SWE 621:
Software Modeling and Architectural Design

Lecture Notes on Software Design

Lecture 9 – Task Structuring

Hassan Gomaa
Dept of Computer Science
George Mason University
Fairfax, VA

Copyright © 2011 Hassan Gomaa
All rights reserved. No part of this document may be reproduced in any form or by any means, without the prior written permission of the author.
This electronic course material may not be distributed by e-mail or posted on any other World Wide Web site without the prior written permission of the author.

SWE 621:
Software Modeling and Architectural Design

Lecture 9: Task Structuring

Hassan Gomaa

Reference: H. Gomaa, Chapters 18 - *Software Modeling and Design*, Cambridge University Press, February 2011

Reference: H. Gomaa, Chapter 14 - *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Addison Wesley Object Technology Series, July, 2000

Copyright © 2011 Hassan Gomaa
All rights reserved. No part of this document may be reproduced in any form or by any means, without the prior written permission of the author.

Steps in Using COMET/UML

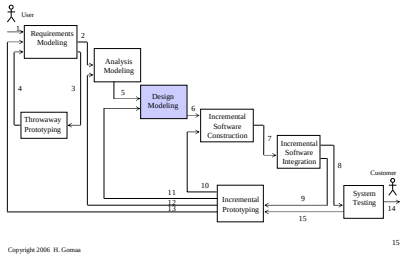
1 Develop Software Requirements Model

2 Develop Software Analysis Model

3 **Develop Software Design Model**

- Design Overall Software Architecture (Chapter 12, 13)
- Design Distributed Component-based Subsystems (Chapter 12-13,15)
- **Structure Subsystems into Concurrent Tasks (Chapter 18)**
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design

Figure 6.1 COMET object-oriented software life cycle model

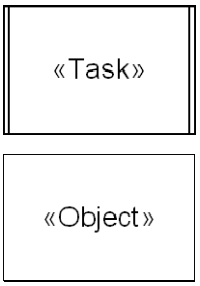


Structure System into Tasks

- Concurrent Design with UML
- Concurrent task structuring criteria
  - Structure analysis model into concurrent tasks
    - Task is an active object
    - Task has thread of control
  - Consider concurrent nature of system activities
  - Determine concurrent tasks
- Define task interfaces
- Support for concurrent tasks
  - Operating system services: multi-tasking kernel

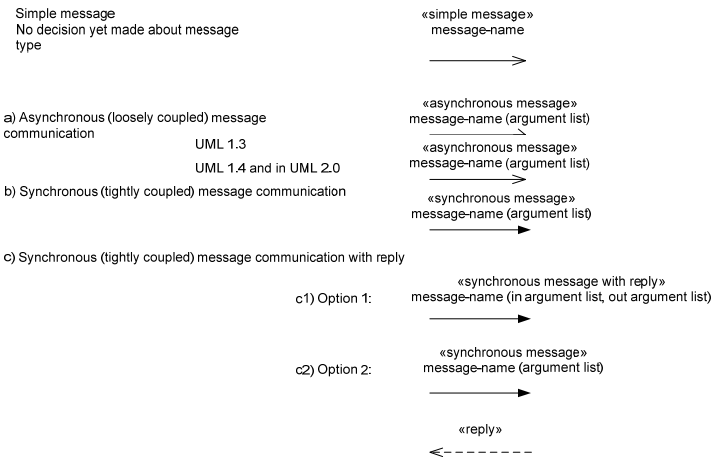
Active and Passive Objects

- Objects may be **active** or **passive**
- **Active object**
  - **Concurrent Task**
  - Has thread of control
- **Passive object**
  - a.ka. **Information Hiding Object**
  - Has no thread of control
  - Operations of passive object are executed by task
  - Operations execute in task's thread of control
    - Directly or indirectly
- Software Design terminology
  - **Task** refers to active object
  - **Object** refers to passive object



Copyright 2011 H. Gomaa

UML notation for messages



Copyright 2011 H. Gomaa

Task Structuring Criteria

- Event driven task
  - Activated by external event (e.g., interrupt)
- Periodic task
  - Activated by timer
- Demand driven task
  - Activated by arrival of internal message

Copyright 2011 H. Gomaa

I/O Task Structuring Criteria

- Event driven I/O task
  - Task for each event (interrupt) driven I/O device
  - Event driven device generates interrupt
- Periodic I/O task
  - Task for each polled I/O device
  - I/O device (usually input) sampled at regular intervals
- Demand driven I/O task
  - Task for each passive I/O device (usually output)
  - Computation overlapped with output

Copyright 2011 H. Gomaa

Event Driven I/O Task

Device Driver

- One task for each event driven I/O device
- Activated by device I/O interrupt
- Reads input
- Converts to internal format
- Disposes of input
  - Sends message containing data
  - Signals event (message with no data)
  - Writes to data store

Copyright 2011 H. Gomaa

Figure 18.5 Example of event driven I/O task

Figure 18.5a Analysis model – communication diagram

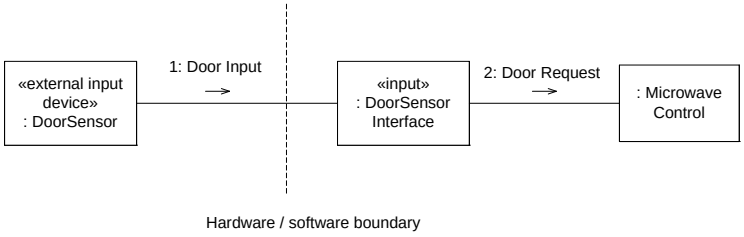
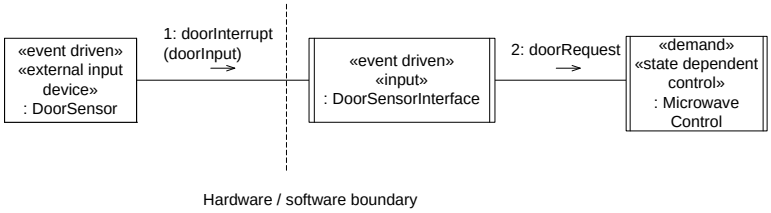


Figure 18.5b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

Periodic I/O Task

Task for each polled I/O device
- Activation of task is periodic
- Samples I/O device

Periodic I/O task
- Activated by timer event
- Performs I/O operation
- Waits for next timer event

Copyright 2011 H. Gomaa

Figure 18.6 Example of a periodic I/O task

Figure 18.6a Analysis model – communication diagram

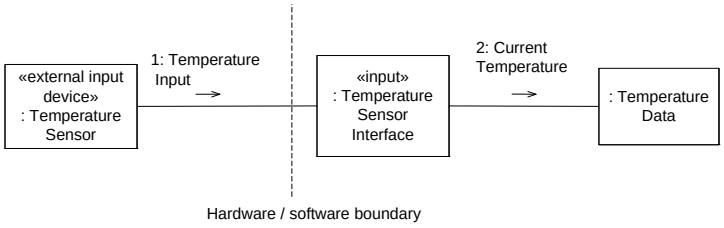
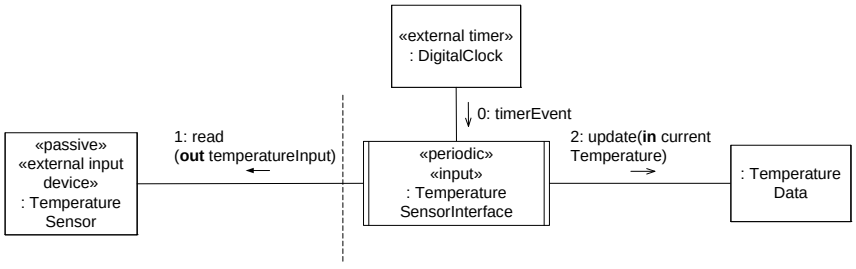


Figure 18.6b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

Demand Driven I/O Task

- Task for each passive I/O device (usually output)
  - Passive I/O device does not need to be polled
  - Computation overlapped with output
    - Task output to device overlapped with
    - Computational task that produces data
- Usually for passive output device
  - Demand driven I/O task
- Passive input devices more likely to be polled
  - Periodic input task

Copyright 2011 H. Gomaa

Figure 18.7 Example of a Demand Driven Output Task

Figure 18.7a Analysis model – communication diagram

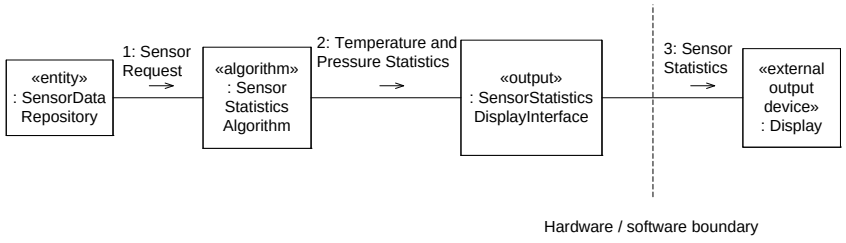
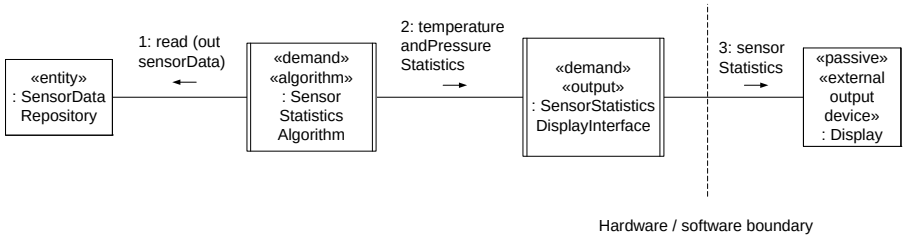


Figure 18.7b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

Internal Task Structuring Criteria

- Periodic task
  - Task for each periodic activity
- Demand task
  - Task for each demand driven internal activity
- Control task
  - Task executes state machine
- User interaction task
  - Task for each sequential user activity

Copyright 2011 H. Gomaa

Periodic Task

Task for each periodic activity

Task activated periodically

Activated by timer event

Performs activity

Waits for next timer event

Copyright 2011 H. Gomaa

Figure 18.8 Example of periodic task

Figure 18.8a Analysis model – communication diagram

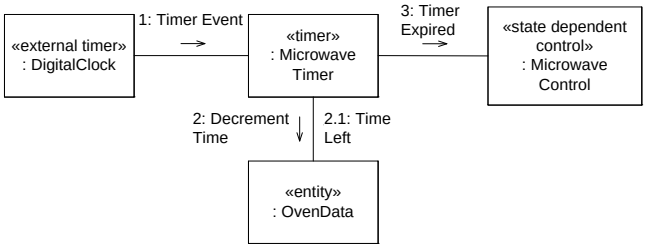
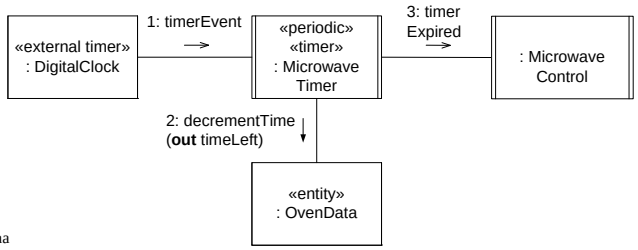


Figure 18.8b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

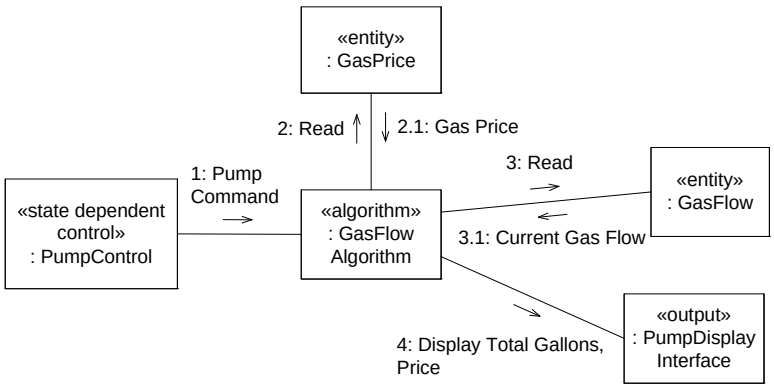
Demand Task

- Activity executed on demand
  - Activated by internal event or message
  - Map to Demand Task
- Demand task
  - Activated on demand by event or message sent by different task
  - Performs demanded action
  - Waits for next event or message

Copyright 2011 H. Gomaa

Figure 18.9 Example of demand task

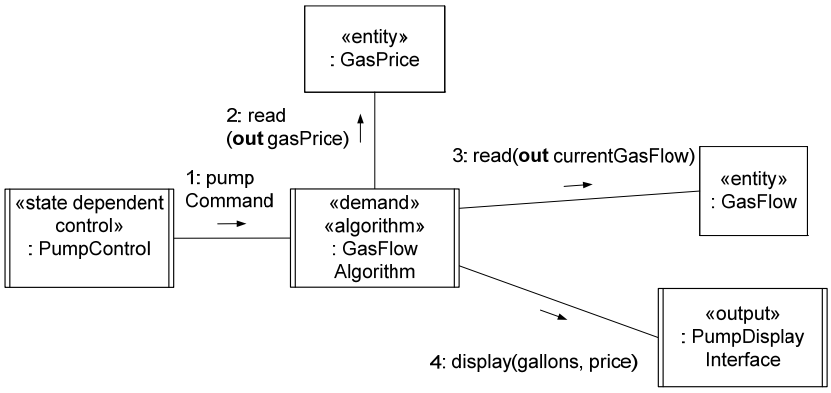
Figure 18.9a Analysis model – communication diagram



Copyright 2011 H. Gomaa

Figure 18.9 Example of demand task

Figure 18.9b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

Control Task

Task executes statechart

- State dependent control object executes statechart
- Execution of statechart is sequential

One task for each control object

Can have multiple tasks of same type

Copyright 2011 H. Gomaa

Figure 18.10 Example of control task

Figure 18.10a Analysis model – communication diagram

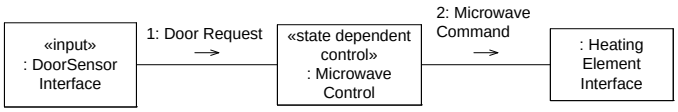
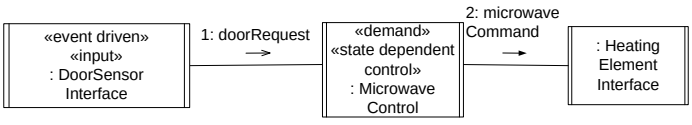


Figure 18.10b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

User Interaction Task

One task for each sequential user activity

Multi-user system

- One task per user
- User may also spawn background tasks

Windowing system

- User engaged in multiple activities
- Each window executes sequential activity
- One task for each window

Copyright 2011 H. Gomaa

Figure 18.12 Example of user interaction task

Figure 18.12a Analysis model – communication diagram

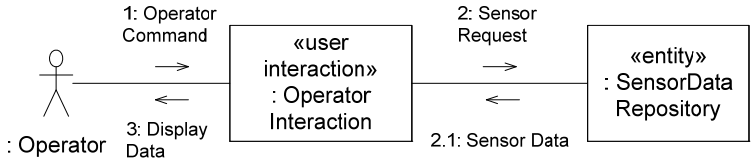
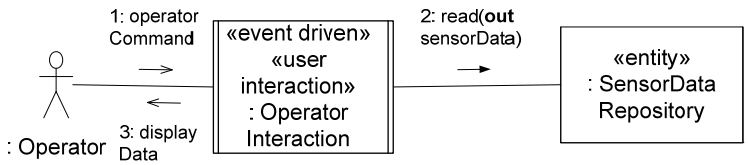


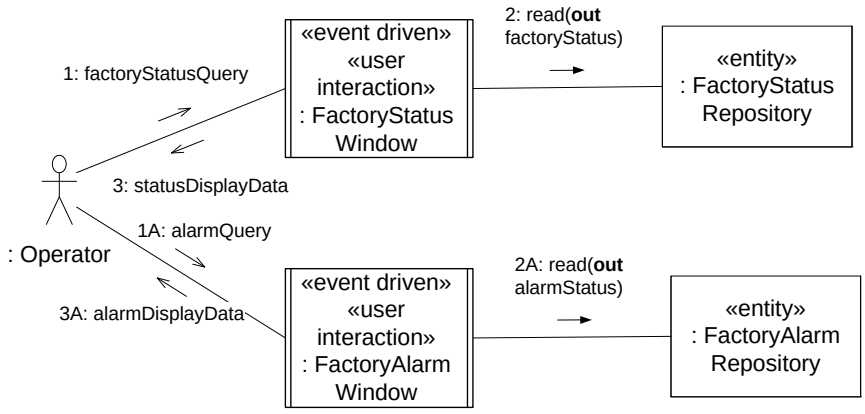
Figure 18.12b Design model – concurrent communication diagram



Copyright 2011 H. Gomaa

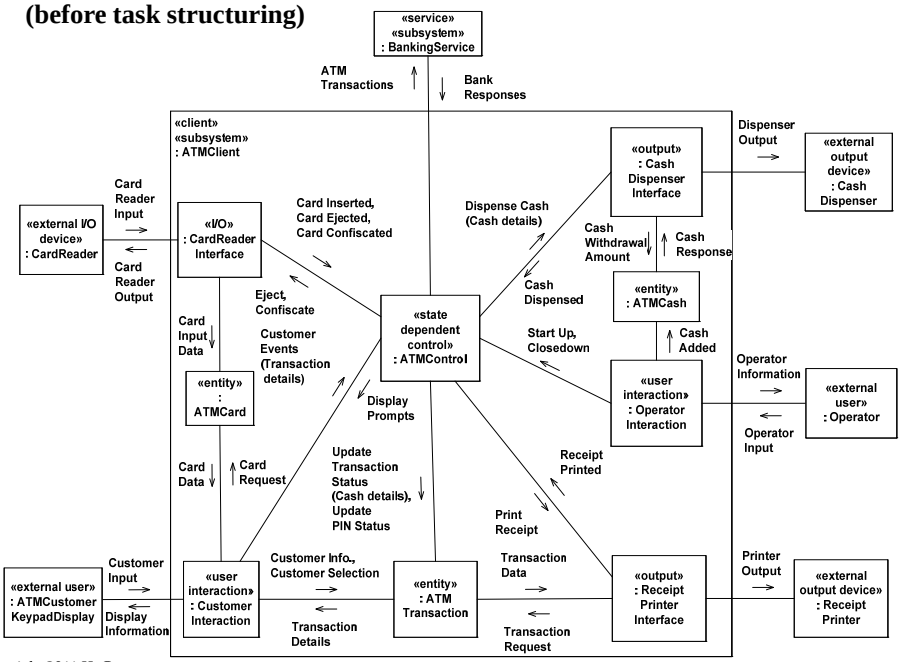
Figure 18.12 Example of user interaction task

Figure 18.12c Design model – concurrent communication diagram



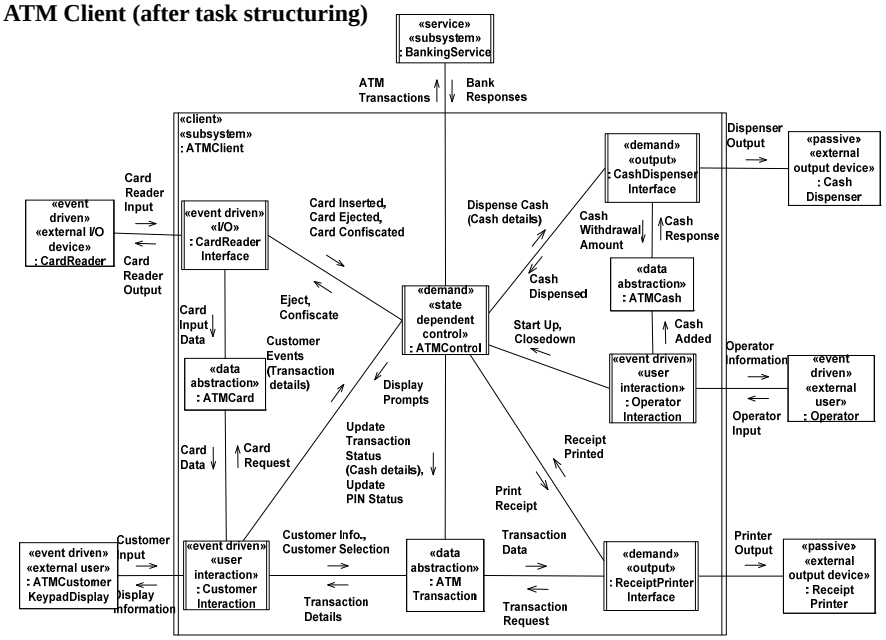
Copyright 2011 H. Gomaa

Figure 21.25 Integrated communication diagram for ATM Client subsystem (before task structuring)



Copyright 2011 H. Gomaa

Figure 18.13 Task architecture – initial concurrent communication diagram for ATM Client (after task structuring)



Copyright 2011 H. Gomaa

Banking System Case Study - Task Structuring Criteria

- Event driven I/O task
  - Card Reader Interface
- Demand driven output task
  - Cash Dispenser Interface
  - Receipt Printer Interface
- Event driven user interaction Task
  - Customer Interaction
  - Operator Interaction
- Demand driven state dependent control task
  - ATM Control
- Service task
  - Bank Service

Copyright 2011 H. Gomaa

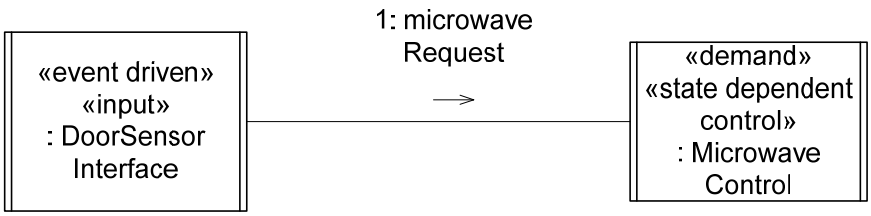
Define Task Interfaces

- Map Analysis Model simple message interfaces to task interfaces
  - Need to determine type of message communication
- Loosely coupled (asynchronous) message communication
- Tightly coupled (synchronous) message communication
  - With reply
  - Without reply
- Event synchronization
  - External event (interrupt)
  - Timer event
- Passive objects
  - Task interfaces to information hiding object
- Update task architecture

Copyright 2011 H. Gomaa

Asynchronous Message Communication (Loosely Coupled)

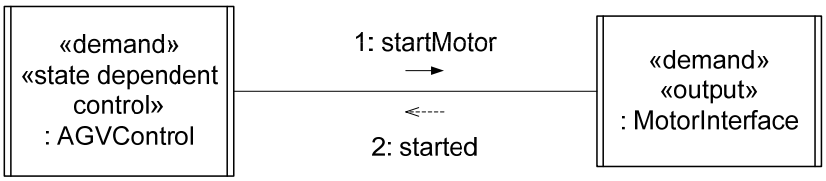
- Producer sends message and continues
- Consumer receives message
  - Suspended if no message is present
  - Activated when message arrives
- Message queue may build up at Consumer



Copyright 2011 H. Gomaa

Synchronous (Tightly Coupled) Message Communication With Reply

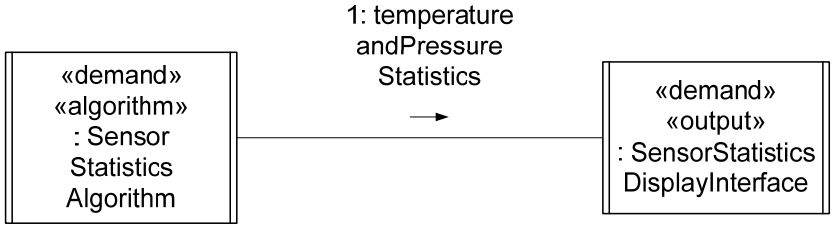
- Producer task sends message and waits for reply
- Consumer receives message
  - Suspended if no message is present
  - Activated when message arrives
  - Generates and sends reply
- Producer and Consumer continue



Copyright 2011 H. Gomaa

Synchronous (Tightly Coupled) Message Communication Without Reply

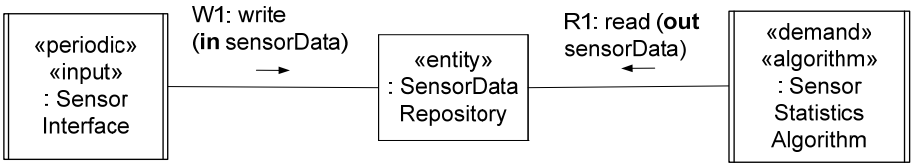
- Producer task sends message and waits for acceptance
- Consumer receives message
  - Suspended if no message is present
  - Activated when message arrives
  - Accepts message, Releases producer
- Producer and Consumer continue



Copyright 2011 H. Gomaa

Information Hiding Object

- Passive object
  - Encapsulates data
  - Hides contents of data structure
  - Data accessed indirectly via operations
- Passive object accessed by two or more tasks
  - Operations must synchronize access to data
  - Use semaphore or monitor object
  - Design of class operations is described in Class Design



Copyright 2011 H. Gomaa

Task Interface Specifications (TIS)

Developed during Task Structuring

Expanded during Detailed Software Design

Describes concurrent task's

- Information hidden
- Structuring criteria
- Anticipated changes
- Task inputs and outputs
- Event sequencing logic

Copyright 2011 H. Gomaa

Task Interface Specification

Task interface
- Message communication
  - Type of interface
  - Message names and parameters
- Events signaled
  - Name and Type of event
- External inputs or outputs

Task structure information
- Task structuring criterion used to design task

Task Behavior spec (event sequencing logic)
- Response to each message or event input
- Described informally in Pseudocode

Copyright 2011 H. Gomaa

Example of Task Interface Specification

Name: Card Reader Interface

Information hidden: Details of processing input from and output to card reader.
Structuring criteria: role criterion: input/output; concurrency criterion: event driven
Assumptions: only one ATM card input and output is handled at one time.
Anticipated Changes: Possible additional information will need to be read from ATM card.

Task interface:

Task inputs:
Event input: Card reader external interrupt to indicate that a card has been input.
External input: `cardReaderInput`.
Synchronous message communication without reply:
- `eject`
- `confiscate`

Task outputs:
External output: `cardReaderOutput`
Asynchronous message communication:
- cardInserted.
- cardEjected
- cardConfiscated.

Passive objects accessed: ATMCard
Errors detected: Unrecognized card, Card reader malfunction.

Copyright 2011 H. Gomaa

Steps in Using COMET/UML

1 Develop Software Requirements Model

2 Develop Software Analysis Model

3 **Develop Software Design Model**

- Design Overall Software Architecture (Chapter 12, 13)
- Design Distributed Component-based Subsystems (Chapter 12-13,15)
- **Structure Subsystems into Concurrent Tasks (Chapter 18)**
- Design Information Hiding Classes (Chapter 14)
- Develop Detailed Software Design

Figure 6.1 COMET object-oriented software life cycle model