Experiences of teaching UML within the information systems curriculum

Huseyin Dagdeviren
Radmila Juric
Patrick Lees

School of Electronics and Computer Science

Experiences of Teaching UML within the Information Systems Curriculum

Huseyin Dagdeviren, Radmila Juric, Patrick Lees
Cavendish School of Computer Science, Department of Information Systems,
University of Westminster, 115 New Cavendish Street, London W1W 6UW
dagdevh@wmin.ac.uk, juricr@wmin.ac.uk, leesp@wmin.ac.uk

Abstract. *The Unified Modelling Language (UML) has been a standard modelling language for the development of software intensive systems since 2000. As a consequence, the Information Systems (IS) curriculum, at the Cavendish School of Computer Science, University of Westminster in London, had UML teaching incorporated two years ago. We have encouraged the introduction to and use of UML in modules that replaced traditional approaches to IS development. In this paper we report on experiences of using UML within the two modules of our undergraduate curriculum, delivered by the IS department. The first module is taught in the second year, i.e. at level 5, and delivers requirements analysis with UML. The second module uses the UML for modelling and designing distributed business applications and is taught in the final year, at level 6. In both modules it is assumed that an introduction to modelling in IS, with the syntax and semantics of a selection of UML modelling elements and diagrams, has been done earlier. We single out some problems and give a rationale for changes in the next academic year.*

Keywords. UML, Modelling in Information Systems, Requirements Analysis, Distributed Business Applications, IS Curriculum.

1. Introduction

The UML specification became the software industry's standard modelling language in 2000 [10]. The advantages of having a common modelling language for visualising, specifying, constructing and documenting the artefacts of software intensive systems [4], has attracted software practitioners, tool vendors, academics and researchers. All of them have had their own 'say' on this *shared graphical language* [13], which ranges from the systems developer's practical needs for UML, to the consternation of researchers about UML [6] and only partially successful UML adoption by tool vendors [8]. Regardless of UML prospects within software engineering (SE) and IS communities, the UML is a standard modelling language adopted by practitioners and researchers at a rate that has surprised even the most optimistic Object Management Group (OMG) expectations [15]. When the Cavendish School of Computer Science, at the University of Westminster in London UK, was considering changes in the IS curriculum, as part of our IS course review program, we made the use of UML explicit [9]. We decided not to train students as UML experts but through our program students will attempt to secure their modelling competence, whilst obtaining a medium through which concepts of "object-oriented", "relational", "process" etc. can be more readily assimilated and communicated. One intention was to build on students' increased willingness to express ideas and opinions about the world and enhance the rigour and precision of their expression [9], [18].

Two years after the adoption of the new IS curriculum, we report on our experiences of delivering and using the UML within our undergraduate program. The purpose of this paper is to

- reflect on two different modules where the UML plays a major role, influences the weekly teaching schedule and affects students' assessment;
- document our experiences in a systematic way that can be shared by our colleagues and serve as a basis for our future curriculum reviews.

We give our experiences of teaching and using UML in two undergraduate modules: Requirements Analysis (RA), which is given in section 2, and Distributed Business Applications (DBA), which is given in section 3. Details of module syllabi are available at [18]. We report briefly on problems related to the UML adoption throughout these two modules and reflect on possible changes in future academic years. In both modules it is assumed that an introduction to modelling in IS, with the syntax and semantics of a selection of UML modelling elements and diagrams has been done in the Modelling in Information Systems (MinIS) module at level 4 (year 1).

This paper does not try to justify our decision to make UML teaching explicit. We leave any evaluation of our students' modeling competence and their ability to express ideas and opinions about the world with rigour and precision, for future works, when the course matures and all modules will have run.

2. Requirements Analysis

2.1. The role of the module, its aims and teaching/assessment patterns

The RA module aims to teach students the knowledge and skills required to perform requirements analysis, from their elicitation to specification. It has replaced the Systems Analysis module from our old programs, where traditional systems analysis and design teaching [5], [19] was essential in the IS curriculum. The RA module's prerequisite MinIS focuses on the syntax and semantics of a selection of UML modelling elements and diagrams. The RA module is (a) a core module for students studying for the BSc degrees in IS and IS with Business Management and (b) an optional module for Computing, SE and Information Product Design students, all at level 5. The module is taught by a combination of lectures (24 hours), supervised tutorial sessions (24 hours), where students can study in groups, and self-study exercises, which, together with coursework preparation, require 96 hours of students' independent work.

The module's assessment includes two pieces of coursework, for 50% of the total marks, and an exam. A group coursework requires students to elicit and model requirements from a given case study using the UML. Students' ability to express their approach to the requirements analysis, adoption of appropriate terminology and modelling principles are reflected on and assessed in the self-assessment part of the coursework report. The closed book exam has more of a summative role that allows the assessment of student's retention and understanding of RA topics drawn from the entire module.

2.2. Teaching RA with UML

The UML plays a very important part in the delivery of the RA module, which is based on student's modelling skills obtained in its prerequisite MinIS. The RA module does not only refine the UML diagrams introduced in MinIS, but also continues with some other diagrams relevant to the RA topics. Consequently, there is an overlapping between the two modules, particularly when *Use Case diagrams* and *Class diagrams* are taught and used. However, the purpose of the MinIS module is to introduce UML diagrams, which could be used in any context, such as conducting RA or designing IS.

The RA module strays slightly from the requirement engineering (RE) teaching known from [11], which can be found within many SE degrees. Consequently, our essential text book [3] is the main source of the teaching material used in both lectures and tutorials for the RA module. It adopts a process that is largely consistent with Unified Systems Development Process (USDP) [7]. We have adopted their approach to the IS development, which has assisted us in placing the UML into the context of a requirements analysis *'process'*. The following outlines how the UML was used within the RA module:

(a) To document the results of the requirements elicitation process using *use case diagrams* and *scenarios*;
(b) To analyse object interactions within each use case separately by using *collaboration diagrams*;
(c) To produce an *analysis class diagram* by integrating all collaboration diagrams;
(d) To perform further analysis by
    i using *sequence diagrams* for object interaction
    ii using *state diagrams* for objects that have complex state dependent behaviour.

Subsequently, the analysis class diagram is refined by reflecting the effects of the further analysis and enhanced by incorporating other modelling elements such as *generalization* and *aggregation relationships*.

We have excluded some UML diagrams such as *component* or *deployment diagrams,* which would fit better within design modules and do not necessarily bond with RA. However, we needed at least 6 lectures and 10 tutorial sessions, i.e. almost 70% of teaching time, to cover (a) to (d) above.

2.3. Problems encountered

As implied above, teaching the UML modelling elements and diagrams constitutes a significant part of the module delivery, i.e. the UML dominates the module. The RA topics covered within the remaining time were

(a) an overview of IS development practices
(b) structured and object-oriented IS developments with their underlying principles and assumptions
(c) problems in IS development and the role of requirements
(d) requirements capture, user involvement and overview of user analysis.

Consequently, we have had two undesirable effects:

i. *We shifted the module's emphasis from requirements analysis to modelling.* While some students were able to perceive that the modelling is a vehicle to capture and analyse requirements rather than being the actual goal of the module, many students viewed it as a continuation of the prerequisite MinIS module and interpreted the purposes and roles of these two modules similarly.
ii. The domination of UML has had a knock-on effect on the module's assessments. Assessment questions were supposed to be centred either on requirements, modelling with UML or both. While it is difficult to label all questions strictly as a *modelling with UML question* or a '*requirement question*' it has become clear that the creation of UML diagrams will constitute a large proportion of the assessment, due to its dominating role within lecture and tutorial sessions. Consequently, more than 60% of the exam marks were allocated to creating UML diagrams and assessing student's ability to interpret the syntax and semantics of UML modelling elements correctly, whereas less than 40% of the marks were related to requirements analysis itself. Furthermore, we could play down the role of UML in an individual coursework, which was a phase test with questions based on basic concept, and terminology within the module, but the modelling tasks within the group coursework uniformly emphasised the UML. *We failed to ensure that not only the modelling skills are assessed but also issues of eliciting and specifying requirements stayed in the core of our assessment strategy.* Shouldn't the latter represent the essence of the RA module delivery?

2.4. Possible solutions

Eliminating some of UML diagrams that contribute the least or relate little to RA can alleviate the problem of the UML domination in the module. This means that we free more lecture and tutorial time for delivering the requirements elicitation and specification topics. We propose to eliminate S*tate diagrams* because:

(a) They can be used to describe the behaviour of control and boundary classes in later stages of a design process, as opposed to their minor role in RA for assessing the behaviour of complex entity classes.
(b) Their tasks of assessing the behaviour of complex entity classes for refining class diagrams may to some extent be done with sequence diagrams.
(c) Their syntax and semantics require a considerable amount of teaching time, which does not make them cost-effective in this module.
(d) Consequently, they would make a good candidate for the design module that follows the RA module.

However, the implications of this change will be in tailoring the currently used '*process*' [3] or employing our own '*process*' that would deal with a strictly prescribed set of UML modelling elements and diagrams suitable for the RA module delivery.

In the light of the above, the assessment will be easy to change. The syntax and semantics of UML modelling elements and diagrams should be secondary within the module's assessment. This can be achieved by forming questions carefully. For example, the *"What are the different types of relationships that can be used in a class diagram?"* question is more related to UML and its role within modelling in general, while the question *"How do the generalisation and aggregation relationships in a class diagram* *contribute to the development of a resilient system?"* is more appropriate for the RA assessment. These kinds of questions will be correctly interpreted by all students as long as the module is not perceived as a continuation of modelling exercises practices in the prerequisite MinIS.

3. Distributed Business Applications

3.1. The role of the module, its aims and teaching/assessment patterns

The DBA module introduces students to traditional and emerging distributed applications: from their business aspects/domain and strategies for development, to their impact on organisational and cultural changes in terms of the rising dominance of the Internet. The problem of modelling and designing such applications is addressed through the issues of heterogeneity and distribution and through various frameworks/architectural models and middleware technologies that deploy distributed applications. Consequently, one of the module's aims is to teach students how to compose a suitable requirement specification which leads towards an architectural model of a distributed application, and the analysis of a suitable technology for the application deployment. The module is (a) a core module for students studying for the BSc degree in Internet Computing and (b) an optional module for IS, Computing and SE students, all at level 6. As it is delivered by the IS department, the module's syllabus and its assessment mirror a specific approach in modelling and designing, where tasks short of final software implementations are not unusual.

The module is taught by a combination of lectures (12 hours) and tutorials (24 hours) and students are expected to undertake their own research to deliver their coursework and facilitate tutorials. The assessment is 100% coursework-based. Students submit three pieces of work:
(i) an essay based on individual research on distributed computing platforms available in industry, (ii) group coursework on *modelling* and deploying an example of distributed business application and (iii) a presentation of a topic in the areas of e-commerce applications, Computer Supported Collaborative Work, component based technologies and web services.

For the *modelling task* students develop their own requirement specification from a given case study, and derive an architectural model which addresses a distribution of data and processes within a given problem domain. This requires adequate modelling skills and the communication of solutions. Clear diagrammatic solutions are required in order to pass this coursework. It is essential to use UML throughout: from conducting requirement analysis and specification, to delivering a component based architectural solution and design which outlines the choice of technology for the deployment and implementation of the application's components.

3.2. Using UML within a particular 'process'

The UML is essential for the module delivery. It is used everywhere: from discussions on diagrams generated from tutorial case studies to a coursework which includes modelling tasks. Tutorials are supported by a text book which advocates the design of distributed applications with UML and the Java platform [1]. This choice of text book also dictates a *process* for the DBA design and implementation. We have not blindly adopted the book's *process* in the module, but we discussed and tailored it into the following steps:

(a) Generate a *use case model* with detailed scenarios, exceptions/commonalities and generalisations;
(b) Derive *sequence diagrams* from each use case with *boundary* and *control* objects as the first two to be revealed, followed by any number of *entity* or any other objects;
(c) Generate *class diagrams* with all classes involved - including entity classes - and group them in order to prepare a system for technology selection;
(d) Create a component based architectural model (we used UML *package modelling elements*) which also shows the technology needed for the implementations of the component's functionality (we used *package dependencies* with technology components);
(e) Decide on the *application design*, i.e. give choices of sessions and entity beans within the Enterprise JavaBeans (EJB) platform [1].

Steps (a) to (e) are used in tutorials when following certain sections from the text book and they also mirror the tasks given in the coursework on *modelling*.

3.3. Problems encountered

Problems were polarised around the following two:

Problem 1: The heterogeneous class in the DBA module consisted of students from all four of our undergraduate degrees: IS, SE, Internet Computing, Computing degrees. Consequently students may have

i. various modelling skills – some students only have experiences of modelling techniques in structured analysis and design methodologies as in [5], [19] and some are familiar with the extraction of abstraction for object oriented systems development [3], [14];
ii. different levels of UML adoption: from UML experts to students who have never been taught UML or who were self-taught;
iii. various perceptions of the purpose and role of UML in the development of IS and distributed application in particular – some students still perceive the UML as a methodology, hence the issue of having a standardised modelling language without having a standardised *process* which prescribes how and where to use modelling diagrams represents a problem;
iv. diverse experiences of applying the UML modelling elements in various application domains - we have students with experience of using UML in SE applications with a strong emphasis on design and code generation [14], [16] which is a contrast to the high level of abstraction needed when employing the same modelling elements and diagrams within requirements and software architecture models [1] ;
v. different views of *processes* where UML modelling elements and diagrams are put in a certain perspective - students are puzzled by questions such as: which diagrams should I use and why? In which order? Is there any process? Is this our process? If there are some processes available where I can use UML, how do I choose which one to use?

Problem 2: The module is 100% coursework-based, which has implied poorer lecture attendance and little communication with the lecturer compared with modules that involve formal exams. Consequently, the assessment tasks required a certain level of lecturer involvement: the coursework on modelling and designing a DBA included a 'check-point' where students were supposed to have their initial modelling solution approved by a lecturer. There were other strong reasons for such a prerequisite:

- Initial system requirements' models given through UML diagrams determine the success of all other coursework tasks for the design and deployment of DBA (see steps (a)-(e) from 3.2) and
- The variability of the UML modelling skills acquired outside this module could have impaired students' performance in this module.

Consultations with a lecturer, in order to pass a 'check-point', took approximately 1-2 hours per group, which culminated in 20 contact-hours outside scheduled lecture and tutorial sessions. This might become unfeasible when the number of students taking the module increases from the current 41.

3.4. Possible Solutions

Problem 1 will be difficult to remedy as long as we allow the DBA module to be a free choice module for all our students. However, it could have helped if our teaching had had a stronger emphasis throughout *all our degrees* on

a. modelling skills with UML,
b. clear perception on the role and purpose of UML when developing software systems,
c. understanding the current trends in methodologies, i.e. '*process*es', that may use UML, and which range from USDP [7] to XP [2] and agile developments [17].

Consequently, we would not have had to exercise any check-points within students' assessments on the scale currently needed.

4. Conclusion

Our experiences of including UML within the IS curriculum may generate many questions. We chose to outline the following two:

In spite of having UML successfully incorporated across the IS curriculum, we feel that introducing students to UML and teaching them how to apply it is not balanced. Students obviously need more time in the first year to familiarise themselves with modelling in general and with the adoption of a specific modelling language, if we want them to apply both later in their studies. We might also think about the synchronisation of UML deployment across all our courses in the school, if we want to keep modules such as DBA a free choice module for all our degrees.

We also share various concerns from practitioners regarding the UML role within a process that supports IS development. It is difficult to choose one, when they range from heavy-weight USDP [7] and light-weight agile development [17] to various selections of UML diagrams generated in order to follow a 'process' that many potential text books advocate [12],[3],[1]. For decades we have taught IS students how essential methodologies and their techniques for IS development are. It appears that not having 'a methodology', where you can place UML diagrams in a certain order and for a certain purpose, might be a problem when teaching UML.

All these issues will be considered when designing weekly teaching programs for all modules with UML involvement in September 2004.

5. Acknowledgements

We are grateful to our colleaque Ms Jackie Croft for her proof-reading and constructive suggestions.

References

[1] Arrington C T, Rayhan S H. Enterprise Java with UML. John Wiley and Sons: 2003.

[2] Beck K. Extreme Programming Explained: Embrace Change. Reading, MA: Addison Wesley; 1999.

[3] Bennett S, McRobb S, Farmer R. Object Oriented Systems Analysis and Design Using UML. Maidenhead: McGraw Hill; 2002.

[4] Booch G, Rumbaugh J, Jacobson I. The Unified Modelling Language User Guide. Redwood City, CA: Addison Wesley Longman Publishing Co.; 1999.

[5] DeMarco T. Structured Analysis and System Specification. Upper Saddle River, NJ: Prentice-Hall Company; 1979.

[6] Dori D. Why Significant UML Change is Unlikely. In CACM, November 2002, Volume 45, No 11, pp 82-85.

[7] Jacobson I, Booch G, Rumbaugh J. The Unified Software Development Process. Reading, MA: Addison Wesley; ACM Press, 1999.

[8] Juric R, Kuljis J. Building an Evaluation Instrument for OO CASE Tool Assessment for UML Support. In R.H. Sprague (ed.) Proceedings of the 32nd Hawaii International Conference on Systems Sciences; 1999 Jan; Hawaii, USA, IEEE Computer Society Press.

[9] Juric R, Lees P, Photos T. Experiences of Revalidating the Undergraduate and Postgraduate Courses Within the Information Systems Curricula at University of Westminster, UK. In Journal of Computing and Information Technology 2003; 11(4), 2003, pp. 243-252.

[10] Kobryn C. UML 2001: A Standardisation Odyssey. In CACM, October 1999, Volume 42, No 10, pp 29-37.

[11] Kotonoya G, Sommerville I. Requirements Engineering: Processes and Techniques, John Wiley and Sons; 1998.

[12] Maciaszek L A. Requirements Analysis and Systems Design: Developing Information Systems with UML. Harlow: Addison Wesley; 2001.

[13] Miller J. What UML should be. In CACM, November 2002, Volume 45, No 11, pp. 70-72.

[14] Priestley M. Practical Object-Oriented Design with UML. McGraw-Hill; 2000.

[15] Selic B, Ramackers G, Kobryn C. Evolution, not Revolution. In CACM, November 2002, Volume 45, No 11, pp. 70-73.

[16] Sommerville I. Software Engineering, Reading, MA: Addison Wesley; 2001.

[17] The Official Agile Modelling Site. http://www.agilemodeling.com/ [15 Apr 2004].

[18] University of Westminster, Cavendish School of Computer Science, Department of Information Systems, BSc (Hons) Information Systems Course Handbook, 2002.

[19] Weaver P L, Lambrou N, Walkley M. Practical SSADM Version 4+. London: Pitman Publishing; 1998.