Architecture Overview Diagram & Component Model

An introduction to these key work products

Learning Objectives

- At the end of this lecture, you should be able to:
  - Understand:
    - What is an Architecture Overview Diagram (AOD)
    - What uses are there for an Architecture Overview Diagram
    - What is a Component Model and how is it represented
    - How an AOD and a Component Model relate to an Operational Models
  - Develop a simple **Architecture Overview Diagram**
  - Identify potential issues when reviewing an Architecture Overview Diagram
  - Identify **candidate components for a Component Model**

Architecture Overview Diagram

What is it?

Where does it fit?

Examples

What is an Architecture Overview Diagram?

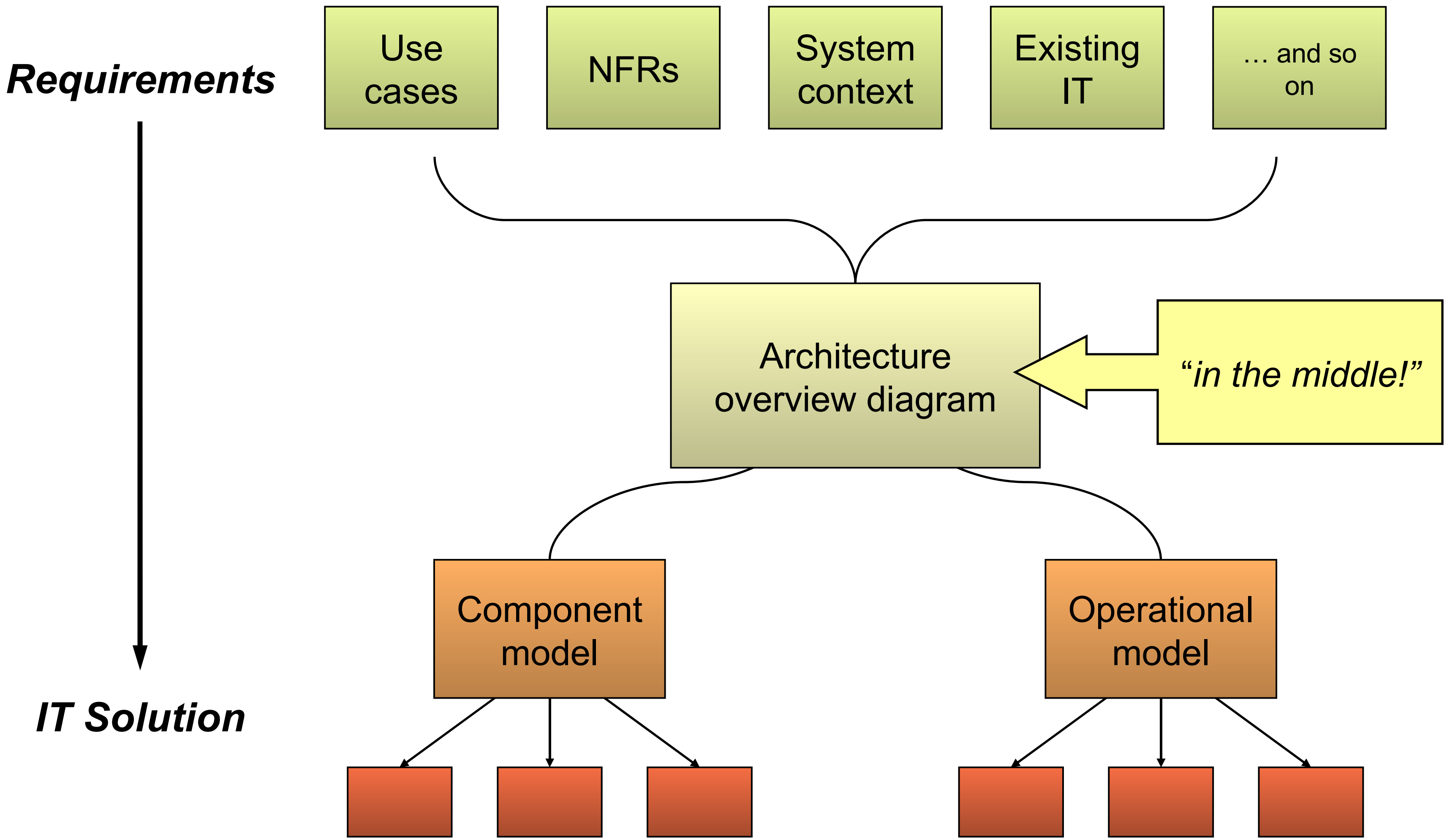
The purpose of this work product is:

- To **communicate** to the **sponsor** and external stakeholders a conceptual understanding of the intended IT system
- To provide a **high-level shared vision** of the architecture and scope of the proposed IT system for the development teams
- To explore and **evaluate alternative architectural options**
- To enable early recognition and validation of the **implications of the architectural approach**
- To facilitate **effective communication** between different communities of stakeholders and developers
- To facilitate orientation for new people who join the project

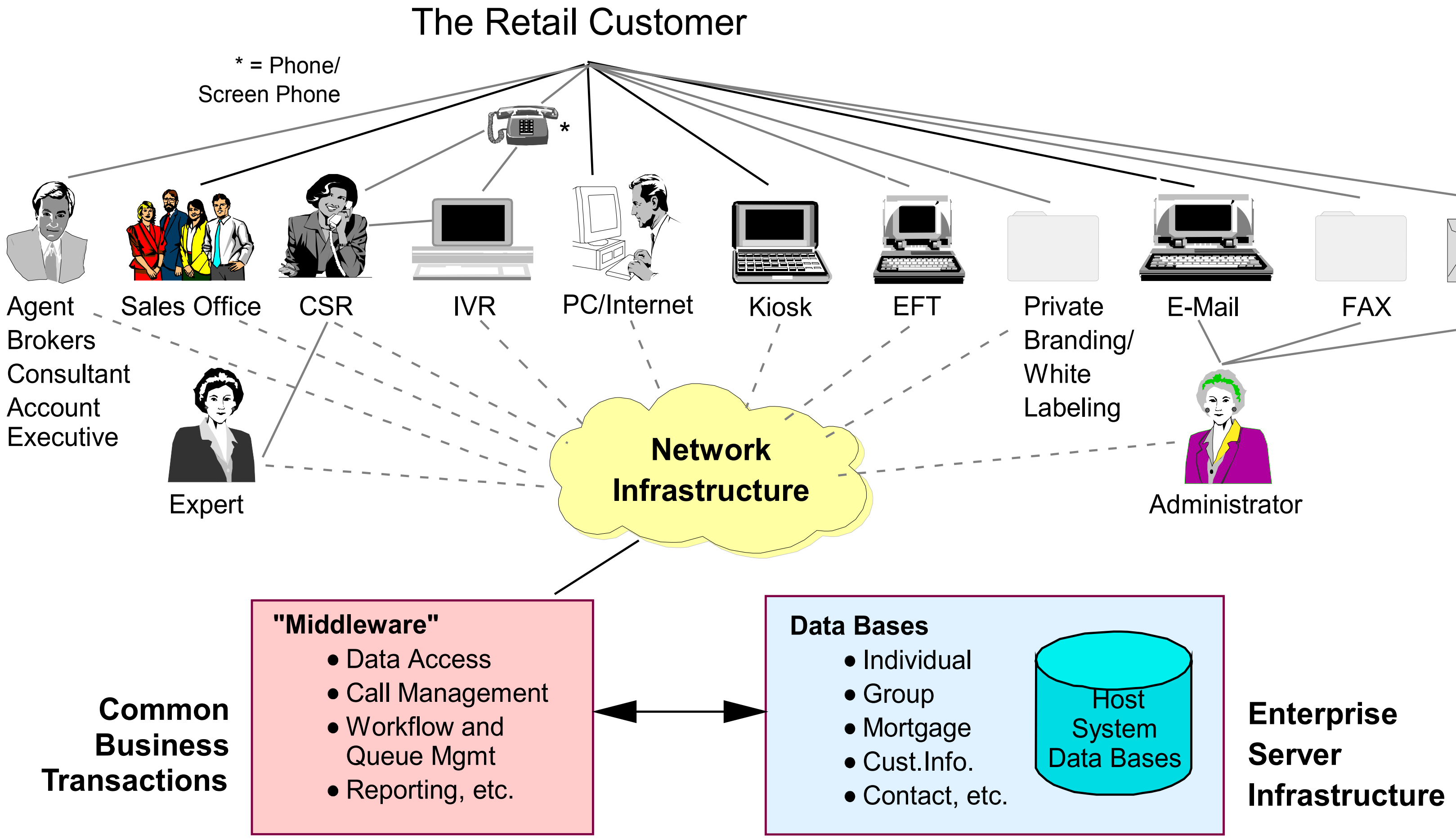
Important things to note:

- An Architecture Overview Diagram contains schematic diagrams that represent the governing ideas and building blocks of an IT system.
- An AOD can include both functional and operational concepts.
- **An AOD is not a model**

Where does the Architecture Overview Diagram fit?



Example 1: Retail multi-channel access



The Retail Customer

* = Phone/
Screen Phone

Agent
Brokers
Consultant
Account
Executive

Sales Office

CSR

IVR

PC/Internet

Kiosk

EFT

Private
Branding/
White
Labeling

E-Mail

FAX

Mail

Expert

Network
Infrastructure

Administrator

"Middleware"

- Data Access
- Call Management
- Workflow and Queue Mgmt
- Reporting, etc.

Data Bases

- Individual
- Group
- Mortgage
- Cust.Info.
- Contact, etc.

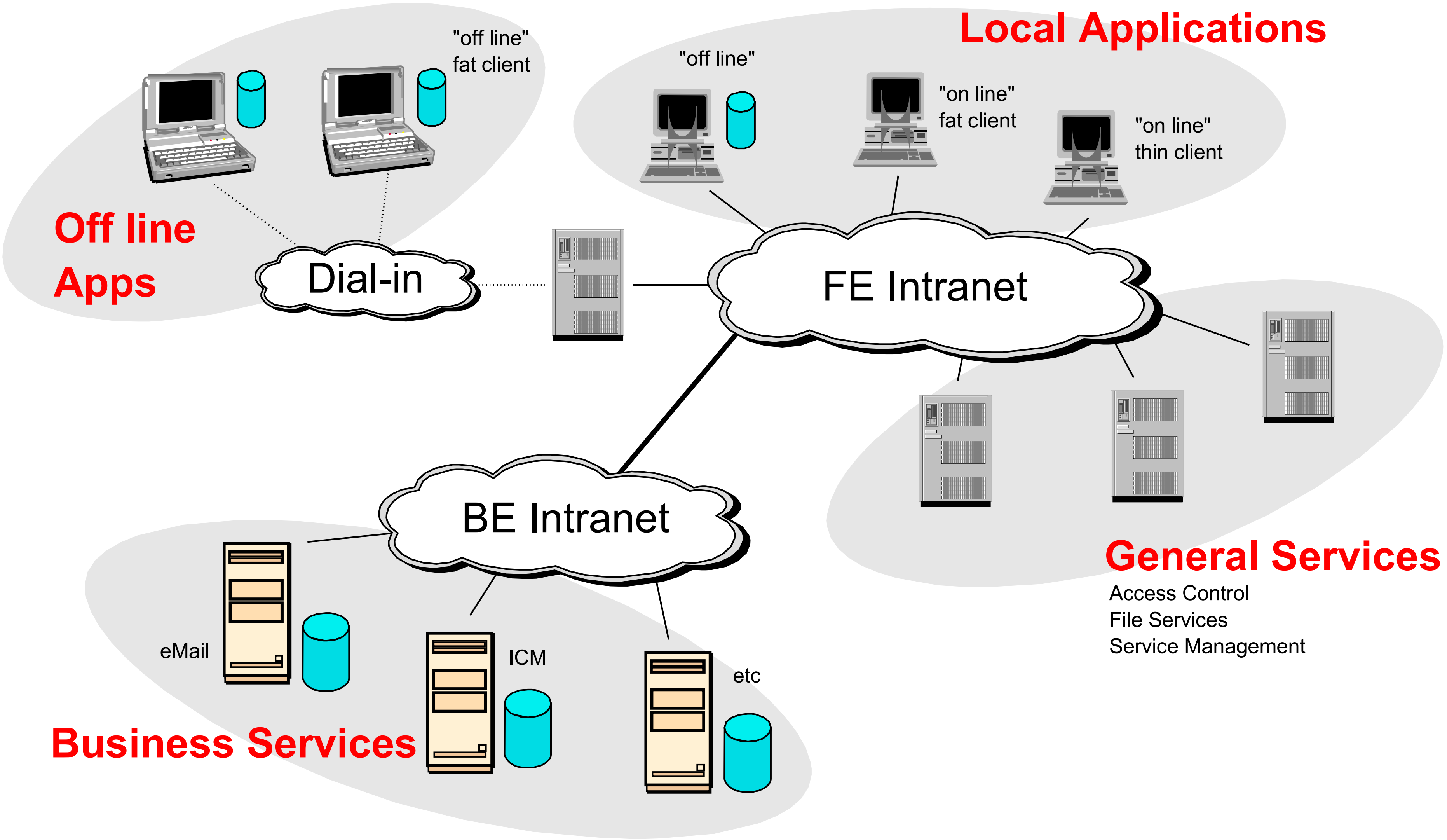
Host
System
Data Bases

**Common
Business
Transactions**

**Enterprise
Server
Infrastructure**

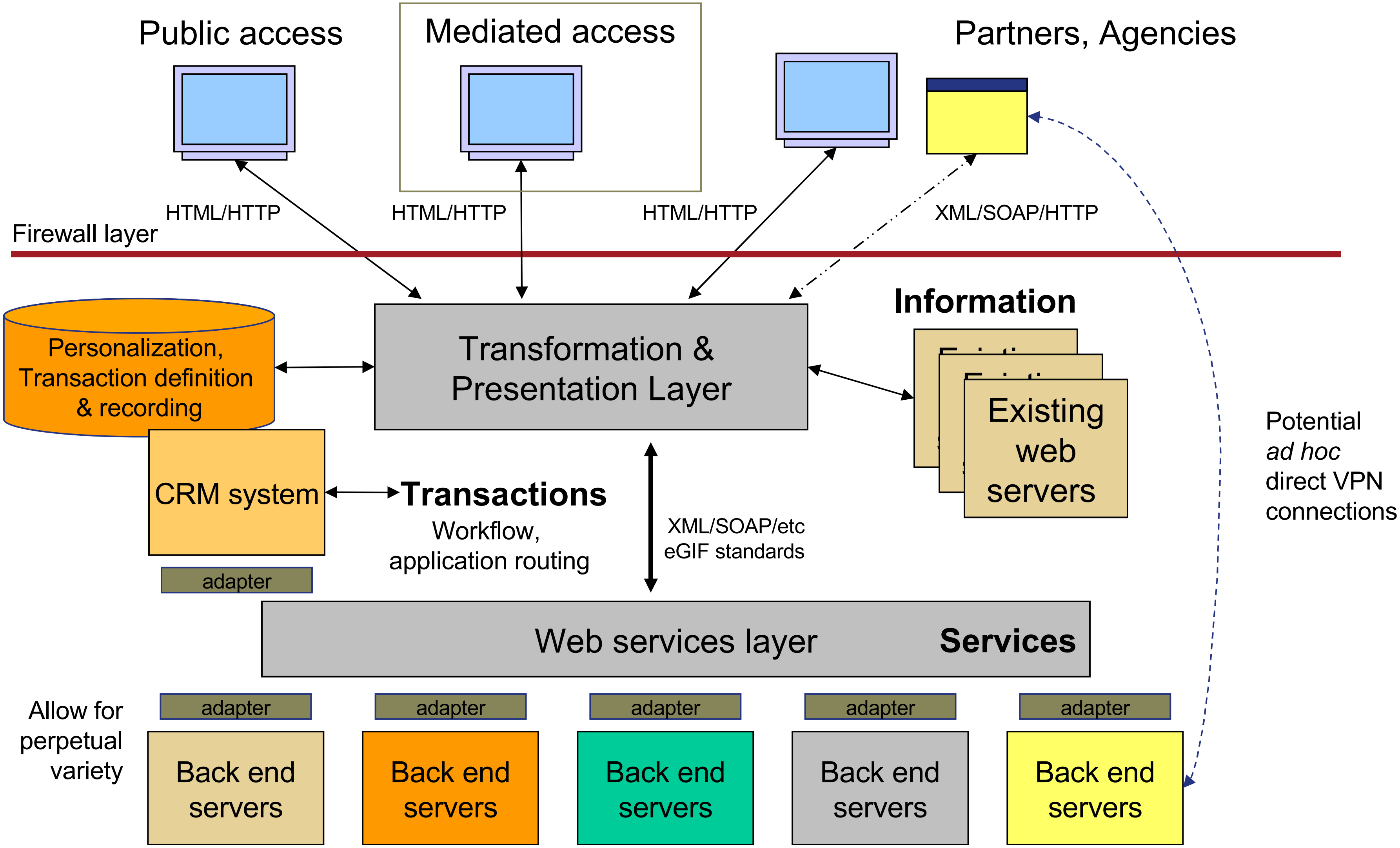
Retail Customer Access Points—The Retail Customer can choose from a variety of ways to interact with the company. The supporting infrastructure should be common whenever possible.

Example 2: Corporate applications

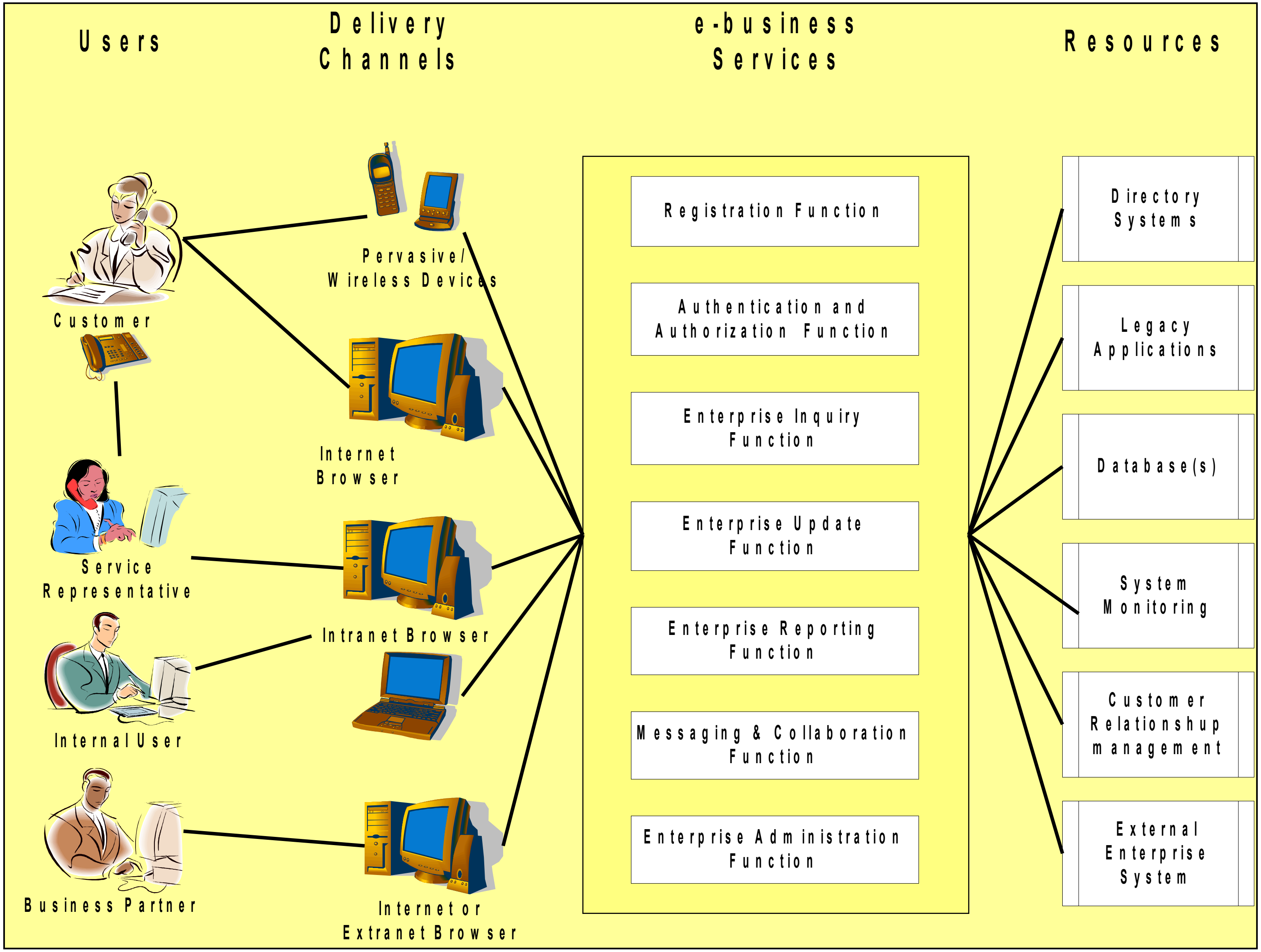


Access Control
File Services
Service Management

Example 3: Local e-government



Example 4: e-business Reference Architecture


Users
Delivery Channels
e-business Services
Resources
Customer
Service Representative
Internal User
Business Partner
Pervasive/ Wireless Devices
Internet Browser
Intranet Browser
Internet or Extranet Browser
Registration Function
Authentication and Authorization Function
Enterprise Inquiry Function
Enterprise Update Function
Enterprise Reporting Function
Messaging & Collaboration Function
Enterprise Administration Function
Directory Systems
Legacy Applications
Database(s)
System Monitoring
Customer Relationship management
External Enterprise System

Component Model

What is a Component?

What is a Component Model?

How do you create one?

The primary concept used for modular design

- Within the software domain, a component can be defined as "…an encapsulated part of a software system that provides a well-defined interface to its services"

- Components are not limited to application components. They can also be:
  - Technical components
  - System software components
  - Hardware components

  H examples?

Components are a formal modelling construct

- Components can be **comprised of other components**

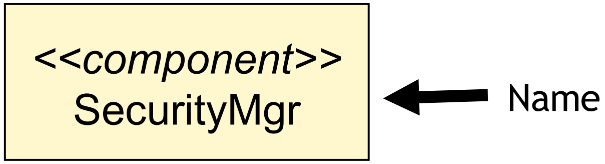
- A subsystem groups components, but cannot be characterized as a component because it does not have interfaces.

- Objects are not very good or useful components

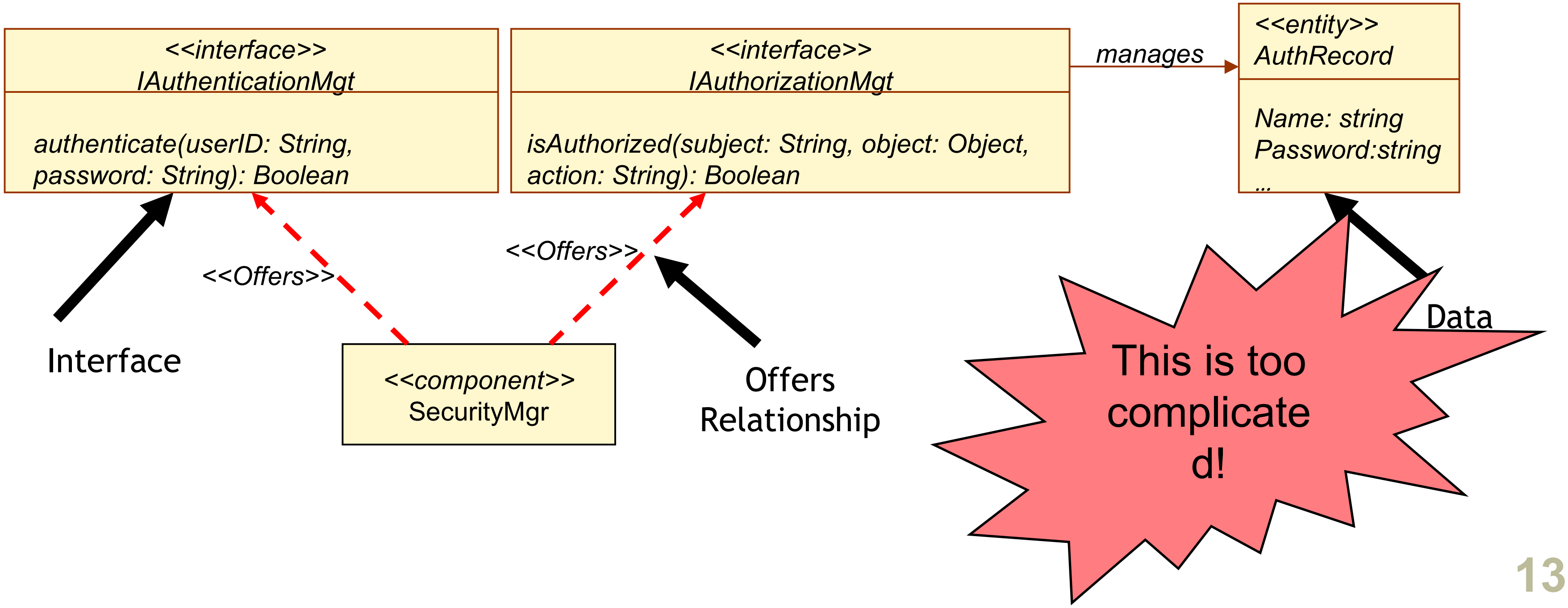
Why?

The notation used to represent components is based on UML

- Component representation uses UML Class notation



- Component interfaces specify their services



The function of an IT System is described by components

- **Components**
  - Are identified based on their responsibilities that collectively achieve the system behavior
- **Component Interfaces**
  - Represent an agreement of the requested services that describes component responsibilities and access to the interfaces' data
- A component is **developed through several stages**, including:
  - Component **identification**
  - Component **specification**
  - Component **realisation**

Component Models include two types of diagram

- **Component Relationship Diagram** (Static Model)
  - Is represented by a variation of the UML Class Diagram

- **Component Interaction Diagram** (Dynamic Model)
  - Depicts component relationships and dependencies
  - Illustrates how components collaborate to achieve system functionality
  - Is represented by a variation of the UML Collaboration or Sequence Diagram

*A Component model is **never just one diagram***

A Component Model is used to describe complex software solutions

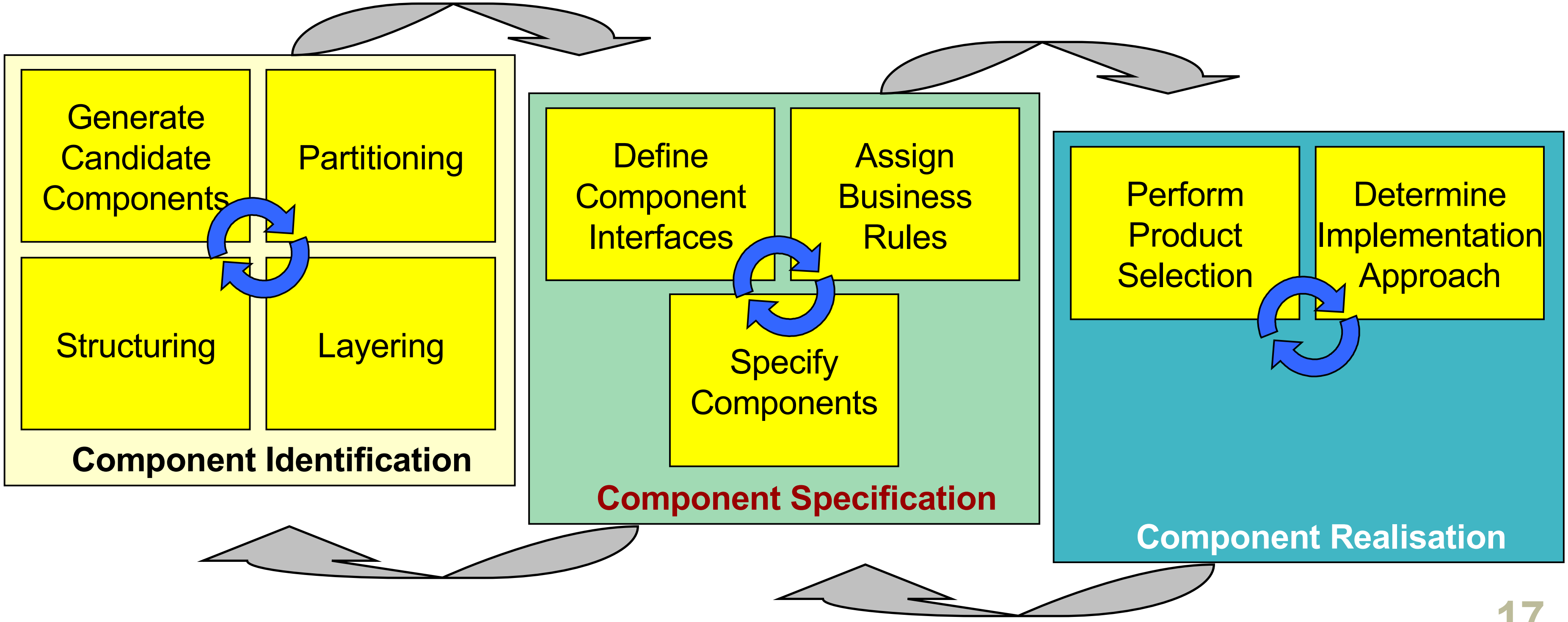
- A Component Model helps to bridge the **gap between requirements and the solution** by:
  - Ensuring that detailed specifications need not be made immediately but can be elaborated over a period of time
  - Mandating the main design principles and overall structure

- The Component Model achieves this by **defining smaller problem scopes** that can be handed to different teams while encouraging reuse.

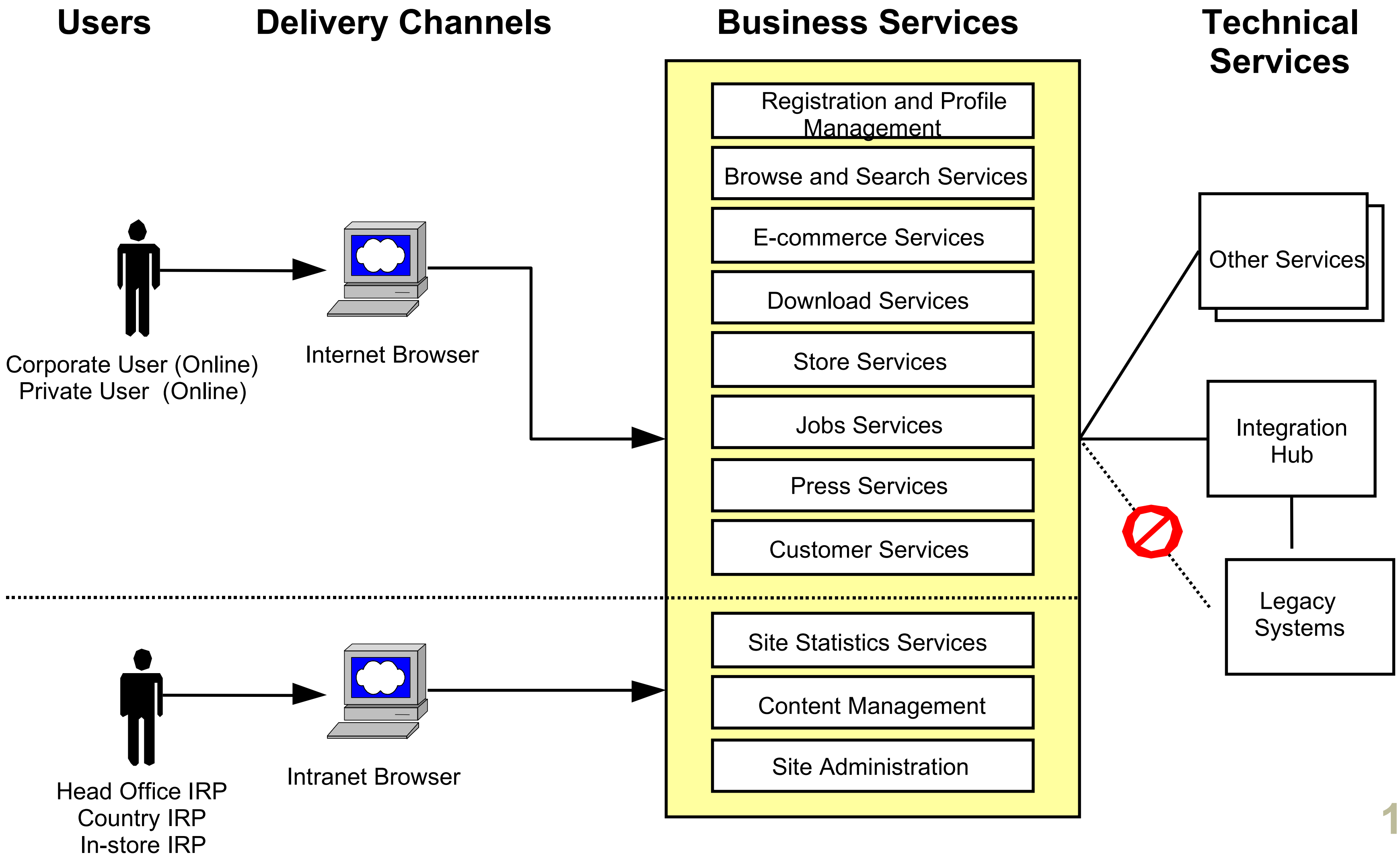
- Each of these problem scopes can then have an associated:
  - Analysis and detailed design
  - Implementation
  - Logical and physical database model

Component modeling is divided into three stages

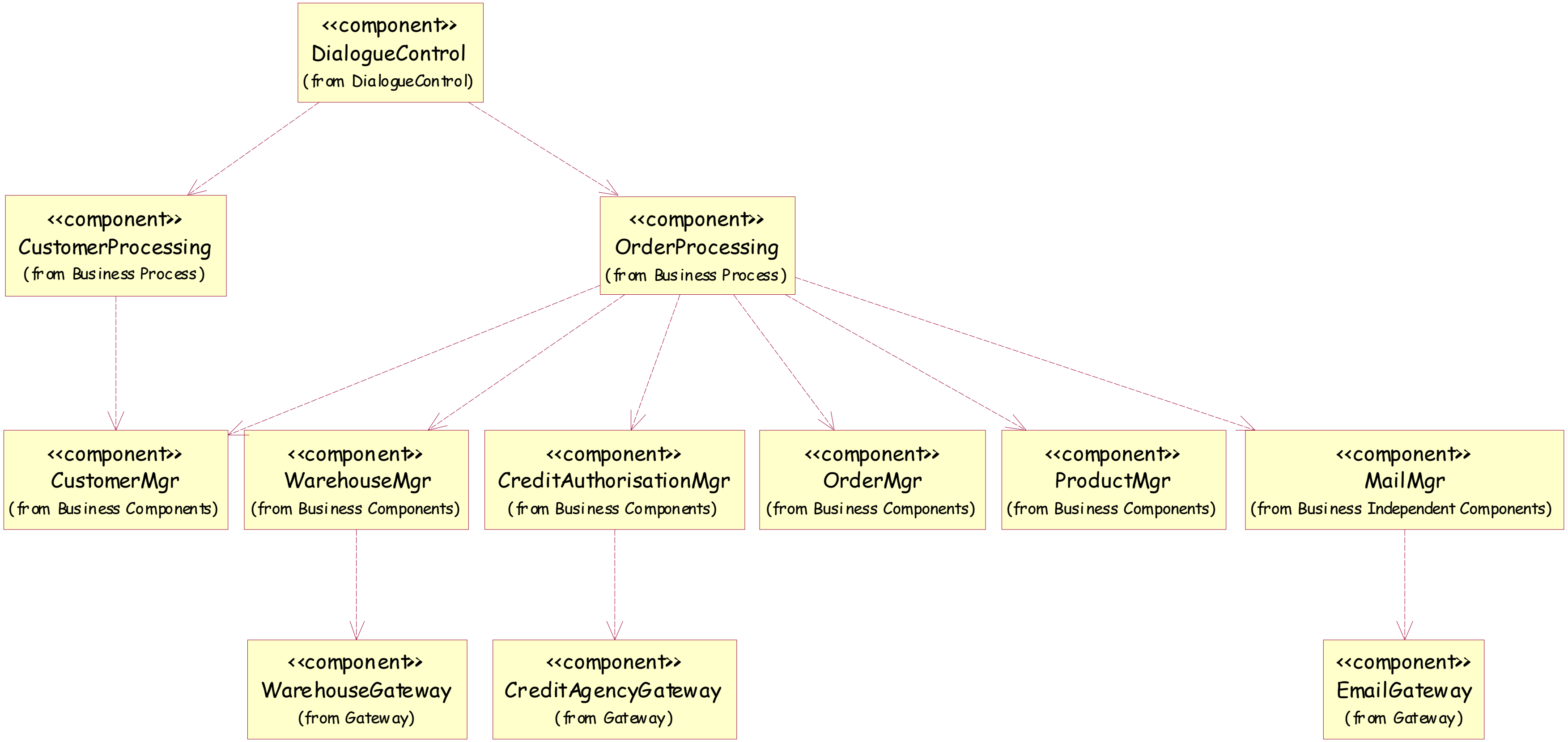
- High level design focuses on component **identification**
- Detailed design deals with component **specification**
- Development deals with component **realisation**



The Architecture Overview Diagram of a Home Shopping Example



The Component Relationship Diagram shows the static relationships among components

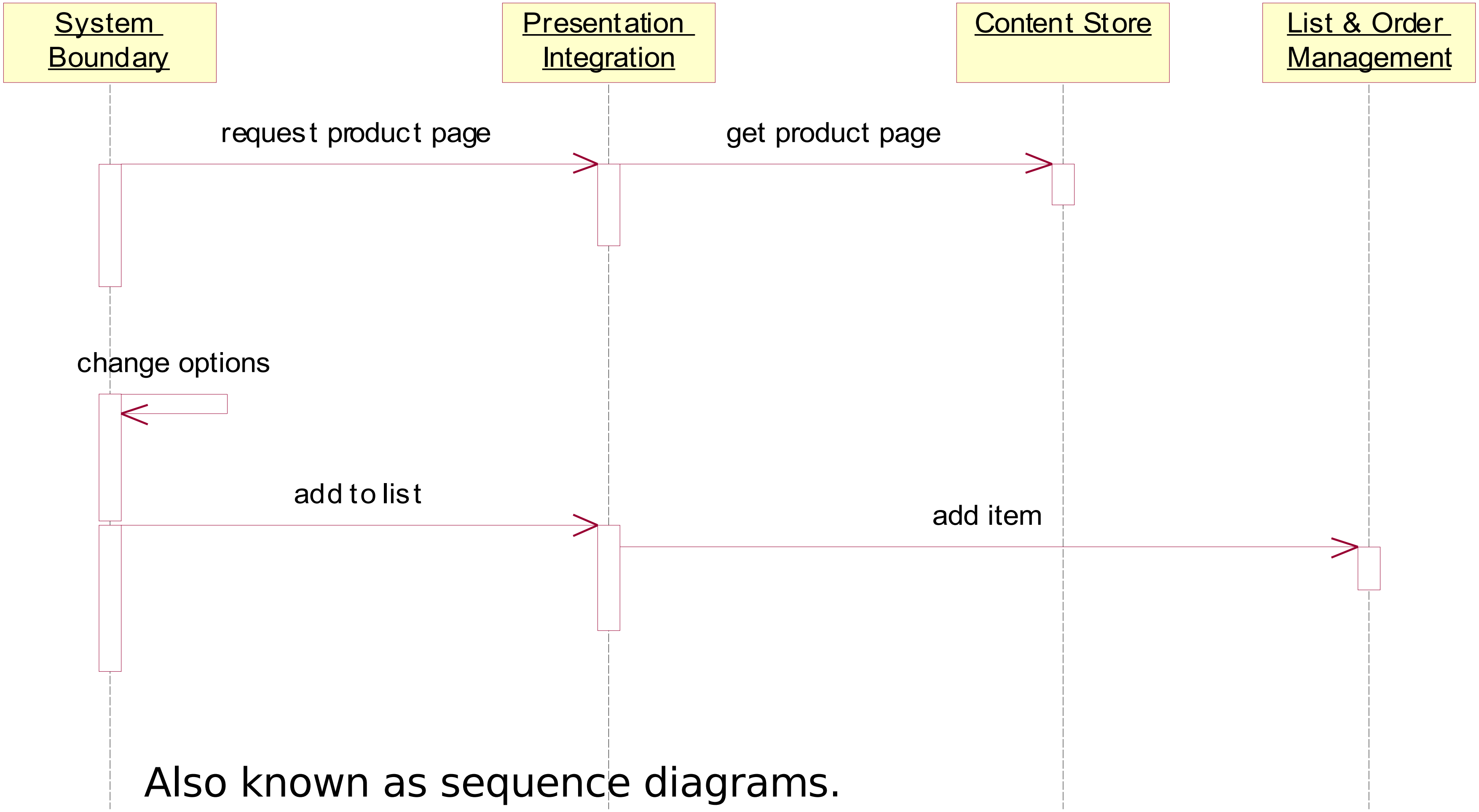


Components are identified, named and their responsibilities are described

- <COMP-001> ProductMgr

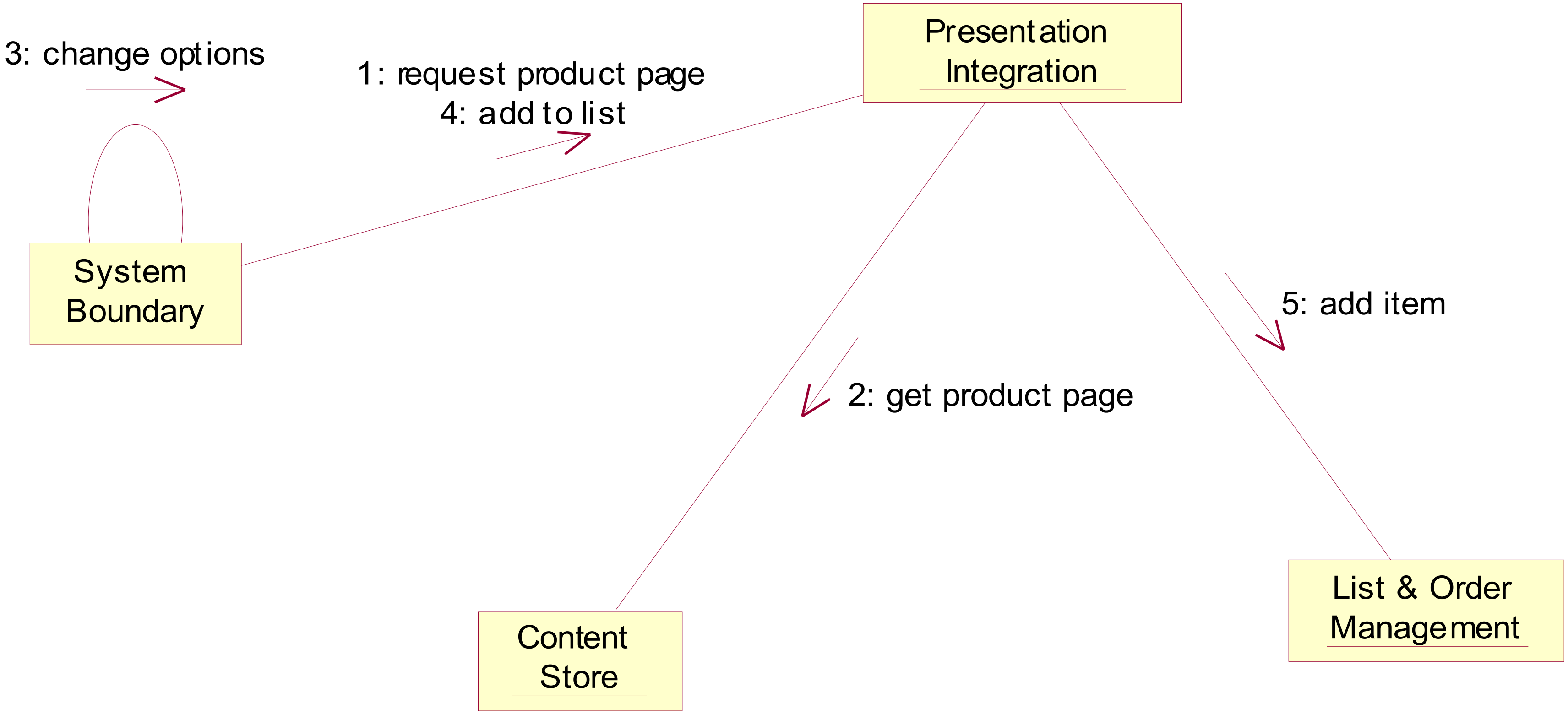
- The product manager component is responsible for interacting with back-end systems and providing product, article, and category information. Conceptually, the component performs a batch job at a set schedule, performing the following actions:
  - Querying back-end systems for new or updated products/articles (items)
  - Extracting information from the back-end system
  - Possibly transforming or filtering the information
  - Responding to real-time queries to provide product information

Component Interaction Diagrams show the dynamic relationships among components



Also known as sequence diagrams.

The Component Collaboration Diagram is a different way of looking at the Dynamic Model



Architecture Overview Diagram & Component Model

Summary

Learning Points

- Use an Architecture Overview Diagram to provide effective communication between different communities of stakeholders and developers

- An Architecture Overview Diagram is not a model

- Components are the software building-blocks of an IT system, providing services through their interfaces.
- Component Models describe the static relationships and the dynamic interactions between components