Extending the Eiffel library
for data structures and algorithms: EiffelBase

Project Plan

Master project

Project period: April 5th - October 4th, 2004

Student: Olivier Jeger <oljeger@student.ethz.ch>

Status: 9th semester

Supervisor: Karine Arnout

1 Project description

1.1 Overview

EiffelBase is intended to be a general, high-quality library covering the basic needs of everyday programming [1]. It offers data structures to organize data and algorithms which can be applied to those data structures. The library design dates back to 1985 in its first form; that it has stood the test of time and is still used in so many new applications is testimony to its solidity.

The goal of this project is to extend EiffelBase in a number of areas not yet covered. In particular, the task is to add library classes for graphs, topological sort and B trees. The classes should comply to the overall design of EiffelBase, thus be of high quality and fit well into the existing class hierarchy.

1.2 Scope of the work

The thesis is divided into several subprojects. Each of them consists in the implementation and integration of one algorithm or data structure into the EiffelBase library. The first step will be to integrate some library classes for which the corresponding design work has already been done (but may need to be refined), and which are very important in practice:

Graphs

Bernd Schoeller has designed some classes [2] which seem quite in line with the general spirit of EiffelBase. Here the idea is to possibly refine these classes and integrate them into EiffelBase.

Topological sort

There is a detailed discussion in chapter of the textboook *Touch of Class* [3]. The goal is to write the classes implementing this design, and a set of test classes. The process might uncover errors in the design or bring up ideas for improvement and generalization. It may also be interesting to compare the final design and implementation with the approach taken in the Gobo Eiffel Data Structure Library [4].

The second step will target the following areas:

- New forms of trees, in particular B trees

- Ideas gleaned from standard textbooks on algorithms, e.g. Widmayer [5], Sedgewick [6].

It is planned that in total 5 concepts are realized in the 6 months project duration: Graphs, topological sort, B trees and two additional algorithms or data structures which are to be defined at a later stage. However, quality goes over quantity and it may be possible that not all concepts can be implemented in this limited time.

1.3 Intended results

EiffelBase extension

The main goal is the integration of the graph classes, topological sort and B trees. Depending on the needed effort for those concepts, there may be no more time to focus on the additional algorithms or data structures.

Thesis report

- Documention of each project step, especially the design considerations

- API of the new library classes

- User guide with instructions on how to use the classes

2 Background material

2.1 Reading list

EiffelBase library, design principles: [1]
Graphs: [2]
Topological sort: [3], [4]
B Trees: [5], [6]
Further algorithms and/or datastructures: [5], [6]

3 Project management

3.1 Objectives and priorities

The goal is to implement high quality library classes. The main priority is on the design phase. The classes must fit well into the existing class hierarchy, the used names should conform to the overall EiffelBase style. Obeying the "Open-Closed principle", the class interface must not be modified after publication, since it would possibly destroy existing work built on the library. A well thought-out class design is therefore indispensable.

According to the principle "quality over quantity", the focus is rather on a few good components rather than on many which are unsatisfying.

3.2 Criteria for success

Implementation

The design of the library classes must be made with special care and should conform to the general spirit of EiffelBase. The classes should also fit well in the existing class hierarchy. Intensive tests should uncover possible implementation errors.

Thesis report

Each algorithm or data structure involves a whole subproject with design phase, implementation and test cases. Especially the design phase is very important. All considerations that lead to the final design will be written in the thesis report.

3.3 Method of work

The duration of each subproject which covers one algorithm or data structure is about 4 weeks. The algorithms and data structures are taken from well-accepted books [5], [6]. For the implementation, *ISE EiffelStudio 5.4* will be used. The documentation will be written with L_YX.

3.4 Quality management

The development process is repetitive for each subproject. Experience from one cycle may be used to improve the next subproject.

The results of one subproject can be reused in subsequent project steps, e.g. graphs for topological sort. Possible weakness may be uncovered that way.

Validation steps

- Weekly report to the supervisor, both source code and documentation

- Periodical meetings with the supervisor with discussion of the design aspects

Documentation

- Weekly report about project progress

- Detailed documentation at the end of each subproject cycle

4 Plan with milestones

4.1 Project steps

- **Graphs**

  - Analyze and/or adapt class design
  - Implementation
  - Test cases
  - *Milestone M1: End of graphs subproject. Friday, May 12th*

- Topological Sort

  - Elaborate class design
  - Implementation
  - Test cases
  - *Milestone M2: End of topological sort subproject. Friday, June 18th*

- B Trees

  - Integrate class(es) into existing hierarchy
  - Implementation
  - Test cases
  - *Milestone M3: End of B trees subproject. Friday, July 16th*

- Additional Algorithm #1

  - Elaborate class design
  - Implementation
  - Test cases
  - *Milestone M4: End of subproject. Friday, August 13th*

- Additional Alorithm #2

  - Elaborate class design
  - Implementation
  - Test cases
  - *Milestone M5: End of subproject. Friday, September 10th*

- Documentation

4.2 Deadline

Project start: Monday, April 5th, 2004

Project end: Monday, October 4th, 2004

4.3 Tentative schedule

Task	04-05 / 15	04-12 / 16	04-19 / 17	04-26 / 18	05-03 / 19	05-10 / 20	05-17 / 21	05-24 / 22	05-31 / 23	06-07 / 24	06-14 / 25	06-21 / 26	06-28 / 27	07-05 / 28	07-12 / 29	07-19 / 30	07-26 / 31	08-02 / 32	08-09 / 33	08-16 / 34	08-23 / 35	08-30 / 36	09-06 / 37	09-13 / 38	09-20 / 39	09-27 / 41
Startup																										
Finish project plan	■																									
Read "Reusable Software"		■																								
Setup environment		■																								
Graphs																										
Analyze Bernd's class design			■																							
Enhance / Redesign classes				■																						
Implementation					■																					
Test cases						■																				
Documentation						M1																				
Topological Sort																										
Read "Touch of class" chapter							■																			
Elaborate class design								■																		
Implementation									■	■																
Test cases											■															
Documentation																										
Comparison with GoboSoft solution											M2															
B Trees																										
Read "Algorithms & data structures"												■														
Class design													■													
Implementation														■												
Test cases															■											
Documentation															M3											
Additional Algorithm #1																										
Read text book																■										
Class design																	■									
Implementation																		■								
Test cases																			■							
Documentation																			M4							
Additional Algorithm #2																										
Read text book																				■						
Class design																					■					
Implementation																						■				
Test cases																							■			
Documentation																							M5			
Documentation																										
Write Thesis report						■					■				■				■				■	■	■	■
Write user guide																								■	■	

References

[1] Bertrand Meyer: *Reusable Software: The Base Object-Oriented Component Libraries.*
Prentice Hall, 1994.

[2] Bernd Schoeller
Graph classes.

[3] Bertrand Meyer: *Touch of Class: Learning how to program well with Object Technology, Design by Contract, and steps to Software Engineering (in preparation).*
Chapter 19: An elegant algorithm family: Topological Sort
Available from http://se.inf.ethz.ch/touch

[4] Eric Bezault, *Gobo Eiffel Data Structure Library.*
Available from http://www.gobosoft.com

[5] Peter Widmayer, *Algorithmen und Datenstrukturen,*
Spektrum Akademischer Verlag, 1996.

[6] Robert Sedgewick, *Algorithms,*
Addison-Wesley, 1998.