Management Brief

Business vs. IT: Solving the Communication Gap

New techniques to bridge the chasm between users and analysts so systems are done right.

January 2002

Author: Dan Drislane

FRONTIER STRATEGIES

frontier-strategies.com
info@frontier-strategies.com

This paper originally published in a slightly different form at Enterprise Agility, Inc. (January 2002).
enterprise-agility.com

Frontier Strategies, Inc.
1106 West Park Street, Suite 444, Livingston, Montana 59047-2955 USA
Phone: (+01) 248-207-9020 Fax: (+01) 406-794-0506
E-mail: info@frontier-strategies.com
frontier-strategies.com

[040300101: FS_Understanding_Rules_v1.docx]

Why Projects Fail

Doing software is tricky business. It is truly a memorable event when a software project, from concept to completion, is heralded as an unmitigated success. "Successful software project" might seem an oxymoron. The Standish Group's CHAOS Chronicles[1] reports that only 28% of software projects succeed—meaning, of course, that almost three-quarters of all projects fail. While this might be a disturbing revelation, it shouldn't surprise anyone that has been directly or even peripherally involved with a software project. Software professionals, users and even customers feel the sting when a project goes south or doesn't meet expectations. And this happens more often than it should.

Standish also reports that American companies annually spend upwards of $275 billion on about 200,000 application software projects. While the research firm roundly points to the lack of skilled project management as the chief cause, they also report on the results of an earlier survey[2] given to IT managers asking them to name their top project success criteria. Of the top ten weighted criteria, the top three were:

1) User Involvement – 19%

2) Executive Management Support – 16%

3) Clear Statement of Requirements – 15%

"Competent Staff" and "Hard-working, focused staff" were factors #7 (8%) and #10 (3%), respectively. Though the first two criteria rely mainly on management practices and cultural values, the third criteria—*Clear Statement of Requirements*—is something IT professionals and users can sink their teeth into. A subsequent exercise by the managers surveyed by Standish was to deconstruct the criteria even further by answering questions identified by the group. For *Clear Statement of Requirements*, the following questions were posed:

- Do I have a concise vision?
- Do I have a functional analysis?
- Do I have a risk assessment?
- Do I have a business case?
- Can I measure the project?

This paper aims to focus on two additional factors not listed by The Standish Group, but certainly implied: *business processes* and *business requirements*. While a project must have good analysis, pragmatic risk assessment, a sound business case and reliable measurement tools if it is to have any hope of succeeding, *business processes* and *business requirements* are inextricably linked to a company's vision and the project itself. Closely coupling business processes and the business requirements of a new application are not only desirable, they are inherently critical. Business software applications are tools to aid business processes.

[1] The Standish Group. *CHAOS Chronicles II*. 2001. www.pm2go.com
[2] Sixty IT managers were surveyed at The Standish Group's CHAOS University.

In the sections that follow, we'll see why this important relationship is a key factor in successfully completing software projects, and why its most prominent stumbling block, the communication gap between users and the IT community, is responsible for a large proportion of project failures.

The Dissociation Game

The successful completion of information technology (IT) projects is often plagued by two persistent problems: (1) differences between users and the IT community in understanding true business requirements, and (2) a lack of understanding the business processes that use the application to be built. This is something we call the *dissociation*[3] *game.* The dissociation game can be divided into two realms:

- Requirements dissociation
- Process dissociation

With requirements dissociation, business requirements stated by users are misunderstood by the team designing and constructing the application under development. The requirements may also be incomplete, but not due to the users' negligence, as we will see. It seems obvious that the IT project team—the analysts, programmers, testers and the project manager—must have a clear understanding of the business requirements as expressed by users, but the team has no hope of launching an application if it doesn't have a clear and comprehensive handle on what the user needs. Unfortunately, those needs are usually expressed in a language altogether alien to the IT team. While users tend to give requirements in terms of their job responsibilities or the metrics of business, IT folks use a vocabulary built on systems.

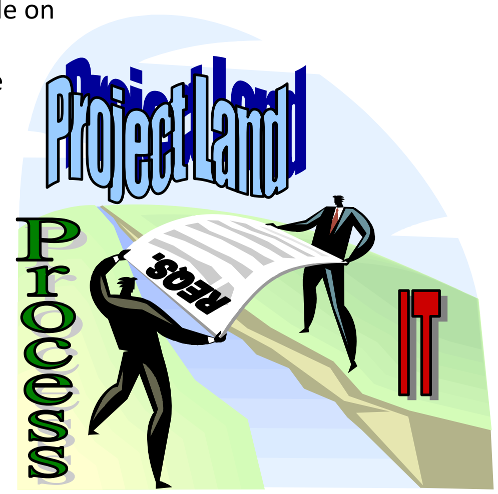

Figure 1: Process, business and IT organizations are often dissociated from each other, making elicitation of true business requirements difficult.

Compounding the problem is process dissociation. There is a close relationship between a company's business processes and the applications that must support those processes. Once a project is underway, the clock is ticking and it's easy to overlook the fact that business processes strongly influence most software application development efforts. Undocumented or misunderstood business processes often lead to inaccurate or incomplete business requirements, which can lead to the wrong system requirements. Relying on business requirements alone is ineffectual because they often lack business context, context that is partly derived from understanding the business processes and operational scenarios that give rise to the requirements in the first place. Standalone business requirements—what are usually termed *line item requirements*—provide the analyst with little knowledge as to how each requirement fits into the business and how they relate to each other.

[3] **dissociation**. n: the state of being separate and unconnected. (Source: www.dictionary.com)

As astonishing as it may seem, many IT teams start a software project with little to no knowledge of the business processes requiring, or using, the application. If the team is lucky, this handicap will be apparent soon after the outset and the schedule can be expanded to include the ramp-up time needed for the analysts to meet with process stakeholders. Unfortunately this is seldom the case and even if the team recognizes it needs to understand the business better, this activity is usually shoehorned into the requirements gathering/analysis phase. Such sloppy practices lead to incomplete analysis and broad assumptions that are only uncovered later in the project. Now in crisis mode, the project can devolve easily and result in earnest but vain attempts to fill in gaps in requirements or functionality where they're needed, regardless of merit or value to the project.

Of course, the root cause of this problem is that IT organizations are often disconnected from the company's process efforts. And though industry leaders have been after companies for years to push responsibility for process to all corners and levels of the organization—including IT—most enterprises dole out their process worries to a dedicated group, or even more common, a process consultant.

With process dissociation, business users are focused on documenting their specific needs for the future system (those *line item requirements* we mentioned earlier) and are sometimes only peripherally aware of what's happening down at "process central." Meanwhile, the IT project team is wrestling with one of several problems of its own: (1) it may not have access to process documentation; (2) even if documentation is available, the *as-is* processes don't accurately reflect what is actually done day-in/day-out; or (3) they will be working with users who may not understand the scope of the business processes involved or, even worse, that may have an outmoded understanding, especially if process improvements are being made by a separate group. (While we recognize that some companies have tuned their practices to anticipate and avoid these problems, they seem to be few and far between.)

Process dissociation is quite common among project teams and its constituencies. Ultimately, it creates a barrier to all parties having (or actually earning) a holistic understanding of the problem to be solved—the original intent of the application to be built.

Tracing Your Roots

There are other challenges for users and the project team. Even if requirements and process dissociation are conquered, translating those business requirements (that are supported by processes) into a useful system is still a daunting task. If all projects were stable and static—with no external drivers such as changing market conditions, competitive developments, and new technology requirements muddying the water—the smooth progression from process to requirements to design and implementation would be achievable. Seldom, if at any time, is that the case. Projects must roll along with the dynamics of the business. And to meet this need, tools should be deployed to react to these external influences and weather the stormy transitions.

Simply put: You can't build a successful system if your design fails to map back to the business requirements and processes.

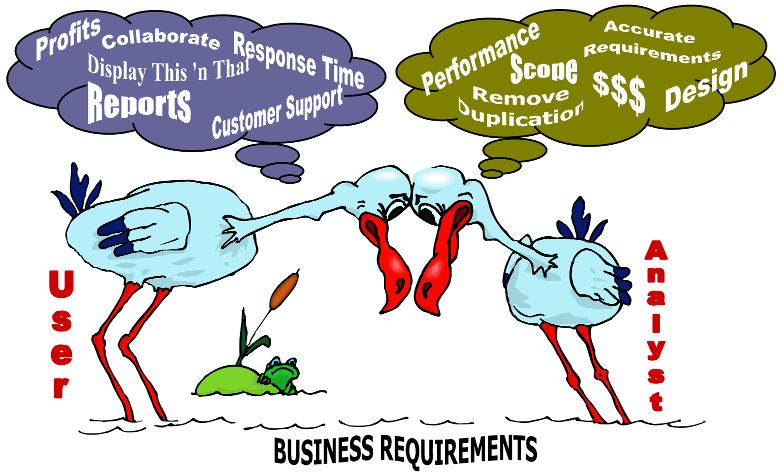

Figure 2: New tools that offer bridging functionality can allay problems caused by accommodating different perspectives regarding the system to be built. End-to-end traceability assures that artifacts produced by project analysts and designers trace back to the users' disparate business requirements.

Considering the dynamics of the typical software project, a natural goal of the project team should be to assure that what it will build supports the agreed upon needs of the users, even after—and despite—the myriad changes that threaten the scope and milestones of the project. This is what is commonly termed *traceability,* the ability to trace back to the roots of what is required of the application: the vision and the original line item requirements.

Traceability earns its keep when things in the project go awry, which are nearly always the case. When market conditions change, the business reorganizes or funding is cut, projects must respond accordingly. Tracing back to the roots of the project—those processes and requirements that guided the analysis and design—is critical if the project is to adjust to new circumstances and see what needs to be re-analyzed. The lack of end-to-end traceability between business requirements and what is actually implemented creates a formidable chasm for both the project team and the project itself, which by now, if not already, is fraught with risk. If change isn't anticipated and traceability isn't built into the project, it's usually at this time that another crisis rears its head. Schedules slip; cost overruns occur; and the resultant architecture will turn out flawed.

All said, these three problems—the stakeholders and IT project teams speaking a different requirements language, not understanding business processes, and the lack of traceability of system design to business requirements to process—accounts for a large share of project overruns and lapses in system performance and acceptance by users. Simply put: You can't build a successful system if your design fails to map back to business requirements and processes.

Bridging the Chasm

In most enterprise organizations where business processes span multiple functional areas and observe complex operational rules, the ability to implement systems that actually meet the users' current and future needs depends, as we have said, on understanding an organization's business processes—basic or complex as they may be. Indeed, to develop system requirements for complex businesses, both users and the project team's business analysts require a collaborative relationship so they can understand the *as-is* business processes and help, where necessary, to define the *to-be* processes. Both parties must perceive and understand how business practices—past and future—will impact the business requirements and ultimate design and implementation of the system to be built.

One challenge that business and IT organizations need to overcome is bridging their different perspectives. As we explained earlier, regardless of the level of intimate collaboration, the language of the user and that of the IT team are quite naturally different.

Until recently, the technology available to the IT industry to *refactor* business requirements into the necessary systems perspective was inadequate, cumbersome, or at worst non-existent. Limitations forced many businesses and IT organizations to rely on methodologies and tools that focused on system development modeling for their business requirements efforts. Software productivity tools were ill suited for understanding the business perspective. When users and analysts work together to define business requirements, use of IT-centric techniques can cause frustration and miscommunication among the collaborators. Both parties need to find common ground and a common modeling language to be able to effectively work together.

There is also a tendency to stereotype business users and business analysts in roles that severely limit their effectiveness and productivity. Users are concerned with solving business problems while IT analysts are predisposed to the technical aspects of solving those problems. This is less true today as companies attempt to align IT efforts more closely with the needs of the enterprise.

Many organizations have implemented more progressive techniques to capture the voice of the customer (i.e. users) and make sure the analysts and designers are involved firsthand. Joint Application Development (JAD) and facilitated requirements sessions are more common among many companies today than even several years ago.

Application software vendors have seen this trend toward more intimate collaboration (and some industry observers would argue they have influenced it). Combined with a maturing IT standards movement centered on object-oriented (O.O.) analysis and programming techniques, vendors are offering more robust system lifecycle application development tools that allow teams to integrate process design features with business (line item) requirements, a capability known as *end-to-end traceability*. In an end-to-end software lifecycle that uses traceability, business processes and business requirements can be traced to the derived system requirements and, ultimately, to the design and implementation of the system. Being able to forward or backward trace to and from all project artifacts is invaluable for all stakeholders of the new system.

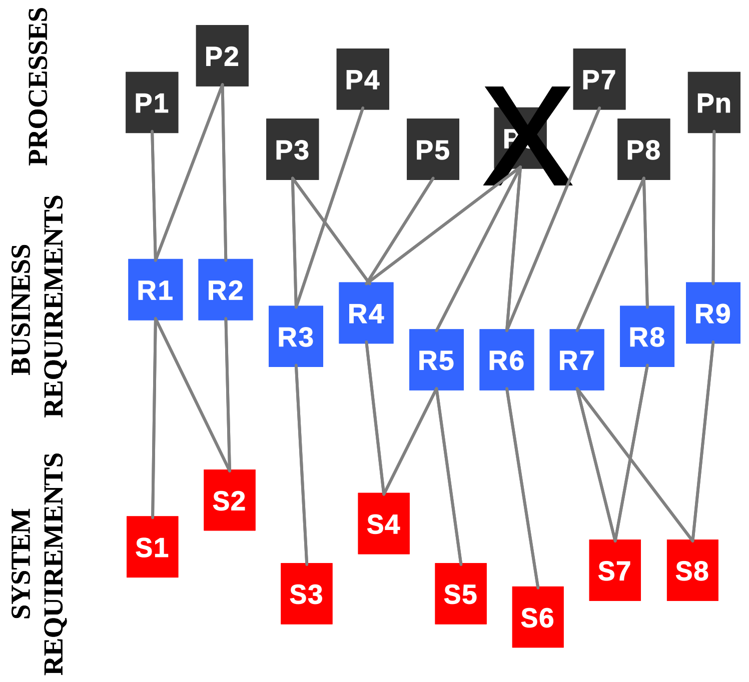

Figure 3: End-to-end traceability affords visibility throughout the lifecycle. If Process 6 (P6) is retired, it affects Business Requirements 4, 5 & 6, which impact System Requirements 4, 5, & 6. Similarly, if a technology glitch affects System Requirement 7 (S7), Business Requirements 7 and 8 should be re-evaluated, as should Process 8.

Why is this important? Were it that projects were small, business processes were few and well understood, and business requirements numbered in the tens, projects might even finish on time and deliver the goods everyone had asked for. The problems reported by the Standish Group might be relegated to the past. Even better, the team could in all likelihood keep track of everything on a whiteboard or a spreadsheet.

When budgets are cut, new requirements are added, or priorities are shuffled, traceability plays a key role.

However, this simple view is far from reality. Projects encompass multiple user groups, numerous business processes, dozens of business rules, and hundreds of disparate line item business requirements from many users. Even after diligent analysis and refactoring of business requirements, the system requirements often number in the hundreds as well.

As complex as projects can be, this might all be well and good if things didn't change. But we now know that's never the case. When budgets are cut, new requirements are added (sometimes because they "just have to be"), or priorities are shuffled, traceability must play a key role. Analysts and designers can't possibly adjust the scope or cut functionality unless they know implicitly how the change will impact the project. Knowing what processes and business requirements are affected is a key advantage for all concerned.

End-to-end traceability has three additional advantages. First, few users and stakeholders realize that they should have at their ready a "crumb line," that barefaced record of how their business requirements are coursing through the software development lifecycle. Being able to show, from one stage to the next, through several or more iterations, how business line item requirements are distilled into detailed system requirements, is a demand that both the users and project team should require of each other. As shown in Figure 4, people have natural learning traits, gaining incremental knowledge of the subject at hand as they probe and analyze different facets, over and over. The same holds true for IT development projects. While infinite analysis is ludicrous, it is equally unrealistic to learn and then develop, in singular succession, business requirements, system requirements and the ultimate design of the system in one fell swoop. (This is largely why traditional system lifecycle approaches, such as the *waterfall* methodology, have often failed.)

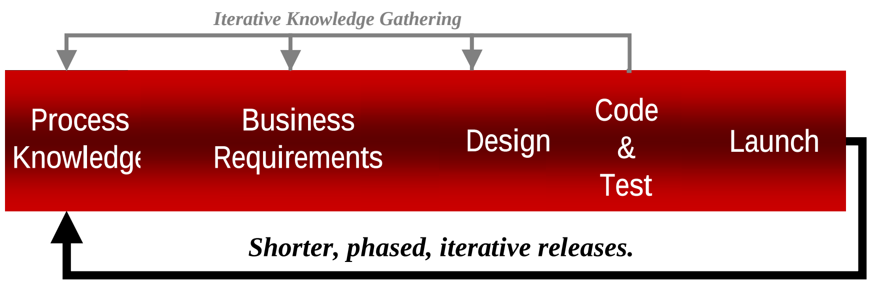
Iterative Knowledge Gathering
Process Knowledge
Business Requirements
Design
Code & Test
Launch
Shorter, phased, iterative releases.

Figure 4: Iterative learning is a natural human trait that can be supported by modern IT tools, which also provide end-to-end traceability—from process to launch of the system. Frequent but smaller releases of software better insure conformance to the organization's business practices and the user community's business requirements.

Second, end-to-end traceability assures that the development and execution of the user acceptance test scenarios accurately satisfy the true requirements expressed by the users. Tying user acceptance test cases to requirements has always been difficult, especially when business requirements are numerous. Testing planners would be forced to select and perform triage on what requirements they would test and build into the plan, and this approach sometimes left the most critical aspects of the system unproven. Tools are available today that provide total forward and backward traceability, which in turn gives the project team and users a comprehensive view of how the test cases will address requirements, based on priority and affinity.

And finally, end-to-end traceability has long-term advantages. After the initial release of an application, the software begins its life as a legacy asset that over time is maintained and enhanced. As bugs and enhancement requests flow in from users, it is important to be able to trace back to original business requirements and legacy business processes to get a holistic view of the impact of each request. With each subsequent maintenance or enhancement release, the IT team can assure that business processes as well as old and newly acquired business requirements are considered in terms of real cost, usability, and customer/user acceptance. Software tools are available today that use traceability to allow change management to be integrated into the system lifecycle process.

Summary

This paper has endeavored to explain how companies can overcome three problems that are all too common in software development projects:

- Lack of communication among process champions, users and the team that will build the new system;
- Lack of sound methods to capture the voice of the users (line item requirements) that enable analysts to produce accurate system requirements and a viable design; and,
- Lack of end-to-end traceability that provides frequent feedback to users, stakeholders and the IT team—which includes the business and system analysts, designers, architects and testers—so that the project can consider priorities, analysis and design decisions, then make adjustments as required.

Companies can solve the communication gap among users, process analysts and IT project teams by recognizing that all stakeholders must have all the cards on the table—the *as-is* processes that are suspect (due to new requirements); the *to-be* processes that are needed; the business line item requirements that are expressed in the voice of the user; the requirements that are validated against *as-is* and *to-be* processes; and, finally, the system requirements that are derived after painstaking (and iterative) analysis of the processes and business requirements. Leaving one of these components out of the puzzle is a risk.

End-to-end traceability can only be accomplished when there are tools that are deployed that work to accommodate dynamic, changing projects, which, we have suggested, is almost all of the time. Software applications are commercially available that provide direct links to business process documentation (steps and tasks), the functionality of the system, the users' disparate business requirements, the subsequent derived system requirements, the design features and, eventually, their components. Both forward and backward linking from one project artifact to the next provide powerful capabilities to react and adjust when changes to the project occur, due to unforeseen market conditions, organization changes, funding adjustments, and new business functionality priorities.

Surely there are scores of other problems that we have not discussed, nor have we offered solutions, and other techniques that deal with requirements and the solution lifecycle should be explored. However it is our experience that the vast majority of projects share one trait: they falter due to the lack of understanding the big picture (processes, business requirements, system requirements, design, and probable solution) and the attendant ramifications when something changes (end-to-end traceability). While these problems are rampant among companies today, there are tools and techniques that can be deployed to help close the communication gap between business and the IT organization. While the tools are available, a carefully planned approach to incorporating these and other best practices into future projects is recommended.

About the Author

Dan Drislane is the founder of Frontier Strategies, Inc. in Livingston, Montana, an IT consulting firm begun in 1991 in Michigan. He has over 20 years of experience in business analysis, business process analysis and project management. Dan's work with clients focuses on two goals: transforming the IT organization's culture so it will be more agile and accountable to business; and integrating an organization's vision and supporting business processes with its enterprise business and system architecture.