

This Lecture

- surveys (9:00–9:20)
 - need volunteer to take back to DCS Office after lecture
- Theme for Assignments 2 & 3:
 - concepts the U/EBoat Simulation System have illustrated
 - lecturers reflections of Eiffel
 - Tournament results will be posted early next week!
- Study Guide:
 - details of the exam paper
 - study Hints

Review: via the U/EBoat Simulation System

- specification: technical English descriptions
- high-level design: overall module decomposition (crucial!)
 - ideally, complex classes should have a high-level design via semi-formal methods
 - ◆ eg. a OOD 'informal strategy' and structure graphs
 - ◆ lack of structure in *OCEAN* class reflects this was not done ...
 - inheritance for code re-use between kinds of boats
- low-level design for main classes:
 - mainly via **require**, **ensure** and **invariant** clauses
 - useful, but complex and exposed subtleties
 - ◆ eg. *real_equal*, *BOAT_DRIVE.move*
 - ◆ could not be proven without an implementation!

U/EBoat Simulation System (II)

- implementation:
 - mainly bottom-up: generic classes, boat classes, boat library, ocean, and boat drivers
 - ◆ drivers could use a 'spiral' approach
 - standard Eiffel coding style used, with heavy use of assertion checking
 - ◆ were there enough comments?
 - was there enough support for general, low-level calculations
 - ◆ eg. for vectors and angles? an *ANGLE* class needed ?
 - accuracy problems: *DOUBLE* would have been better than *REAL* !
- testing:
 - unit testing (mainly specification testing), for all but *OCEAN* & boat drivers
 - ◆ had separate test programs; seems reliable once past all test cases
 - boat drivers could only be tested at the system-level
 - ◆ interactive nature of boats made this harder
 - ◆ had at least a mode to create an initial situation of interest
- version control (CVS): only somewhat useful at this scale

Some Reflections on use of Eiffel for the Theme

- previous system used Modula-2
 - combined procedural (eg. libraries, function parameters) and OO (via 'opaque types') styles
 - a pure OO language made this more difficult
 - 'portng' meant a complete rewrite! (even had some impact on the design)
 - a *BOAT_ALARM* parameter used to imitate a Boolean function parameter in M-2;
 - but *BOAT_ALARM* had to be implemented separately from *UDRIVER*
 - ◆ a little awkward, esp. accumulating history-related state (eg. count depth charges released)
 - functional-type constructors (*TWOVEC.add_v*) proved very useful
- however, Eiffel versions of codes were generally shorter

Reflections on Eiffel (II)

- inheritance via **deferred** classes worked very well for boats; permitted generic boat libraries to be written
 - could not exploit this for *EDRIVER* due to lack of co-routines
 - *EDRIVER* had to be written in a 'state-based' fashion: as short, but more tricky
- assertion checking support was nice (but you can do it in Modula-2 or C)
- seems to be as secure as Modula-2
- in summary, only a little better than Modula-2;
a functional language, eg. Miranda, would be my choice!
- your comments?

Details of the Exam Paper (well, some of them)

Format & degree of difficulty much like the COMP1110 exams:

- 15 minutes reading; 3 hours writing
- 4 questions (each is split into parts).
- Questions total 80 marks; each worth 20 marks
- Answer all questions, on the exam paper
- No aids!

note: differences from COMP1110 papers:

- predicate calculus questions are N/A, or give an Eiffel expression instead of on in predicate calculus
- ignore Q1(a) [2002,2001]; Q1(a-c,g,h) [2000]

How Do I Make Sure I Pass? **STUDY!**

Revise everything in the DA/SE modules:

- Lecture notes from all lectures.
- Laboratory exercises from all labs.
- Tutorial exercises from all tuts.
- Assignment work.
- Relevant Chapters of the textbook.

Collect marked Assignment 3 from box outside CSIT N330 (expect by Jun 23)

See Peter or Carol (should be available most days before exam) with:

- specific questions
- but not on content of the exam
- and not between the exam and notification of results

Do the exam!

- <http://timetable.anu.edu.au/>