There's no such thing as an IT Project

A HANDBOOK FOR INTENTIONAL BUSINESS CHANGE

BOB LEWIS & DAVE KAISER

THERE'S NO SUCH THING AS AN IT PROJECT

This page intentionally left blank

There's No Such Thing as an IT Project

A HANDBOOK FOR INTENTIONAL BUSINESS CHANGE

BOB LEWIS & DAVE KAISER

Berrett–Koehler Publishers, Inc.

There's No Such Thing as an IT Project

Berrett-Koehler Publishers, Inc.
1333 Broadway, Suite 1000
Oakland, CA 94612-1921
Tel: (510) 817-2277, Fax: (510) 817-2278
www.bkconnection.com

Ordering information for print editions
Quantity sales. Special discounts are available on quantity purchases by corporations, associations, and others. For details, contact the "Special Sales Department" at the Berrett-Koehler address above.
Individual sales. Berrett-Koehler publications are available through most bookstores. They can also be ordered directly from Berrett-Koehler: Tel: (800) 929-2929; Fax: (802) 864-7626; www.bkconnection.com
Orders for college textbook/course adoption use. Please contact Berrett-Koehler: Tel: (800) 929-2929; Fax: (802) 864-7626.
Distributed to the U.S. trade and internationally by Penguin Random House Publisher Services.

Berrett-Koehler and the BK logo are registered trademarks of Berrett-Koehler Publishers, Inc.

First Edition
Paperback print edition ISBN 978-1-5230-9883-5
PDF e-book ISBN 978-1-5230-9884-2
IDPF e-book ISBN 978-1-5230-9885-9
Digital audio ISBN 978-1-5230-9887-3

2019-1

Set in Palatino by Westchester Publishing Services.
Cover designer: Adam Johnson

To the many longtime readers of InfoWorld*'s "IS Survival Guide" and its successor, my weekly* Keep the Joint Running *e-letter. They have, over the years, kept me motivated, honest, and most important, better informed.*

—Bob

To the second boss in my career, Dave Crowley, who once told me, "Kaiser, if you ever stop complaining you will have a great career." Dave was a great mentor and taught me the importance of being open and honest with those you care about.

—Dave

This page intentionally left blank

Contents

Foreword

Anita Cassidy

Partner at ITDirections (www.itdirections.com)
and author of five books, including *A Practical Guide to Information Systems Strategic Planning*.

Why is this book important? It's because today, technology is at the core of all businesses. It is difficult to find a business strategy or business process that does not depend on technology for its success. In fact, for differentiating, innovating, or disrupting organizations, the business and the technology are typically inseparable. Just look at companies such as Uber, Airbnb, Amazon, Houzz, Netflix, Zipcar, Tesla, Instacart, Salesforce, Facebook, Google, or Apple; it is difficult to see the business and the technology as separate entities. The business is the technology, and the technology is the business.

For decades, IT practitioners have hammered on the idea that the business and IT need to be aligned. In today's world, with technology and the business essentially being one, alignment isn't enough anymore. As Bob and Dave point out, IT can't just be aligned with the business—it has to be integrated into it. Which is why many are coming to see that there is no such thing as an IT project. They're business-change projects that are using technology as an enabler.

Digital transformation is the current discussion and hype in the industry. What is Digital transformation? It is nothing more

than business change, effectively using technology throughout all aspects and touchpoints of the business.

This book is important because it's about change. Companies in all industries and of all sizes must continually change to be relevant and successful. Change must happen quickly, intentionally, and flawlessly. Yet, change of any kind is difficult. Whether you are talking about changing a culture, a business, a business process, a toolset, a skill, a personal habit, or a mind-set, change is tough. There is no surefire solution or recipe for change.

This book provides excellent thoughts and advice for navigating a course of change in your organization. Change in every organization and every project is different. What may be successful in one organization or project could be disastrous in another. This book provides many creative ideas and suggestions for business change.

Bob and Dave have a unique and entertaining writing style. Not only is their writing enjoyable, but they make you think. They make you question the obvious and the not so obvious. They provide original, concrete, and pragmatic suggestions and advice that you can put to immediate use. At the beginning of each chapter, they provide a real-life story, or interlude, to help make the material relevant. At the end of each chapter, they provide a succinct summary of the key points, titled "If You Remember Nothing Else," and make it actionable with a section titled "What You Can Do Right Now." Whether your job function is in the business or IT, I think you will find thought-provoking value and useful advice in this book that will help you navigate your unique path of, as Bob and Dave call it, *intentional* business change.

Prologue

What the Mess Is and How We Got into It

An analogy isn't the same thing as being the same thing.
—The Economist

There's a famous cartoon in IT circles[1] (figure 1) that shows a succession of swings, as specified in the project request, as explained by the systems analyst, as built by the programmers,

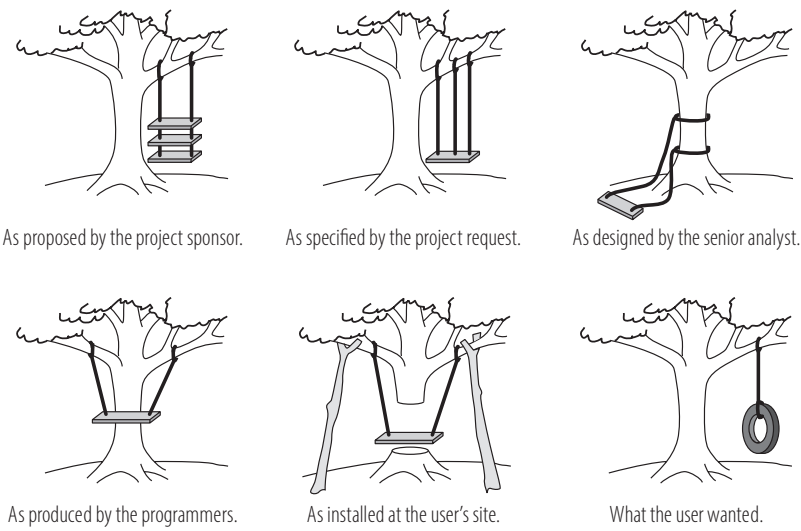

FIGURE 1 An application development metaphor.

and so on. None of them could actually work, and all of them look entirely ridiculous.

Except, that is, for the last panel of the cartoon, which shows what the users really wanted.

It shows a tire, hanging by a rope from a tree limb.

Only that doesn't tell the story as this book would tell it. Our view: what the users wanted wasn't the tire swing itself. The final panel of our cartoon would show something quite different.

It would show children playing on the swing, having fun.

Two Anecdotes

A rather excitable former client once became quite agitated when I (Bob Lewis) asked which of the six dimensions of optimization (explained in chapter 2 if you want to read ahead) were most important for a critical business process we were discussing.

"I don't have to choose!" he shouted (not an exaggeration). "I do Six Sigma! I can improve all of them at the same time, and I don't even need information technology to do it!"

He very well might have been right at that. If your business processes are bad enough you probably can improve them on all fronts at the same time. What you can't do is optimize for all of them at the same time.

In any event, I was struck by his proud assertion that he wouldn't need any information technology, because this has become an important talking point among business improvement methodologists. It's an attractive selling proposition, built around expectations that if you need information technology you're in for IT projects, which are notorious for being expensive at the start, with inevitable cost overruns later on if outright failure doesn't happen first.

Whatever the reason, management consultants have, for the most part, done everything they can to decouple business improvement from information technology delivery.

Meanwhile . . .

Once upon a time a decade or so ago there lived an online retailer. The company was quite successful—successful enough that its leaders were on a constant lookout for what the next level might look like.

Then they heard a presentation from a purveyor of eMerchandising software (call it EMS). The online retailer's decision-makers liked what they saw, signed a contract, and launched the EMS Project.

The company's resident strategic consultant* urged the project sponsor not to do this. It wasn't that EMS was inferior to its competitors or that installing it would be a bad idea.

It was the project's name. The EMS project was all about installing, configuring, and integrating software into the company's existing portfolio.

What the consultant unsuccessfully recommended was to call the effort the "Online Merchandising, Oh, by the Way, Using EMS" project instead.

The project was successful according to the Project Management Institute's definition of success: it was completed on time, with unchanged scope, and within its original budget.

And yet, a month after the project had finished, the consultant heard two of the company's senior web designers arguing about which of their new home page layouts was better.

"Why," asked the consultant, "don't you use EMS to A/B test them, to see which one drives more sales?"

"It does that?" they asked.

---

* Your loyal lead author again.

Why IT Projects Are a Bad Idea

You could stop reading right here and get value from this book, because this is the point of it: there are lots and lots of IT projects going on right now, all over the world.

That's the problem, because with few exceptions they're mistakes. It isn't that there is no such thing as an IT project. It's that there shouldn't be. Because if you're undertaking an IT project—if all you're going to do is install software—what you're going to deliver is shelfware.

Or, almost as bad, you'll deliver something reminiscent of an old Steven Wright joke: "I dreamed someone stole everything I owned and replaced it with an exact duplicate."

Only here what's going on isn't theft. It's far stranger than theft. What happens a lot is that companies buy new and expensive software to replace the old legacy systems they can't do without but that cost a lot and require the talents of programmers who are nearing or beyond retirement age to maintain.

And . . . this is the punchline . . . they do everything they can to make the shiny new software act just like an exact duplicate of the software they're retiring. Why? It's easier, less risky, and nowhere near as disruptive. Except that just about all of the business benefit comes from the disruption.

And so, after decades of this praxis, skeptical business-opinion influencers make a big fuss about how little return businesses get from their investments in information technology—opinions that demonstrate it's possible to be simultaneously accurate and completely wrong. Because often, businesses get too little return from their investments in information technology because they choose to get too little return.

See, the purveyors of these expensive software suites don't charge their sky-high license fees because their sales reps are extraordinarily persuasive, conning IT decision-makers who are too naive to figure out these are overpriced behemoths.

They cost as much as they do because they provide an enormous set of potential capabilities many businesses consciously decide to ignore.

They ignore them because, as you'll read in the pages that follow, the methodologies available to them . . . convincingly presented, so long as you consider the phrase "best practice" to be convincing . . . provide little or no guidance for how to use new information technology capabilities to improve how their employees get things done.

How did we get into this mess? The first two steps on the path to this particular circle of perdition were (1) drawing an analogy and (2) taking it seriously.

The analogy was that because IT organizations deliver technology to the rest of the business, they're "like" any other organization that delivers technology to someone else, which is to say, they're like software businesses.

And so they are, in that both they and software businesses create applications for someone else to use.

Interestingly enough, that's where the analogy ends, or should, not that you'd know this from reading the IT industry press.

It should end right there because software companies—think SAP, Oracle, Microsoft, Salesforce, and their brethren—create software products they sell to thousands and sometimes millions of business customers. When an IT organization writes software, in contrast, that software is designed for one business to use.

There are more than a few differences between having one metaphorical "customer" and having ten thousand real, paying

ones. For example, with one customer you can tailor the software for your customer's exact situation. With ten thousand you can't.

On the other hand, with ten thousand customers you can afford to spend a lot more on your product. Beyond that, you're competing for business with other companies that sell similar products. Which means you can and have to spend a lot more designing and building your software package than any internal IT organization can or should afford to.

One more difference: Internal IT ought to, and usually does, have the best interests of the organization it serves at heart. Software vendors, in contrast, while they generally do want their customers to succeed, care more about making a sale than about what's truly in their customers' best interests.

Nonetheless, the analogizers were (and still are) persuasive. And so we're suffering from what so often happens with analogies: rather than recognizing where the analogy holds and where it breaks down, we've taken every characteristic of software businesses and insisted IT adopt identical practices.

In particular, IT's project methodologies almost universally start with the notion that IT's job is finished when the software "satisfies requirements" and "meets specifications." They finish with the proposition that business cost centers should pay IT for its products and services. Why? Because that's what businesses do: charge and pay for the products and services they provide and consume.

Except that when you run a business the point of the exercise is to sell products and services at a profit. When you run internal IT, the point of the exercise is, or rather should be, to collaborate in and provide support for intentional, beneficial business change.

How We Got into This Mess

Once upon a time, deep in the dim past when computers were first starting to expand beyond their initial accounting toehold in the world of business, programmers didn't understand their job was to deliver a product to their internal customers. For that matter, they didn't understand they weren't supposed to be capable of talking to their nontechnical counterparts elsewhere in the organization.

Quite the opposite. In the early days of IT, business managers were quite comfortable asking programmers if they could get the computer to do something or other to help them run their part of the business better, and programmers were equally comfortable either making it do thus-and-such or explaining that no, computers don't do that, but they could do some other thing that might improve the situation.

After a bunch of conversations like this, the programmer had the computer doing so much that it would have been hard for the business area that used these programs to go back to pencil-and-paper techniques.

Without even necessarily meaning to, the programmer and the business manager had built a big honkin' system that ran a significant chunk of the enterprise. These big honkin' systems vastly improved the business processes they touched, long before Lean, Six Sigma, Theory of Constraints, and Business Process Reengineering became part of our lexicon.

If they (and "they" includes "we") had only known, we and they would have written up "talking with each other a lot" as a methodology, called it "Agile," and made a fortune.

The big honkin' systems (we now call them "legacy systems") built through this series of can-you-get-the-computer-to-do-this conversations did sometimes get a bit messy. That's

because neither the business manager who asked nor the programmer who delivered knew where things were going to end up. So they built a puzzle piece, then a second puzzle piece that fit into the first. The third piece had to fit together with the first two, and every additional piece increased the danger they'd paint themselves into a metaphorical corner.*

And so, a bunch of methodological theorists came along to make sure this never happened again. They managed this by tossing around another analogy that doesn't work very well: developing software, they explained, is exactly like building a skyscraper.

Or it should be. Because you never see skyscrapers with internal design inconsistencies.

This is how the so-called Waterfall methodologies came to be, with their legendarily high rates of failure and their usual mismatch between what the business needs and what projects built around Waterfall methodologies deliver.

Not that it matters from the perspective of this book, but among the many analogy breakdowns these fine theorists might have noticed but didn't is an important difference between computer software and skyscrapers: users of software but not of skyscrapers never run out of new requests for "Can you get the computer to do this too?"

What matters a lot from the perspective of this book is their definition of success, taken whole cloth and without much scrutiny from the Project Management Institute's skyscraper guidelines.

Success, according to Waterfall methodologies, means the project came in under budget, on schedule, and with all the originally planned features intact.

---

* Yes, the metaphor police are going to hunt me down like a dog.

Please understand—your loyal authors have nothing against a project coming in on time, staying within budget, and delivering what it's supposed to deliver.

We just object to calling this success.

Come to think of it, this isn't much good as a definition of skyscraper success either: when someone builds a skyscraper, it's a failure unless lots of people and businesses want to live or work in it.