

1 CSB31/EngCS21: Program Design & Construction

- 0.5 point, semester 1, 2nd-year computing unit
- involves the further development of design, implementation and testing skills using systematic approaches
- modest-sized, multi-module software systems in Module 2
- emphasis on design strategies using procedural and data abstraction, using ‘information hiding’
- features:
 - Abstract Data Types (ADTs) for higher-level modeling of data objects (data abstraction)
 - Object-Oriented Design techniques
 - implementation & testing strategies using version control (SCCS)
 - improving software’s ‘resilience to change’ (‘Defensive Programming’ and full documentation)

Each piece of practical work involves building a component of a software system with a theme which should:

- be appropriate to the unit’s objectives
- integrate well into the lecture material
- be motivating for the students

2 Description of the U-boat Simulation Theme

- scenario: U-boat negotiates channel, avoiding:
 - collision with the sides
 - detection by a patrolling E-boat (surface boat, also autonomous)
 - destruction by the E-boat (via depth charges)
- system parameters:

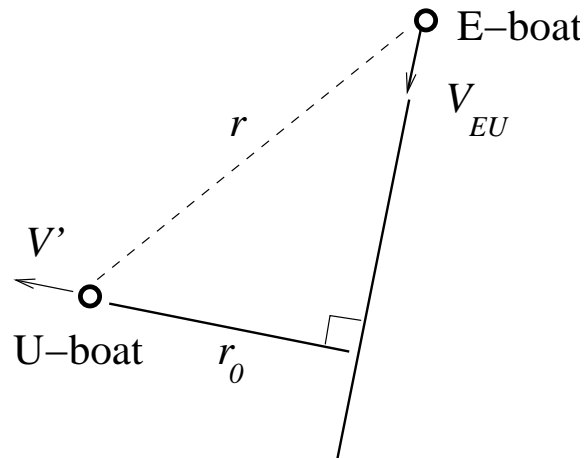
	(U-boat)	(E-boat)
channel:	8 km × 6 km	
time limit:	4800 s	
max. speed:	5 ms ⁻¹	12.5 ms ⁻¹
max. turn rate:	6 deg s ⁻¹	10 deg s ⁻¹
max. sonar range:	1650 m	1000 m
min. sonar range:	0	$r_{min} = 125$ m
depth:	15 m	0 m
# depth charges:		7
charge radius:		60 m
charge delay:		$t_d = 7$ s

- intention of the simulation:
 - chosen to be simple (2-D) but potentially ‘interesting’, asymmetric but balanced
 - to determine what evasion/hunting *tactics* are successful for U-boat/E-boat

3 Autonomous U-boat Evasion Tactics

- evade detection:

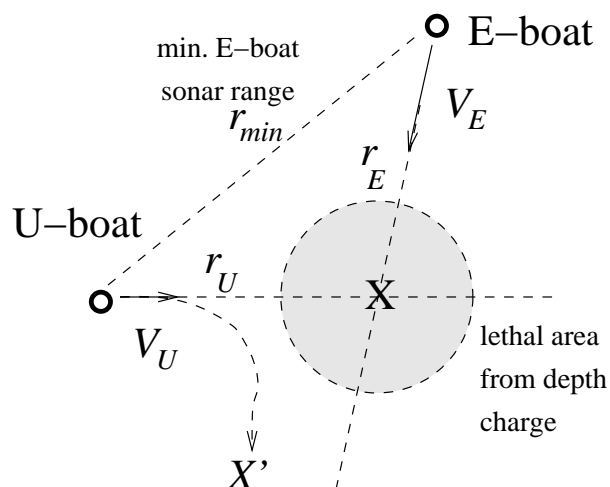
V_{EU} is *relative* velocity; $r \approx$ max. U-boat sonar range;
 U-boat should change velocity to direction V'



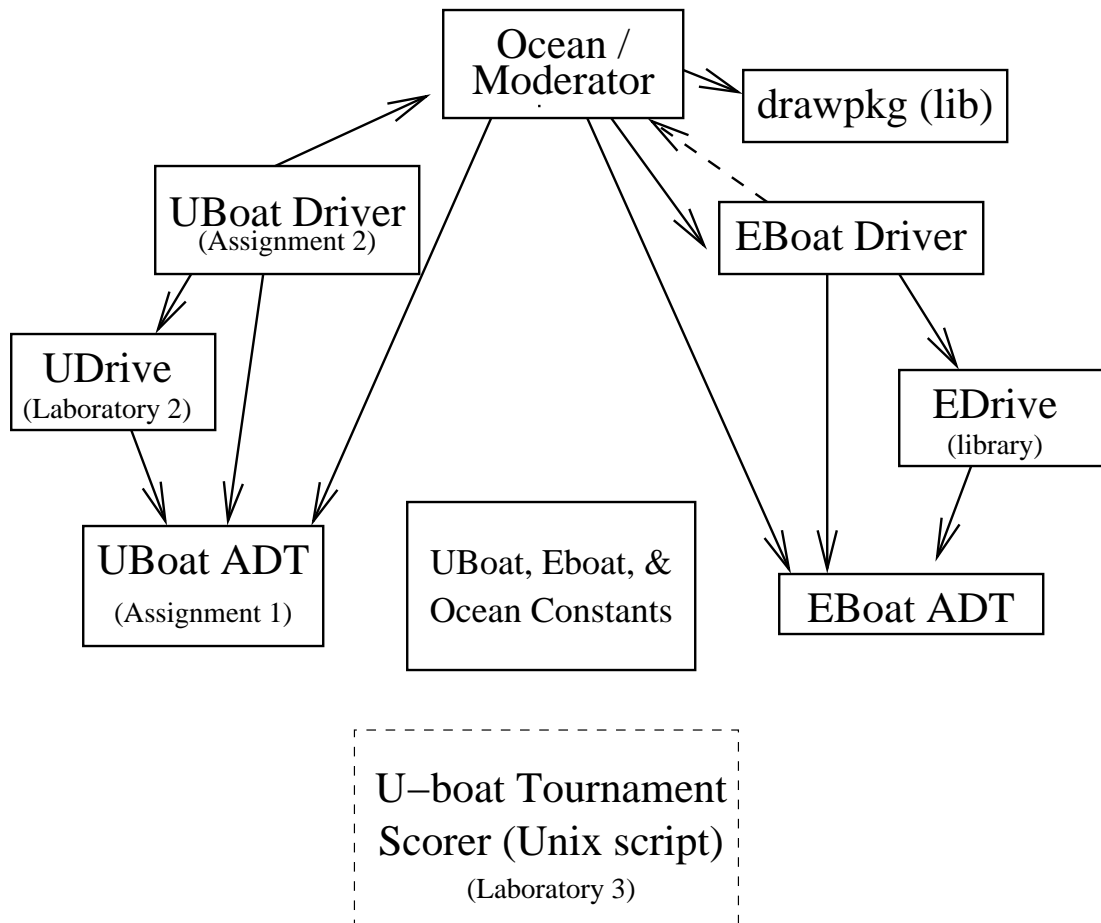
nb. if r_0 small, evasion is impossible as E-boat is much faster

- evade destruction:

E-boat must commit itself in r_E/V_E s to drop charge at U-boat's projected position X in $(r_U/V_U = r_E/V_E + t_d)$ s; U-boat should make a sudden turn to be at X' instead



4 Overall Design of U-boat Simulation System



- fully functional U-boat Abstract Data Type (ADT):
destroy, set sonar; set rudder & throttle, increment time-step operations
- Ocean creates U- & E-boat objects s.t. they interact through ‘side effects’ of the time-step operation
- security enforced via ADTs & time-stamp checking in the Ocean module only

5 Object-Oriented Design of the U-boat Driver Module

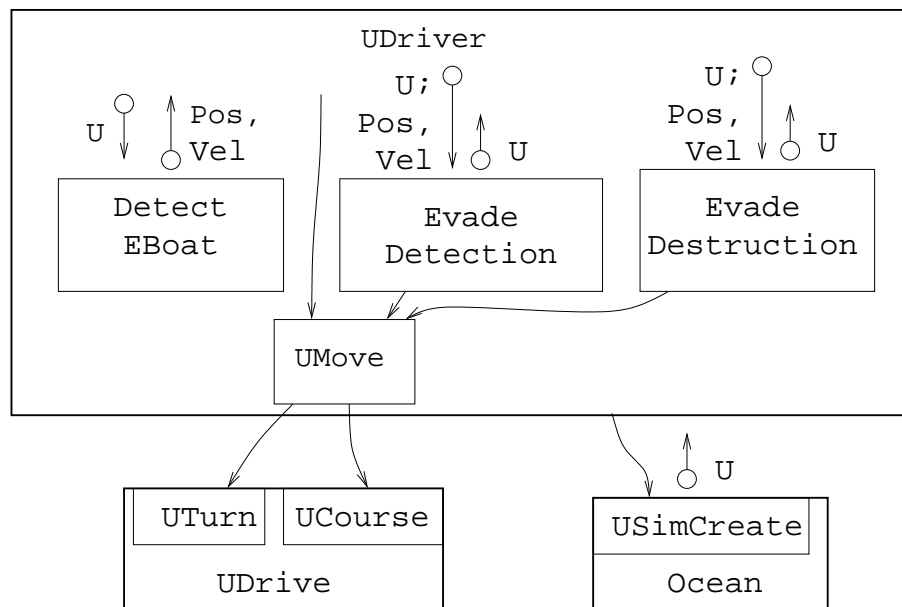
1. Develop an 'Informal Strategy':

The U-boat ² is driven ^{2A} as follows. It ² moves ^{2B} from its initial position ^{2b} to its destination. When it ² first detects ^{2C} an E-boat ³ [via sonar ^{2c}], giving its position ^{3b} and velocity ^{3c}, it ² tries to evade detection ^{2D}, by making appropriate turns ^{2D1} and setting a new course ^{2D2}. When it ² detects nearby ^{2C'} the E-boat ³, it ² tries to evade destruction ^{2D'}.

2. Analyze objects & actions, form an 'Object Table':

<u>object:</u>	<u>attributes:</u>	<u>operations:</u>
U-boat (2)	time-step position (2b) velocity sonar (2c) is-destroyed	drive (2A), moves (2B) turns (2D1), sets course (2D2) detect E-boat (2C,2C') evades detection (2D) evade destruction (2D')

3. Analyze operations' relationships, form a Structure Graph:



6 Formal Specification of the U-Boat ADT

- U is a U-boat object
- time evolves by applying a time-step operation:

$$U\text{Time} (U\text{TimeStep} (U)) = U\text{Time} (U) + 1$$

new position depends on old position, speed and orientation:

$$\begin{aligned} U\text{Position} (U\text{TimeStep} (U)) = \\ (x + v * \cos (th), y + v * \sin (th)) \\ \text{WHERE } (x,y) = U\text{Position} (U) \\ v = U\text{Speed} (U) \\ th = U\text{Orientation} (U) \end{aligned}$$

new orientation depends on old, speed and rudder setting:

$$\begin{aligned} U\text{Orientation} (U\text{TimeStep} (U)) = \\ th + c * v * \text{Sign} (u\text{Rudder} (U)) \end{aligned}$$

the rudder can only change by a set rudder operation:

$$\begin{aligned} u\text{Rudder} (U\text{TimeStep} (U)) &= u\text{Rudder} (U) \\ u\text{Rudder} (U\text{SetRudder} (U, rd)) &= rd \end{aligned}$$

(the throttle can be specified similarly to the rudder)

- the U-Boat ADT was also specified using pre- and post-conditions

7 Graphical & Textual Display of the Simulation

8 Motivations / Advantages of the Theme

- + simulations are an increasingly important area of application, especially in Engineering
- + shows Abstract Data Types (ADTs) can successfully model 'real-world' (non-mathematical) objects
- + Modula-2 module interfaces model real-world information hiding
- + Defensive Programming:
 - ADT pre- and post-condition & data structure invariant checking also motivated by security
 - procedure tracing necessary for debugging
- + a 'natural' for Object-Oriented Design
- + version control simply involves adding extra tactics
- + illustrates well the limitations of procedural languages & single processes, motivating Functional Programming, Object-Oriented Programming and Concurrency
- + good potential for discussing design decisions
- + enabled a successful U-boat tournament to be run

9 Conclusions

- the resulting system is 'unstable': a minor change can make behavior change dramatically (this makes debugging difficult)
- other problems encountered are surmountable, due to the novelty of the theme
- + student motivation high; many liked the idea of competing against the lecturer (author of the E-boat Driver)
- + U-boat tournament: 25 simulations (scores from -25 to 100):
 - system parameters were changed (in favor of E-boat!):
this motivated generality of design and proper parameterization of code
 - non-security breaking category
(1 entrant with score 100, 10 entrants over 80)
 - security breaking category
(1 successful entrant only, with score 100)
- + solutions implementing well simple tactics were the best!
- + many solutions realistically modeled a non-trivial 'real-world' phenomenon
- + plan to extend theme next year to the unit CSB32: Principles of Concurrent Programming