BOOK REVIEWS

Jigang Liu
Editor
Metropolitan State University
St. Paul, MN 55106
Jigang.Liu@metrostate.edu

If you have any suggestions on books for review, or you would like to write a book review for us, or you have any comments and concerns on the book reviews published on this column, please feel free to send an email to Jigang Liu at Jigang.Liu@metrostate.edu.

BOOK REVIEW

Zeidman, B. (2011). *The Software IP Detective's Handbook: Measurement, Comparison, and Infringement Detection*. Boston, MA: Pearson Education, Inc. 480 pages, ISBN-10: 0137035330; ISBN-13: 978-0137035335, US$49.99.

Reviewed by **Diane Barrett**, American Military University.

Do not the book title fool you into thinking that the book is only for those looking to detect software infringement detection. It is a comprehensive look at software intellectual property. The book covers a wide range of topics and has something to offer for just about everyone from lawyers to programmers.

The book is divided into ten sections. There is a small overview segment at the beginning of each section providing a brief synopsis, objectives, and the intended audience. The first two chapters contain the book introduction. What I like in particular about Chapter 1 is that there is a nice table titled "Finding Your Way Through This Book". The matrix cross-references the Chapter number, titles, and the particular group or occupation that would have an interest in each Chapter. This is a very handy chart and it shows the amount of thought Ziedman put into how each part of the book suits the needs of different readers. Since the book includes a Safari Online license, I looked at this version also. The Safari version has a nice five minute introductory video by Ziedman. The rest of the introduction, Chapter 2, provides an overview of intellectual property crime along with statistics.

Part II, comprised of software concepts covers Chapters 3, 4, and 5. This section helps the reader understand basic software principles and how different types of software code relate to each other,. Chapter 3 explains source code and provides a nice table with common programming languages including a description of each language. Ziedman uses an example of source code for a small function to explain what source code does and how it is written. The example provides enough detail without being overwhelming for a novice particularly since the

intended audience is nontechnical managers and lawyers. Chapter 4 covers object code and assembly code. It is short, to the point and not overly technical. Since some embedded devices use assembly code, it is good to see an explanation of what the code does along with an example. Chapter 5's format is similar to that of Chapter 4 in that it is short, to the point and not overly technical. In the chapter, Ziedman covers important concepts such as scripts and macros.

Part III is one of the most extensive areas and spans over the course of four chapters. This part delves into intellectual property, including copyrights, patents, trade secrets, and software forensics. This section helps the reader understand basic intellectual property terms and how these terms relate to one another. According to the matrix in Chapter 1, this is a "must read" section for all readers, with the exception of lawyers, who should already be versed in all areas but software forensics. Chapter 6, which covers copyrights, begins with the origin of copyrights as far back as Great Britain's Statute of Anne. A highlight of the chapter is the case examples emphasizing different types of software copyright issues. The reverse engineering section is particularly interesting as is the Atari case example. The Safari version has a short video to go along with the chapter. Chapter 7, Patents, is laid out similar the copyright chapter with history as case examples intertwined. The Safari version has a short video to go along with the chapter. Chapter 8 on trade secrets is a short concise chapter that contains a very useful table evaluating the advantages and disadvantages of patents and trade secrets. The Safari version has a short video to go along with the chapter. The final chapter in this part was quite intriguing. Ziedman explains the practice of software forensics, demonstrates how it fits into the field of forensic science, and clearly defines how it differs from digital forensics. The chapter concludes with a discussion of expert certification and tool testing.

Part IV describes the basic methods of comparing and measuring software via chapters on theory, implementation, and applications. The purpose of this section is to provide the reader with a theoretical foundation for characteristic comparison of software source code. This part may have more of an appeal to computer programmers and software engineers but can be useful to anyone wishing to understand or implement techniques for measuring software changes. This section covers how to measure the amount of code that has been directly copied between programs. Chapter 10 is theory and tends to be somewhat mathematical but not so much as to lose the reader. There are many tables to help with the explanations. Ziedman shares a very important observation about the location of files and measurement algorithms. Chapter 11 covers the implementation of source code differentiation methods. This chapter is similar to the previous one in that it contains numerous mathematical formulas and tables to explain the concepts. Ziedman does a good job of providing a concise explanation of a complex topic. The final chapter covers the application of source code differentiation. This chapter is less mathematical than the previous two chapters. It contains a nice graphic showing the evolution of the lines of code and files of a

project from the original version to a subsequent version making the process easy to follow for the reader. The case story at the end of the chapter clearly describes why this part of the book is essential.

Part V is the other extensively covered area, spanning over the course of five chapters. This part delves into more complex methods of comparing software. It is similar to Part IV in purpose and audience. The difference between the two parts is that correlation takes the concepts one-step further and finds similarities even though the code has changed. The topics covered in these five chapters are the same as in the source code differentiation part but Ziedman adds two chapters on the front end covering software plagiarism detection and source code characterization. While these two chapters are short, they contain a lot of good information. In chapter 13 Ziedman weaves personal experience into his writing in a story about theory formation and publication. I found Mike Flynn's comment to him a bit humorous yet laced with truth. Chapter 14 on characterizing source code, lays the groundwork for the next chapter on theory. The chapters on theory and implementation are somewhat mathematical but contain enough tables and case stories to keep the content interesting. The last chapter starts with a table summarizing the many applications of source code correlation then provides a detailed explanation of each type in the following pages.

Part VI consists of three chapters. It is similar to Part IV and V in purpose and audience. Following the same pattern as the previous two parts, it contains theory, implementation, and application chapters. This section examines the theory and mathematics of object code correlation and source/object code correlation. These concepts help the reader understand how object code is compared to object code to find signs of copying and how source code is compared to object code to find signs of copying. As in the previous chapters on theory, the text is somewhat mathematical. Chapter 19 provides a description of the algorithms for creating a program to measure both correlations. I found Chapter 20 on applications to be the most fascinating chapter in this section partly because of my interest in malware and because there was an interesting case study included.

Part VII It is similar to Parts IV, V, and VI in purpose and audience. This is a relatively short section with theory, implementation, and applications combined into one chapter with most of the chapter containing the mathematical theory and the last two pages containing implementation, application, and a case study. This section arms the reader with the theory and mathematics of source code cross-correlation and may be especially useful programmers who want to understand how to implement this measure. This type of correlation compares functional source code statements to nonfunctional source code comments for finding signs of copying. The implementation section contains some good observations about the use of commented-out code.

Part VIII consists of three chapters. The objective is to explain methods and tools

used by expert witnesses and technical consultants to determine IP theft or infringement. The intended audience changes for this part and attracts managers and lawyers more than programmers and scientists especially since the discussion of techniques is from the point of view of an expert witness. Chapter 22 on copyright infringement provides a nice figure to describe Ziedman's software copyright infringement methodology, an abstracted block diagram of the CodeSuite program and a Copyright Infringement Checklist at the end of the chapter. In Chapter 23, techniques for detecting whether a program infringes on a patent are discussed. Ziedman shares his experience and the general process used at his consulting company. It includes a case study, discussion on a Markman hearing, and the landmark case of Phillips v. AWH Corporation. The DMS Software Reengineering Toolkit (SRT) and invalidity by proof based on Sections 102, 103, and/or 112 of the Manual of Patent Examining Procedure (MPEP) issued by the U.S. Patent and Trademark Office are also covered. Chapter 24 covers techniques for detecting whether a program incorporates another's trade secret. It includes an interesting case study and a section addressing tools.

Part IX consists of three chapters and applies to all audiences. The objective of this section is to provide some insight to key areas not directly related to software intellectual property or software forensics but that anyone involved in the field of software forensics will most likely come across. Chapter 25, which covers clean rooms, has a detailed figure with clean room procedures, includes cases such as of Nordstrom Consulting, Inc. et al. v. M&S Technologies, Inc. et al and the Safari version has a short video which I really enjoyed. Chapter 26 is very timely since the use of open source software is increasing due to budget constraints and rising costs of commercial software. Topics covered include the Open Source Initiative (OSI)'s formal definition of open source code and the Free Software Foundation coinage of the word copyleft. Numerous lawsuits are discussed including SCO v. IBM, IP Innovation LLC v. Red Hat and Novell and Network Appliance, Inc. v. Sun Microsystems, Inc. Ziedman provides his own thoughts on the open source movement to end the chapter. Chapter 27 is a short chapter that provides the particulars of the DCMA, arguments for and against the DCMA, and three noteworthy lawsuits.

Part X contains a conclusion by Ziedman covering what has been done to date, speculation on areas of future research that build on the book concepts and a look toward new applications. The points I found most admirable are that Ziedman is a strong believer in formal procedures and he feels that the mathematical frameworks described in the book can be applied to many things outside of software forensics.

According to the report compiled by SAIC and MacAfee titled *Underground Economies: Intellectual Capital and Sensitive Corporate Data Now the Latest Cybercrime Currency*, 25% of organizations suffer from intellectual property theft. We have recently seen efforts to control some of this loss through the Stop

Online Piracy Act (SOPA) and the Protect IP Act (PIPA). In light of these occurrences, Ziedman's book could not come at a more opportune time. For someone who is more of a networker than a coder, I found it to be a good read and was through 100 pages before I knew it. I will be implementing the book into some of my classes. I believe it adds value and clarification to an area that is rapidly growing. It also enhances the skills of forensic students by introducing software forensics. Ziedman did a great job on the explaining the concepts so they are understandable by a wide audience and did it in a unique way.