URL: http://www.elsevier.nl/locate/entcs/volume72.html 3 pages

Case Study: Simulation of a LAN [1]

Dirk Janssens and Serge Demeyer [2,3]

Department of Mathematics and Computer Science
Universitaire Instelling Antwerpen
Antwerp, Belgium

Tom Mens [4]

Programming Technology Lab
Vrije Universiteit Brussel
Brussels, Belgium

1 Introduction

The aim of this contribution is to present a case study - the simulation of a Local Area Network - that has been used at Vrije Universiteit Brussel (VUB) and the University of Bern to illustrate various aspects of the evolution of Object-Oriented programs. We believe that it can be useful as a common test case for the further investigation of this subject. The basic version of the case, as presented here, is rather simple, but the core features of OO languages are used in it, and it seems obvious that it can be extended to more complex versions in a smooth way.

2 Domain information

The case is mainly intended to illustrate modeling support for software evolution, and in particular refactoring. It might also be relevant for the domain of network design. From a programming point of view, the domain is object-oriented programming. The interplay between incremental development and refactoring has been explored, using this example, in a number of lectures at VUB and at the University of Bern.

The case consists of an initial version of the simulation program, and a series of changes in the specification that require it to evolve. Straightforward

[1] Partially supported by the EC TMR Network SegraVis (Syntactic and Semantic Integration of Visual Modeling Techniques) through Universitaire Instelling Antwerpen.
[2] Email: Dirk.Janssens@ua.ac.be
[3] Email: Serge.Demeyer@ua.ac.be
[4] Email: Tom.Mens@vub.ac.be

adaptations of the program undermine the original design, and hence refactorings are used to improve the structure. As diagrammatic representations have become a useful element in OO design, it seems natural to extend this kind of visualization to the topic of refactoring. As demonstrated in our contribution to ICGT [1], at least some of the refactorings used in the case can be adequately described by graph grammar productions.

The case requires knowledge of a standard OO programming language, such as JAVA or Smalltalk. It uses only basic OO programming concepts such as late binding, dynamic invocation, inheritance, super calls. Concerning semantic concepts, the case study focuses on behaviour-preserving properties, which are crucial for refactoring. The relevant meta-level objectives from SegraVis are in the first place M4 (modularity, refinement and transformation of models), but it may also serve as an example for M5 (integration of visual modeling techniques) and M6 (integrated meta CASE support). The development of suitable graph representations of OO programs may be relevant for M1 (static and dynamic semantics). The sample code is available at http://prog.vub.ac.be/FFSE/Cases/Toy%20Examles/LAN/

3 The case study

A simulation of a Local Area Network in a standard OO language is considered. So far, Java and Smalltalk have been used. Starting from a simple network (a ring of nodes where a single type of message is transmitted), more complex versions are developed in an incremental style: extra functionality is stepwise added, but for each step a straightforward adaptation leads to a program that can be improved by refactoring.

In the initial version of the program, the simulated LAN is very simple, and its simulation is straightforward: the network consists of nodes, organized in a ring. There are two types of nodes (workstations and printservers). Packets are sent around the ring, and these packets must travel from the workstation where they originate to their destination (Fig. 1). The simulation pro-

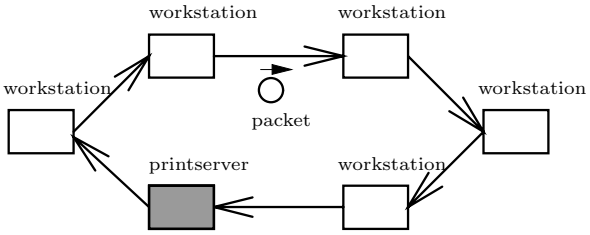

Fig. 1. A simple LAN.

gram consists of the obvious four classes (`node`, `workstation`, `printserver` and `packet`), where `workstation` and `printserver` are subclasses of `node` (Fig. 2). The evolution of the program corresponds to a series of four consecutive requirements at the level of the LAN, which must be incorporated into

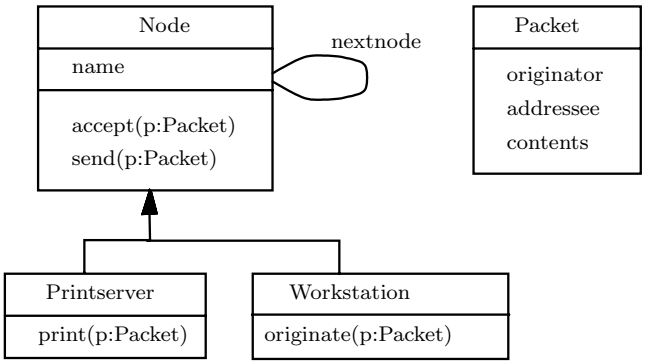

Fig. 2. The simulation program.

the simulation program. For each of them, a number of refactorings are used.

(i) Provide a logging facility: each time a packet is sent from a node, a message identifying the node and the packet is printed. (Refactorings: Extract-Method, Encapsulate-Field)

(ii) Extend the LAN so that documents of different types (Postscript, ASCII) can be printed on a corresponding printer. (Refactorings: Extract-Superclass, Pull-up-Field, Pull-up-Method, Extract-Method, Form-Template-Method)

(iii) Printservers need to register the author and the title of the document that is being printed. (Refactorings: Extract-Superclass, Replace-Conditional-with-Polymorphism)

(iv) Extend the LAN so that messages can be broadcast. (Refactorings: Extract-Method, Move-Method, Inline-Method)

4 Closing remark

We propose a LAN simulation as a case study for the evolution of Object Oriented programs. In its current form, the case study is too small to serve as a full-scale experimental basis. However, there are various obvious ways in which the test case can be improved and extended. To mention but a few, it would be desirable to use more relevant parts of the UML to express the specification of the program as well as the new requirements to which it has to be adapted. Also, if one restricts attention to the refactoring aspect, one may try to find changes that require, or at least enable one to illustrate, a wider range of refactoring operations.

References

[1] T. Mens, S. Demeyer and D. Janssens, Formalizing Behaviour Preserving Program Transformations, technical report 02-03, Dept of Math. and Computer Science, University of Antwerp - UIA, 2002. Also Proceedings of ICGT 2002.