An Approach to Teaching Object-Oriented Analysis and Design

Pavle Bataveljic
Marian Eastwood
Heinz Seefried
School of Computing, Mathematical and Information Sciences
University of Brighton
Brighton, BN2 4GJ, UK
p.s.bataveljic@brighton.ac.uk m.e.eastwood@brighton.ac.uk h.g.seefried@brighton.ac.uk

ABSTRACT

This paper presents a syllabus that attempts to address the problem of teaching systems analysis and design in the changing world of today. In the first part of the paper, major issues and constraints that affect the development of a syllabus for this discipline are identified and analyzed. The second part of the paper focuses on the key points of a methodology constructed from traditional and object-oriented techniques, designed to satisfy the academic demands of the subject and reflect current practice, while providing students with a coherent and organized approach to systems analysis and design. Analysis of the outcomes and experience of implementing the syllabus provide the basis for conclusions and identification of possible areas for future research.

Keywords: Approaches for teaching SA&D, Curriculum design and implementation issues, Object-oriented analysis and design, Transition from structured approaches to object-oriented approaches.

1. INTRODUCTION

This paper presents an attempt to analyze and propose a solution to the problem that currently faces academic institutions in the development of a syllabus for the discipline of Systems Analysis and Design (SA&D). The problem can be defined as one of finding an adequate balance between an academic approach that should incorporate the underlying principles of the discipline and the needs and interests of two other major stakeholders in the education process, namely organizations and students.

The next section of the paper discusses major issues and constraints that define a context within which a solution to the problem can be sought. A possible solution to the described problem is given in Section 3. The syllabus and the tailored methodology within it represent an attempt to: (i) follow a commonly accepted object-oriented development method (hence the choice of a lightweight version of the Unified Software Development Process (USDP) (Jacobson, Booch and Rumbaugh, 1999)); (ii) incorporate proven traditional modeling techniques, namely Data Flow Modeling (DFM) and Entity-Relationship Modeling (ERM) to demonstrate alternative modeling views of a business system; (iii) demonstrate that it is beneficial to perform the initial steps of USDP by capturing information and knowledge of the business system in data flow models and E-R models. Section 4 describes some of the problems and dilemmas that we encountered. Conclusions and objectives for future research are given in the last section of the paper.

2. ISSUES IN TEACHING ANALYSIS AND DESIGN

The goal of this section is not to compare different methods and associated models but rather to discuss some of their strengths and weaknesses that are relevant to the adoption of an approach to teaching SA&D. An exhaustive survey and comparison of methods and models can be found in Wieringa (1998).

Both structured (traditional) methods that have dominated the eighties and the now prevalent object-oriented methods have the same goal – to analyze, design, and build complex software systems. However, the sets of models and associated modeling techniques of these approaches differ. Jackson (1995) observes that the models that we develop are descriptions of three subsets of phenomena: two that are specific to the problem domain (the "real world") and the solution domain (the "machine") respectively and a subset of shared phenomena (Figure 1).

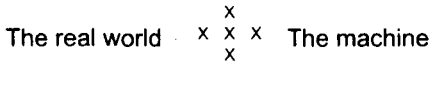


Figure 1. Phenomena of the real world and the machine (adapted from Jackson, 1995, p. 170)

Structured methods were developed under the influence of the structured programming paradigm, which led to top-down decomposition, the foundation of data flow models (Jackson, 1995). The data flow model is convenient for describing the problem domain but not the solution domain, which is described by a different set of models, most often structured charts (Whitten, Bentley and Dittman, 2001; Page-Jones, 1988). The transition from one set of models to the other has often been referred to as the "gap" between analysis and design. Bridging this gap is an inherent problem of structured methods.

In contrast, object-oriented methods use the same set of models to describe both the problem domain and the solution domain enabling a "seamless" transition from analysis to design. The "unity" of models was achieved by "migration" of object-oriented programming concepts to object-oriented design (OOD) and then to object-oriented analysis (OOA).

Some aspects of both approaches have been criticized. For example, Larman (1998) states that functional decomposition leads to procedural software architecture, while Ourusoff (2004) observes that OOA is solution rather than problem oriented. Jackson criticizes the top-down approach in general, arguing that the real world "hardly ever has a single hierarchical structure" (Jackson, 1995). He also criticizes OOA and observes "that the world outside the machine" is too complex to be described by a single phenomenon – the object.

Although it is apparent that object-oriented analysis and design (OOA&D) is currently the dominant approach Hay points out that "... disciplines that have been dominant for several decades have simply been ignored." (Hay, 2003). Applied in a controlled manner use of DFM in the analysis phase can enhance understanding of a complex real world system, define its boundaries, identify external events that initiate its responses, and enable partitioning of the system by applying (and limiting) decomposition to higher levels of abstraction.

The E-R model is probably one of the most important and influential data-oriented models developed during the 1980s. The strength of the E-R model is that it can be readily applied to describe phenomena in both the real world and the machine. The E-R model represents a semantically rich model that captures relevant characteristics of the system under consideration. At the same time, it is the design model for relational databases.

This leads to another important issue: the impedance mismatch between object-oriented technology and persistent data stored in a relational database. Ambler points out that "It is clear that object and relational technologies are in common use in most organizations and both are here to stay for quite a while ..." (Ambler, 2005). The data driven approach of traditional methods produces a logical E-R model that maps well into a physical design for a relational database. Conversely, OOA&D translates seamlessly into a programming language such as Java, while the class model does not map so readily onto a relational database.

The previous discussion suggests that E-R models and object-oriented models based on the Unified Modelling Language (UML), a standard supported by Object Management Group (OMG, 2006), should be included in the syllabus of SA&D. The issue of including some structured modeling concepts, namely DFM, remains open and decision rests on the judgment of academics that design the syllabus.

Another important issue in teaching SA&D concerns software process models. The traditional waterfall model defines the phases of the System Development Life Cycle (SDLC) as analysis, design, construction, implementation, and maintenance. While this definition of phases may be useful from an education point of view, in practice the waterfall model has been criticized for its inflexibility and predominantly replaced by iterative-incremental approaches such as Dynamic Systems Development Method (DSDM, 2003) and USDP (Jacobson, Booch and Rumbaugh, 1999)). It is important that a syllabus for SA&D includes a compatible set of modeling techniques, current development methods and software process models.

In the process of education, organizations are obviously major stakeholders. Driven by the competition, organizations need cost-benefit effective solutions. Object-oriented technologies offer more reliable and re-usable solutions, and combined with iterative-incremental system development these solutions can be achieved in a shorter period of time. On the other hand, organizations are not going to discard proven techniques and legacy systems that still satisfy their daily requirements and were built and documented during the traditional methods era. Ideally, new graduates should be proficient in both.

Other major stakeholders are the students. Throughout their education, they have to achieve two partly conflicting goals. On one hand, they need good understanding of and the ability to apply contemporary models and methods that will make them employable. On the other hand, graduates of today will still be working forty years from now. They have to develop a good understanding of the technology independent underlying principles of SA&D. This knowledge is an essential foundation upon which they can build understanding of the technologies that are yet to come.

Academic institutions are the stakeholders that have to balance the needs of industry and students. More often than not these objectives have to be achieved with limited resources, one of them being the time that can be devoted to the discipline of SA&D within a three year degree. Selection of an appropriate textbook to support the proposed curriculum is also an important constraint. Most textbooks that provide detailed presentation of models and modeling techniques tend to be confined to one particular approach and/or method (Bennett, McRobb and Farmer, 2006; Larman, 2002; Satzinger, Jackson and Burd, 2005). Textbooks that attempt to incorporate different methods, models and software process models, for example Avison and Fitzgerald (2006) by necessity tend to be cursory in their coverage of individual methods. Not surprisingly, empirical evidence shows that there is a discrepancy between the

recommended textbooks and the material actually delivered to students (Burns and Klashner, 2005).

In this section, we have presented the issues that we consider to be relevant in designing a SA&D syllabus. The approach we have taken and the elements of the syllabus we teach in an attempt to resolve these issues are given in the next section of the paper.

3. OUR METHODOLOGY

The computing program at the University of Brighton has a common level 1 for all computing students, which includes a single semester module on Requirements Analysis where basic modeling and requirements gathering techniques are introduced. Only students on the Business Information Systems (BIS), Internet Business Computing (IBC) and Business Software Development (BSD) degrees continue to study SA&D in the second year. This paper focuses on the level 2 analysis and design module for these students who also study related modules in Databases, and Web Application Development in the Microsoft .NET environment (Microsoft, 2006). In the final year, a variety of models and methods are introduced for comparison and evaluation.

3.1 Overview

The workflows and artifacts of USDP (Jacobson, Booch and Rumbaugh, 1999) form the basis of our approach. The main workflows identified in USDP are Requirements Capture, Analysis, Design, Implementation and Testing. We incorporated the first three as the core of the module, as illustrated in figure 2, with issues of Implementation and Testing being discussed towards the end of the year. We have deviated from the object-oriented paradigm of USDP with the retention of the traditional techniques of ERM and DFM in the requirements gathering phase.

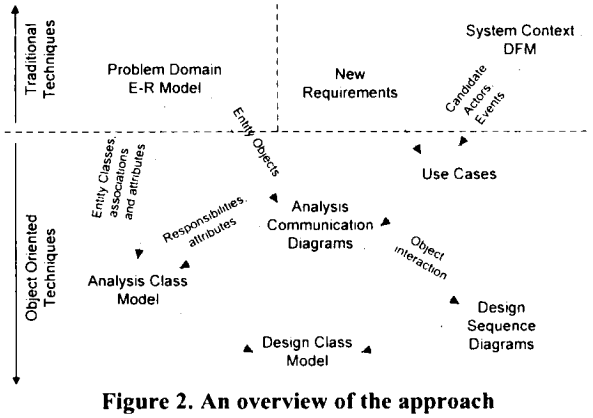


Figure 2. An overview of the approach

Figure 2 shows an overview of the methodology we use for teaching. The figure suggests a waterfall approach to analysis and design but the arrows are intended to represent the general flow of the analysis and design process, and there is likely to be considerable iteration between the artifacts. It should also be noted that there is no direct translation implied from one artifact to another; for example, the Analysis Communication Diagrams do not translate directly into the Design Sequence Diagrams because more detail is added in the transition.

The problem domain is modeled using an E-R model rather than a class diagram. The entities identified become candidates for the objects of the UML entity class stereotype required for the analysis use case realizations, specifically the analysis communication diagrams. Attributes and entity relationships identified in the problem domain also inform the Analysis Class Model. Although not shown in figure 2, the E-R model is developed into the design of the relational database. The database design model may also include persistent classes identified from the Design Class Model.

The data flow model enhances understanding of the context of the system and assists in identifying events. There may be a question as to whether current and/or required, logical and/or physical data flow models are most appropriate but whichever is chosen events can be identified that in turn suggest potential use cases that are the system's responses to these events. The external entities and locations/roles on the data flow model may suggest candidate actors for the required system. The candidate actors and use cases derived from the data flow model then inform the use case model along with any additional new requirements.

The approach then more closely follows USDP as described by Jacobson et al. (Jacobson, Booch and Rumbaugh, 1999). Use Case Realizations are initially represented by Analysis Communication Diagrams. The three UML class stereotypes of boundary, control and entity objects are introduced, and their representation in the communication diagrams facilitates the view of the three-tier architecture (Presentation layer, Business Layer and Data layer). The business logic for a use case is represented by the control object and is described in an activity diagram to ensure business rules are understood and captured correctly.

The Analysis Class Model is constructed from the Analysis Class Diagrams for individual use cases. The responsibilities of each class are identified from the object interactions shown on the analysis communication diagrams. The transition from analysis to design involves adding more detail to the classes already identified, and the identification of additional classes needed for construction in the chosen technology. For example, an analysis boundary class may become a number of interface classes in design, and an analysis entity class may become a data access class and persistent data.

3.2 Rationale

For a number of years Structured Systems Analysis and Design (SSADM) (CCTA, 1996), was used for teaching SA&D to Information System undergraduates. As a prescriptive method, it provided a development framework and well defined modeling techniques for analysis and design. Undoubtedly, the method was well engineered with clear guidelines as to when and how the modeling techniques should be applied. As previously discussed, modeling the real world was a strength of structured methods. However,

for students the transition from one phase of the SDLC to the next was not facilitated by the associated change of modeling techniques.

Although OOA&D has played a role in our final year for several years, the demise of structured methods and the ascendancy of object-oriented methods needed to be reflected in the emphasis of our syllabus over all three years. We therefore came to the decision to base the level 2 module on an object-oriented approach. The uniformity of modeling with UML would facilitate the transition between development phases specifically addressing the issue of moving from analysis to design described in the previous paragraph.

Not wishing to discard all the benefits that SSADM had brought to our old syllabus, we wanted to integrate an object-oriented method into our teaching rather than just presenting students with a 'tool box' of modeling techniques. Our review of current practice (and textbooks) suggested that various flavors of USDP are now commonly used in the IT industry. USDP is a "generic process framework that can be specialized ..." (Jacobson, Booch and Rumbaugh, 1999), so by tailoring a 'lightweight' version of USDP into a methodology to satisfy our teaching needs students could still experience the basic life cycle of development.

Our rationale for retaining DFM and ERM in the Requirements Capture phase requires more explanation.

We have already mentioned that we perceive a weakness of object-oriented modeling is in representing the 'real world'. Consequently, to help students understand the organizational context of a computerized information system, and therefore to clarify the boundary of the system, we have retained the structured technique of DFM in the requirements gathering phase of our approach. Decomposition is a necessary and powerful tool for managing the complexity of a system and a high level data flow diagram (a top level DFD and possibly level 2 DFDs, depending on the complexity of the system) suggests a logical partitioning of a complex domain into a set of smaller, more manageable domains (potential subsystems of the analyzed system). By limiting decomposition to higher levels of abstraction, design of procedural software is not implied. Finally, as mentioned earlier, data flow models facilitate identification of events and potential actors and consequently facilitate identification of use cases.

The design for an information system should consist of both a design of the object-oriented software and a design of the relational database. In the context of teaching, a number of OOA&D textbook authors, (Bennett, McRobb and Farmer, 2006; Satzinger, Jackson and Burd, 2005), resolve the problem by using a class model to represent the problem domain, refining it through analysis and design and then normalizing it into a relational model. However, Ambler (2006) demonstrates that some aspects of a class model can be mapped into more than one relational design, suggesting this may not be the best technique for designing a relational database. Conversely, Ambler (2006) also argues against using data models to drive object-oriented design, contesting that the class model should be derived during domain or analysis modeling and not based on a conceptual data model. This is a complex and well known issue and it is not the purpose of this paper to discuss it at length, however it does present a pedagogic dilemma: should the problem domain be represented using a class model or an E-R model? Our decision to use ERM was taken because we believe that at this level the semantics of the data model and those of the class model are equally appropriate for modeling the problem domain. However, the discipline of producing a normalized E-R model appears to assist students in understanding the data requirements of the system. In addition, the E-R model evolves into the database design while providing a basis for identifying the analysis entity classes. Hay provides some justification for our stance:

> "... data modeling during analysis (whether of the object or entity/relationship kind) is intended to do one thing: describe the things about which an organization wishes to collect data, along with the relationships among them." (Hay, 2003).

4. OBSERVATIONS

We have now used this approach to SA&D teaching for two years, and the following issues have arisen.

Within the constraints of a single module, there is effectively only time to work through one iteration of the USDP workflows, perhaps portraying a waterfall life cycle. However, the iterative and incremental nature of USDP is addressed in lectures.

The artifacts of the analysis phase clearly demonstrate some degree of design. Indeed Jacobsen, Booch and Rumbaugh consider one of the important aspects of the analysis model to be that it "... can be viewed as a first cut at a design model ..." (Jacobson, Booch and Rumbaugh, 1999). The introduction of boundary and control stereotypes into communication diagrams is surely making an early design decision to adopt a three-tier architecture. This is contrary to Hay's view that "the models developed during analysis must be technologically neutral" (Hay, 2003).

'First cut' design in the analysis model is also demonstrated by a simple example from a case study of a private library. The library has a number of business rules governing the eligibility of a member to borrow a book. The member must have current membership of the library and the maximum number of loans permitted must not be exceeded. In addition, the library will not allow members to borrow a book if they have an overdue loan that has been recalled to satisfy a reservation placed by another member. The use case that represents this functional requirement is 'Check a Member's Borrowing Status'. The normal scenario of this use case is represented in two different ways in the communication diagrams in figure 3. Both communication diagrams capture the business rules, but each solution represents a different allocation of responsibilities to the MemberStatusHandler and Member classes, and hence an early design decision.

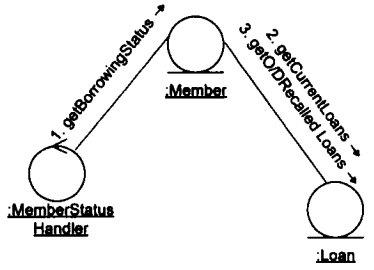


Figure 3a

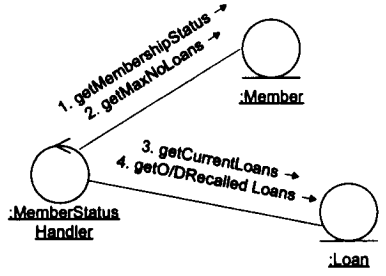


Figure 3b

Figure 3. Communication diagrams for the normal scenario of 'Check Member's Borrowing Status' use case

Although the distinction between a technology independent analysis model and a technology dependant design model has become obscured, this facilitates the transition from analysis to design for students. If we accept that some degree of design is inevitable during object-oriented analysis, and acknowledge the advantage to students, we are faced with a dilemma: should we adopt a purer object-oriented approach to distributing responsibilities among the objects (figure 3b), an early design decision that may have to be reversed when designing for a particular implementation technology; or should the control object be responsible for all the business rules in analysis (figure 3a), postponing the distribution of responsibilities until detailed design?

5. CONCLUSIONS

Our approach based on USDP with traditional techniques employed in the requirements capture phase has enabled us to retain most of the pedagogic benefits we previously gained from SSADM: a framework of system development; a well-defined set of modeling techniques; the underlying principles of the SA&D discipline, and an industry-recognized method. However, we are still grappling with one issue in particular.

We have previously discussed that SSADM clearly separated conceptual analysis from design, a differentiation that has become obscured with the object-oriented approach. While this initially seemed an advantage for students, the complex issues of design clearly impinge upon analysis at an early stage. We wish to further research where the boundary between analysis and design should lie, and/or how the analysis phase can be more independent of the technology.

6. REFERENCES

Ambler, S. W., (2005), "The Object-Relational Impedance Mismatch", http://www.agiledata.org/essays/impedanceMismatch.html Accessed 27/01/06.

Ambler, S. W., (2006), "Why Data Models shouldn't drive Object Models (and Vice Versa)", http://www.agiledata.org/essays/drivingForces.html Accessed 27/01/06.

Avison, D., Fitzgerald, G. (2006), Information Systems Development: Methodologies, Techniques and Tools, (4th ed.), McGraw-Hill.

Bennett, S., McRobb, S., Farmer, R. (2006), Object-Oriented Systems Analysis and Design using UML, (3rd ed.), McGraw-Hill.

Burns, T., Klashner, R. (2005), "A Cross-Collegiate Analysis of Software Development Course Content", SIGITE'05, October 20-22, 2005, Newark, New Jersey, USA.

CCTA (1996), Central Computing and Telecommunications Agency. SSADM4+ Reference Manual v4.3, London: The Stationary Office 1996. ISBN 0-11-330844-2.

DSDM (2003), Dynamic Systems Development Consortium http://www.dsdm.org, Accessed 31/01/06.

Hay, D. (2003), Requirements Analysis, Prentice Hall PTR, Upper Saddle River, New Jersey.

Jackson, M. (1995), Software requirements & Specifications: a lexicon of practice, principles and prejudices, Addison-Wesley; ACM Press.

Jacobson, I., Booch, G., and Rumbaugh, J. (1999), The Unified Software Development Process, Addison-Wesley.

Larman, C. (1998), Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, Prentice-Hall.

Larman, C. (2002), Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, (2nd ed.) Prentice-Hall.

Microsoft.NET (2006), http://www.microsoft.com/net/default.mspx Accessed 15/02/06.

OMG (2006), UML Resource Page, http://www.uml.org Accessed 31/01/06.

Ourusoff, N. (2004), "Reinvigorating the Software Engineering Curriculum with Jackson's Methods and Ideas", The SIGCSE Bulletin, Vol. 36, No 2, June 2004.

Page-Jones, M. (1988), Practical Guide to Structured System Design, (2nd ed.), Prentice-Hall.

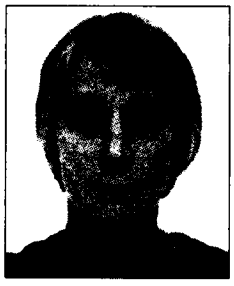
Satzinger, J., Jackson, R., and Burd, S. (2005), Object-Oriented Analysis and Design with the Unified Process, Thomson Course Technology.

Whitten, J., Bentley, L., Dittman, K. (2001), System Analysis and Design Methods, (5th ed.), McGraw-Hill.

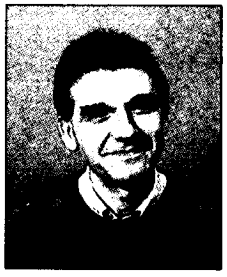
Wieringa, R. (1998), "A Survey of Structured and Object-Oriented Software Specification Methods and Techniques", ACM Computing Surveys, Vol. 30, No. 4, December 1998.

AUTHOR BIOGRAPHIES

Pavle S. Bataveljic is a Senior Lecturer in the School of Computing, Mathematical and Information Sciences, University of Brighton, UK . He is the award leader for the degrees of Business Software Development and Internet Business Computing. His research interests include requirement analysis and modelling techniques. Before joining the University of Brighton in 2000 he was a lecturer at University of Belgrade, Serbia. He received his MSc degree in Applied Systems Theory from University of Belgrade. He also studied computer science at the University of Virginia, USA supported by the Fulbright Program.



Marian E. Eastwood is currently a Senior Lecturer in the School of Computing, Mathematical and Information Sciences at the University of Brighton, where she teaches Analysis and Design. She is the course leader for the postgraduate programme in computing. Before joining the University in 2001 she worked in the IT industry as an analyst and programmer. Her research interests are Analysis and Design, and Information System Development Methodologies. She received her MSc degree in Information Systems Development from the University of Brighton.



Heinz G. Seefried is currently a Senior Lecturer in the School of Computing, Mathematical and Information Sciences at the University of Brighton. He is also the coordinator of the Information Systems Development subject group. His research interests include Information System Development Methodologies. Prior to joining Brighton University, where he teaches Databases, and Analysis and Design, he was an independent consultant in Information Management. He received his BA degree in Business Studies from the University of Stuttgart, Germany.

Information Systems & Computing
Academic Professionals

STATEMENT OF PEER REVIEW INTEGRITY

All papers published in the Journal of Information Systems Education have undergone rigorous peer review. This includes an initial editor screening and double-blind refereeing by three or more expert referees.

ISSN 1055-3096