



Software Systems Construction with Examples in Ada

Bo Sanden

Software Systems Construction with Examples in Ada is written by Bo Sanden, and published by Prentice-Hall, Inc., 1994, ISBN 0-13-030834-X, 433 pages, \$ unknown by reviewer

This book certainly lives up to its title. The author delivers a text "intended for readers with practical experience in software construction. It targets graduate or upper-level undergraduate students of software engineering as well as practitioners who want to familiarize themselves with modern trends in software construction applied in a mainstream language environment." As the author notes, the book could be complimented by an Ada primer and a software engineering text. This reviewer found the text well structured, filled with appropriate references, expanded with a number of design examples, and supported by a large amount of Ada source code fragments (specs and bodies). Additionally, each chapter concludes with a number of problem statements. I am sure with the author's attention to detail, an updated edition covering Ada95's new object oriented (OO) and real-time features will be forthcoming.

The text includes nine chapters that cover software construction in Chapters 2-7, present entity-life modeling in Chapter 8, and conclude with Chapter 9 by walking through a case study of a flexible manufacturing system from problem statement to Ada implementation.

Discussions on software construction build through the text and cover different aspects such as control structuring, modularization, information hiding, abstract data types, finite automata, concurrent tasks, and resource sharing. Chapter 8 presents entity-life modeling as an approach for development of concurrent programs. The discussion builds nicely on the earlier topics and covers the properties of concurrent programs. Then moves on to the analysis and design methods that support the modeling approach. Two well-known problems (the buoy and elevator examples) are provided with their ensuing entity-life solutions. Chapter 9 provides a detailed case study by applying the modeling approach to a non-trivial flexible manufacturing system. This culminating chapter also serves to tie all of the earlier discussion together.

I would strongly recommend this book for a curriculum that covers software construction and the Ada language. The introduction of entity-life modeling broadens the target community to the OO realm; however, the experienced OO practitioner may need only to focus on the final two chapters.

Reviewed by Frank Hollenbach —
frankhol@nadc.nadc.navy.mil.

An ISO 9000 Approach To Building Quality Software

Östen Oskarsson and Robert L. Glass

An ISO 9000 Approach To Building Quality Software is written by Östen Oskarsson and Robert L. Glass and is published by Prentice Hall PTR 1996, hardbound, ISBN 0-13-228925-3, 274 pages, \$49.95.

In this self-styled "schizophrenic" analysis of the ISO 9001 and 9000-3 standards, Oskarsson and Glass take contrasting (but not incompatible) positions on the standard's value for engineering quality software. In Part 1, Oskarsson is primarily concerned with exegesis and application of ISO 9001 - how to interpret the production-oriented standard into the design-oriented world of software engineering and how to apply the standard in the creation of a certifiable quality system. In Part 2, Glass compares the stipulations of the ISO 9000-3 standard with the tools and techniques available for software quality assurance and repeatedly concludes that the ISO document, *qua* quality "procedure" (in the terminology of auditing), is deficient.

In his monograph, Oskarsson takes the conventional auditor's view that ISO 9001 stipulates the minimal *management* requirements for a quality system. He explains the context in which the ISO standards were developed, introduces some of the terminology of audits and certification, and then explains each of the 20 ISO 9001 elements in turn. To alleviate the tedium of the exegesis, Oskarsson uses anecdotes from his auditing experiences to illustrate the standard's requirements and to identify those policies and procedures that are particularly prone to non-conformance citations. He ends with a brief summary of the necessary components of a quality system and the steps along the way to certification.

Much of the current ISO literature is content to parrot the standard; Oskarsson interprets it using his TickIT expertise and auditing experience, examining fine distinctions and problematic cross-references that other works ignore. (E.g., what are the implications of the traceability and inspection requirements for testing policies?) Readers interested in detailed analysis of what ISO 9001 says and doesn't say would be better served by struggling through Allan Sayle's *Meeting ISO 9000 in a TQM World*, and those concerned with how to get a certificate will find Charles Schmauch's *ISO 9000 For Software Developers* and Lawrence Wilson's *Eight-Step Process to Successful ISO 9000 Implementation* more useful. But as an *overview* of the *standard*, Oskarsson's discussion is unparalleled — just the sort of thing one would want to hand to one's engineering manager when s/he comes by and says "We've been told we have to go for ISO. What's that mean?"

Glass's Part 2 will disappoint readers who view ISO 9001 and ISO 9000-3 as an auditing standard, a verbose checklist for assessing project management capability. For one hundred and fifty pages he seems to belabour the obvious: the standards specify a quality management system, they don't guarantee quality software; they address the principles used to manage

software, they don't address the techniques required to develop quality software.

But Glass is simply taking seriously the ISO requirement that a quality system must be "effective" (ISO 10011, 3.1 & 4.1): the system must be practicable and it must result in a quality product. On his view, ISO 9000-3 describes the "necessary, but not sufficient" components of such a quality system. To predictably and consistently produce a quality product, software development teams must look beyond the ISO documents to the literature on software process. And Glass intends Part 2 to be a 'guideline for the application of defensible software process engineering to ISO 9000-3' in the same way ISO 9000-3 is a 'guideline for the application of 9001 to a software quality management system'.

Glass begins by explaining his classification system for development projects using size, application domain, criticality and innovation. He then presents the ISO 9000-3 stipulations for several development areas (Requirements, Design and Implementation, Testing and Validation, Replication/Delivery/Installation, and Maintenance), summarizes the tools and techniques available to software engineers for meeting those ISO requirements, and then maps those tools and techniques to the project types. Readers familiar with his *Building Quality Software* will notice (intentional) similarities in organization and content between the monograph and the book.

Glass warns in his Introduction that the material will be controversial, and he makes no apologies for the fact that we cannot "confirm or deny [his] taxonomy." As a rhetorical stance, this is expedient: Glass can cite whom he likes to support what he likes, ignore contrary evidence, wax polemic when it suits him, and still manage to fit a field survey into the limited space allotted him. Adherents to various schools will no doubt find some of this section annoying, and those familiar with the current work by Charles Engle and Michael Deck will find the Cleanroom discussion, shall we say, 'typical'.... Still, if one wanted to get an introduction to what tools are available to help developers put ISO principles into practice, Part 2 of *An ISO Approach To Building Quality Software* is the place to start. To be followed we presume, based on Glass's own statements, with a jaunt through *Building Quality Software*.

Introductory, Schizophrenic, and sometimes Oblique, this book provides an excellent summary of the intentions, limitations, and applications of those standards with which it shares some characteristics.

Reviewed by Dr. Robert Bruce Kelsey, Storage Management Software, DEC — robert.kelsey@cxo.mts.dec.co

A Quantitative Approach to Software Management: The ami Handbook

Kevin Pulford, Annie Kuntzmann-Combelles, Stephen Shirlaw

A Quantitative Approach to Software Management: The ami Handbook is written by Kevin Pulford, Annie Kuntzmann-Combelles, Stephen Shirlaw, and is published by Addison-Wesley 1996, softbound, ISBN 0-201-87746-5, 179 pages, \$24.69.

The Preface promises that this book will provide a "simple, straightforward, and understandable" software metrics program for those "who have yet to start on software measurement." The program proposed is based on the experiences of a European consortium of 9 organizations, but this is a handbook for newcomers to metrication and not an extended case study.

ami [sic] stands for "Application of Metrics In Industry" - and appropriately the first chapter is spent trying to persuade the reader that software metrics are indeed your friend. Chapter two summarizes the Goal-Question-Metric approach **ami** uses, and chapters 3 through 6 describe step-by-step how to implement an **ami** system. The reader learns how to create a metrics team, how to adjust a metrics program to organizational goals, how to write a metrics plan, etc. Since metric selection is organization- and goal-specific, the text itself offers few examples of what data one might want to collect in an **ami** project; however, Appendix 3 provides a "Basic Set" of metrics and Appendix 4 summarizes some sample implementations under the title "Case Studies." The book also contains a glossary, a short annotated suggested reading list (surprisingly short, given the literature available on metrics), and contact information for a user group and further training in this method.

The authors have set themselves a difficult task: how-to books for beginners are seldom both comprehensive and practicable and more often risk being simplistic and banal. With the actual metrics relegated to an appendix, the authors provide no comparative analysis of metrication programs (except for a curt dismissal of Kitchenham and Boehm in Chapter 7). Although the SEI Capability Maturity Model is frequently alluded to (and Appendix 1 contains SEI's 1987 checklist), the authors never explicitly correlate their metrics or implementation program to the requirements of the CMM. The section on "exploiting" the data mentions causal analysis but neither describes the techniques for root cause analysis nor provides any examples of conclusions one might draw from the defect typology metrics in the "Basic Set." Yet the authors are careful to share with their readers such maxims of Project Management 101 as "someone should be available for trouble-shooting in the early days of data collection," or why it is important to write a report summarizing the results of the findings. If the readership needs that level of hand-holding in project management, then one might expect the handbook to be as careful in its presentation of metric use and interpretation.

Inexpensive compared to some of its better crafted and more detailed peers, this book might be a good place to start if you know little about software quality management. But it is certainly no substitute for the works by Grady, Boehm, Gilb,