Why New Systems Fail

Theory and Practice Collide

By Phil Simon

Trying is the first step to failure.

-Homer Simpson

CHAPTER 1: INTRODUCTION

With regard to new systems, Homer Simpson is usually right.

Specifically, more than three in five IT projects do not do what they were supposed to do for the expected costs and within the expected timeline.[1] More gloomy stats include the following:

- 49 percent suffer budget overruns.
- 47 percent result in higher than expected maintenance costs.
- 41 percent fail to deliver the expected business value and ROI.

Ouch. Or, as Homer says, "D'oh!"

The process of implementing systems can hardly be called a recent business development. If it were, then perhaps the failure rate would be understandable. It is not. Organizations have been using major enterprise systems for decades but, as a percentage, few have met their original goals and promise. Reflecting on more than a decade of implementing systems in many countries, industries, and organizations, I have come to one conclusion:

The statistics above are probably understated.

I have many reasons to question these statistics. For one, many executives do not want to admit that their pet projects ran over budget and/or missed deadlines. What's more, the simple time and money criteria mask less obvious system failures. The project that "makes its date and hits its numbers" may still not give the organization the expected bang for its buck. Many end-users fail to use the new system's enhanced functionality and revert to old habits. The average CIO probably does not have the following in mind prior to implementing a

[1] http://advice.cio.com/remi/two_reasons_why_it_projects_continue_to_fail

new system: spending an enormous amount of money essentially to replicate the problems of the previous system.

Speaking from years of personal experience, I have been involved with very few system implementations that could be categorized as unqualified successes. This begs the obvious questions:

- Why do so many system implementations fail so spectacularly?
- What can be done to increase the chances that organizations' use of new technologies will be successful?

In a nutshell, those two questions drive this book, divided into the following five parts:

- Deciding to Take the Plunge
- System Selection
- The System Implementation
- The Brave New World of Post-Production Life
- Maximizing the Chance of Success

Fantasy World: The Theoretical System Implementation

At the onset of a project, senior management usually believes that their organizations will implement and activate new systems as follows:

- The system is selected.
- The system is implemented, during which time all issues are found and promptly addressed.
- The system is activated.
- Consultants remain on site for a few weeks of "post-production" support, ensuring complete end-user knowledge transfer.
- IT and functional end-users can independently maintain the new system after consultants leave.

Based on this expected sequence of events, executives approve these massive projects. Unbeknownst to them, however, the process of implementing and activating a system is almost never this easy. After over a decade in the field, I have *never* come across a system implementation that has gone this smoothly.

This book does *not* attempt to list every possible cause of system failures. There are too many variables: type of system, industry, size of

organization, budget, timeline, dates, and so on. Rather, this book delves into the "usual suspects," to borrow a phrase from one of my very favorite films. To the extent that each implementation is different—if not unique—the causes of system failures can and do vary on each project. For example, consider data issues: One organization may have pristine data in its legacy system[2] while a second may have wildly divergent data. Still, a third may have very minor data issues.

This book focuses on the most commonly encountered reasons for system failures.

The Anatomy of a Typical System Failure

While the specifics vary quite a bit based on industry, type, and scope of the new system, many enterprise resource planning[3] (ERP) failures have similar root causes, issues, costs, and ultimately, consequences. The similarities do not end there. Organizations also tend to experience these problems and disappointments whether they purchase and implement an existing system or if they build a system—internally or via an independent service provider[4] (ISV).

Table 1.1 represents a typical step-by-step failure of a fictitious system, Bonham Software. The client in this example is Page and Plant (P&P).

---

[2] A legacy system is defined as "an old computer system or application program that continues to be used because the user (typically an organization) does not want to replace or redesign it." (Source: Wikipedia)

[3] Enterprise resource planning (ERP) is an enterprise-wide information system designed to coordinate all the resources, information, and activities needed to complete business processes such as order fulfillment or billing. An ERP system supports most of the business systems that are maintained in a single database, the data needed for a variety of business functions such as Manufacturing, Supply Chain Management, Financials, Projects, Human Resources and Customer Relationship Management. (Source: Wikipedia)

[4] Independent software vendor (ISV) is a business term for companies specializing in making or selling software, designed for mass marketing or for niche markets. Such markets may be diverse including software for real estate brokers, scheduling for healthcare personnel, barcode scanning, stock maintenance and even child care management software. (Source: Wikipedia)

Table 1.1: The Typical System Failure

Task	Notes
System Selection	
The senior management of P&P decides that it needs to buy or build a new enterprise system.	
Vendor selection begins. Formal RFIs (requests for information) and RFPs (requests for proposals) are sent to major vendors or ISVs.	
Vendors or ISVs send formal responses, hoping to make the final cut.	
P&P personnel evaluate vendor/ISV responses. Decisions about "next steps" are made and communicated to these organizations.	
Vendors and ISVs make formal presentations to key P&P constituents.	
P&P personnel evaluate proposals and references provided by vendors and/or ISVs.	
P&P narrows down its selection to two vendors or ISVs based on perceived functionality, cost, time to implement, references, track record in the industry, and so on.	
P&P selects Bonham Software as its future enterprise system. For the sake of simplicity, assume that Bonham wins the consulting business as well.	Typically, Bonham will also attempt to win the consulting business. Many times, however, vendors such as Bonham recommend certified partners for the work in the event P&P deems Bonham too expensive or Bonham does not have the resources to complete the implementation.
System Implementation	
The implementation of Bonham Software formally begins.	
Unanticipated data and conversion issues manifest themselves, causing significant project delays. The climate becomes adversarial within among key internal and external players.	The project begins to fall behind.

Task	Notes
Testing is consequently delayed and additional "parallel" tests are added to see if fixes have resolved identified issues.	Delays mount.
Delays and "rework" typically feed into employees' previously approved vacation time or key deadlines (month-end, quarter-end, year-end closing, W-2 printing, and the like). The net result is that delays of two weeks can quickly become two months.	These delays should not be taken lightly, as key players may be unable or unwilling to move their personal commitments.
Because of certain hard deadlines such as January 1st or the beginning of a new quarter, P&P devotes less attention and fewer resources to properly document current and future business processes.	The downstream impact of this can be enormous, especially if a key resource leaves the organization without sufficiently documenting or training others in how to conduct a key business process.
Functionality and features promised for Phase I of the implementation are pushed to later dates.	
Data challenges cause issues with interfaces to vendors. Vendor delays in returning test files exacerbate the problem, as they typically have a multitude of clients and cannot focus exclusively on P&P.	Interfaces may include health coverage for employees, direct deposit for employee checks, employment verification, etc. These must work upon going live; there's no room for error here.
A "go/no-go" meeting is held at which point key players determine if they are ready to proceed with system activation.	Despite the objections of several key players, P&P forges ahead, ostensibly aware of the risks.
P&P activates its new system.	Unfound and unsuspected functional, data, training, and interface issues manifest themselves causing emergency "patchwork" or corrections.
Post-Production Support	
P&P end-users are unprepared for issues that may arise. They still have much room for improvement many areas: error correction, reporting, and so on.	
Bonham consultants plan to remain at P&P for two weeks to ensure a smooth transition and to address any system issues.	Ultimately, Bonham's stay lasts months afterward (at considerable additional expense to P&P).

Types of Failure

Not all failures are created equal and there certainly are *degrees* of failure. This book defines four major types of system failures, one of which may not become apparent until months after a new system has been activated. Each case study is classified in terms of the following failure scale:

- The Unmitigated Disaster
- The Big Failure
- The Mild Failure
- The Forthcoming Failure

The Unmitigated Disaster

The most egregious failure occurs when an organization spends millions of dollars implementing a system and misses deadlines repeatedly. It ultimately junks the new system for a different one altogether or reverts to the legacy system. Relationships between consultancies and clients are often severed. Lawsuits in such cases are not completely out of the question. Fortunately, these abominations are atypical.

The Big Failure

These types of failures are less severe but more common. Perhaps an organization initially budgets $2M and one year on an implementation and ultimately spends $4M over the course of three years, getting much less functionality than expected in the process.

The Mild Failure

Very often, a system failure is so mild that one can hesitate to even call it a failure, especially relative to the two types just mentioned. By comparison, these are rousing successes! For the sake of consistency, however, this book uses the term "failure." An example of the Mild Failure is the company that initially budgets $2M and a year on an implementation and ultimately spends $2.2M over the course of fifteen months, getting slightly less functionality than expected in the process.

The Forthcoming Failure

Sometimes a system failure is not immediately apparent. At first, this notion may seem perplexing. **If an organization has met its goals with respect to both its budget and deadline, then how can it consider the system a failure?**

Budget and deadline are only two criteria for a system failure, as the statistics at the beginning of the chapter illustrate. The answer to this question lies within the organization's data, documentation, processing, and people. Examples of latent failures include the following:

- The implementation team has made a key mistake that will come back to haunt the organization down the road.
- End-users may not completely understand the system and, as a result, make significant errors or revert to "old ways," negating one of the major benefits of the new system.
- The organization is vulnerable to employee attrition on two fronts:
    - End-user documentation is deficient and, if key staff members leave, their replacements will need significant time and/or training to do their jobs.
    - Knowledge is not dispersed; only a few employees understand the system in sufficient breadth and depth.

In other words, these are failures waiting to happen. Organizations are not prepared for shocks to their systems.

A Prime Example of the Forthcoming Failure

I am reminded of an organization—Oates Healthcare—that activated its ERP in 2003 with a fundamental but unknown problem with the way in which it calculated employee overtime. No one identified this issue during setup or testing. Only when an ex-employee filed a lawsuit did the problem come to light, five years *after* Oates had gone live.

For Oates, fixing the problem in the system involved two things, one simple and one very difficult. The first merely entailed changing some flags in the system, allowing it to begin calculating overtime correctly *from that point forward.* However, checking those flags did not retroactively go back and recalculate overtime for all employees paid incorrectly over the past five years. A breakdown of those errant employee records is presented in Table 1.2:

Table 1.2: Breakdown of Payroll Records Requiring Analysis at Oates Healthcare

Employees paid per year	6,500
Average types of pay per week per employe	6
Weeks per year	52
Checks per year	338,000
Payroll records per year	2,028,000
Years of data requiring analysis	5
Total records (over a five year period)	10,140,000

The enormity of this task—recalculating employee overtime—was beyond the time and skill of Oates' existing end-users (arguably a failure in itself). Even if an internal super-user knew how to do this on over 10,000,000 records, he or she would not have been able to do it. For *ad hoc* analyses, Oates provided its end-users only with Microsoft Excel, a very valuable tool but one not nearly robust enough to handle a task of this magnitude. As a result, Oates hired Bishop Consultants to perform this task at considerable expense.

Had Oates' end-users properly tested the system prior to going live, it may have avoided the lawsuit. To be sure, it would have not have had to spend the time, internal resources, and funds on Bishop to fix the problem. Bishop recalculated employee overtime pay but Oates' end-users were not available during the remediation project. This prohibited Bishop consultants from transferring any knowledge during the error-resolution process.

Ironically, Oates did not learn from its mistakes. Despite the recommendation from Bishop, Oates did not seriously consider adding a more powerful reporting tool for end-users to conduct similar kinds of analyses (e.g., Crystal Reports, Microsoft Access, or Business Objects). Oates also failed to actively recruit more technical end-users to use these very tools, should it one day have decided to purchase them. If Oates encounters a similar problem in the future, then it will be at the mercy of external consultants such as Bishop once again.

Consequences of a Typical System Failure

The consequences for a failed implementation go beyond mere dollars and cents. Let's return to the P&P example. Bonham Software may forever have a tarnished reputation within the organization among both end-users and employees who actually do not even use the system

on a regular basis. Bonham may always be known as "that system that screwed up payroll." Data could be lost or altered in such a way that it will be impossible to retrieve. Due to lack of training or documentation, employees' jobs may actually become *more* difficult than they were with P&P's legacy system.

For Bonham Software, as a company, the project was a disaster. Bonham now has a tarnished reputation in the industry resulting from this highly publicized failure. It may have difficulty collecting the hundreds of thousands in accounts receivable from P&P and lose key consultants. P&P may also refuse to provide a reference for its new partner.

As stated before, many types of issues typically haunt system implementations. While each of the case studies detailed in the book differs in terms of the way in which the organizations employed technology and even in the technologies themselves, there is considerable overlap among the issues encountered.

The Expectations Gap

Table 1.1 illustrates that system implementations typically leave many parties disappointed in the ultimate outcomes. Senior management expects the new system to be implemented smoothly, on time, and within the planned budget. Client end-users expect to learn how to properly use the new system and be self-sufficient when consultants leave. Consultancies expect strong client references. For each party, these expectations are often unmet, many times by significant degree.

Disappointments often give way to disasters. A less-than-inconsequential percentage of these projects have their plugs pulled mid-implementation. Organizations sometimes go live when they are wholly unprepared to do so. Issues abound and the benefits and cost savings once promised by the vendor and consultancy may be significantly less pronounced than what clients ultimately see. In retrospect, after system activation many clients opine that the new system is a far cry from what they expected when senior management signed the original contracts.

Risks for Mature Organizations

While this book focuses organizations implementing new systems, the content applies to the maintenance, enhancement, and support of existing systems as well. Mature systems can fail in several ways. First,

successfully-activated systems often begin show signs of future problems. Second, a Mild Failure could easily become either a Big Failure or, in extreme cases, an Unmitigated Disaster. In other words, just because a new system goes live on time and under budget does *not* mean that an organization is out of the woods. There is still significant risk. Systems can and often do begin to experience major difficulties after even successful activation, attributable to:

- Key employee turnover
- System upgrades and the decommissioning of older versions of the application
- The introduction of additional functionality within a system
- Changes to business processes
- Acquisition of a company and the integration of additional legacy systems
- Unwise expansion

A Balanced Approach: Theory, Case Studies, and Examples

Throughout this book, theory and practice are given equal weight with respect new systems. Consider system testing for a moment. The book does not simply espouse the virtues of system testing; to do so would be facile. After all, all consultants and implementation teams intend to run proper parallel tests.[5] How, then, should the importance of—and frequent missteps associated with—testing be illustrated? By drawing upon extensive examples and detailed case studies, the book manifests the *essential* questions that cause testing to be compromised. These include:

- What causes system testing to produce unexpected results?
- What are the effects of failed testing on the project's timeline, budget, and ultimate outcome?
- Most important, what specifically can an organization do from the beginning—and during—a project to promote accurate, timely, and comprehensive testing?

---

[5] Parallel testing is the process of feeding test data into two systems—the modified system and an alternative system (possibly the original system)—and comparing results. Source: www.astrainfotech.com.

The examples and case studies in this book stem from actual system implementations but, for reasons of confidentiality, the names of the organizations, consultancies, and individuals have been changed. Specific names are not nearly as important as the lessons they provide. As we will see, many ostensibly different organizations face similar—if not identical—challenges implementing systems.

It's commonly said that one learns more from failures than from successes. To that end, this book's case studies and examples will examine in great detail system implementations that failed, identifying the specific individuals, decisions, and events responsible for the outcomes. This book contains eight detailed case studies:

Table 1.3: List of Case Studies in Book

Case Study	Chapter	Description	Result
Costanza	9	A Flexible Client and a True Consulting Partner	Averted Failure
Lifeson	11	Replicating the Old in the New	Unmitigated Disaster
Portnoy	11	A Square Peg and a Round Hole	Unmitigated Disaster
Elton	13	The Stubborn Client	Big Failure
Julian Marketing Partners	15	Building its Own System	Mild Failure
Wilson	15	Good Design but Lack of Resources	Big Failure
Petrucci	18	Trying to Boil the Ocean	Big Failure
Tate	22	Adding on to a Poor Foundation	Forthcoming Failure; Became Unmitigated Disaster

Who Should Read this Book?

This book has no one intended audience. As a full-time independent consultant, I wish that I had known many of the things in this book when I started working with systems back in the mid-1990s. (Maybe I wouldn't have gone into the field, but that's a separate matter!) Many current and aspiring consultants would benefit from the advice dispensed in this book, both on a general level as well as in terms of any specific projects on which they may be working.

Beyond consultants, this is a book for end-users at all levels within an organization. First, CIOs thinking about implementing a major new system could learn a few things about vendors and consulting companies *before* they plunk down hundreds of thousands of dollars. Second, many organizations are already somewhere in the middle of such a project. The subject matter, if applied properly and in a timely manner, may help these organizations avoid many of the outcomes detailed in the case studies and examples. Regardless of the stage of the implementation, the practical tips have the potential to right the ship. Understanding the causes of system failures should help organizations avoid them.

Summary

The reasons typically vary but the outcomes of many system activations are sadly the same. Many differ only to the degree to which they have disappointed or outright failed. Rare is the implementation that meets its deadline at or under budget and gives end-users the functionality promised from day one.