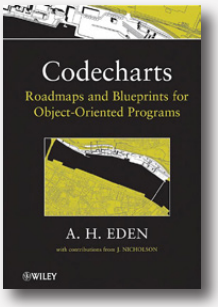
linking field notes to the maps generated *in situ*. Neatness, completeness, and factual display of the information collected are emphasized, while details such as the use of dashed versus solid lines for inferred or known contacts are scattered throughout. The second chapter, "Fair Copy Maps and Other Illustrations" is aimed at creating a clean map that shows selective information. Though not explicitly stated in the book, the authors mean data generalization, and describe which information from a field map transfers to the fair copy map. Some considerations that a geologist makes are familiar to any cartographer, including map function, clutter control, and scale.

The authors emphasize the use of hand-drawn maps in this book; while they do consider the use of GPS, GIS, stereographic aerial photography, and even Google Maps, the majority of instruction is aimed at the use of paper and pencil for the creation of both the field map and fair copy map. There is brief mention of the use of computer drafting and modeling in these two chapters. The chapter titled "Field Maps and Field Notes" covers the use of drafting software for the generation of fair copy maps, but warns that the use of GIS for generating geologic maps is very time consuming. Despite this, the authors list the basic steps in transferring their hand drawn field maps into a GIS. In the chapter titled "Fair Copy Maps and Other Illustrations" briefly introduces the use of geological modeling software such as RockWorks and Datamine for creating three-dimensional models from data collected in the field.

The book concludes with a chapter covering geologic report writing. The material covers report structure, referencing systems, and some general comments about the importance of clear and well-written reports for conveying geologic information. This chapter serves as a nice reminder for students of how to compose a report, and offers useful information on writing specific to geologic information.

Overall, I would recommend this book as a useful introductory textbook for geology students heading out for their first field mapping experience, or as a general reference handbook for the more advanced geologist. The book is succinctly written and contains many useful tricks and tips for a wide array of geological measurement techniques. I would recommend students using this book for geology courses outside the United Kingdom to explore the guidelines and map resources available to them in their home countries.

CODECHARTS: ROADMAPS AND BLUEPRINTS FOR OBJECT-ORIENTED PROGRAMS



By Amnon H. Eden, with contributions from Jonathon Nicholson.

John Wiley & Sons, 2011. 243 pages, no maps, 123 code charts, many diagrams. $93.95, hardcover.

ISBN: 978-0-470-62694-8

Review by: Jed Marti, Artis LLC

If you're interested in the design and structure of large-scale programs such as Geographic Information Systems from the perspective of designing your own, you might find this book of some interest. As the author points out, software systems are the most complex things our civilization has created, and their sordid past is filled with failure. Codecharts define the components and structure of object-oriented programs in simple graphical terms, somewhat like how antiquated flow charts mapped decisions, loops, and computations into something easier to understand than Fortran code. The aim is to create a visualization that will "fit on the side of a van." It is hoped that programs developed from Codechart descriptions will have better structure that those that are not. By providing a mapping between LePUS3, the language of Codecharts, and Java (other languages are possible), it is possible to verify that a Codechart models a Java program.

There are 18 chapters divided into 3 sections. The first section compares Design Description languages and Codecharts (and its language LePUS3) to the more widely known Universal Modeling Language (UML), of which the author is very critical. The second, and largest, section describes modeling the various components of Object Oriented programs and presents examples of common practice and structures. The final theory section proves the completeness of Codecharts and provides an entrance for formal verification of programs so described.

I approached this book with three questions:

1. What, if anything, does this book have to do with Cartography and Geographic Information Systems?
2. Can the methodology proposed be of help to the practitioners thereof?
3. Is this a good textbook?

Sadly, I conclude that the answers are nothing, no, and not likely for any programs or systems we might find interesting. A Cartographer will find no interesting maps or charts; there are no dragons marked on the Codecharts, there are no route numbers on the Roadmaps. Black, grey, and white are sufficient for all diagrams presented. The book has a high-resolution map of an urban manufacturing area on the cover but that is the extent of the cartography.

Can learning this practice provide any assistance to the learned cartographer? Here I think Codechart's generalization level fails the potential user. The language of Codecharts is object-oriented programming, not the *lingua franca* of Cartography and Geographic Information Systems. Contrast this with Scalemaster diagrams (Roth et al. 2011), where the description is in terms of geography and geometry—the abstraction is in terms of the problem to be solved, not the underlying implementation. Furthermore, this abstraction drives the implementation in the way that a Codechart written before the implementation cannot. One might hope to derive a Codechart from this description, but the promised benefits are unlikely to be useful or cost effective for the program's users or implementers as this involves learning an unfamiliar representation.

The authors make the assumption that programmers are competent machinists and cannot be relied upon to create things without the benefits of a fancy flowchart. While this may be true in enterprise systems with organized hordes cranking out Java, many of the great systems of our time such as SAS (statistics), ARCInfo, Maple (Computer Algebra), and Matlab, amongst others, were designed and at least initially implemented by leaders in their fields with a goal and firm grasp of the field's priniciples and perhaps set of *ad hoc* flow charts and state diagrams at best. The success of these enterprises invites emulation, not replacement.

If the methodology isn't particularly useful for GIS, can it aid the broader community of programmers? The laudable goals are object structure correctness, re-usability, and visualization. Following Codechart strictures may very well lead to such programs but only for certain applications—those similar to the case studies for support libraries for XML, 3D graphics, and other similar application program interfaces. It's very hard to see where a new, stand-alone program, perhaps even using these Codecharted APIs, would benefit. A Codechart may define the 3D texture mapping method in the class hierarchy but has little to say about what it does and how to use it—you're still required to read the manual to find this out. What have you gained?

Though the Universal Modeling Language has achieved a certain level of acceptance for complex systems it is unclear that the rigorousness of Codecharts is required or supportable in a research or academic environment. The support for only Java limits applicability to systems that don't require great efficiency for large amounts of data. The authors wistfully hope that support for C++ and other useful object-oriented languages may follow, but with minimal return on investment, I think few would find this worth their effort.

Is this a textbook that one might use in the Geography and Computer Science classroom? This is unlikely for three reasons: 1) There are no aids or even suggestions for exercises, 2) There is no indication of how to accurately judge a Codechart's quality, 3) Large scale acceptance of UML. Contrast this with the extensive exercises in the classroom staple *The Art of Computer Programming* (Knuth 2011), where the exercises are the most important part of the text and problems range from obvious to PhD theses. Though the book presents case studies for the usual design patterns, the methodology lets you produce functional yet clumsy object structures. Finally, there's the inertia of large-scale acceptance of UML with all its flaws. *Codechart*'s goals are laudable but must provide more added value to leap this hurdle.

REFERENCES

Knuth, D. E. 2011. *The Art of Computer Programming*. 5 vols. Boston: Addison-Wesley

Roth, R. E., C. A. Brewer, M. S. Stryker. 2011. "A Typology of Operators for Maintaining Legible Map Designs at Multiple Scales." *Cartographic Perspectives* 68:29–64.