given the vast dependence most companies now have on technology. All three of the cases cited at the beginning of this article had the same ending: "And then the business leaders took over…"

Business Process Architecture Linkages to Enablers and Management System

Copyright 2005 – The Performance Design Lab

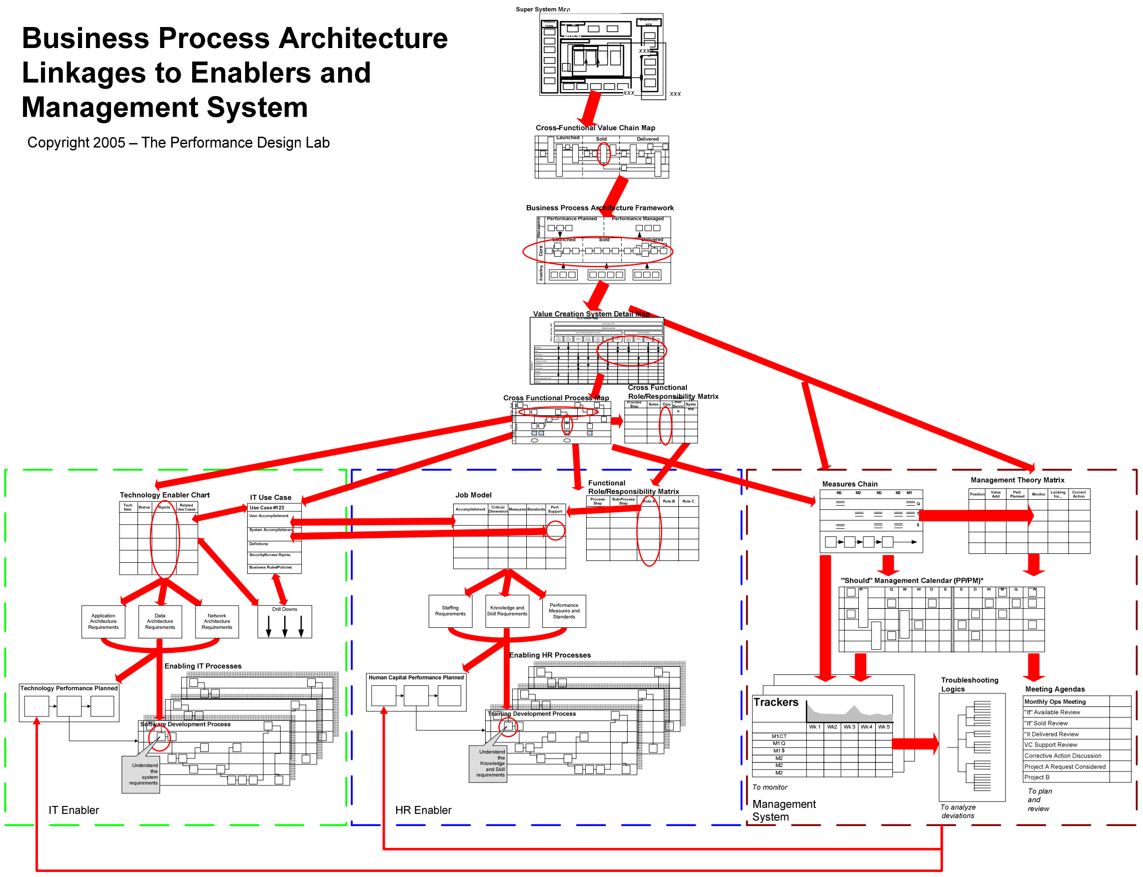
Super System Map
Cross-Functional Value Chain Map
Business Process Architecture Framework
Value Creation System Detail Map
Cross Functional Process Map
Cross Functional Role/Responsibility Matrix
Technology Enabler Chart
IT Use Case
Job Model
Functional Role/Responsibility Matrix
Measures Chain
Management Theory Matrix
Application Architecture Requirements
Data Architecture Requirements
Network Architecture Requirements
Drill Downs
Staffing Requirements
Knowledge and Skill Requirements
Performance Measures and Standards
"Should" Management Calendar (PP/PM)*
Enabling IT Processes
Technology Performance Planned
Software Development Process
Understand the system requirements
Enabling HR Processes
Human Capital Performance Planned
Training Development Process
Understand the Knowledge and Skill requirements
Trackers
Troubleshooting Logics
Meeting Agendas
Monthly Ops Meeting
"It" Available Review
"It" Sold Review
"It Delivered Review
VC Support Review
Corrective Action Discussion
Project A Request Considered
Project B
To monitor
To analyze deviations
To plan and review
IT Enabler
HR Enabler
Management System

Figure 1