Preface

The main objective of this book is to teach students and practitioners to analyze and design information systems (IS) using the functional and object oriented methodology (**FOOM**),[1] which combines the functional (process-oriented) approach with the object oriented (OO) approach.

The **functional approach** to IS development (sometimes also termed the *traditional approach*) was popular in the 1980s and 1990s of the 20th century. The development life cycle of this approach is based on the *waterfall* model (or its variations), according to which the IS development is essentially a sequential process, with the possibility of repetitions and iterations, thus making it look more like a spiral process. This approach views the IS as made of functions (processes), interconnected in a complex manner. The analysis of the IS focuses on discovering and defining the functions which the system needs to perform, and the flow of data to and from those functions. Two of the notable methodologies supporting this approach are structure system analysis (SSA) and system structure design (SSD) (DeMarco, 1978; Gane & Sarson, 1979; Yourdon, 1989). The SSA methodology is based on the use of data flow diagrams (DFDs) which describe the various functions of the system; the data stores in which the data are saved; the external entities which are the source of data input to the system and the destination of output information; and the dataflows which connect all of these components. According to the SSD methodology, the DFDs created in the analysis phase are transformed into a modular description of application programs, expressed by structure charts (Yourdon & Constantine, 1979).

With the development of the relational data model on the one hand, and conceptual data models on the other hand, more emphasis was given to the analysis and design of the system's database. The entity relationships (ER) model and its entity relationship diagram (ERD) (Chen, 1976) had become a common mean

for modeling the data and creating a conceptual data model, thus playing a complementary role to the role of DFDs in functional analysis. In the design phase, the ERD is mapped into a relational database schema. Simultaneously, the functional model is mapped, as mentioned previously, into structure charts of the application programs.[2]

One of the main problems with the traditional development methodologies such as SSA and SSD is the difficulty of transition from the analysis phase to the design phase. The transition is not smooth and causes difficulties because of the need to translate DFDs, which are a network structure, into structure charts, which are hierarchical. Another problem is the gap between the functional modeling aspect on one hand (leading to the creation of the application programs), and the data modeling aspect on the other hand (leading to the creation of the database schema of the application). In order to address these issues, Shoval developed the ADISSA methodology, which closes the gap between analysis and design phases and enables a smooth transition from the former to the latter phase (Shoval, 1988, 1991). The smooth transition from analysis to design is made possible by introducing a new construct in the DFDs: transactions. From a user's point of view, a transaction is a process performed by the IS to support a user's task, which is activated as a result of an event in the real (user) world. The transactions of the system are identifiable in the DFDs, and based on them it is possible to design all components of the system as a natural continuum of the analysis phase. The products of the design include, according to ADISSA, detailed descriptions of the application programs; a database schema; the user interfaces (menus trees) and the input/output screens; and reports.

The OO approach for IS development became popular in the early 1990s. The success of object oriented programming languages (OPL) motivated the penetration of the objects approach also to the area of analysis and design methodologies. In the last 15 years many OO analysis and design methodologies have evolved, and many techniques and diagram types which support these methodologies have been created, enabling the modeling of a system from various perspectives. Some examples of early OO methodologies can be found in Booch (1994), Coad and Yourdon (1990, 1991), Jacobson (1992), Martin and Odell (1993), Rumbaugh (1995), Rumbaugh, Blaha, Premerlani, Eddy, and Lorensen (1992), Shlaer and Mellor (1992), and Wirfs-Brock, Wilkerson, and Wiener (1990).[3]

The huge number of techniques and diagram types which evolved until the mid 1990s was a main driving force for proposing and adopting the unified modeling language (UML) as the "standard" for OO systems modeling.[4] UML is a collection of visual notation, that is, diagrammatic techniques. In spite of its great popularity and the advantage of having standardized techniques, UML has limitations. One of them is that UML includes many techniques with a certain

degree of overlapping between them. Some techniques enable developers to achieve the same goal in different ways;[5] but it is not always clear which technique should be preferred. Clearly, multiplicity of techniques and notations makes learning UML difficult and complicates the development process because of the need to move from one model/diagram type to another while keeping all models consistent (Siau & Qing, 2001).

Ever since the use of development methodologies for the creation of IS, software developers had to deal with two main problems: (1) the gap between process and data; and (2) the gap between analysis and design. The gap between process and data was manifested in traditional methodologies by the fact that DFDs emphasize process (functional) modeling, neglecting somewhat the modeling of data. A remedy to this gap came with the introduction of conceptual data models, notably the ER model, which complement DFDs as tools for defining the users' requirements. In early OO methodologies, the gap between process and data modeling was manifested by putting most of the emphasis on data (objects) modeling, while process modeling played a minor role only. To compensate for this deficiency, various techniques were added over time to deal with the functional aspects; but the result was, as said, a multitude of techniques with no well-defined interconnection among them.

The gap between analysis and design is expressed by the fact that the transition from analysis to design is not always clear and natural. In the analysis phase we define what the system ought to do as based on the users' needs, while in the design phase we deal with *how* the system will do that. Although it is clear that the design should be a direct continuation of the analysis, analysis and design methodologies have not always succeeded in doing so; some methodologies do not make it clear what "belongs" to analysis and what to design, or when does one phase end and the other begins, or (especially) what to do with the products of the analysis phase in the design phase. A solution to this void was offered, as said, by the ADISSA methodology, which defines and derives transactions from the DFDs and uses them as the basis for designing the application programs, the user interface, and the inputs and outputs of the system (Shoval, 1988, 1990, 1991). Some OO methodologies have tried to bridge the gap between the analysis and design by making the borders between the two phases "fuzzy," that is, treating the design as a refinement of analysis (e.g., Coad & Yourdon, 1990, 1991). Some OO methodologies do not specify what activity belongs to which phase, or where one phase ends and the other begins, or which of the possible techniques should be used in each of these phases. Yet, some methodologies view design as a refinement of analysis.

FOOM methodology (initially presented in Shoval & Kabeli, 2001) combines the functional and objects approaches and gives them an equal stand in both phases. In the analysis phase, the users' requirements are defined by creating two complementary models: a data model, expressed in the form of an *initial*

class diagram, and a functional model, expressed in the form of object oriented DFDs (OO-DFD). The two models are synchronized and used in the design phase in order to design the various components of the system. The design products include a complete class diagram; detailed descriptions of the class methods; user interfaces and input/output screens; and reports. The products of the design phase facilitate the construction (programming) of the system in an OO development environment.

Organization of This Book

This book is aimed for students of IS, computer science, management, and other fields which include a concentration on IS. It is intended to be a textbook of an advanced course (possibly in an undergraduate or graduate program), after the students have had at least one course in the fields of computer science or IS. In addition, it is recommended that the students take a course on databases (mainly being familiar with the relational model, data normalization, and the ER model). A course on IS analysis and design is not a prerequisite. However, familiarity with IS development methodologies, either from the functional or objects approach, is an advantage.

The book is divided into three learning sections, each consisting of three to five chapters. The first section deals mainly with the objects model and class diagrams; the second section deals with system analysis, and the third with system design. The material in each chapter includes many examples. At the end of each chapter there are review questions, which are meant to help the students in digesting and understanding the material. In some chapters there are also assignment questions which require solving practice-oriented problems. In addition to working on such assignments, it is recommended to include in the course a guided project, in which teams of two to three students perform the tasks of analysis and design of an IS for an organization in a real-world environment (as much as possible). If this is not possible an alternative is to perform a similar project on a case study that will be prepared for the students.[6]

The content of the book is as follows:

Section I (The Objects Model and Class Diagrams) provides a preview of the objects approach in general, and elaborates on the objects model and class diagrams in particular. The section consists of five chapters.

- **Chapter I (Introduction to the Objects Approach in Software)** presents the principles and characteristics of OO software in the objects approach, and common terms in OO programming.

- **Chapter II (The Objects Model and the Class Diagram)** describes in detail the components of the objects model (including objects, classes, attributes, relationships, and functions), and the class diagram which represents them.
- **Chapter III (Creating Class Diagrams)** discusses considerations and rules for identifying classes, attributes, relationships, and functions and presents case study examples (problems), that is, descriptions of users' data requirements, along with their class diagram solutions.
- **Chapter IV (Mapping Entity Relationship Diagrams to Class Diagrams)** explains why it might be preferred to first create an ERD and then map it to a class diagram. The chapter then describes the mapping rules and demonstrates the mapping process with several comprehensive examples.
- **Chapter V (Mapping Class Diagrams to Relational Schemas)** explains the need to map a class diagram to a relational schema. Most of the chapter is dedicated to presenting and demonstrating the mapping rules for converting a class diagram into a relational schema which is made of normalized relations. The mapping process is demonstrated with several comprehensive examples.

Section II (Functional and Object Oriented Analysis) starts with presenting a background for the development of UML, and then explains the motivation for the development of FOOM, which combines the objects and functional approaches. Most of the section is dedicated to learning how to analyze a system according to FOOM. The section consists of four chapters.

- **Chapter VI (Object Oriented Methodologies and UML)** reviews the evolution of OO methodologies and UML. Most of the chapter is dedicated to presenting and demonstrating the various techniques and diagrams which make up UML, and then it provides a detailed example of IS modeling using a UML-based methodology.
- **Chapter VII (Combining the Functional and Object Oriented Approaches: Introduction to FOOM)** starts by introducing the motivation for the development of a combined methodology. Then it presents the stages, substages, and products of FOOM.
- **Chapter VIII (Information Systems Analysis with FOOM)** elaborates on the activities and products of the analysis phase. The products of analysis include a data/objects model (in the form of an initial class diagram) and a functional model (in the form of hierarchical OO-DFDs. The two diagram types are synchronized in order to verify the correctness and completeness of the two models. The chapter presents various examples of diagrams of both types.

- **Chapter IX (Data Dictionary)** explains the roles of a data dictionary in the development of the IS and describes its components. The chapter presents a possible implementation of the data dictionary both with the relational and with the OO models.

Section III (Information System Design with FOOM) is about the design phase. The products of the design include: (1) a complete class diagram, containing (in addition to the data classes) the interface, inputs, outputs, and transactions class; (2) detailed descriptions of the various class methods; (3) the menus of the user interface; (4) the input and output screens and reports. The section includes three chapters.

- **Chapter X** (**Transactions and Their Top-Level Design)** describes what a transaction is and explains how to identify and extract the transactions from the OO-DFDs. Then it explains how to map transaction diagrams to top-level descriptions which details their components and process logic.
- **Chapter XI (Designing of the Man-Machine Interface: Menus, Inputs, and Outputs)** presents a method for the design of user interfaces—menus trees—for the entire system as well as for its subsystems. Then it describes how to design the inputs and outputs/reports of the systems.
- **Chapter XII (Detailed Design of the Transactions and Class Methods)** describes how to map top-level descriptions of transactions to detailed descriptions, and then how to "decompose" these detailed descriptions into various methods, which are attached to proper classes. Two equivalent techniques for the description of methods are provided: pseudo code and message charts. The chapter ends with a review on the products of the design phase, which serve as input to the system construction (programming) stage.

References

Avison, D., & Fitzgerald, G. (1988). *Information systems development: Methodologies*, *techniques and tools.* Oxford, UK: Blackwell.

Booch, G. (1994). *Object-oriented analysis and design with applications* (2nd ed.). Redwood City, CA: Benjamin/Cummings.

Chen, P. (1976). The entity-relationship model—Toward a unified view of data. *Transactions on Database Systems, 1*(1), 9-36.

Coad, P., & Yourdon, E. (1990). *Object oriented analysis.* Englewood Cliffs, NJ: Prentice Hall.

Coad, P., & Yourdon, E. (1991). *Object oriented design.* Englewood Cliffs, NJ: Prentice Hall.

DeMarco, T. (1978). *Structure analysis and system specification.* Englewood Cliffs, NJ: Prentice Hall.

Gane, C., & Sarson, T. (1979). *Structured systems analysis, tools and techniques*. Englewood Cliffs, NJ: Prentice Hall.

Jacobson, I. (1992). *Object-oriented software engineering: A use case driven approach.* New York: Addison Wesley.

Jayaratna, N. (1994). *Understanding and evaluating methodologies: NIMSAD, a systematic framework.* London: McGraw Hill.

Martin, J., & Odell, J. (1993). *Object-oriented analysis and design.* Englewood Cliffs, NJ: Prentice Hall.

Olle, W., Sol, H., & Verrijn-Stuart, A. (Eds.). (1986). *Information system design methodologies—Improving the practice.* North Holland: Elsevier Science Publishers; IFIP.

Rumbaugh, J. (1995). OMT: The dynamic model, the functional model, the object model. *Journal of Object-Oriented Programming*, 7(9), 6-12; *8*(1), 10-14; *7*(8): 21-27.

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., & Lorensen, W. (1992). *Object-oriented modeling and design.* Englewood Cliffs, NJ: Prentice Hall.

Shlaer, S., & Mellor, S. (1992). *Object lifecycles—Modeling the world in states*. Englewood Cliffs, NJ: Yourdon Press, Prentice Hall.

Shlaer, S., & Mellor, S. (1992). *Object-oriented systems analysis: Modeling the world in data*. Englewood Cliffs, NJ: Yourdon Press, Prentice Hall.

Shoval, P. (1988). ADISSA: Architectural design of information systems based on structured analysis. *Information System, 13*(2), 193-210.

Shoval, P. (1990). Functional design of a menu-tree interface within structured system development. *International Journal of Man-Machine Studies, 33,* 537-556.

Shoval, P. (1991). An integrated methodology for functional analysis, process design and database design. *Information Systems, 16*(1), 49-64.

Shoval, P. (1998). *Planning, analysis and design of information systems* (Vols. 1-3). Tel-Aviv, Israel: Open University Press. (Original work published)

Shoval, P., & Kabeli, J. (2001). FOOM: Functional and object-oriented analysis and design of information systems—An integrated methodology. *Journal of Database Management, 12*(1), 15-25.

Shoval, P., & Kabeli, J. (2005). Essentials of functional and Object-oriented methodology. In M. Khosrow-Pour (Ed.), *Encyclopedia of information science and technology* (pp. 1108-1115). Hershey, PA: Idea Group.

Siau, K., & Qing, C. (2001). Unified modeling language (UML)—A complexity analysis. *Journal of Database Management, 12*(1), 26-34.

Wieringa, R. (1998). A survey of structured and object-oriented software specification methods and techniques. *ACM Computing Surveys, 30*(4), 459-527.

Wirfs-Brock, R., Wilkerson, B., & Wiener, L. (1990). *Designing object-oriented software*. Englewood Cliffs, NJ: Prentice Hall.

Yourdon, E. (1989). *Modern structured analysis*. Englewood Cliffs, NJ: Prentice Hall.

Yourdon, E., & Constantine, L. (1979). *Structured design.* Englewood Cliffs, NJ: Prentice Hall.

Endnotes

1 FOOM was developed by Peretz Shoval, the author of this book, with the assistance of his doctoral student Judith Kabeli (Shoval & Kabeli, 2001, 2005). FOOM is based on and expands the ADISSA methodology, which Peretz Shoval has developed as a functional development methodology (Shoval, 1988, 1991, 1998).

2 More background and surveys of traditional IS development methodologies can be found, among others (in Avison and Fitzgerals (1988), Jayaratna (1995), Olle, Sol, and Verrijn-Stuart (1986), and Wieringa (1998)).

3 For a survey of both structured and early object-oriented methodologies see Wieringa (1998).

4 UML Web sites are detailed in the References.

5 For example, sequence diagrams and collaboration diagrams.

6 It is also recommended that the students will build (program) the system (or parts of it) in a proper development environment. This can be done in a follow-up course or exercise.