Book Reviews

E. F. Wolstenholme, S. Henderson and A. Gavine

The Evaluation of Management Information Systems: A Dynamic and Holistic Approach

Wiley, Chichester, 1993, pp. iv+242, ISBN 0-471-93090-3

This book has two characteristics that makes it rather unusual. First, it presents a close connection of two radically different methodologies and approaches: management information systems (MIS) and system dynamics (SD). MIS methodology includes various computer-assisted methods that enable control of information flow and analysis of information with the goal to present it to managers in such a way to enable proper management of systems resources. SD, on the other hand, is a method that enables quasi-continuous modelling and simulation of complex dynamic systems with feedback.

Secondly, the book is written by authors coming from both academic and professional organizations. The first author, Professor Eric Wolstenholme, is a well-know academician whose previous textbook *System Enquiry: A System Dynamics Approach* (Wiley, 1990) is one of the best and richest texts written in the SD field so far. Simon Henderson and Allan Gavine are, on the other hand, specialists working in the Defence Research Agency in UK. Such mixture of authors has led to a proposal of problem solution which is both theoretically grounded and evaluated on real complex problems, a combination which occurs rather rarely.

A central problem tackled by the book is a MIS evaluation problem. Although MIS is one of the computer methodologies that had attracted extremely high investments, the field of MIS evaluation is not very advanced: MIS assessment is actually not included in the most MIS development methodologies. And when it is included, it typically supports a low-level function evaluation. Such evaluation is, however, not able to tackle the strategic role of MIS tending to cover strategic needs of the organization.

This book demonstrates how SD brought novel perspectives to MIS validation. It enables development of the SD model of an organization even before MIS is implemented, and only after that MIS is included in the organization model. The model is a strategic one, and takes into account MIS effects on the organization processes. This enables validation of MIS on a strategic level, taking into account performance of the organization as a whole.

The first chapter of the book presents an overview of MIS, in which key features of MIS are described as well as problems with it. Chapter 2 presents a review of the methodology of MIS development and evaluation. The authors point out the tendency of MIS to concentrate on solving details, which too often leads to evaluation on a rather low level. Instead of only assessing the ability of MIS to process information, MIS should be assessed according to how it affects the organization which is using it. Chapter 3 presents the overview of SD, and describes qualitative and quantitative aspects of SD as well as simulation with SD.

Chapter 4 describes the evaluation approach called BISEM (Bradford Information System Evaluation Methodology) proposed by the authors for more appropriate MIS evaluation. BISEM tends to assess strategic aspects of MIS through its holistic effects on the whole organization rather than on its parts. The basic idea of BISEM approach is to build a model of the organization capable to assess the effect of managerial policies on overall performance of the organization. In the first modelling stage a strategic SD model of the organization is built, while in the second stage the model incorporates expected changes in its physical resources and information flows appearing as a consequence of MIS installation. The same scenarios and performance measures are used in both modelling stages.

Chapters 5–7 present two comprehensive military case studies at different stages of their life cycles. The first case study evaluates the impact of MIS on a military logistic organization. The SD model was built using the DYSMAP/386 simulation language. The proposed MIS was a distributed database system serving corps, divisions and brigade levels. The BISEM approach was capable to suggest improved operational procedures and policies for stocking ammunition. The second stage evaluated the effects of implementing MIS on a military battlefield operation. Modelling of this rather unstructured problem was done using the STELLA simulation language. The SD model enabled understanding the dynamic of the situation modelled, and thus helped in evaluating the proposed MIS.

Finally, Chapter 8 presents an evaluation of the BISEM methodology itself, and gives some guidance for its application.

This book presents a novel approach to MIS evaluation. It offers a strategic evaluation approach and enables formulation of an organizational model including its feedback processes and dynamics of its behaviour. This was possible due to a happy marriage of SD and MIS approaches.

The book has an outstanding balance of theoretical background and comprehensive case studies which fully demonstrate the characteristics and potential of the new MIS evaluation approach. It is well written and full of relevant information. Because of using two radically different methodologies in concordance, this book requires a considerable effort for comprehension. The book is a valuable tool both for practitioners and students in information systems and management oriented disciplines.

Vlatko Čerić

University of Zagreb
Faculty of Economics
Zagreb, Croatia

Grady Booch

Object-Oriented Analysis and Design with Applications, 2nd Edition

The Benjamin/Cummings Publishing Co., Inc., Redwood City, CA, 1994, pp. x+589 (ISBN 0–8053–5340–2)

Grady Booch has been an object-oriented analysis and design advocate for some time. He's written books such as *Software Engineering with Ada, Software Components with Ada*, and *Object-Oriented Design (OOD) with Applications*. The first two are Object-Based (OB), the third is primarily Object-Oriented (OO), but all use OB and OO notations and methodologies. Over the past several years many OB and OO methodologies have been proposed. Even though G. Booch appears as a pioneer OO methodologist, there are still diverse opinions and definitions of many terms within this dynamic topic. His current notation is often referred to as simply the "Booch" method. The first edition of *Object-Oriented Design with Application* traced the path of OO design methodology into the mainstream of industrial-strength software development. In his second edition, G. Booch draws upon the earlier results to offer a more unified notation through numerous examples, now implemented exclusively in C++.

The author claims that this book provides practical guidance to the construction of OO software systems, in particular those that fully satisfy requirements and which are delivered on time, within the given budget. This approach, rather than trying to prove conceptual theorems, ensues a more pragmatical course. The book is divided into three major sections (concepts, the method, and applications), augmented with an appendix on OO programming languages, the glossary, classified bibliography, and the index.

The first section (Concepts) embodies four chapters, and examines the complexity of software, and how this complexity manifests itself. A way of bringing an order to this chaos is by the invention of common abstractions and mechanisms of similarity. To exhibit an organised complexity, the author introduces in the first chapter "the canonical form of a complex system". This is an orthogonal view to classes and objects. The canonical form gives the most clear conception of a system architecture by identifying relations among classes as an "is_a" hierarchical structure, and relations among grouped objects as a hierarchical "part_of" structure. The view takes into account that the class structure and the object structure are not completely independent, rather, each object represents a specific instance of some class. This view shows an inherent redundancy of a system. To overcome the fundamental limitations of the human capacity for dealing with complexity, the author suggests an object-oriented (instead of an algorithmic top-down), decomposition of a system in question, resulting with an "object model" abstraction.

The second chapter presents an evolution of the object model. The author chronologically follows the generations of programming languages, ending with the topology of OB and OO programming languages. Because the object model derives from many (disparate) sources, a clarification of terminology is offered. These parts of the book are often cited as "Booch conceptions". The focal point in every concept definition embraces collections of objects, each of which represents an instance of some class. These classes are members of a hierarchy of classes, united via inheritance. With the introduction of elements of the object model, the author illustrates the concept of abstraction by utilising C++ language formalism (a very brief exposure to C++ is given in the appendix). The reader can follow the usual material on class derivation with private, protected and public members, and its instantiation in a concrete object. The given examples illustrate the process of encapsulation, separating the contractual interface from the abstraction and its implementation. In addition, the concepts of modularity, hierarchical ranking, and typing are thoroughly discussed. With the unavoidable proliferation of multi-processing systems, the author emphasizes the concurrency and persistence property of objects.

Chapter 3 encompasses a detailed study of the nature of classes, objects and their relationship. To craft an object model, the software designer must develop certain skills that help in recognising what is and what isn't an object. The definition of the state of an object is given, augmented with C++ declarations. The meaning of the object behaviour, as a function of the state, is explained, together with various kinds of operations (some of them expressed as methods). Other properties of an object (identity and life span) are demonstrated. Since objects do collaborate with one another, there are various kinds of relationships among them. Two sets of object hierarchies are of particular interest: links and aggregation. Since the concepts of a class and object are tightly interwoven, the author goes into the details of class construction. I regard this chapter as a core of OO programming, since classes and objects are not independent static entities, but rather highly dynamically interrelated components of the object model. It takes a great skill and care to define generalisation/specialisation, whole/part, and associational relationship through hierarchy of classes, their interfaces and implementations. Single and multiple inheritance are discussed, with associated concepts like virtual functions, polymorphism, typing, procedure invoking (methods), aggregation, instantiation, and metaclass. Throughout the process of building classes and objects, one is faced with metrics of quality. The author gives some hints on choosing operations, relationships, and implementations during the design phase of high quality classes and objects.

The last chapter of the first section is devoted to knowledge ordering through classification. Clearly, there is no simple rule for identifying classes and objects. Since a project design is a compromise of many factors, there is no single solution either. The difficulty of classification lies in its subjective measure of complexity. Different observers will classify the same objects in different ways. Classical categorization will try to find common properties that define a category. Conceptual clustering first formulates the concept and then tries to classify the entities. For the situations where these approaches are inadequate, one may try the prototype theory, where an object belongs to a class if it resembles the prototype in a significant way. Finding classes and objects that form the vocabulary of the problem domain, is an OO analysis activity. The type of categorization (classical, conceptual, prototypical) highly influences the analysis. The Concepts section of the book is concluded with discussion on key abstractions that give boundaries to the problem domain.

The second section (The Method), encompassing Chapters 5, 6, 7, presents a methodology for the development of complex systems based on the object model. Chapter 5 introduces a graphic notation for the OO analysis and design. A notation is a vehicle for capturing the reasoning on the behaviour and architecture of a system. The presented notation is largely language independent, and exhibit multiple views. The differences between logical and physical models, as well as between static versus dynamic semantics, are explained. The existence of classes and their relationship is shown by the class diagram and the icon metaphor. All notions regarding the object model, and presented in the Concepts section, are mapped into a set of graphic icons. A basic notation set ("Booch Lite") is expanded to cover more advanced concepts like: parameterized classes, metaclasses, class utilities, nesting, export control, properties, physical containment, roles and keys, constraints, and attributed associations and notes. A graphical form is expanded by non-graphical specifications. The dynamics of the model is captured in state transition diagrams, expanded with the semantics of statecharts. However, even more additional views are introduced through interaction, module, and process diagrams.

Chapter 6 examines the incremental, iterative, process of OO analysis and design. Following the first principles of a successful project development, the author provides a description of micro and macro developing processes. The micro process captures the daily activities of a programmer. It involves the identification of classes and objects, together with their semantics and relationships. The milestones and measures are clearly stated. The implementation phase is performed as late as possible. The macro process serves as the controlling framework for the micro process, and characterizes the concern of the team technical management. This process follows traditional phases: conceptualization, analysis, design, evolution, maintenance. Again, through milestones and measures, validation criteria are established.

The pragmatics of OO development are examined in Chapter 7. No matter how sophisticated the development method is, one cannot ignore the practical aspects of a system design. In this context, the author urges for strong leadership, and stresses the following activities: risk management (micro process is inherently unstable), task planning (weekly team meetings), and walkthroughs (project reviews). The OO design approach calls for careful staffing of various development roles (project manager, analysts, reuse engineer, quality assurance (QA), integration manager, documenter, toolsmith, and system administrator). The interactive and incremental process of OO development results in releasing of multiple prototypes, hence release and code reuse management are needed. In complying with QA requirements, many traditional measures of quality are also applicable to OO systems. The documentation of a system's architecture should include: the high-level architecture, key abstractions and mechanisms, and scenarios that illustrate the behaviour of the system. OO development calls for a set of tools with a richer semantics than traditional (graphics-based editor, class browser, class librarian, incremental compiler, sophisticated debugger, etc.).

The third section, encircling 5 chapters, presents a collection of five examples from diverse problem domains, to illustrate how the object model scales up to a complex application. The development of a software cannot follow the cookbook style prescriptions, hence incremental approach is emphasized. The sets of applications embody data acquisition (weather monitoring station), foundation class library (domain-neutral collection of classes), client-server computing (inventory tracking), artificial intelligence (cryptoanalysis), and command and control (traffic management). The author doesn't present the complete implementation of any problem, rather, he focuses on the analysis and design phase, with enough material to indicate mapping from abstraction to implementation.

The book is packed with supplemental material. The appendix presents a brief overview of OO programming languages. A glossary of common terms and an extensive classified bibliography (structured into eleven domain sections) are included. Cover pages provide a quick reference to the notation and process of OO development method.

In summary, this is one of a few books that are fundamental to the complete OO analysis and design domain. I found it informational, inspiring, and interesting. It covers a lot of ground that could be arranged in many different ways. Current structure and contents would best match the interest of a computer professional with proficiency in programming (most likely in C, and some familiarity with C++), who would like to employ the "object model". Those programmers who are knowledgeable in C++, should have already mastered the basics of OO analysis and design. The presented material span vertically over several levels of granularity, covering topics not equally appealing and essential to both programmers and managers. I found my preferred pages in Chapter 3, presenting beautifully the three- dimensional universe of discourse (objects, classes, and their relationship). An unrestrained recommendation of the book for undergraduate or graduate courses would need an expanded chapter on C++ placed after Chapter 1, to serve as the common communication coherency. Finally, I consider the book as a necessity for every computer professional, and in particularly software developer. It should not be placed on his/hers bookshelf, but rather on the desk.

Nikola Bogunović

Electronics Laboratory
R. Bošković Institute
Zagreb, Croatia