UML – a tutorial

UML – a tutorial.................................................................1
1 The history of object-oriented analysis and design methods..........................2
2 Software engineering .............................................................................7
  2.1 Responsibility-driven versus data-driven approaches ......................12
  2.2 Translational versus elaborational approaches ...............................13
3 Object-oriented analysis and design using UML .....................................13
  3.1 Types and classes ............................................................................13
  3.1 Object structures .............................................................................18
  3.2 Using use cases to discover types .................................................28
  3.3 Invariants and rulesets .................................................................35
  3.4 Invariants and encapsulation .......................................................43
  3.5 State models .................................................................................45
  3.6 Moving to component design ......................................................48
  3.8 The design process .......................................................................55
  3.9 Documenting models .....................................................................57
  3.10 Real-time extensions .................................................................58
4 Identifying objects ............................................................................60
  4.2 Task analysis ................................................................................65
  4.3 Kelly grids ....................................................................................68
5 CASE tools ........................................................................................71
6 Patterns, architecture and decoupled design ........................................73
  6.1 Design patterns for decoupling .....................................................87
7 Designing components .......................................................................97
  7.1 Components for flexibility .........................................................100
  7.2 Large-scale connectors ...............................................................101
  7.3 Mapping the business model to the implementation .....................103
8 Notation summary ............................................................................105
  8.1 Object modelling symbols ..........................................................105
  8.2 Action (use case) modelling symbols ..........................................110
  8.3 Sequence and collaboration diagram symbols ................................110
  8.4 State modelling symbols .............................................................112
  8.5 Action or activity diagram symbols ..............................................114
  8.6 Implementation and component modelling symbols .....................115
  8.7 Collaborations and patterns ........................................................116
  8.8 Real-time notation: ports and connectors .....................................116
9 Further reading ................................................................................117
10 Exercises ......................................................................................118

UML is the international standard notation for object-oriented analysis and design. It is defined by the Object Management Group (www.omg.org) and is currently at

Contacts: Ian Graham or Alan Wills +44-161-225 3240; clive@trireme.com
Copyright © MMI – Trireme International Ltd. All rights reserved
1 The history of object-oriented analysis and design methods

The development of computer science as a whole proceeded from an initial concern with programming alone, through increasing interest in design, to concern with analysis methods only latterly. Reflecting this perhaps, interest in object-orientation also began, historically, with language developments. It was only in the 1980s that object-oriented design methods emerged. Object-oriented analysis methods emerged during the 1990s.

Apart from a few fairly obscure AI applications, up until the 1980s object-orientation was largely associated with the development of graphical user interfaces (GUIs) and few other applications became widely known. Up to this period not a word had been mentioned about analysis or design for object-oriented systems. In the 1980s Grady Booch published a paper on how to design for Ada but gave it the prophetic title: Object-Oriented Design. Booch was able to extend his ideas to a genuinely object-oriented design method by 1991 in his book with the same title, revised in 1993 (Booch, 1994) [sic].

With the 1990s came both increased pressures on business and the availability of cheaper and much more powerful computers. This led to a ripening of the field and to a range of applications beyond GUIs and AI. Distributed, open computing became both possible and important and object technology was the basis of much
development, especially with the appearance of n-tier client-server systems and the web, although relational databases played and continue to play an important rôle. The new applications and better hardware meant that mainstream organizations adopted object-oriented programming and now wanted proper attention paid to object-oriented design and (next) analysis. Concern shifted from design to analysis from the start of the 1990s. An object-oriented approach to requirements engineering had to wait even longer.

The first book with the title *Object-Oriented Systems Analysis* was produced by Shlaer and Mellor in 1988. Like Booch’s original paper it did not present a genuinely object-oriented method, but concentrated entirely on the exposition of extended entity-relationship models, based on an essentially relational view of the problem and ignoring the behavioural aspects of objects. Shlaer and Mellor published a second volume in 1992 that argued that behaviour should be modelled using conventional state-transition diagrams and laid the basis of a genuinely OO, albeit data-driven, approach that was to remain influential through its idea of ‘translational’ modelling, which we will discuss. In the meanwhile, Peter Coad had incorporated behavioural ideas into a simple but object-oriented method (Coad and Yourdon, 1990; 1991). Coad’s method was immediately popular because of its simplicity and Booch’s because of its direct and comprehensive support for the features of C++, the most popular object-oriented programming language of the period in the commercial world. This was followed by an explosion of interest in and publication on object-oriented analysis and design. Apart from those already mentioned, among the most significant were OMT (Rumbaugh et al., 1991), Martin-Odell (1992), OOSE (Jacobson et al., 1992) and RDD (Wirfs-Brock et al., 1990). OMT was another data-driven method rooted as it was in relational database design, but it quickly became the dominant approach precisely because what most programmers were forced to do at that time was to write C++ programs that talked to relational databases.

OMT (Rumbaugh et al., 1991) copied Coad’s approach of adding operations to entity-type descriptions to make class models but used a different notation from all the previous methods. Not only was OMT thoroughly data-driven but it separated processes from data by using data flow diagrams separately from the class diagrams. However, it emphasized what Coad had only hinted at and Shlaer and Mellor were yet to publish: the use of state-transition diagrams to describe the life cycles of instances. It also made a few remarks about the micro development process and offered very useful advice on how to connect object-oriented programs with relational databases. Just as Booch had become popular with C++ programmers because of its ability to model the semantic constructs of that language precisely, so OMT became popular with developers for whom C++ and a relational database were the primary tools.

Two of OMT’s chief creators, Blaha and Premerlani (1998), confirm this with the words: ‘The OMT object model is essentially an extended Entity-Relationship approach’ (p.10). They go on to say, in their presentation of the second-generation
End of ebook preview
Download the full PDF tutorial from the link below:

Click Here