- S. 69.X2;

Books and Proceedings

A. General Literature

A.1 INTRODUCTORY AND SURVEY

See: 0402-0150 [D.2.2—*Visual Basic*]; 0402-0153 [D.4.0—*Linux*]; 0402-0167 [I.7.2—*XML*]

C. Computer Systems Organization

C.0 GENERAL

See: 0402-0168 [I.7.4]

System Architectures

See also: 0402-0168 [I.7.4]

BORGER, E.; AND STARK, ROBERT F. 0402-0141

Abstract state machines: a method for high-level system design and analysis.

Springer-Verlag, Secaucus, NJ, 2003, 420 pp., ISBN 3540007024.

One of the well-known issues incidental to systems engineering and software development today is the relative lack of sophistication of the level of training many students receive in appreciating the nitty-gritty of real-world projects. This can be contrasted with the impressive depth of theoretical understanding, engendered in these same students, in classical curricula in the computing sciences. In recent years, this tendency has, fortunately, been subjected to vigorous efforts at correction.

This book addresses the theory of abstract state machines (ASM), a salutary effort to impart rigor to the entire span of systems development, from requirements analysis to implementation, testing, and maintenance. The existing standard practices of unified modeling language (UML), though impressive when compared with the chaos they replaced, are often limited, namely, by the cumbersome, yet limited graphical notation of UML.

The basis of the method is the use of abstraction and stepwise refinement, well-known syntactic concepts, in a semantic domain. This is possible because the method rests on ASMs, which are, at the basic level, finite sets of transitions rules that map or transform abstract states to abstract states. (This is, in some ways, similar to Dijkstra's well known guarded command language). This formulation allows for a greater mathematical rigor, and consequent ease of proofs, than is possible under UML. (Standard concepts from mathematical logic, such as completeness and compactness, are proved with respect to the ASM model).

Borger and Stark do an admirable job of documenting and extending a method for bridging the considerable gap between theoretical system models, which often only allow for toy systems to be modeled and require proofs to be done only by hand, and real-life systems and practices. While the ASM method is still young (though the authors valiantly strive to illustrate it with many real-life examples), it is no doubt a stride in the right direction.

The authors have provided an accompanying CD, with carefully created lecture slides, and have also presented student problems throughout the book. This considerably eases the burden on a prospective instructor who is considering adopting this book as a course text. Considering the breadth of the material covered, however, it seems unlikely that it can be presented entirely in one semester, even in a graduate class (which is probably the only place for material of this depth). For the same reason, it is also unlikely that there will be many students able to absorb all of it. Practitioners may be unfamiliar with the mathematics, and computer science students may be unfamiliar with the practical issues involved.

—*Shrisha Rao*, Cedar Rapids, IA

GENERAL TERMS: DESIGN

KLIR, GEORGE J.; AND ELIAS, DOUG 0402-0142

Architecture of systems problem solving.

Da Capo Press, Inc., New York, NY, 2002, 349 pp., ISBN 0306473577.

One of the major characteristics of science in the second half of the twentieth century was the emergence of a number of related intellectual areas, such as cybernetics, general systems research, information theory, control theory, mathematical systems theory, decision theory, operations research, and artificial intelligence. All of those areas, whose appearance and development are strongly correlated with the origins and advances of computer technology, have one thing in common: they deal with such systems problems in which informational, relational, or structural aspects predominate, whereas the kinds of entities that form the system are considerably less significant. It has increasingly been recognized that it is useful to view these interrelated developments as parts of a larger field of inquiry, usually referred to as systems science. A course that covers systems fundamentals is now offered not only in systems science, information science, or systems engineering programs, but in many programs in other disciplines as well. This book could serve as a text for a first-year graduate or upper-division undergraduate course covering the fundamentals of systems problem solving.

A unique feature of this book is that the concepts, problems, and methods are introduced within the context of an architectural formulation of an expert system, referred to as the general system's problem solver (GSPS). The GSPS architecture, which is developed throughout the book, facilitates a framework that is conducive to a coherent, comprehensive, and pragmatic coverage of systems fundamentals.