including luminaries from Catmull and Smith to Bill Kroyer and Sherry McKenna. These interviews shed new light on such computer animation milestones as the making of *Tron* (1982), the implosion of Digital/Omnibus/Abel (1987), Disney's adoption of a Computer Animation Production System (1989) and its role in their animation revival in the early 1990s, the birth of DreamWorks SKG (1995) and its acquisition of Pacific Data Images to make *Antz* (1998), and of course, the rise of Pixar to the computer animation throne with the runaway success of *Toy Story* (1995), the first completely computer-generated full length animated film.

Moving Innovation ends with a twinge of melancholy, as Sito recognizes the end of the pencil-drawn era, but he also acknowledges that animation is evolving rather than dying as, for example, the combination of CG motion blur and traditional stop-motion animation resulted in two 2009 Academy Award nominated films—*Coraline* and *Fantastic Mr. Fox*. Sito's history will appeal to anyone with an interest in animation, traditional or computer generated, particularly those interested in the personalities who helped make computer animation "magic" happen.

Sarah A. Bell *is a doctoral candidate in communication at the University of Utah. Contact her at sarah.bell@utah.edu.*

Donald E. Knuth and Edgar G. Daylight, *The Essential Knuth*, Lonely Scholar Scientific Books, 2013, 89 pp.

Peter Naur and Edgar G. Daylight, *Pluralism in Software Engineering: Turing Award Winner Peter Naur Explains*, Lonely Scholar Scientific Books, 2011, 127 pp.

Edgar G. Daylight, *The Dawn of Software Engineering: From Turing to Dijkstra*, Lonely Scholar Scientific Books, 2012, 239 pp.

Reading these three books by Edgar Daylight and surfing his website makes clear that he is on a substantial mission to study the early days of software engineering. Much of his era of study is before the term "software engineering" was in common use, back when pioneering computer people were taking early steps in trying to figure out how to write computing programs.

From my point of view (as a non-professionally trained computing historian with a career as a computing practitioner behind me), Daylight is working at an interesting place in computing history. He is writing scholarly researched computing history, but it's tilted a bit toward the world of computing people rather than directly at computing historians. To a considerable extent, Daylight is aiming at university teachers of computing science because he believes that learning the history of the field can help improve the quality of the field. Furthermore, by including the edited interviews themselves, rather writing his own narrative based on traditional (barely edited) oral history interviews, Daylight provides a first-person-account component that is likely to make the books more popular with people from the computing world itself. Daylight has trained as both a computer scientist (PhD) and as a historian (multiyear mentorship under Gerard Alberts) and thus is well-qualified to undertake the research and publication program he has set for himself.

I review these three books in the order I found and read them.

Knuth Booklet

The Essential Knuth primarily consists of the interview Daylight did with Donald Knuth in November 2012 in Frankfurt, Germany. The interview (74 pages) is accompanied by a preface in which Daylight gives background on the interview and notes some themes he sees in the interview. The book also includes a 66 item bibliography of books and papers mentioned during the interview and an index. The title page states that the booklet was edited by Kurt De Grave. In other words, the interview is longer, more richly documented, and carefully edited than many more journalistic interviews of Knuth.

This interview complements the many other interviews that have been done with Knuth. It covers some things I don't remember reading about elsewhere, and it has a different slant on Knuth than many of the other interviews. (A fairly comprehensive list of other interviews exists on the website http://tug.org/interviews/#knuth of the TeX Users Group, which venerates Knuth as its founding father.)

Consistent with Daylight's interest in the history of programming and his goal of extensively interviewing high-profile and retired computer scientists, his interview of Knuth deals with Knuth's work with computer languages and language processors; Knuth's interest in the methodology of structured programming; and Knuth's personal

involvement with or knowledge of other pioneers of the 1950s, 1960s, and 1970s who were involved with computer language and programming research. The bulk of this discussion is in Chapters 3 through 5 of the book's six chapters.[1]

In the interview, Knuth expresses views on a variety of early software researchers and their points of view, including Ole-Johan Dahl, Edsger Dijkstra, C.A.R. (Tony) Hoare, Peter Naur, and many others. Many of his assessments are positive—for instance, he calls Joel Erdwinn the "main genius behind BALGOL," the Burroughs version of Algol that greatly impressed Knuth, and the study of which showed him the use of linked lists in programming. But he is uniformly candid; for example, Knuth seems more impressed with Harlin Mills's accomplishments than those of Frederick Brooks in the domain of structured programming, and he lauds Bernard Galler for founding this journal but not as a researcher. Knuth is also candid about his own characteristics. He sees himself as born to do discrete mathematics and computer science, poor at forecasting future developments in computing, unable to write an overview of a topic without first completely surrounding the topic, and wishy-washy compared with Dijkstra and Naur, of whom he says that their strength lay in the strength of their ideas.

With the exception of Chapter 5, the discussions of other people are in the context of the software topics about which Daylight queries Knuth. Some of these topic areas of discussion are formal methods versus practical understanding, machine independent programming in the absence of a specification for floating-point arithmetic, and what is and what isn't structured programming.

In the booklet, Daylight states his belief that such historical accounts can "drastically help computer professionals understand their own research problems" (pp. 3–4). I'm not sure how much of an audience he will find in this domain. However, I believe the book may be of interest to two other groups of people who read this journal. The first group is people like me who entered computing in the era Daylight and Knuth discuss and who themselves read and admired the writings of the computer software pioneers of the era. This booklet sheds light on how these pioneers interacted with each other and what they thought of each other's ideas as well as reminding us of some of the issues of the day.

A second group of people who may find this booklet useful are fledgling professionals and their university teachers. I can imagine that college history majors beginning their study of the history of computer software will find this booklet particularly interesting. It recounts some of the key issues and the evolution of the discussion of the issues of which such historians, not having lived through the era as practitioners, may be unaware. The book's bibliography is also an excellent starting point into the source papers of the time.

Equally interestingly, the booklet has a definite slant on the practice of computing history that could be a basis for interesting discussions in a history course context. For instance, covering a narrow portion of Knuth's life and work and having been heavily edited and augmented, how does this booklet differ from an oral history, and what is the value of such a document compared with typical oral histories that are a mainstay of professional computing history researchers?

Also, Knuth views himself as something of a historian of computing history and has strong opinions on how it should be done. In Chapter 6, "Historiography," Knuth objects to what he has read by Martin Campbell-Kelly,[2] which Knuth sees as encouraging a move in computing history writing away from including technical content. Knuth is also dismayed by this trend more generally in science history writing. (Daylight also takes a swipe at Campbell-Kelly in this discussion of historiography.) As is typical of Knuth, he communicated his concerns to Campbell-Kelly and gave a presentation at a 2009 conference attended by Campbell-Kelly in which he contrasted mathematics history writing (which Knuth says continues to include technical content) with computing history writing. (Unfortunately, the attempt to record this talk failed.) Thus, here's another possible discussion question: To what extent should the history of computing focus more squarely on technical matters?

If I were a beginning student of computer software history, I'd enjoy starting my study with this booklet. In any case, I recommend the booklet. Daylight has clearly done his homework, his questions are insightful, and Knuth is highly articulate in this written version of his answers to Daylight's questions. The booklet is easy to read and a pleasure to read.

Naur Booklet

The following words appear above the title on the front cover of *The Essential Knuth*: "Lonely Scholar Conversations Issue 3." Clearly, the Knuth booklet is part of a series,

which led me to look up the two prior interview books from Daylight.

As with the Knuth booklet, *Pluralism in Software Engineering: Turing Award Winner Peter Naur Explains* is an extended interview by Edgar Daylight. The front cover says, "Conversations Issue 1." This interview of Naur was conducted on three days in April and May 2011. This booklet is divided into 18 short chapters with a 79 item bibliography of documents mentioned in the interview. The chapters are divided into three parts, which Daylight describes in his preface:

> Part I should be of interest to most readers [reviewer note: this is the most computery part of the booklet]. Part II delves into the details of philosophy, program development, and Dijkstra's "pleasantness problem." Part III elaborates on Naur's recent research on psychology and helps the computer programmer, with hindsight, Parts I and II with much greater precision.

As with the Knuth volume, Daylight appears to have been superbly prepared for the interview with many specific and sometimes subtle questions. The Naur booklet sheds some light on how various computing pioneers interacted with each other and what they thought of each other's ideas. However, all in all, I found the interview slanted too much toward philosophy and psychology given my background in computer programming and its history and lack of background in philosophy and psychology. Nonetheless, I see the Naur volume as a useful contribution to the history of computing and believe that computer history libraries and archives should have a copy.

Software Engineering Book

The Dawn of Software Engineering contains four interviews (about 80 pages total): Chapter 4, "Tony Hoare and Mathematical Logic"; Chapter 5, "Niklaus Wirth and Software Engineering"; Chapter 6, Barbara Liskov and Data Abstraction"; and Chapter 7, "Peter Naur and Turing's 1936 Paper." Some or all of the Naur chapter is drawn from the same interviews upon which Daylight's *Pluralism in Software Engineering* was based.

The book also contains the equivalent of another four chapters (about 85 pages total) from Daylight himself and includes his thinking about how software developed in the early days, starting with Turing. A good bit of Daylight's own material relates to his views on Edsgar Dijkstra's role and are based on his extensive study of Dijkstra's writings and writings about him. (Because Dijkstra died in 2002, he could not be interviewed for this book.) These chapters are titled as follows: Chapter 1, "Introduction (From Turing to Dijkstra)"; Chapter 2, "Turing's Influence on Programming"; Chapter 3, "Dijsktra's Rallying Cry for Generalization"; and Chapter 8, "Deromanticizing Turing's Role in History." The book also includes an index, a 316 item bibliography, and a long set of endnotes.

I liked this book and recommend it, although the Wirth and Liskov interviews interested me more than the Hoare and Naur interviews. I also find Daylight's intellectual tussle with Turing's influence on and Dijkstra's role in software engineering to be engaging and thought provoking.

This book is not marked as an issue in Daylight's conversation series, so apparently between the Naur and Knuth conversations is another interview book in the making.

Daylight Collection

Perhaps this is the best order to read these three books: first, *The Dawn of Software Engineering* to understand how Daylight views his mission in the history of software engineering and to read what four pioneers say; second, the Knuth volume to see what another pioneer says; and third, the Naur volume to get the rest of Daylight's interview of Naur.

Despite my doubts that Edgar Daylight's historical accounts can "drastically help computer professionals understand their own research problems," I admire him for pursuing his mission and I certainly find his approach engaging and the computing history fascinating. Daylight's website (dijkstrascry.com) has much additional interesting content from his area of computing history research.

References and Notes

1. The booklet also provides an introduction to other work of Knuth for those who don't know much beyond his series on *The Art of Computer Programming,* published by Addison-Wesley. See http://www-cs-faculty.stanford.edu/~uno/books.html
2. M. Campbell-Kelly, "The History of the History of Software," *IEEE Annals of the History of Computing,* vol. 29, no. 4, 2007, pp. 40–51.

David Walden *retired from BBN in 1995, after working as a computer programmer, technical manager, and general manager. Contact him via waldenfamily.com/ieee.*