The Business Manager's Guide to Software Projects

A Framework for Decision-Making, Team Collaboration, and Effectiveness

—

Jonathan Peter Crosby

Apress®

The Business Manager's Guide to Software Projects

A Framework for Decision-Making, Team Collaboration, and Effectiveness

Jonathan Peter Crosby

Apress®

The Business Manager's Guide to Software Projects: A Framework for Decision-Making, Team Collaboration, and Effectiveness

Jonathan Peter Crosby
Baden, Switzerland

ISBN-13 (pbk): 978-1-4842-9230-3 ISBN-13 (electronic): 978-1-4842-9231-0
https://doi.org/10.1007/978-1-4842-9231-0

Copyright © 2023 by Jonathan Peter Crosby

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Editorial Assistant: Gryffin Winkler

Cover image designed by Isaac Soler at eStudioCalamar.

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub.

Printed on acid-free paper

Table of Contents

About the Author

Jonathan Peter Crosby is a software developer, performance engineer, and consultant who has worked in the field for over 20 years.

Having gained his professional experience at a range of companies from start-ups to blue chips, he also founded and cofounded three small tech companies—the first one at the age of 33. Jonathan works at the crossroads of technology and business. The best project outcome, he finds, is achieved through sharing the business and technical knowledge—a reason that he likes to involve all team members in important project decisions. Jonathan believes that effective communication is the cornerstone of every successful software project.

About the Technical Reviewer

Juval Löwy is the founder of IDesign and a master software architect. Over the past 20 years, Juval has led the industry with some of his ideas such as microservices serving as the foundation of software design and development. In his master classes, Juval has mentored thousands of architects across the globe, sharing his insights, techniques, and breakthroughs, and has helped hundreds of companies meet their commitments. Juval participated in the Microsoft internal strategic design reviews and is a frequent speaker at major international software development conferences. He is the author of several bestsellers, and his latest book is *Righting Software* (Addison-Wesley, 2019), which contains his groundbreaking ideas on system and project design.

Juval published numerous articles, regarding almost every aspect of modern software development and architecture. Microsoft recognized Juval as a Software Legend, as one of the world's top experts and industry leaders.

Acknowledgments

First and foremost, I'd like to thank my wife, Mirjam, for all her support over the years. This book would not have been possible without her. I am also very much indebted to everyone who has shared their knowledge with me such as teachers, colleagues, friends, family, authors, and countless others. Many people dedicate so much of their time to help others learn and progress; this is my chance to thank you all.

No one would have understood what I'm trying to say without the fantastic help of my main copy editor, Leila Johnston. Thank you so much for your skill, expertise, and attention to detail. You helped me with every aspect of this book and have been an absolute joy to work with.

Visually, the book came to life thanks to your beautiful illustrations, Danira Spahić—you understood exactly what I was looking for. I often outlined a rough idea in just a few words, and based on this you created illustrations even better than I could have imagined. Since finishing the final image, I've really missed working with you!

As a first-time author, I find it incredible how many people are prepared to spend their spare time helping with such a book project. A great big thank you to my mom, Carolyn, for always having emphasized the importance of languages. Your time and expertise in helping me realize this book are highly appreciated. My sincere gratitude goes out to Ben Smith for sharing his invaluable hands-on project experience. A massive thank you also to my dad, Tony, and my brother, James, as well as to Minh Vuong, Peter Smith, Charles Smith, Philipp Ochsner, Jon P. Smith, and Marc Mettler for proofreading the book. Thanks also to Ymyr Osman and Reto Scheiwiller for your wonderful ideas and inspirations.

Preface

Understanding what technology can and can't do has become a core competency that every part of the business must have.

—Gene Kim, Kevin Behr, and George Spafford
[Kim, Behr & Spafford 2013]

The meeting room had cleared when Daniel leaned forward and asked, "Johnny, what is a system interface exactly?" I was so glad he asked. The question lit a spark and inspired me to write this book. Daniel is a friend and former colleague who's well-educated in finance. He's the author of some of the best business requirements I've come across. We had cooperated on projects before, and I'm certain I had used the term "interface" a few times previously. Had I kept him in the dark all this time by overusing IT jargon? On a broader scale, how can we fill this knowledge gap between IT teams and business specialists? What resources are out there to help business people pick up the essentials? Not many, I discovered.

What, then, would be the best way to explain the basics of software projects? I began taking note of the examples I used for making technical topics clearer. After 8 years of collecting explanations, I've finally found the time to compile these into a book, and I'm delighted to now present my ideas to you. I aim to demystify the underlying concepts of software projects to let you in on the act.

Almost every company today is becoming a software company to some extent, yet software projects still suffer a high failure rate for a multitude of reasons. I'm convinced, though, that a better understanding between IT and business teams will help avoid common pitfalls.

As I've aimed to make this book an easy read, I must apologize now for any oversimplifications. The goal is to get the message across in simple terms rather than to cover every possible permutation.

In answering Daniel's preceding question, I used the metaphor of pipes and cables leading to a house. Instead of electricity or fresh water, a system will typically send data. I'll elaborate on this example later in the book.

Experience has taught me just how effective metaphors can be in illustrating many aspects of software. In discussions between techies and the business side, the right metaphor has the power to portray a complex technical topic in an instant. Examples based on the design and construction of buildings work particularly well. No one would dispute the idea. *If you want to add a new level to the Leaning Tower of Pisa, you'll need to secure the foundation first*. Likewise, before adding new visible features to a piece of software, you may first need to invest in the underlying code structure.

Introduction

What we do not understand we do not possess.

—Goethe

Helping teams achieve higher success rates in software projects is one of the main goals of this book. The key is to establish common ground on software project concepts among all the stakeholders in your project.

When creating software, some poor decisions are made that would never be made when building or renovating a house. Would you, for instance, create a beautiful new bathroom in a house that would be torn down in 18 months? No? I've worked on software projects where people made decisions just like that. If you were working on your own house project, you'd certainly take the time to check every aspect of it. Your business team really needs to do the same in a software project. Too often though, these teams lack the time necessary to collaborate well with the IT teams. Assigning them sufficient time to focus on the project will enable the business teams to think things through more carefully and make a fuller contribution to the project—optimizing the chances of project success. Also, while any serious company will keep its buildings in good condition, many tend to neglect some of their core business systems for years. By bringing the non-techies into the act, it follows that the mixed teams will make better decisions.

Metaphors, illustrations, and genuine examples can help reveal the core concepts. Some good metaphors can be found in techie books, yet I doubt any non-IT people would buy these and skip the technical stuff just to read the metaphor section. I therefore decided to write this guide specifically for the audience that would benefit most.

I'd like to mention a book my new boss at a finance company sent me shortly before I started the new job. The book, on asset management, was incredibly helpful in giving me a grasp of many important topics and common terms. It sped up my learning quite dramatically when I started in the new role. Similarly, this guide is full of practical advice for people with little or no background knowledge in software projects. On the other hand, you may have participated in such projects already, but haven't felt quite at ease with all the technicalities. I've seen business people showing a keen interest when someone makes an effort to explain technical terms in an interesting way.

Having worked in the field for over 20 years, I've drawn on my experience and that of my extended network to present a collection of ideas that will be relevant to your software project, alongside working methods that I know to be effective. So far in my career, I've been lucky to work for some fantastic clients, from start-ups to blue chips. Although most of the software projects were implemented successfully, my focus here is mainly on the problematic ones. The examples here serve well to show you what went wrong and how you can prevent any similar issues in your own projects. As the book aims to cater for a wide audience, I find myself treading a fine line between describing things in plain language and trying not to alienate or bemuse the software community.

Finally, I'd like to highlight the importance of *communication*. Though relevant in all types of projects, this factor is, I believe, especially critical in software projects. How can teams make good decisions if not through effective communication and coherent terminology that's clear to everyone involved?

Who This Book Is For

> *Technology and digitalization is neither a threat nor an end in itself, but will provide us and our clients with added value at various levels. So, let's go for it together!*
>
> —Christoph Hartgens

Written with business people in mind, the book offers you a key to the world of software projects. An essential part of the digital transformation is about involving the business teams much more. The software projects need you because your decisions and involvement will be crucial to the project outcome. The book will not teach you how to program but will give you an overview of the steps and processes involved in creating a piece of software. Concrete, real-life examples will introduce you to the basic concepts, with a focus on your role and your deliverables at the same time. Whether you're a subject matter expert, a manager, or a user representative, you'll find this guide invaluable. Your newly acquired knowledge will help you reach the market faster and meet your customers' needs far more effectively.

While researching for this volume, I was astonished to find that very few books on software address this audience. That's why I aim to give you a complete picture of how the various parts of a project fit together.

Has your boss nominated you to represent your department in a software project? Do you need to review and sign off formal IT project documents? Have you felt overwhelmed in meetings when an IT specialist reels off a stream of IT jargon?

The information here will also enable project sponsors and line managers to gain a better insight into project best practices. The rich set of illustrated themes will help visualize the common steps. Even beginner software developers can enhance their knowledge of the more practical side of projects. A project manager, meanwhile, could present copies of this book as essential reading to the team.

Company executives know they need to understand both the potential and the pitfalls of IT. A company may need to reshape its digital strategy, for example. This digital transformation must be carried by everyone in the company. One of the most important developments is in bringing traditional businesses into the digital age. IT cannot do this alone—the business teams must also be involved. At the same time, business-critical software projects will require some executive decisions. Therefore, specific sections focus on helping executives make decisions that are well-informed.

The technical project members involved in a software project are often outnumbered by the business professionals—and their input is a great asset in any project. To fully utilize this business knowledge, the team must know how to apply it best. Team discussions and decisions will be very effective when all members understand the core concepts of both the business and technology sides. One of the biggest barriers that people face is, in fact, the techie language, which is mostly incomprehensible to anyone without a background in computing. You may feel too embarrassed to comment or ask questions in the face of it, even if you have a valid contribution to make.

Learning the basics of software projects is therefore a bit like learning a new language. The more you understand, the more involved you can be. Similarly, by broadening your software project know-how, you'll be able to participate effectively—who knows, you might even start to enjoy the projects (more)!

Business and IT teams that communicate well together are incredibly powerful—this essential element of good communication is often the missing link in unsuccessful projects. Also, by applying best practices, you'll enable your business to adapt better to changes and keep its competitive edge. These days, we witness how traditional businesses such as book stores, taxi services, and record companies are shaken by global software solutions. Which sector will be next? All businesses really need to be software savvy now to survive.

I sincerely hope this book will help you build a strong foundation for your software projects. Lastly, I welcome your feedback and would highly appreciate your participation in this exciting topic. Please visit the book website at www.SoftwareGuide.blog.

How to Use This Book

In using a conversational tone, I imagine that I'm interacting with you directly. My aim is to make the information as accessible as possible—enabling you to rapidly increase your knowledge and engage effectively in your next project.

Structure of the Book

The book is divided into three main parts. Part 1, "Conceptual Guide," will help you understand the main concepts behind software development. The metaphors offer a high level of abstraction and allow you to understand something new much faster. Not only will you feel more at home in an unfamiliar place, but you'll also gain a more holistic perspective on software projects. I make comparisons between software development and physical construction projects, as everyone can relate to building a house. After identifying the commonalities between renovating and building from scratch, we'll look at the differences between the two. In software as in physical construction, creating something new can vary distinctly from modifying a structure that already exists. Finally, we'll explore some further metaphors as we extend beyond the construction comparison.

Part 2, "Practical Guide," focuses on best practices in the hands-on side of software projects, both large and small. We'll look at the whys and then the hows. This part of the book, which runs through the various

stages of a project, focuses on the topics most relevant to you, the business professional. A brief outline of the technical side of things will give you a bird's-eye view of what goes on behind the scenes at the same time.

Part 3, "Technical Guide," digs a little deeper into some of the common technical topics that all projects need to address. The relevance of this part of the book to you will depend on your role and interest. Again, metaphors and analogies help describe the technical concepts in a clear and interesting way.

I introduce the necessary terminology gradually, giving you the chance to become familiar with the terms. "Appendix B: Glossary" at the end of the book explains the technical terms in plain English. Additionally, the appendixes expand on some of the topics introduced in the main part of the book.

References

The quotations and extracts, all referenced, are based on best practices or on research findings. Also, when you find a topic of particular interest, you can delve into some of the related materials listed in "Appendix C: References and Further Reading." There, you'll find a selection of books, online resources, and videos to choose from.

The references to source materials are presented as: [Horowitz 2014], for example. Additionally, if an author has asked me to include the page numbers, then this format is used: [Brooks 1995 p. 55].

The Value of Metaphors

> *The value of metaphors should not be underestimated. Metaphors have the virtue of an expected behavior that is understood by all. Unnecessary communication and misunderstandings are reduced. Learning and education are quicker. In effect, metaphors are a way of internalizing and abstracting concepts, allowing one's thinking to be on a higher plane and low-level mistakes can be avoided.*
>
> —Fernando J. Corbató

Before digging deeply into any unknown topic, it's important to build a mental framework that helps put things in place. The powerful effect of metaphors and analogies makes them indispensable when explaining software projects. Here, we'll take a brief look at the general topic of metaphors.

As a leadership coach and author, Dr. Peter Fuda states

- "metaphors stimulate creative thinking by inviting the reader to discover complementary and related meanings and applications
- metaphors make complex stuff simple by introducing you to an idea and making it much easier to explore once you're inside it
- metaphors use familiar imagery and hence make a topic easier to recall"

[Fuda 2012]

Now, without further ado, let's dive in and begin with a look at the main metaphor used here—**a software project is like a construction project**.

PART I

Conceptual Guide

CHAPTER 1

Architecture and Construction

In Part 1, "Conceptual Guide," we look at the tasks involved in building software that are similar to those in physical construction. Software architecture is named thus for a reason—you'll see many similarities between designing software and physical buildings. The architect's team will create plans for all the construction work including things like the electrical wiring, the plumbing system, the landscape design, and so on. The same goes for software. In this case, the architect considers data flows, user interaction, sequence diagrams, and many other elements in the process of planning, discussing, and building. You'll find more information on the various roles in software projects in "Appendix A: Collaboration."

What happens if you radically change your mind about the house you're having built during the planning phase? That's doable. Parts of the building will have to be redesigned, and costs will need to be adjusted; but it's certainly achievable. How about if you change your mind when the building is half completed? That's a lot trickier to deal with. At best, the changes can be artfully worked into the ongoing construction; but in the worst case, most of the building might have to be torn down and rebuilt. That's not to say it can't be done—just that it'll be very costly and time-consuming. Although modern software projects cater better for late changes, radical ones will continue to cause higher costs and delays.

© Jonathan Peter Crosby 2023
J. P. Crosby, *The Business Manager's Guide to Software Projects*,
https://doi.org/10.1007/978-1-4842-9231-0_1

> *Treating software construction as similar to building construction suggests that careful preparation is needed and illuminates the difference between large and small projects.*
>
> —[McConnell 2004]

As the home buyer, you may not care much about the exact route of the drainage pipes in your new house, but you'll be very interested in the layout of the kitchen. The construction plans will be adjusted to reflect your choices. Similarly, some areas and details in software design are more relevant to the non-IT professionals than others—you probably won't need to know about all the "plumbing" or the "under the hood" techie stuff.

In construction, the interconnection of the various building elements and materials is fundamental. The materials must be weatherproof and also easily replaceable. The paint, windows, wood, and bricks all have different lifespans, for instance. The windows may need replacing after 25 years, but the bricks will typically last much longer. Software parts have different lifespans too. The support for a software component may end next year and will therefore need to be replaced, for example. In both areas of work, the architect and development team need to make sure the structure will be sufficiently functional to keep all the parts independent and updateable.

Additionally, just as a residential area isn't designed to be converted into a theme park later, software cannot easily be converted into something much bigger either. The architect needs to start from a new plan in both cases.

Inevitably, the fiddly bits end up taking the most time. Any professional floor tiler will confirm how the shaping of the small pieces takes a lot longer than laying out the whole tiles. The trimming work is also a big part of software development.

Despite all the planning and preparation, unexpected factors will almost certainly crop up and create extra work. In a rather extreme case,

the house of a former colleague of mine began to sink while still under construction. The architect arranged for 12 large concrete piles to be driven into the ground around the house to stabilize the foundation. Another kind of complication would be the discovery that the building you were about to renovate was a heritage building. The structure would be subject to certain laws, and you'd probably need to comply with additional rules, adding further costs. You might be inclined to think that nothing comparable could happen in software projects, but you'd be surprised. Many unforeseen factors can suddenly appear and necessitate extra work. The team may have overlooked an important stakeholder or a critical business case, new regulations may have come into force, or a security breach could be discovered, to name just a few.

Note To ensure that the terms used throughout the book are clear, I will use the word *construction* when referring to physical construction and *development* when describing software development.

As in most human activities of any complexity, a software project begins with planning…

CHAPTER 2

Planning and Scope

All projects require some degree of planning, depending on the size and complexity of the project. At its core, a project is about managing an abundance of small decisions and dependencies. This requires high-quality communication on all levels. And just as a physical model can help us visualize a new building, creating a working prototype is also useful in software development. This prototype gives users and stakeholders a good idea of how the product will look. Modern software methodologies usually create screen mockups rather than prototypes—these mockups are easier to produce and give a good visual impression.

The planners, at the same time, will need to conduct a survey before embarking on a project plan. Just as a feasibility analysis should first be carried out to check the viability of digging a tunnel, the same needs to be done for a large software project.

> *Feasibility studies are preliminary studies undertaken in the very early stage of a project. They tend to be carried out when a project is large or complex, or where there is some doubt or controversy regarding the proposed development.*
>
> —[Feasibility Studies 2017]

© Jonathan Peter Crosby 2023
J. P. Crosby, *The Business Manager's Guide to Software Projects*,
https://doi.org/10.1007/978-1-4842-9231-0_2

The Right Dimensions

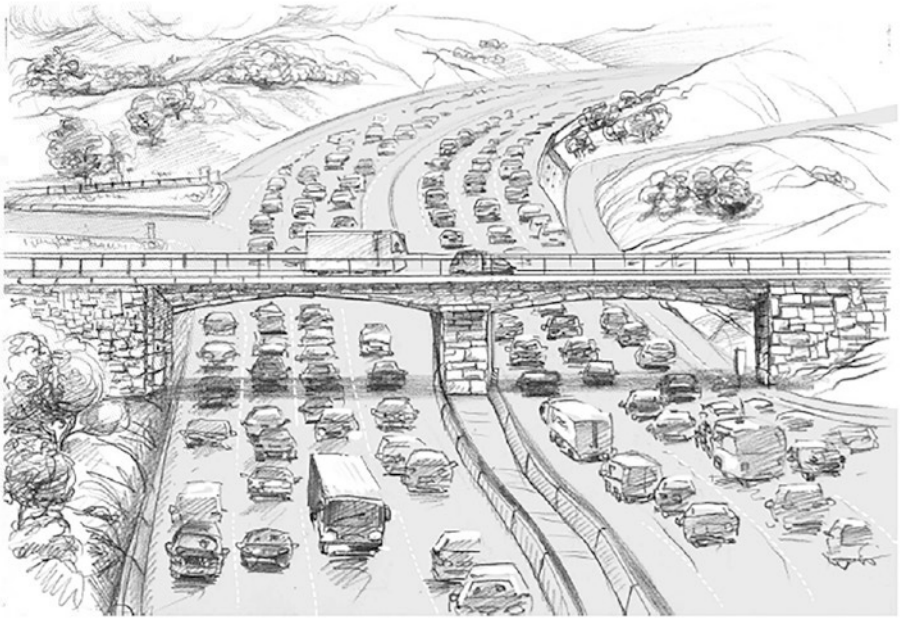

Figure 2-1. *How many lanes should you plan for?*

When building a new highway (Figure 2-1), one of the things the planners will need to decide on is the width of the bridges that will cross it. Let's say that the highway will initially comprise three-lane roads in both directions. Should the planners *future-proof* all the bridges to leave scope for adding another three lanes later? How would these wider bridges affect costs?

If the roads can currently cope with around 2,000 vehicles per hour but the number increases to 10,000 after a few years, then further construction may be required. This situation is comparable to the number of concurrent users of a software application. Software for creating team reports would naturally require a leaner technical setup than an online ticket system where 200,000 people may be trying to buy concert tickets at the same time, for instance.

Although future-proofing might seem a good idea, it can sometimes backfire. Here's one example: Some 30 years ago, a highway tunnel was built under the main railway station in Zurich, Switzerland, in anticipation of a future road. Today, there is still no road connected to this segment, and there probably never will be. Imagine the complexity of building a tunnel underneath a large station, as well as the high cost incurred. Trying to foresee the distant future is equally hard in software projects. Making absolutely everything configurable to allow for future changes is not always the best approach—it will make the software a lot more expensive to produce and maintain.

When building a new house, the homeowner may request the builder to place empty tubes in the walls that will be useful later if additional cables need to be laid. If not required, these tubes can usually just remain in place with no extra work or cost involved. The opposite is true for software, however. Because every new feature needs to be tested and maintained, features that aren't needed now should not be built.

Hogwarts Castle—Keeping Within a Budget

Some people may dream of living in a castle or a palace (Figure 2-2), but common sense usually dictates that you build something more modest with an affordable budget. If you plan to build your own house, you'll naturally think very carefully about what you need and what you can afford. The initial outline will include the size and type of house you want and the number of bedrooms and bathrooms you require. This process will involve some hard decisions. Can the kids share a bedroom? Do I really need an office? Can I afford a pool? How much can we spend on the kitchen? You'll also consider hundreds of minor details. Do I need a power socket here or a light switch there? Your house project will include an abundance of decisions both large and small. Some decisions will be hard to make, and not every wish can be fulfilled.