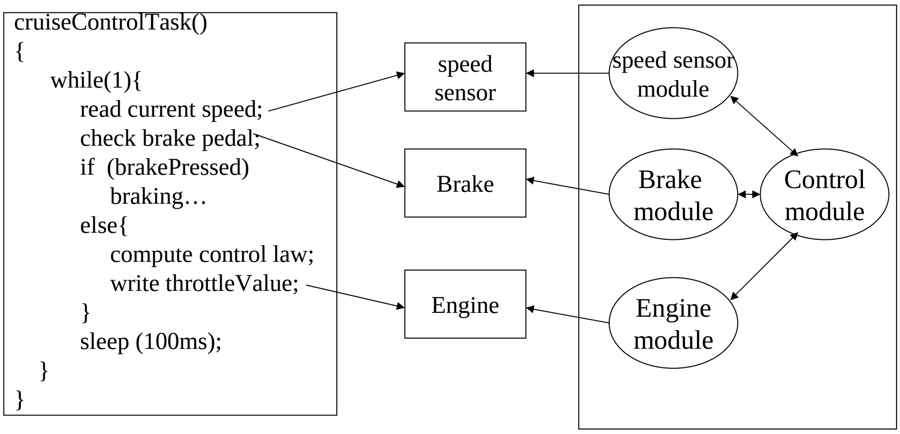
Objected-Oriented Real-Time System Design

Motivations

- Next-Generation real-time systems become
  - Complicated
  - Distributed
  - Networked
- Examples
  - Military unmanned command/control system
  - City-wide disaster monitoring and management system
  - Hospital patient monitoring system
  - Assisted-living
- System specification is very difficult in traditional way
- **Object-Oriented Design Paradigm needed**

Time-Driven vs Event-Driven(OO)



Time-Driven SW Design

Event-Driven (OO) SW Design

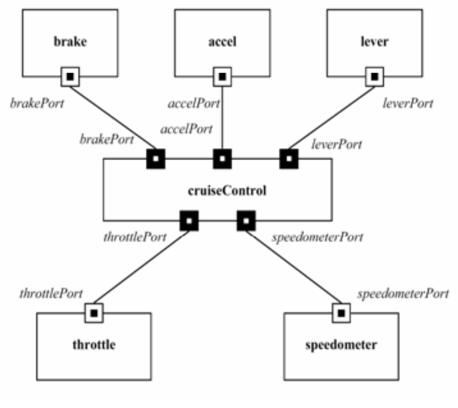
Object-Oriented Real-Time Design Approach

- OO-design more naturally reflect the actual system
- Easy to think
- We can focus on each component and specify event-driven operations with a stateChart
- Reusable
- Portable
- Flexible
- Extendable

Emerging RT designs use OO paradigm

- Real-Time OO design support languages and tools
  - Chaos (Honeywell)
  - Cadena (Kansas State Univ.)
  - Geodesic (CMU)
  - ROOM: Real-Time Object Oriented Modeling
  - UML (Universal Modeling Language) – RT
  - Real-Time JAVA
  - Real-Time CORBA
- Even a small system follow OO paradigm
  - TinyOS (Set of commonly used object modules)

Cruise control system example



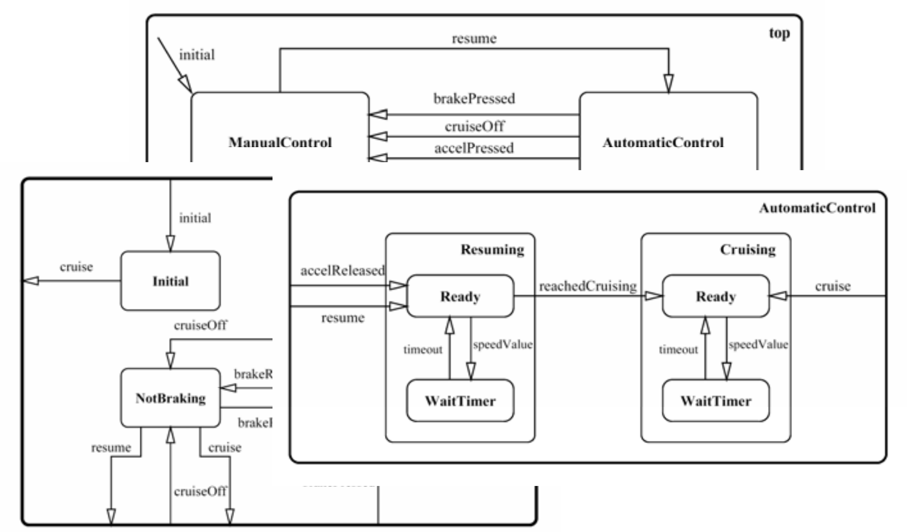
-Start with manual mode

-When Cruise lever set, goto automaticControl mode

- In automaticControl mode, regularly check the current speed and actuate the engine accordingly to maintain the setSpeed.

- In automaticControl mode, goto manual mode when brakePressed, accelPressed, cruiseOff

State Description of the cruiseControl object



Transactions and timing constraints

At each rotation of drive shaft
(at every rotation of the wheels)

Periodically invoked an calculate the speed using the number of shaft rotations in the previous period

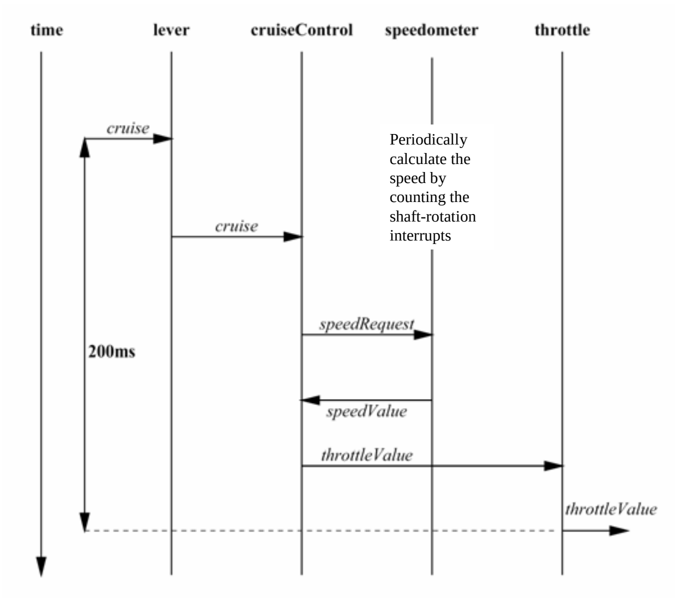
When 6000rpm

Transaction	Stimulus	Response	Period	Deadline
Shaft Interrupt (SI)	*shaftInterrupt*	-	(min) 10ms	10ms
Determine Speed (DS)	*timeout*	*speedValue*	50ms	10ms
Control Loop (CL)	*timeout*	*throttleValue*	100ms	100ms
Enter Cruise (EC)	*cruise*	*throttleValue*	-	200ms
Resume Cruise (RC)	*resume*	*throttleValue*	-	200ms
Accel Released (AR)	*accelReleased*	*throttleValue*	-	200ms
Brake Pressed (BP)	*brakePressed*	-	-	50ms
Accel Pressed (AP)	*accelPressed*	-	-	150ms
Cruise Off (CO)	*cruiseOff*	-	-	100ms

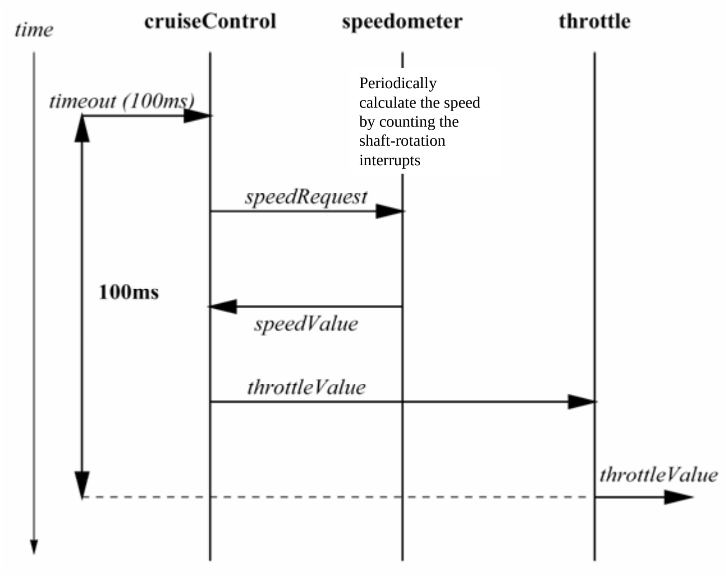
Enter Automatic Control Mode

Enter Manual Control Mode

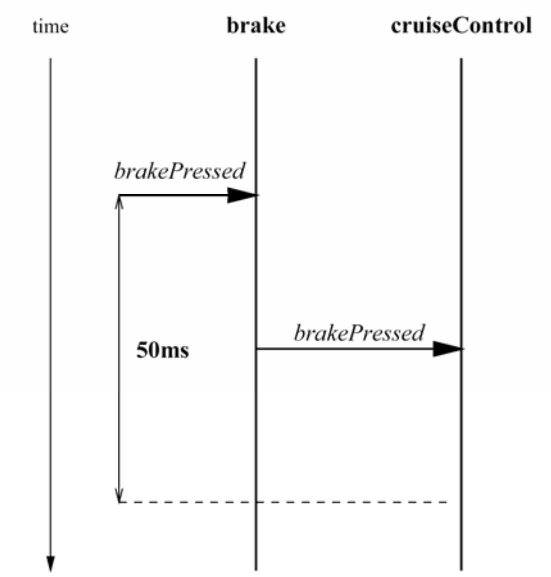
Message Sequence for Entering Cruise mode



Message Sequence for Cruise Control Loop



Message Sequence for Leaving Cruise Mode



Challenge is how to implement the system and validate the timing

- Real-Time theory (including schedulability analysis) is built on Time-Driven Model
- Real-Time Operating Systems have been evolved with Time-Driven Model in mind
- Mapping is required from OO-design to Time-driven implementation over RTOS platform
- How to reuse Real-Time Theory for the schedulability check after the mapping?

Mapping Objects to RTOS threads

- Map a group of objects into an RTOS thread
  - For example
    - map speedometer object to a RTOS thread
    - map all other objects to another RTOS thread
  - Optimal mapping is a challenging problem
- Priority
  - Transaction priority is determined based on e2e deadline. We give higher priorities to the aborting transactions (BP,CO,AP > CL)
  - Event priority is determined by the highest priority of the transactions that it belongs to
  - Thread priority: dynamically determined by the priority of event currently being handled (RTOS will dispatch a thread according to the thread priority)
  - This is just heuristic. The optimal priority assignment is an open issue.

Priorities of Cruise Control Transactions

Transaction	Period	Deadline	Priority
SI	(min) 10	10	1
DS	50	10	2
CL	100	100	6
EC	-	200	7
RC	-	200	7
AR	-	200	7
BP	-	50	3
CO	-	100	4
AP	-	150	5

- Consider each transaction as a Virtual Task.
- Our concern is whether each virtual task can meet its deadline if it is executed on the above (thread implemented) run-time system

Schedulability Analysis

- Calculate the worst case response time of each transaction
- High-priority transaction can be blocked by low-priority event handling
  - Blocking due to Run-To-Completion of a thread

$$B_i^{RTC} = \max_j C_i^{event} :: \text{Thread}(j) = \text{Thread}(i)$$

Worst case event processing time that belongs to transaction j

  - Sharing message queue (Mutual exclusio
- Revised response time equation fo

A high priority thread will not be delayed by a low priority thread due to preemptive thread scheduling
However, it may blocked by the event handling procedure of the same thread but belonging to a different transaction.

$$R_i = \sum_{\tau_j \in hp(i)} \left(\left\lceil \frac{R_i}{T_j} \right\rceil C_j \right) + C_i + B_i^{RTC}$$

Response times for transactions

cruiseControl	speedometer	Other
$C_{timeout}$ = 2ms	C_{shaft} = 2ms	$C_* = 2ms$
$C_{speedValue}$ = 10ms	$C_{timeout}$ = 3ms	
$C_* = 5ms$	$C_{speedRequest}$ = 3ms	

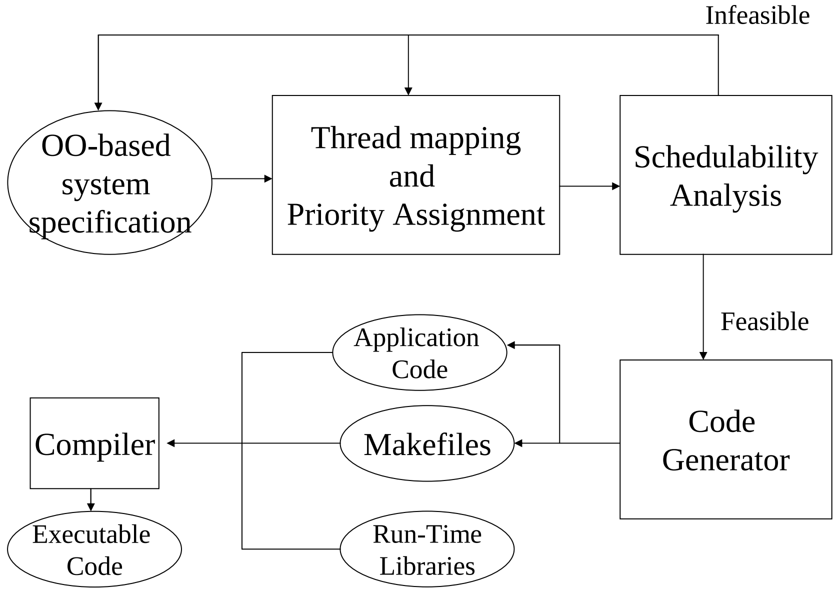
SI (Shaft Interrupt) : $2+3$

DS (Determine Speed) : $R = \left\lceil \frac{R}{10} \right\rceil 2 + 3 + 3$

CL (Control Loop) : $R = \left\lceil \frac{R}{10} \right\rceil 2 + \left\lceil \frac{R}{50} \right\rceil 3 + (2+3+10+2) + (5+3)$

Transaction	Period	Deadline	Priority	Execution	Blocking	Response
SI	(min) 10	10	1	2	3	5
DS	50	10	2	3	3	8
CL	100	100	6	17 (2 + 3 + 10 + 2)	8 (5 + 3)	36
EC	-	200	7	22 (2 + 5 + 3 + 10 + 2)	8 (5 + 3)	43
RC	-	200	7	22 (2 + 5 + 3 + 10 + 2)	8 (5 + 3)	43
AR	-	200	7	22 (2 + 5 + 3 + 10 + 2)	8 (5 + 3)	43
BP	-	50	3	7 (2 + 5)	10	26
CO	-	100	4	7 (2 + 5)	10	35
AP	-	150	5	7 (2 + 5)	10	44

Automate the Overall Design Flow



References

- M. Saksena, P. Freedman, and P. Rodziewicz, “Guidelines for Automated Implementation of Executable Object Oriented Models for Real-Time Embedded Control Systems, IEEE RTSS 1997
- Z. Gu and Z. He, “Real-Time Scheduling Techniques for Implementation Synthesis from Component-Based Software Models, ACM SIGSOFT 2005
- W. Deng, M. B. Dwyer, J. Hatcliff, G. Jung, Robby, and G. Singh, “Model-checking Middleware-based Event-driven Real-Time Embedded Software”, The 1st International Symposium on Formal Methods for Components and Objects, 2003
- T. E. Bihari and P. Gopinath, “Object-Oriented Real-Time Systems: Concepts and Examples, Computer 1992

Still open problems .....

- Component chains in distributed resources?
- Communication costs?