Slide Deck: Lecture 1 - Requirements Elicitation & Modeling

1. Title Slide: The Foundation

- **Title:** Software Architecture: Lecture 1
- **Subtitle:** Requirements Elicitation & Modeling
- **Focus:** Laying the Architectural Foundation for **ShopSphere** (E-commerce Platform)

---

2. What is Software Architecture?

- **Definition:** Architecture is the **set of design decisions** that, once made, are difficult to change. It defines the structure, components, relationships, and principles/guidelines governing the design and evolution of the system.
- **The Architect's Goal:** Satisfy the system's **Quality Attributes** (Non-Functional Requirements) while meeting the functional needs.
- **Analogy:** It's the blueprint of a building, not the interior decorating.

---

3. The Starting Point: The ShopSphere Scenario

- **Case Study: ShopSphere** - A modern, high-traffic, secure, online retail platform.
- **Business Goal:** Provide a seamless shopping experience, handle high transaction volumes, and integrate reliable payment processing.
- **Key Challenge:** Must be **highly available** and **scalable** to manage seasonal peak loads (e.g., Black Friday).

---

4. Knowledge Base: The Four Types of Requirements

Requirement Type	**What is it?**	**Example (ShopSphere)**
Functional (FRs)	What the system **must do** (features).	Process a payment; Search the product catalog.
Non-Functional (NFRs)	How well the system performs a function (Quality Attributes).	Response time must be under 2 seconds.

Requirement Type	What is it?	Example (ShopSphere)
Domain	Constraints imposed by the operating environment (business rules).	All transactions must be secured via PCI compliance.
ASRs (Architecturally Significant Requirements)	NFRs/FRs that critically impact the choice of architecture.	**High Availability** (99.9% uptime).

---

5. Activity 1: Eliciting Functional Requirements (FRs)

- **Goal:** Define the system's external behavior.
- **Practical Activity:** Students list core features for the **Customer** and **Administrator**.
- **Customer FRs (Top 3):**
    - F1: Register/Log in securely.
    - F2: Add/Remove items from a persistent shopping cart.
    - F3: Complete multi-step checkout with payment processing.
- **Admin FRs (Top 2):**
    - F4: Manage product inventory (CRUD).
    - F5: View and update order statuses (e.g., Shipped).

---

6. Activity 2: Eliciting Non-Functional Requirements (NFRs)

- **Goal:** Define Quality Attributes that constrain the system's design.
- **Practical Activity:** Defining measurable metrics.
- **NFR Examples:**
    - **Performance:** Search results must load in **< 1.5 seconds**.
    - **Scalability:** Must support **10,000 concurrent users** during peak load.
    - **Security:** All financial data must be encrypted **end-to-end** (PCI DSS compliance).
    - **Availability:** The payment system must have **99.99% uptime**.

---

7. Knowledge Base: Architecturally Significant Requirements (ASRs)

- **The Filter:** ASRs are the requirements that *if not met*, the system is deemed a failure, and they directly force a choice between major patterns (Monolith, Microservices, etc.).
- **Why ASRs Matter:** They become the **primary drivers** for architecture design.

---

8. Activity 3: Defining the ShopSphere ASRs

- **Goal:** Select three core NFRs/FRs that dictate architecture.

ASR	Requirement Statement	Architectural Impact
ASR 1: Scalability	The system must handle **10,000 concurrent sessions** with peak latency under 2s.	**Forces Distributed Architecture:** Requires load balancing, potential service decomposition (Microservices).
ASR 2: Fault Isolation	Failure in the Notification System must **not prevent** order completion.	**Forces Decoupling:** Requires asynchronous communication (Queues/EDA) to ensure core transaction completion.
ASR 3: Data Security	Customer passwords and payment details must comply with **current industry standards** (e.g., OAuth 2.0).	**Forces Gateway Pattern:** Requires a single point for authentication/authorization checking before routing.

---

9. Knowledge Base: The Four Views of Architecture

- **Purpose of Views:** To document the architecture from different perspectives for different stakeholders.
- **The 4+1 View Model (Kruchten):**
    - **Logical View:** Structure (Components/Classes).
    - **Process View:** Concurrency and synchronization.
    - **Development View:** Organization (Modules/Packages).
    - **Physical (Deployment) View:** Mapping software to hardware.

    - **+1 Use Case View:** The focus of today's modeling exercise.

---

10. Knowledge Base: The Use Case View (UML)

- **Purpose:** Shows the system's boundary and how **Actors** interact with the system's functionalities (**Use Cases**). It's the simplest view, defining *what* the system does from an external perspective.
- **Key Elements:**
    - **Actor (Stick Figure):** A user or external system interacting with the system.
    - **Use Case (Oval):** A sequence of actions performed by the system to yield an observable result of value.
    - **System Boundary (Box):** Defines the scope of the software.

---

11. Modeling Relationships: Include and Extend

- **Include $\ll\text{include}\gg$ (Required):**
    - *Concept:* A mandatory sub-functionality.
    - *Example:* "Make Purchase" **includes** "Secure Payment." (You must pay to purchase).
- **Extend $\ll\text{extend}\gg$ (Optional/Alternative):**
    - *Concept:* An optional step that may be executed under specific conditions.
    - *Example:* "Make Purchase" **extends** "Apply Coupon." (Applying a coupon is optional).

---

12. Activity 4: Modeling the ShopSphere Context

- **Goal:** Define the main actors and the system boundary.
- **Practical Activity:** Sketching the initial diagram.
- **System:** ShopSphere E-commerce Platform (the big box).
- **Actors:**

1. Web Customer
2. Administrator
3. (External) Payment Gateway

---

13. Activity 4 (Cont.): Defining Core Use Cases

- **Customer Use Cases:**
    - Browse Products
    - Manage Shopping Cart
    - Make Purchase
- **Administrator Use Cases:**
    - Manage Catalog
    - Process Customer Orders

---

14. Activity 5 (Proof of Concept): Detailing the Checkout Process

- **Goal:** Model the complexity of the "Make Purchase" use case using relationships.
- **Steps:**
    1. Draw the primary Use Case: **Make Purchase**.
    2. Draw the mandatory step: **Secure Payment** $\ll\text{include}\gg$ from Make Purchase.
    3. Draw the mandatory external step: **Process Credit Card** $\ll\text{include}\gg$ from Secure Payment (External dependency on Payment Gateway).
    4. Draw the optional step: **Apply Discount Code** $\ll\text{extend}\gg$ Make Purchase.

---

15. Practical Activity (Hands-On): Drawing the Diagram

- **Tool Setup:** Install **draw.io Desktop** or navigate to **diagrams.net** online.
- **Steps:**

1. Select the **UML** stencil set.
2. Create a rectangle for the **System Boundary**.
3. Drag and drop **Actor** shapes and name them.
4. Drag and drop **Oval (Use Case)** shapes.
5. Use **Dashed Arrows** for relationships, labeling them $\ll\text{include}\gg$ or $\ll\text{extend}\gg$.

---

16. Self-Correction/Verification (Debugging the Model)

- **Checklist:**
  - Did I include at least one external actor (e.g., Payment Gateway)?
  - Is the dependency flow correct? (The main case points to the include/extend cases).
  - Is the system boundary clearly separating internal functionality from external interaction?
  - Are the Use Cases representing **value** to the actor (e.g., "Manage Shopping Cart" is better than "Delete Item")?

---

17. Project Integration: Linking to Subsequent Lectures

- **The ASRs (Scalability, Fault Isolation, Security)** defined today will be the *metrics* used to choose and evaluate our architectures in Lectures 2, 4, and 8.
- **Lecture 2 Focus:** We will use the **Make Purchase** Use Case to model the internal structure of the Monolith (Layered Architecture).

---

18. Recap: Three Key Takeaways

1. **Architecture is Driven by NFRs:** The "how well" (NFRs) is often more important than the "what" (FRs).
2. **ASRs Dictate Patterns:** Requirements like high scalability force choices away from simple solutions.

3. **UML Use Case View** provides the initial contract between the system and the outside world.

---

19. Summary of Deliverables for Submission

- **Document 1:** Table listing the **three ASRs** and their justification/architectural impact.
- **Document 2:** The final **UML Use Case Diagram** (.png or .pdf export from draw.io) detailing the *Make Purchase* process, including $\ll\text{include}\gg$ and $\ll\text{extend}\gg$.

---

20. Q&A and Next Steps

- **Questions?** (Open floor for discussion on ASR justification).
- **Pre-work:** Research the **Layered Architecture Pattern** (Monolith) vs. **Microservices Pattern**.
- **Next Lecture: Lecture 2: Layered Architecture Pattern (Monolith Design)**. We will design the internal components for the Monolith.