

**SWE 621:
Software Modeling and Architectural Design**

Lecture Notes on Software Design

Lecture 7 – Software Architecture

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.

Copyright 2011 H. Gomaa

Overview

- Collaborative Object Modeling and architectural design mETHod (COMET)
 - Object Oriented Analysis and Design Method
 - Uses UML (Unified Modeling Language) notation
 - Standard approach for describing a software design
 - COMET = UML + Method
- Provides steps and guidelines for
 - Software Modeling and Design
 - From Use Case Models to Software Architecture
- H. Gomaa, *Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures*, Cambridge University Press, February 2011

Copyright 2011 H. Gomaa

2

Overview of Software Architecture

Lecture 7

Hassan Gomaa

Reference: H. Gomaa, Chapters 12 - *Software Modeling and Design*, Cambridge University Press, February 2011

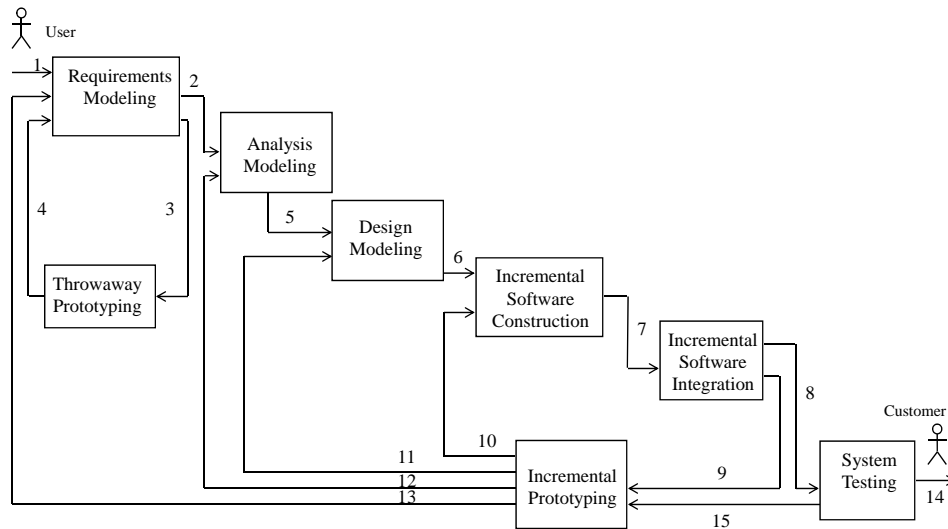
Copyright © 2011 Hassan Gomaa

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.

Copyright 2011 H. Gomaa

3

Figure 6.1 COMET object-oriented software life cycle model



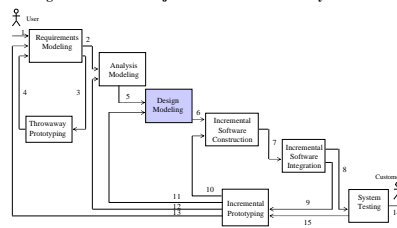
Copyright 2011 H. Gomaa

4

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 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



Copyright 2011 H. Gomma

Copyright 2006 H. Gomma

15

5

Design of Software Architecture

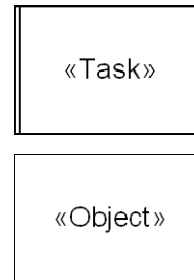
- Software Architecture
 - Structure of software system
 - Software elements
 - Externally visible properties of elements
 - Relationships among elements
- Develop initial software architecture
 - Synthesize from communication diagrams
 - Structure system into subsystems
- Subsystems determined using subsystem structuring criteria
 - Use stereotypes for subsystem structuring criteria
 - E.g., <<client>>, <<service>>
 - Depict subsystems on subsystem communication diagrams

Copyright 2011 H. Gomma

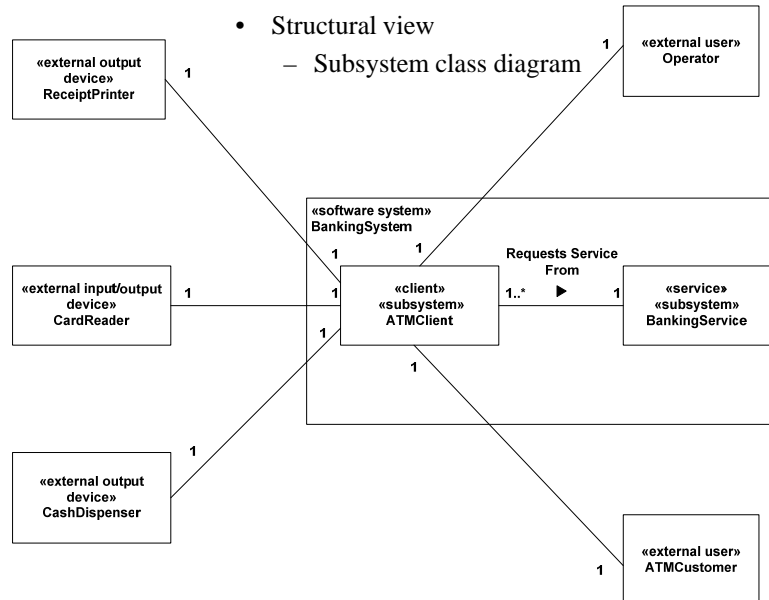
6

Active and Passive Objects

- Objects may be **active** or **passive**
- **Active object**
 - **Concurrent task or component**
 - Has thread of control
- **Passive object**
 - a.k.a. **Information Hiding Object**
 - Has no thread of control
 - Operations of passive object are executed by task

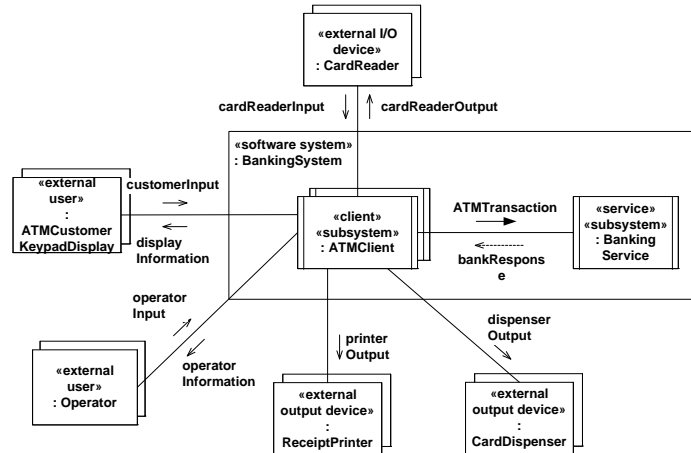


Multiple Views of Software Architecture



Multiple Views of Software Architecture

- Dynamic view
 - Subsystem communication diagram

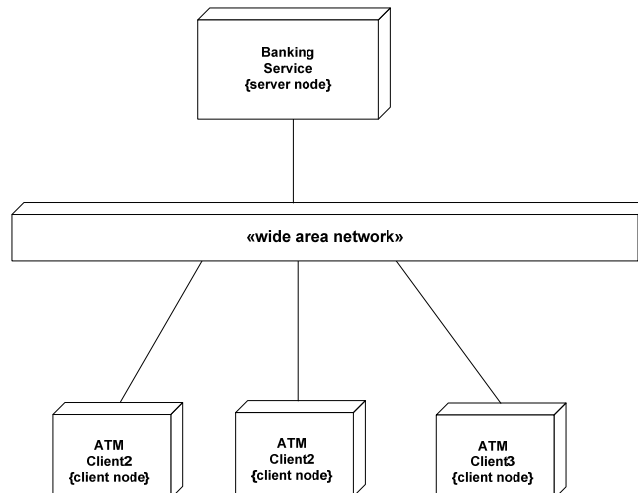


Copyright 2011 H. Goma

9

Multiple Views of Software Architecture

- Deployment view
 - Physical configuration on deployment diagram



Copyright 2011 H. Goma

10

Transition from Analysis to Design: Integration of Communication Diagrams

- Used to determine overall structure of system
- Merger of communication diagrams
 - Start with first communication diagram
 - Superimpose other communication diagrams
 - Add new objects and new message interactions from each subsequent diagram
 - Objects and interactions that appear on multiple diagrams are only shown once
 - Consider alternative scenarios for each use case
- Integrated communication diagram
 - Shows all objects and their interactions

Figure 11.1 Communication diagram: ATM Client – Validate PIN use case – Valid Pin

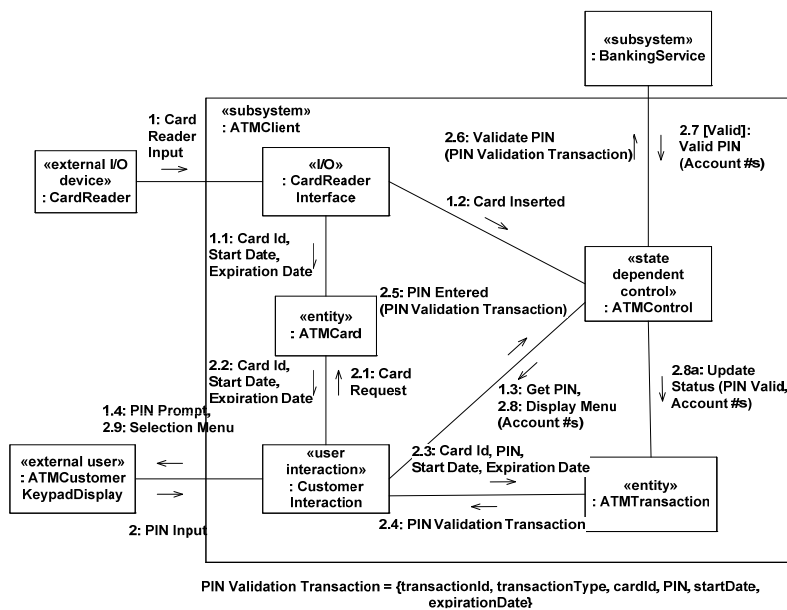
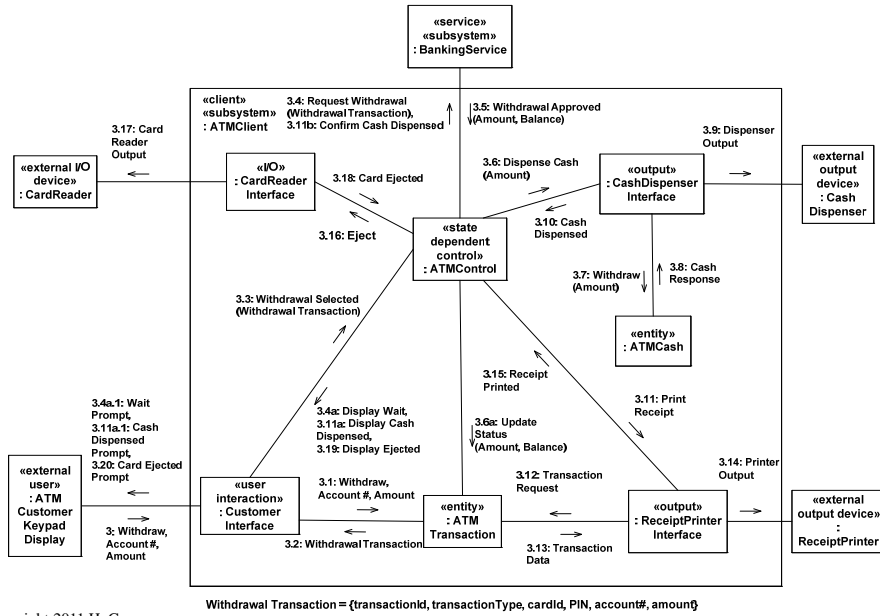


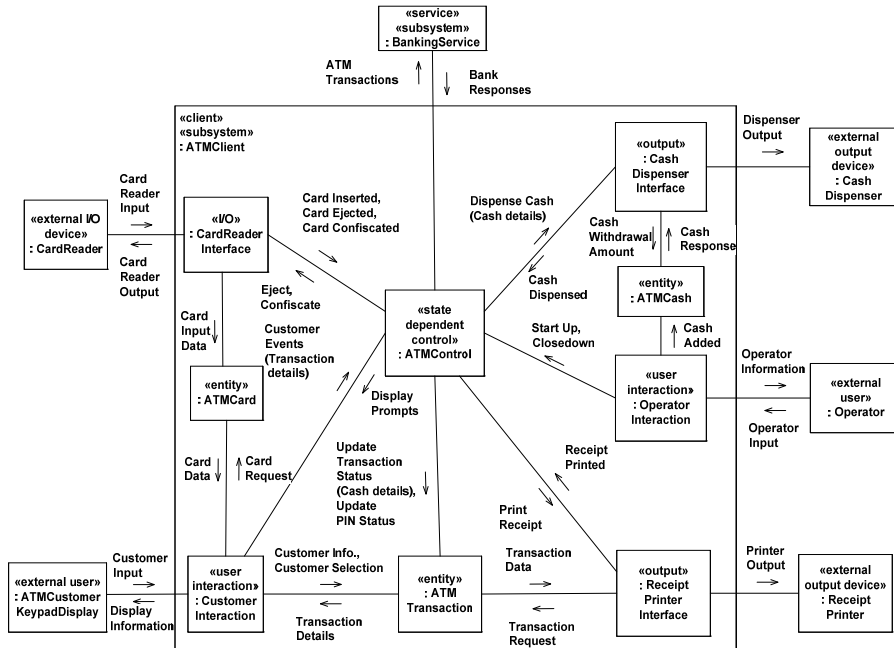
Figure 21.16 Communication diagram: ATM Client – Withdraw Funds use case



Copyright 2011 H. Gomma

13

Figure 13.2 Integrated communication diagram for ATM Client subsystem



Copyright 2011 H. Gomma

14

Integration of Communication Diagrams

- Subsystem communication diagram
 - High-level communication diagram
 - Shows subsystems and their interactions
- Integrated communication diagram
 - If there are too many objects for one integrated communication diagram
 - Develop subsystem communication diagram
 - Develop integrated communication diagram for each subsystem

Design of Software Architecture

- Software Architecture
 - Define overall structure of system
 - Component interfaces and interconnections
 - Separately from component internals
- Each subsystem performs major service
 - Contains highly coupled objects
 - Relatively independent of other subsystems
 - May be decomposed further into smaller subsystems
 - Subsystem is aggregate or composite object
- Candidates for subsystem
 - Objects that participate in same use case

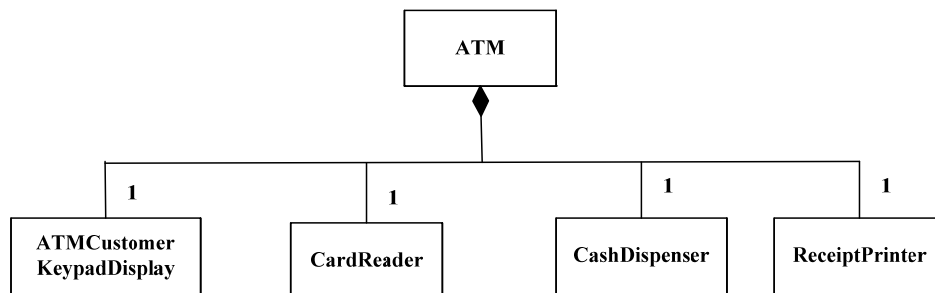
Separation of Subsystem Concerns

- **Aggregate/composite object.**
 - Objects that are part of aggregate/composite object
 - Structure in same subsystem (e.g., Fig. 13.3)
- **Interface to external objects**
 - External real-world object should interface to 1 subsystem (e.g., Fig. 13.7)
- **Scope of Control**
 - Control object & objects it controls are in same subsystem (e.g., Fig. 13.2)
- **Geographical location**
 - Objects at different locations are in separate subsystems (e.g., Fig. 13.5)
- **Clients and Services**
 - Place in separate subsystems (e.g., Fig. 13.5, 13.7)
- **User Interaction**
 - Separate client subsystem (e.g., Fig. 13.5, 13.6)

Copyright 2011 H. Goma

17

Figure 13.3 Example of composite class



Copyright 2011 H. Goma

18

Figure 13.7 Interface to external classes – Banking System

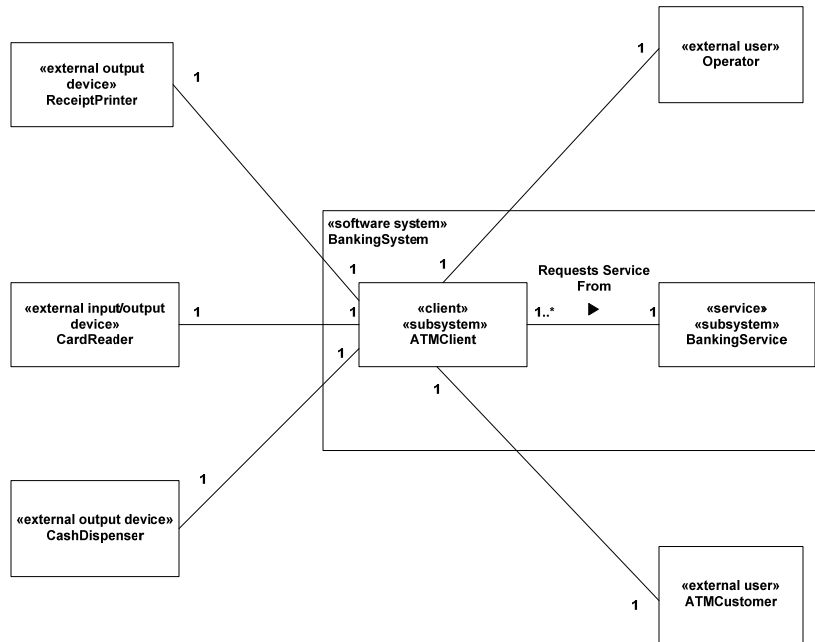


Figure 13.5: Example of geographical distribution: Emergency Monitoring System

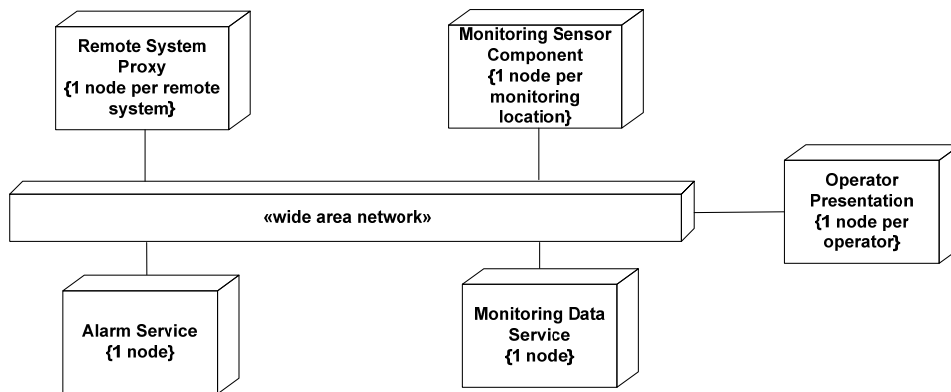
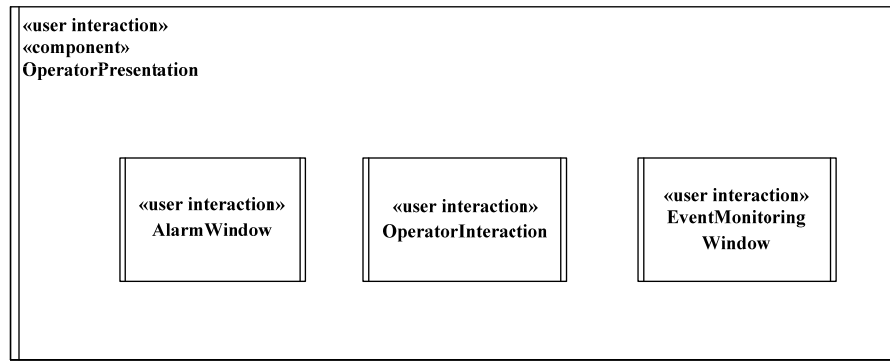


Figure 13.6 Example of user interaction subsystem: Operator Presentation component



Subsystem Structuring Criteria

- Client
 - Requester of one or more services (e.g., Fig. 13.7)
- User Interaction
 - Collection of objects supporting needs of user (e.g., Fig. 13.6, 13.10)
- Service
 - Provides service for client subsystems (e.g., Fig. 13.5, 13.7)
- Control
 - Subsystem controls given aspect of system (e.g., Fig. 13.10)
- Coordinator
 - Coordinates several control subsystems (e.g., Fig. 13.10)
- Input / Output
 - Performs I/O operations for other subsystems (e.g., Fig. 13.5)

Figure 13.10 Example of coordinator and control subsystems - Factory Automation System

