



Johann Rost • Robert L. Glass

# THE **DARK SIDE** OF SOFTWARE ENGINEERING

Evil on Computing Projects

 **WILEY**

 **IEEE**

 IEEE  
computer  
society



*THE DARK SIDE OF  
SOFTWARE  
ENGINEERING*



## Press Operating Committee

### Chair

Linda Shafer  
*former Director, Software Quality Institute  
The University of Texas at Austin*

### Editor-in-Chief

Alan Clements  
*Professor  
University of Teesside*

### Board Members

Mark J. Christensen, *Independent Consultant*  
James W. Cortada, *IBM Institute for Business Value*  
Richard E. (Dick) Fairley, *Founder and Principal Associate, Software Engineering  
Management Associates (SEMA)*  
Phillip Laplante, *Professor of Software Engineering, Penn State University*  
Evan Butterfield, *Director of Products and Services*  
Kate Guillemette, *Product Development Editor, CS Press*

### IEEE Computer Society Publications

The world-renowned IEEE Computer Society publishes, promotes, and distributes a wide variety of authoritative computer science and engineering texts. These books are available from most retail outlets. Visit the CS Store at <http://computer.org/store> for a list of products.

### IEEE Computer Society / Wiley Partnership

The IEEE Computer Society and Wiley partnership allows the CS Press authored book program to produce a number of exciting new titles in areas of computer science, computing and networking with a special focus on software engineering. IEEE Computer Society members continue to receive a 15% discount on these titles when purchased through Wiley or at [wiley.com/ieeecs](http://wiley.com/ieeecs)

To submit questions about the program or send proposals please e-mail [kguillemette@computer.org](mailto:kguillemette@computer.org) or write to Books, IEEE Computer Society, 10662 Los Vaqueros Circle, Los Alamitos, CA 90720-1314. Telephone +1-714-816-2169.

**Additional information regarding the Computer Society authored book program can also be accessed from our web site at <http://computer.org/cspress>.**

---

# *THE DARK SIDE OF SOFTWARE ENGINEERING*

Evil on Computing Projects

**JOHANN ROST and ROBERT L. GLASS**

IEEE  
 **computer  
society**

 **WILEY**

A JOHN WILEY & SONS, INC., PUBLICATION

Copyright © 2011 by IEEE Computer Society. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.

Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400, fax 978-646-8600, or on the web at [www.copyright.com](http://www.copyright.com). Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

**Limit of Liability/Disclaimer of Warranty:** While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department within the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

***Library of Congress Cataloging-in-Publication Data is available.***

ISBN 978-0470-59717-0

Printed in the Singapore

ePDF: 978-0-470-90994-2

oBook: 978-0-470-90995-9

ePub: 978-0-470-92287-2

10 9 8 7 6 5 4 3 2 1

---

# CONTENTS

---

## FOREWORD ix

*Linda Rising*

---

## INTRODUCTION 1

---

- I.1 What's the Dark Side? 1
  - I.1.1 Why the Dark Side? 2
  - I.1.2 Who Cares About the Dark Side? 3
  - I.1.3 How Dark is the Dark Side? 5
  - I.1.4 What Else is on the Dark Side? 7
  - I.1.5 Ethics and the Dark Side 8
  - I.1.6 Personal Anecdotes About the Dark Side 11
- Reference 14

---

## PART 1

---

### DARK SIDE ISSUES 15

---

## CHAPTER 1 SUBVERSION 17

---

- 1.1 Introductory Case Studies and Anecdotes 17
  - 1.1.1 A Faculty Feedback System 18
  - 1.1.2 An Unusual Cooperative Effort 21
  - 1.1.3 Lack of Cooperation due to Self Interest 22
  - 1.1.4 An Evil Teammate 22
  - 1.1.5 Thwarting the Evil Union 24
- 1.2 The Survey: Impact of Subversive Stakeholders On Software Projects 24
  - 1.2.1 Introduction 25
  - 1.2.2 The Survey 26
  - 1.2.3 The Survey Findings 27
  - 1.2.4 Conclusions 34
  - 1.2.5 Impact on Practice 35
  - 1.2.6 Impact on Research 35
  - 1.2.7 Limitations 35
  - 1.2.8 Challenges 36
  - 1.2.9 Acknowledgments 37
- 1.3 Selected Responses 37
  - 1.3.1 Sample Answers to the Question: "What Were the Motivations and Goals of the Subversive Stakeholders?" 37
  - 1.3.2 Sample Answers to the Question "How Were the Subversive Attacks Discovered?" 45
  - 1.3.3 Sample Answers to the Question "How Can Projects be Defended Against Subversive Stakeholders?" 49

1.4	A Follow-Up to the Survey: Some Hypotheses and Related Survey Findings	56
	References	80

---

**CHAPTER 2 LYING 81**

2.1	Introductory Case Studies and Anecdotes	81
2.2	Incidents of Lying: The Survey	86
2.2.1	The Survey Results	87
2.2.2	General Scope	87
2.2.3	An Overview of the Problem	88
2.2.4	Clarification of Terms	89
2.2.5	Discussion	93
2.2.6	Conclusions	93
2.2.7	Limitations	94
2.3	Qualitative Survey Responses on Lying	95
2.4	What Can Be Done About Lying?	96
2.5	The Questionnaire Used in the Survey	107
	References	112

---

**CHAPTER 3 HACKING 113**

3.1	Case Studies of Attacks and Biographies of Hackers	113
3.2	Cyber Terrorism and Government-Sponsored Hacking	118
3.3	The Hacker Subculture	121
3.3.1	Why They Are Called “Hackers”	121
3.3.2	Motivation of Hackers	121
3.3.3	Hacker Slang	122
3.3.4	Hacker Ethics	123
3.3.5	Public Opinion about Hackers	130
3.4	How a Hacker Is Identified	132
3.5	Time Line of a Typical Malware Attack	135
3.6	Hacker Economy: How Does a Hacker Make Money?	136
3.7	Social Engineering	142
3.7.1	Social Engineering Examples and Case Studies	143
3.7.2	Tactics of Social Engineering	151
3.8	A Lingering Question	153
3.9	Late-Breaking News	154

---

**CHAPTER 4 THEFT OF INFORMATION 157**

4.1	Introduction	157
4.2	Case Studies	158
4.2.1	Data Theft	158
4.2.2	Source Code Theft	161
4.3	How Do the Victims Find Out That Their Secrets Are Stolen?	164
4.4	Intellectual Property Protection	166
4.4.1	Trade Secret Protection	167
4.4.2	Copyright Protection	169
4.4.3	Patent Protection	169
4.4.4	Steganography	170
4.5	Open Versus Closed Source	170

**CHAPTER 5 ESPIONAGE 175**

---

- 5.1 Introduction 175
- 5.2 What Is Espionage? 176
- 5.3 Case Studies 177
  - 5.3.1 Sweden Versus Russia 178
  - 5.3.2 Shekhar Verma 178
  - 5.3.3 Lineage III 179
  - 5.3.4 GM versus VW: Jose Ignacio Lopez 179
  - 5.3.5 British Midland Tools 179
  - 5.3.6 Solid Oak Software 180
  - 5.3.7 Proctor & Gamble versus Unilever 181
  - 5.3.8 News Corp Versus Vivendi 181
  - 5.3.9 Spying: Was A TI Chip Really Stolen by a French Spy? 181
  - 5.3.10 Conficker 183
- 5.4 Cyber Warfare 185
- Reference 187

**CHAPTER 6 DISGRUNTLED EMPLOYEES AND SABOTAGE 189**

---

- 6.1 Introduction and Background 189
- 6.2 Disgruntled Employee Data Issues 192
  - 6.2.1 Data Tampering 192
  - 6.2.2 Data Destruction 194
  - 6.2.3 Data Made Public 196
  - 6.2.4 Theft Via Data 199
- 6.3 Disgruntled Employee Software Issues 199
  - 6.3.1 Software Destruction 199
- 6.4 Disgruntled Employee System Issues 200
- 6.5 What to Do About Disgruntled Employee Acts 203
- 6.6 Sabotage 206
- References 212

**CHAPTER 7 WHISTLE-BLOWING 213**

---

- 7.1 A Hypothetical Scenario 215
- 7.2 Whistle-Blowing and Software Engineering 217
- 7.3 More Case Studies and Anecdotes 220
  - 7.3.1 Jeffrey Wigand and Brown and Williamson Tobacco 220
  - 7.3.2 A Longitudinal Study of Whistle-Blowing 221
  - 7.3.3 An Even More Pessimistic View 222
  - 7.3.4 Academic Whistle-Blowing 223
  - 7.3.5 The Sum Total of Whistle-Blowing 224
- References 225

**APPENDIX TO CHAPTER 7 PRACTICAL IMPLICATIONS OF THE RESEARCH INTO WHISTLE-BLOWING 227**

---

- References 240

---

**PART 2**

---

**VIEWPOINTS ON DARK SIDE ISSUES 243**

---

Introduction 243

---

**CHAPTER 8 OPINIONS, PREDICTIONS, AND BELIEFS 245**

---

8.1 Automated Crime 246

*Donn B. Parker*

Information Sources 257

8.2 Let's Play Make Believe 258

*Karl E. Wieggers*

Reference 260

8.3 Dark, Light, or Just Another Shade of Grey? 261

*Les Hatton*

8.4 Rational Software Developers as Pathological Code Hackers 264

*Norman Fenton*

---

**CHAPTER 9 PERSONAL ANECDOTES 269**

---

9.1 An Officer and a Gentleman Confronts the Dark Side 270

*Grady Booch*

9.2 Less Carrot and More Stick 273

*June Verner*

References 275

9.3 "Them and Us": Dispatches from the Virtual Software Team Trenches 276

*Valentine Casey*

9.4 What is it to Lie on a Software Project? 281

*Robert N. Britcher*

9.5 "Merciless Control Instrument" and the Mysterious Missing Fax 284

*A. H. (anonymous)*

9.6 Forest of Arden 289

*David Alan Grier*

9.7 Hard-Headed Hardware Hit Man 292

*Will Tracz*

9.8 A Lighthearted Anecdote 294

*Eugene Farmer*

---

**CONCLUSIONS 299**

---

---

**INDEX 303**

---

---

# FOREWORD

Dr. Linda Rising

Robert Glass has always been one who “boldly goes” where the more cautious fear to tread. I have been a fan of his writing for, well, let’s just say, a long time. I remember when he started telling the truth as he saw it about software development and was forced to change the names of the companies and products that he was discussing—he even changed his own name to conceal authorship of published accounts. I remember teaching a course on structured design (using the green book by Yourdon and Constantine—that’s how long ago that was!) and if I finished a class early, I would say to my students, “You can go now or I can read another story by Robert Glass.” No one ever left before the story was finished. “Cornbelt Shakedown” (from Glass and DeNim [1980]) was a favorite. Many of these stories are the kind of humor that leads you to wonder, “Why am I laughing? To keep from crying?”

Later, as I was working in the industry, I led a study group on *Software Runaways* (Glass 1997) and experienced the serious side of Robert Glass. Very little of the wry and witty here, but, instead, a lot of lessons for serious consideration.

Robert Glass, joined in this book with Johann Rost, is still at it. He continues to be (I can’t resist) fearless! (The reference is to my own book, Manns and Rising [2005]). I don’t know Johann except through his work on this book, which is excellent, and from what I’ve been told—that he’s a German former IT consultant now living in beautiful Romania, the land of Transylvania, Dracula, and Ceaușescu ... it’s no wonder the book has a “dark side” theme! This book is also full of stories about real projects at real companies. Names are named. The result is a compelling look at the dark side of computer programming. We are all hardwired to learn from stories, especially when we can identify with the protagonists.

Hacking, espionage, sabotage, theft, whistle-blowing, subversion, disgruntled employees who want to get even—and, of course, the dance of deception. We’ve all seen it—where *we know* and *they know*, in fact, *everyone knows*—but we all smile and keep dancing as long as we can. The authors cut in on this charade and force us to wake up and take stock.

Robert and Johann also include the results of their serious research. They have certainly done their homework. There’s an abundance of citations to back up their observations. The survey data on sabotage is fascinating!

This reporting is way out of the box; in fact, these authors are standing on the box and they share with us a good look at the terrain—something most of us just don’t take the time to do; we prefer to rush ahead and ignore the lessons of the past.

So, take a moment. We need a breather now and then. We need to step back and retrospect on the history of our industry and think about a better way of working

within it. Robert Glass and Johann Rost are offering us a chance to do just that. Stop. Listen. Think. Is this the road that will serve us best for the next part of our journey?

## **REFERENCES**

---

Glass, Robert and DeNim, Sue. "The Second Coming: More Computing Projects Which Failed," *Computing Trends*, 1980.

Glass, Robert. *Software Runaways: Monumental Software Disasters*. Prentice-Hall, 1997.

Manns, Mary Lynn and Rising, Linda. *Fearless Change*. Addison Wesley, 2005.

---

# INTRODUCTION

---

## I.1 WHAT'S THE DARK SIDE?

---

The dictionary doesn't give a definition for "dark side." Not even my heavyweight dictionary that I can barely lift. Oh, it defines words such as "dark" ("secret, mysterious, evil," among other things), "darken" ("perplex, make foul, sully, cast a gloom upon") and "darksome" ("dark, dismal"). So you get the idea—things that are on the dark side tend to be evil, gloomy, dismal.

That's not a surprise to most of our readers, we suspect. The "dark side" has a sort of intuitive meaning that we all grasp and is (pretty much) in synch with those related dictionary definitions. Things that are on the dark side of the computing profession would be things that we wouldn't necessarily want to be a part of or approve of.

I, Robert, remember an incident from my days of child-raising, when one of my sons played on a little league baseball team. There was a pitcher on that team whose father, like me, attended nearly all of the games. When his son was pitching, the father would shout to his son, from time to time, "Throw the dark one." I never knew exactly what he meant by that cry. But I always assumed that it wasn't so much about a particular pitch his son could throw but about intimidating the opposing batter, who might become convinced that the pitch to come was somehow evil and be less likely to make contact with it because of that.

In any case, even on the baseball diamond, the words "dark" and therefore "dark side" have an intuitively universal meaning.

It's interesting that, if you know the software literature—be it the popular computing press, the academic journals, or even the general popular press—you would be aware that it doesn't say very much about dark side issues. Oh, it says a lot about project success and project failure but that's a different kettle of fish. Projects that fail may be in a sense "dark" but not in the sense of "evil." We tend to assume, without ever saying so, that even projects that fail, do so largely because of some kind of ineptitude, not because of some kind of evil.

Let me be perfectly clear about what we are doing here. This is NOT a book about software project failure, or about prescriptive thinking about how to build software better. This is a book about the EVIL THINGS that happen on computing and software projects—what the kinds of evil are, how they manifest themselves, and what we good guys can do about them. I emphasize this point because a lot of folks we've asked to review the book's material keep thinking that this is "Yet

Another Book About Project Failure” (YABAPF) or “Yet Another Book About Doing Software Engineering Right” (YABADSER)!

Where might we find discussions of dark side matters in the traditional software engineering literature? Look at the topics that literature on computing and software tend to be divided into. They are usually organized into these topics:

- Problem-solving
- Computer hardware
- Systems/software
- Data/information
- Application problem domains
- Systems/software management
- Organizations
- Society
- Disciplinary issues

This list is derived from the computing research topics explored in the series of papers culminating in Glass, Ramesh, and Vessey (2004).

Where in that list of topics would you look to find “dark side” topics? Perhaps in “systems/software management.” Perhaps in “disciplinary issues.” It doesn’t fit comfortably into either of those topics, but it could be forced to fit—inconveniently—into them. But the fact of the matter is, any taxonomy of computing topics you choose is unlikely to provide a convenient home for this issue of the dark side. It is, in other words, a topic that people writing about computing have not only avoided over time; they have avoided it because it doesn’t fit nicely into any list of topics that describe the field.

And that brings us to the topic of the next section.

### 1.1.1 Why the Dark Side?

Both authors of this book have been intrigued by the lack of discussion of dark side issues in computing literature. We were both aware, from personal experience, that dark side things happened in the field. But hardly anyone seemed to be talking about them. Perhaps more importantly, hardly anyone was researching them. For example, how often did dark side matters affect computing and software projects?

I, Johann, had initially thought about exploring this issue. I knew from personal experience the effect of dark side behavior: For example, subversion on software projects, while it does not occur often, has serious repercussions when it does. Because of that, and because of the lack of any appearance whatsoever of “subversion” in computing literature, I conducted a study to determine its prevalence, its effects, and ways of overcoming it. That survey is presented as a chapter later in this book. It is a pioneering study in the software field; to this date, no one else has explored this topic.

My co-author, Robert, came at the subject from a different direction. He was surprised while presenting a topic at a software seminar; the seminar attendees hijacked the session and diverted it to talking about lying as a problem in the software project world. The attendees were vehement—lying was a big-time problem in the projects on which they had been involved. Because of that, and because—once again—of the lack of any significant appearance of the topic of “lying” in the computing literature—he began to explore that topic in more depth.

It was about then that we met one another. (It is interesting to note, in this day of electronic communication, that we have only met on the Web, never in person!) I was having trouble finding a leading journal willing to publish my subversion paper, that is, the one that resulted from his survey. I asked Robert for help, and—to make a long story shorter—the result became a co-authored paper that eventually was published in a leading journal.

Intrigued by the subversion study, Robert suggested that we conduct a similar study about lying. As we have said, neither topic was discussed much in any of the literatures surrounding the field. So the two of us, together with another contributor named Matthias Matook, performed a study in the form of a survey about the prevalence of lying, its effects, and ways of overcoming it. Eventually, to make this long story also shorter, that too was published. Variations and enhancements of the two published papers are presented later in this book.

By then, we had become thoroughly intrigued by these topics, and we began to see them as part of a broader issue: “dark side” issues on computing projects. We expanded the topic into more and more sub-topics, eventually identifying seven dark side matters that affected these projects: subversion, lying, hacking, theft of information, espionage, disgruntled employees and sabotage, and whistle-blowing. There is a chapter of this book devoted to each of those topics.

We considered doing thorough research into the latter five topics, but decided that there was sufficient material in the literature of those more-often covered topics; so we relied on already published case studies, not the survey research that we conducted about subversion and lying, to cover them. (To be honest, that research was extremely laborious and time-consuming, and we were reluctant to engage in it beyond what we had already done!)

And then there is another final fact that brought the interest in dark side matters to a head: Robert has published a number of books and articles on the subject of failed computing projects. (As we said earlier, there is not a direct link between failure and dark side matters, but the two are similar enough to draw the same kind of interest.) He had been intrigued by failure and became equally intrigued by dark side matters!

### **1.1.2 Who Cares About the Dark Side?**

The short answer to this question, of course, is that we hope YOU do! We chose to write about the dark side because we were interested in the subject and because we felt we had some contributions to make on the subject. Our fervent hope is that you, our intended reader, will also be interested in what we have to say.

But that raises the question, “Exactly who is our intended reader”? Usually, both of us have a preferred reading audience, namely experienced software practitioners who have an interest in broadening their knowledge on the topic. And—we can’t help it—that’s who we’ve been thinking about as we did the research and the initial writing of this book.

But it would be disadvantageous and perhaps even disingenuous of us to leave it at that. There aren’t that many experienced software practitioners in the broader world, and if we restrict our readership to those folks we won’t sell very many copies of this book! So, as we developed our material, we increasingly began to think about others who might like to know about dark side matters in computing and software engineering.

For example:

- **Software managers.** When we started thinking about broadening our readership, we began by expanding the material to appeal to a management-focused audience. Certainly, if the problems of dark side matters in software engineering are ever to be addressed (and perhaps even solved!), managers will have to be involved. We have many reasons to wish that software managers become interested in reading this book. And the same goes for managers of those managers. And so on, on up the hierarchy!
- **Academics.** We believe that the same spirit of intellectual inquiry that prompted us to look into this topic in the first place will also engage academics. And we believe that, given the absence of this topic from the textbooks on computing and software subjects, there are some unique pieces of academic insight to be had in our book.
- **Researchers.** To be honest, we have some self-interest here. We discuss the absence of relevant research findings on matters dark side. We hope that this book will stimulate other computing researchers into delving more into this topic. We believe the field will be the richer for it.
- **Novice software practitioners.** The people most likely to be stunned by dark side matters are the field’s “greenies,” those who have no reason to believe—going in—that dealing with dark side matters is going to become part of their job description. So welcome, novices, to the word involving more evil than you might ever have considered being a part of!
- **Software engineering students.** If those greenies we discuss above need to be warned about dark side matters, so do students, who typically are greener than green. Both of us were students once, both of us have taught tons of students, and both of us realize that “dealing with the dark side” doesn’t occur anywhere in an academic curriculum. We don’t intend to scare you off, student—we both believe that software engineering is a career with its own many faceted rewards. But beware: You will run into evil, and evil folk, even in the otherwise wonderful world of computing and software.
- **The general public.** Now this one is tricky. When you’re a professional in some subject matter, there’s a tendency to write for readers who understand your lingo, and in the process of doing that you make your material inacces-

sible to a broader, non-computing, professional audience. And to overcome this limitation, you need to think carefully every time you put finger to keyboard. It's not at all a matter of "dumbing down" your material (that's a term I've always found to be particularly offensive); it's a matter of writing in such a way that you can be understood. And, to be honest, you the reader get to cast the final vote on how successful this particular quest has been. I believe that the general public will find our thoughts and research about the dark side in computing and software engineering interesting, but I don't know whether we have succeeded in working that particular problem successfully. We'd be interested in hearing from you on this: [rlglass@acm.org](mailto:rlglass@acm.org).

So there you have it. We've defined "dark side," we've explained why we chose to write about it, and we've tried to introduce not only ourselves to you, our intended audience, but you to us. It's time to get specific about what all this dark side stuff is really about. For example, how often does it really happen?

### 1.1.3 How Dark is the Dark Side?

It would be nice if there were a clear-cut, straightforward answer to the question we just asked as we concluded the previous section of our book—how often do dark side matters arise? But the fact of the matter is this: The truthful answer is "it depends."

"It depends" is not a very satisfactory answer, especially to academics. For decades now the software engineering field has been hoping for a universal solution to the problems of building software. Each new methodology, invented by an academic, a guru, or a practitioner, is touted as the new be-all end-all for software. And, as we slowly begin to realize through the experience of using these methodologies, each of them has its own "sweet spot" of applicability and its own areas where applying it is an exercise in frustration. Structured programming was the "solution" for all applications in the 1980s; object-orientation in the 1990s; and Agile approaches more recently. There are those who still believe in the claims of universality for methodologies; most of us by now can see that each approach is wonderful in its proper context, but not so wonderful in others. We have arrived at the point where one software engineering expert says "anyone who believes in one-size-fits-all belongs in a pantyhose commercial." In other words, "it depends" is becoming the watch-phrase for such methodologies.

In order to talk about how often the dark side issues arise, we need to break the dark side topic down into its constituent elements. We have some fairly crisp and clear answers for each of the dark side topics that emerge as chapters later in this book—but as we will see in what follows, those answers don't spread well over the "dark side topic" as a whole.

So let's look at each dark side issue one at a time.

- **Subversion.** Here, our survey produced some nicely definitive numbers. Slightly over 50% of our survey responders had seen episodes of subversion, whereas 35% had not. Asked to estimate how often such subversion occurs,

the predominant answer was “on 20% of projects.” In other words, subversion is an occasional, not a frequent, problem on software projects.

- **Lying.** Again, we have some survey results that allow us to speak with some confidence on this matter. The results showed that 86% of survey responders said they had seen lying on software projects, on perhaps 50% of such projects. The majority of lying is about either cost/schedule estimation, status reporting, or is for political maneuvering (these causes of lying were nearly equivalent in their frequency; nearly all other causes lagged those numbers considerably). Based on these numbers, we feel we can say that lying is a common problem on software projects.
- **Hacking.** Here we enter into the dark side issues where we do not have any data on the frequency of occurrence. If you believe this figure can be judged by incidents reported in the popular press, then hacking is a very common problem. But if you ask questions of this kind to experts who specialize in studying hackers and hacking, you get a fairly strong “I have no idea.” Such experts go on to say they have no idea, either, about what percentage of computer systems are hacked and what percentage of hacks go undetected, except for educated guesses such as “less than 50% of hacks go undetected (which tends to be immediately followed by its own kind of “it depends”—it depends on what kind of hack we’re talking about).
- **Theft of information.** Like hacking, information theft is discussed somewhat often in the popular press, but we are not aware of any data on its frequency. There is a suspicion that most corporate employees who leave an enterprise, either under duress or otherwise, may take information (data or software code) with them, but again there is little data to support this belief. But see the discussion below about disgruntled employees and the frequency with which they take things. In any case, we suspect that there is a problem with theft of information, but it is not a very common one.
- **Espionage.** Stories on espionage in the computing field tend to splash in big headlines in the popular press. But it is important to remember that the press goes in for “exception reporting”—if something is common, it is covered with much less emphasis than if it seldom happens. That’s why it is important to avoid an attempt to translate splashy headlines into frequency information. Here again, we have no data on the frequency of occurrence of espionage on computing projects, but we suspect it is uncommon.
- **Disgruntled employees and sabotage.** Although we have no survey data to rely on in this matter, the popular press has done a nice job of studying the frequency of this particular problem. For example, 60–70% of data theft is conducted by disgruntled employees, according to a recent study, and to make matters worse responders in that study believed that 82% of companies who had had such data stolen would not even realize it. Another study said that perhaps only 1 in 400 such data thefts get reported. Based on that data, we feel safe in saying that disgruntled employees cause mischief quite frequently. Sabotage, sometimes engaged in by disgruntled employees along with other dark side acts, is by contrast infrequent.

- **Whistle-blowing.** Whistle-blowing is an interesting topic in the context of this book. For one thing, it is not at all clear that whistle-blowing is a dark side activity. But it is certainly a reaction to a dark side activity, and that is why we include it. For another thing, there has been little research into whistle-blowing in the broader literature and none whatsoever in the software engineering field. We are happy to report that we include one of the few research surveys of whistle-blowing in general later in this book, but we have to admit that it doesn't help us in judging how often whistle-blowing occurs on software projects. For reasons that we will explain in the chapter on whistle-blowing, we suspect that it seldom if ever occurs on software projects.

There you have it. How often do dark side issues arise on software projects? Anywhere from “seldom if ever” to “quite frequently.” If ever there was a case where “it depends” was the correct answer to the question, this is it!

It is interesting to compare this discussion of dark side matters with a comparable discussion of software project failures. Dark side discussions tend to occur, but only in a particular context (as we have seen, based on our chapter topics above). But with software project failure, it is common to see cries of “software crisis” as whoever is writing about the matter bemoans that software is always “behind schedule, over budget, and unreliable.” The general belief is that software is a field with a big-time problem) at least based on such discussions of project failure). Robert believes that cries of “crisis” are bogus, and that the software field, which is the basis for the obvious success of the “computing age,” is a field with far more spectacular successes than spectacular failures.

In any case, whereas software project failure gets enormous attention from the press, dark side matters slide under the press radar for the most part. What that means in practice is that there are few biases to overcome among you readers regarding how often dark side matters arise.

#### I.1.4 What Else is on the Dark Side?

When we first conceived of this section for our introductory chapter, we envisioned a small section with a discussion of those few other books and articles that pertained to dark side issues. Big hah!

If you Google “dark side,” you are returned 620,000,000 results. With a number that big, we quickly gave up on even trying to categorize the uses of the term “dark side.”

Note that when we first introduced the topic of the dark side, we noted that the dictionary didn't define the term *per se*. But we guessed that most people already had an idea of what it meant. Little did we know! You don't get 620,000,000 Google results for a term that people don't understand.

If you look up “dark side” on Amazon, you get another big number. Not as big as the one you get from Google. Still, 94,500 books with dark side or something related to it in the title?! (And that's not even counting ours, which Google doesn't know about as of this writing). Books on the dark side deal with topics ranging from

religion to psychology to politics to dating to leadership (this one, interestingly, links the subject of failure to matters of the dark side, an issue we presented rather tenuously above!). Based on books people have written, you'd guess there is a dark side to nearly any subject you can think of—and someone has likely written a book about it!

Closer to home, in the software engineering and computing literature, the topic comes up far less often, as we have already mentioned. It does arise, of course, disguised under different terms. For example, consider the subject of “ethics.”

### 1.1.5 Ethics and the Dark Side

Ethics is a topic that is explicitly addressed by almost all professional societies. For example, the IEEE, the ACM, and the German computing society all have codes of ethics, that is, relatively brief statements of how their members should behave. These codes tend to focus more on professionalism than outright misbehavior and overlap to a large extent. Here is an overview of those codes

- **Contribute to society and human well-being.** Avoid injuring others, their property, reputation, or employment by false or malicious action. Make decisions consistent with the safety, health and welfare of the public. Disclose factors that might endanger the public or the environment.
- **Be honest and trustworthy.** Give unbiased estimations, reject bribery. Undertake technological tasks for others only if qualified by training or experience. Seek out, accept, and offer honest criticism of technical work and acknowledge and correct errors. Honor confidentiality. Honor contracts, agreements, and assigned responsibilities.
- **Acquire and maintain professional competence.** This includes technological skills, legal skills, and communication skills.
- **Improve the IT understanding of others.** Train students to assist colleagues and coworkers in their professional development. Promote and strive for excellence. Improve public understanding of computing and its consequences. Accept and provide appropriate professional review.
- **Honor property rights including copyrights and patents, and credit properly the contributions of others.**
- **Be fair and take action not to discriminate.** Treat fairly all persons regardless of such factors as race, religion, gender, disability, age, or national origin.
- **Respect the privacy of others.** This includes the refusal to support the implementation of control and surveillance technology without informing the affected persons.
- **Access computing and communication resources only when authorized to do so.**

It is interesting to compare these codes with our dark side topics. There is barely any intersection between these codes and subversion and espionage, for example.

There is more of a link between lying, hacking, theft of information, and disgruntled employees and sabotage. For example, the material on injuring property, public welfare, honoring property rights, respecting privacy, and access authorization has a more or less direct relationship to our topics. Whistle-blowing, interestingly enough, has almost no linkage to these codes. Once again, although we see a deep societal concern for ethical behavior, we see an odd sort of mismatch between our topics—behaviors seen on actual computing projects—and the professional and philosophical content of these codes. Clearly, not only has research tended to ignore these topics, but so have the more ethical foci of our field.

Curiously, twice in the six months preceding the writing of this material, the societal journal *IEEE Computer* has done something on the subject of software engineering ethics. In the first such article (“Professional and Ethical Dilemmas in Software Engineering,” by Brian Berenbach and Manfred Broy, published in the January 2009 issue) the key word from the title is “dilemmas.” The article describes nine specific ethical and professional dilemmas for software engineers:

1. Mission impossible: accepting a schedule that is obviously impossible
2. Mea culpa: delivering a product that lacks key functionality
3. Rush job: being more concerned about product delivery than product quality
4. Not my Problem: showing no inclination to improve productivity or quality
5. Red lies: making statements about a project/product known to be untrue
6. Fictionware versus Vaporware: “fictionware” is signing up for features known to be infeasible; vaporware is announcing a product that does not exist
7. Nondiligence: failing to review key documentation
8. Canceled vacation: management overly pressuring employees to meet short-term deadlines
9. Swept under the rug: ignoring key issues in the hope that they will go away

Interestingly, and as we saw above in analyzing ethical codes, these ethical dilemmas do not overlap well with our dark side issues. Several of them are about lying, but most of our other dark side categories simply do not appear on this list. This serves to reinforce our belief that dark side matters appear all too seldom in the computing literature. There is little doubt in our minds that dark side issues are strongly related to ethical matters, and therefore should somehow appear in any discussions of ethical dilemmas in our field.

The second recent *IEEE Computer* coverage of software engineering ethics was actually a special issue devoted to the topic (“Software Engineering Ethics,” edited by Awais Rashid, John Weckert, and Richard Lucas, published in the June 2009 issue). There were four articles and a point/counterpoint debate in the special issue. The articles dealt with items such as these:

- Addressing certain values in software applications, such as trust, privacy, identity, user content control, green technology, and public welfare

## 10 INTRODUCTION

- Ways of discouraging harmful uses and encouraging beneficial uses of a software product
- The social impact of information systems failures (note that here, again, the topic of failure is coupled with the topic of ethics)
- How a code of ethics, such as that of the IEEE, can be used to aid decision making
- The point/counterpoint debate: whether and how software engineering and ethics can mix

Again, there is not much overlap with dark side topics.

The subject of ethics is one way that the software field explores matters of this kind. But often, humorous treatments of related subjects can be, in effect, discussions of ethical matters in disguise.

For example, years ago some philosophical readers of the humor publication *Mad* magazine noted that most of the material in *Mad*, although funny, was also moralistic. There were lessons to be learned and morals to be grasped in the madcap world of the pages of *Mad*.

But the Back Page section of *Computerworld* (June 27, 2005) contained something in the same vein but more specific to the computing field. While noting that lots of publications discussed the best places to work in the IT field, no one was talking about the worst places! To alleviate that problem, the Back Page article offered 10 ways to make the “worst places to work in IT list.” The list, for all its humorous intentions, becomes a nice list of things not to do, a sort of ethical violations recap:

1. Hide information. Don’t give employees the information they need to do their work.
2. Blame. Name and shame employees publicly when they do something wrong.
3. Go slow. Postpone anything that’s postponeable.
4. Distrust. Make it clear that you don’t trust your employees.
5. Reduce visibility. Don’t share broader corporate information with IT employees.
6. Block opportunities. Don’t reward successful employees.
7. Stifle arguments. Put a lid on discussions of relevant but controversial matters.
8. Outlaw play. Maintain discipline at all costs.
9. Discourage experiments. Don’t allow failure of any form, even under carefully controlled circumstances.
10. Don’t listen. If it’s worth hearing, your boss will have said it to you.

Once again, there would seem to be little overlap between these discussions of software engineering ethics and our dark side issues. For whatever reason, the issues we raise are simply not yet on the radar of most authors of software engineering materials.

### 1.1.6 Personal Anecdotes About the Dark Side

We thought it would be relevant to share with you at this point in our book some incidents of dark side issues that have occurred to us authors personally. For one thing, the dark side—up to this point in our book—has been a sort of distant concept, not well fleshed out with actual anecdotal material to make it come alive. For another, as we describe our personal experiences with dark side matters, it may help you envision times when you, too, have been involved in such matters. So here we go. (Note: For certain purposes, the “I” in what follows refers (indiscriminately and ambiguously) to either of us authors!)

- **Subversion.** I don’t really have a totally relevant subversion story to share. But I had a couple of episodes in my career where it felt like my work was being subverted.

In the first, a manager for whom I worked but with whom I felt terribly uneasy because I couldn’t figure out where I stood with him, eventually told me that he never gave me sufficient direction to do my work properly because he was afraid that if I performed well I might go after his job! I guess he was worried that I would subvert him, but in responding to that he actually subverted both me and the work I was supposed to be doing.

In the second episode, I was given the task at a major research facility of writing a particular document, one for which I had a good background because what I was asked to write about was very similar to a book I had already written.

When I had my first meeting with my colleagues on the project, it gradually emerged that they were totally opposed to what I proposed to do (or at least how I proposed to do it). Somehow I staggered to a completion of that work, in spite of the subversive road blocks they kept throwing up in my path. But when I left the facility, the first thing they did was throw out my work and redo as they had wanted it done all along.

- **Lying.** The subject of lying on software projects kind of snuck up on me. I was conducting a seminar on something or other (it doesn’t matter much what it was, after all these years). I gave my seminar attendees a task to do: one with a deliberately impossible schedule. My intent at the time was to see what they would do if they could not finish their project by the scheduled completion time. Would they override the schedule and take whatever time it took, or would they short-circuit the project goals and try to finish on time?

I think it is significant that those seminar attendees took the schedule as a requirement and downplayed quality in their rush to finish “on time.” I think much of the evidence since then supports the notion that that’s just the way it is on software projects, at least at this point in time. Schedule trumps quality, even if it shouldn’t.

But as the attendees and I discussed what had happened, they also quickly swung the conversation around to “lying on software projects.” (My schedule requirement had been, in a very real sense, a lie). They said things such as “I have to lie 30–50% of the time to get my work done”; “I had to

check my ethics at the door when I went to work here”; “I make wildly optimistic promises to get my management off my back”; and “managers who don’t tolerate failure I especially lie to.” Lying, as these seminar attendees saw it, was a conflagration destroying the profession of software engineering.

I have chosen to talk about the subject of lying rather than to cite personal examples of my lying on software projects. You didn’t really think I would talk about lies I personally have told, did you?!

- **Hacking.** I have an account with a leading investment firm, one where I keep my retirement funds. It is, as you might imagine, ID and password protected.

But, a couple of years ago, a hacker managed to steal my identity and begin performing transactions in my account. I noticed the sale of a big bundle of corporate stock and queried it to the investment company. They immediately froze my account; the stock had been sold, but the money had not yet been transferred out of my account.

As we pursued this matter further, we could see that the hacker had provided an overriding address for the delivery of the check for his sale of my stock and a contact phone number. Fortunately, that was as far as he had gotten.

There were two things to be done. The first was to change my ID and password, and I did that (even that kind of change is not simple when you are under attack). Then I needed to decide what legal steps to take.

In the end, I notified the police department in the city whose address had been provided. The last I heard, the police were going to contact whoever lived at that address. But it is a characteristic of our legal system that there is no follow up; for a variety of reasons, you are never told what came of the matter. So although I would like to end my anecdote by talking about how the rotten character who did this to me got his comeuppance, I will never in fact know if that was the case(!).

- **Theft of information.** There was a time in my life when I supplemented my full-time income by doing legal consulting on the side regarding theft of software. In one case that I remember quite clearly, the situation was that one company had produced a software product and another had put a similar product on the air not long after hiring a former employee of the first company.

In such cases, the legal system provides for the lawyers involved to get access to the listings of both products, and I in turn was given those listings to examine and analyze. I found that in some parts of both versions of the software product, the code was noticeably different but the design structure (as reflected in the product’s call trees) was nearly the same. And, to make matters more interesting, there were a few places where marginally relevant comments included in the first company’s product code showed up in the second company’s version as well!

Now at this point, I’d like to say that the second company was appropriately punished for taking the first company’s code. But what actually happened was that the companies at that point settled out of court, with a provision

in the settlement that the result could not be disclosed. As a result, I never knew what actually happened!

- **Espionage.** I have never really encountered espionage as such on any software projects I have been involved with. Or perhaps what is really true is that I never recognized any espionage that may have been going on around me!
- **Disgruntled employees and sabotage.** Probably the most disgruntled employee I ever worked with was a pacifist I'll call Harley Dove who for some reason I have never fathomed found himself working on military projects at an aerospace company.

I was the project lead; Dove was one of my workers. And, as time passed, it became next to impossible to get any useful work out of Dove, whatsoever. Project due dates came and went, and Dove continued to do almost nothing, and nothing I could do would motivate him to do any work. If not contributing to a project is a special form of sabotage, then Dove was a (pacifist!) saboteur as well as a disgruntled employee!

My own personnel review, at the end of that unfortunate period in my career, reflected my total failure to be able to get any work out of Dove. It's no wonder that this is a pretty memorable episode to me!

I'd like to say that the company converted Dove into a disgruntled ex-employee, but to the best of my knowledge they never fired him. I sometimes wonder what he is doing today, and whom he is doing it to or for!

- **Whistle-blowing.** I once had a consulting contract with a leading banking company. I was invited to do the job by one of the bank's top technical people, whom I will call Top Tech, and at the end I was to present a report on my findings to the top manager, whom I will call Senior Manager. The problem that caused the bank to call me in was that they were falling badly behind on a key project.

During the course of my information gathering, Top Tech told me that one of the problems they had was that they were paying bonuses for the fixing of key bugs, that their best programmers were hoarding those key bugs in order to achieve those bonuses, and that therefore the backlog of bugs to fix was huge and growing. And then, having told me that, Top Tech went on to ask me to keep that finding confidential!

Time passed, and it came time for me to present my findings to Senior Manager. Top Tech attended that briefing with me. I had wrestled all along with the dilemma of reporting that key bug finding or suppressing it to keep my commitment to Top Tech. Going in to the meeting, I still wasn't sure what I was going to do. In the end, what I did was this: I hinted at the problem in my presentation, and Senior Manager picked up on the hint and asked me about it. I hesitated to see if Top Tech would speak up, and when he didn't, I glossed over the whole thing.

Here is my conclusion from a whistle-blowing point of view: I had a golden opportunity to blow the whistle at a point that would have counted,

and I failed to do it. Whistle-blowing, I have to conclude based on my own failings, is not an activity for the faint of heart!

## REFERENCE

---

Glass, Ramesh, and Vessey. "An Analysis of Research in Computing Disciplines," *Communications of the ACM*, June 2004.

# *DARK SIDE ISSUES*

You have just finished reading our introduction to our dark side book. We have told you what we mean by the dark side, why we chose to write about it, how prevalent it is, and who else is talking about it. And we shared with you some personal anecdotes about our own experiences with dark side matters.

We also pointed out that, although we'd like to say some generic things about these dark side issues, in fact it is nearly impossible to do so. All these dark side issues have some things in common—they are all evil manifestations of computing behavior—but on the other hand, they differ enormously in how often they occur and what they are about.

It's high time, at this point in our book, that we stop waving our hands about these dark side issues, and get down to brass tacks about them. In the seven chapters that follow this introduction to Part 1 of our book we get very specific about each of these matters. Welcome to the many-faceted worlds of subversion, lying, hacking, theft of information, espionage, disgruntled employees and sabotage, and whistleblowing. We hope you will find it as fascinating to learn about those issues as we did.



# SUBVERSION

We use several approaches to explore subversion. The first section covers case studies and examples of subversion on software projects; the background information for the material is drawn from the computing and popular press. In the second, and longest, section of this chapter, we present the findings of our unique research survey, one in which we surveyed practitioners to determine how often subversion happened in the software world, and in what ways. We are particularly proud of this section, in that we present the results of exploring a major topic in the field of computing and software that no one else has explored. (An abbreviated version of this material was published earlier in a leading computing journal). Finally, in the third major section of the chapter, we present the hitherto unpublished results of a follow-up survey, one in which we asked responders to the first survey for additional input.

Now, on to the case studies.

## 1.1 INTRODUCTORY CASE STUDIES AND ANECDOTES

Some Motivational Examples. A sprinter is preparing to break a one-hundred-meter record. During the race someone on the edge of the track disturbs him by throwing pebbles at him and holding up funny pictures. The sprinter's chances of breaking the record are diminished because of the distractions. If the person who is causing the disturbances is an experienced sprinter himself, he might do it in a more sophisticated way—for example, tens of seconds before the official start he might imitate the sound of the starting signal. The sprinter is thus likely to fail in breaking the record and, what is more, he may even fail before the start of the race. The analysis of the failure of the project concludes the following: “Study after study reveals that sprinters have the most problems in the fractions of a second around the start, that is, at the very beginning of the race” (also known as the “requirements phase”!).

Such a situation in sports verges on the ridiculous. However, it happens quite frequently in software projects. A great number of software projects involve people who wish the project to fail. How is this possible?

### 1.1.1 A Faculty Feedback System

A college wanted to introduce an online system that allowed students to give anonymous feedback to their teachers. The feedback system was intended to provide an outlet for the evaluation of the quality of lectures and even reveal possible problems. It was hoped that, in the long run, the system would help to identify ways to improve the average quality of lectures.

A superficial analysis revealed three stakeholder groups: the students, the professors, and the college management. The students and the management supported the planned system for obvious reasons: The quality of the lectures and thus the reputation of the college were expected to improve through this project. In theory, both the students and the management would benefit from increased influence; the management would have gained access to additional ways of control. The students were concerned, however, that the anonymity could be broken one way or another, resulting in potential disadvantages for students who had given negative feedback.

A broad consensus of opinion indicated a concern that the feedback needed to be secured against potential manipulation. To prevent the possibility of results tampering by the students, each student was provided with only one opportunity to vote for each lecture. The chances for a very angry student to submit the same negative feedback more than once (thus dramatically lowering the average evaluation feedback for that particular lecture) were reduced. To prevent the possibility of a professor illicitly tampering with the system (for example, giving excellent feedback to his own lecture by pretending that he was a student), other safety measures were introduced. The system had to prevent all these kinds of potential manipulations of information. However, the aspect of protection against falsification required some authentication, which could be a conceptual conflict to the prerequisite of anonymity.

The professors' responses were multifaceted and therefore required further analysis. Some professors who were well known for their outstanding lectures welcomed the plans for the feedback system enthusiastically for quite obvious reasons: They expected excellent feedback for their lectures. Officially, there was no connection between the students' feedback and the career opportunities of the professors. It was obvious, however, that continuous good feedback would be taken into consideration if a higher position in the college management became vacant.

Other professors were more reluctant about the feedback system. Some teachers bore the responsibility of teaching difficult (and mandatory) lectures such as math. Since these lectures were known to be unpopular with many students, the teachers expected negative feedback: Even an excellent lecture of this type (for example, in statistics) would never get feedback as good as a "special interest group" lecture in which only students who are fascinated with the topic participate.

Additionally, some teachers, who were running a small consulting business in addition to their teaching duties, were concerned that the feedback system might force them to spend more time preparing the lectures, something that could eat into their time for professional engineering consulting. This college allowed additional consulting income as long as the teaching duties were not affected. However, this type of sideline was only tolerated but not fully accepted.