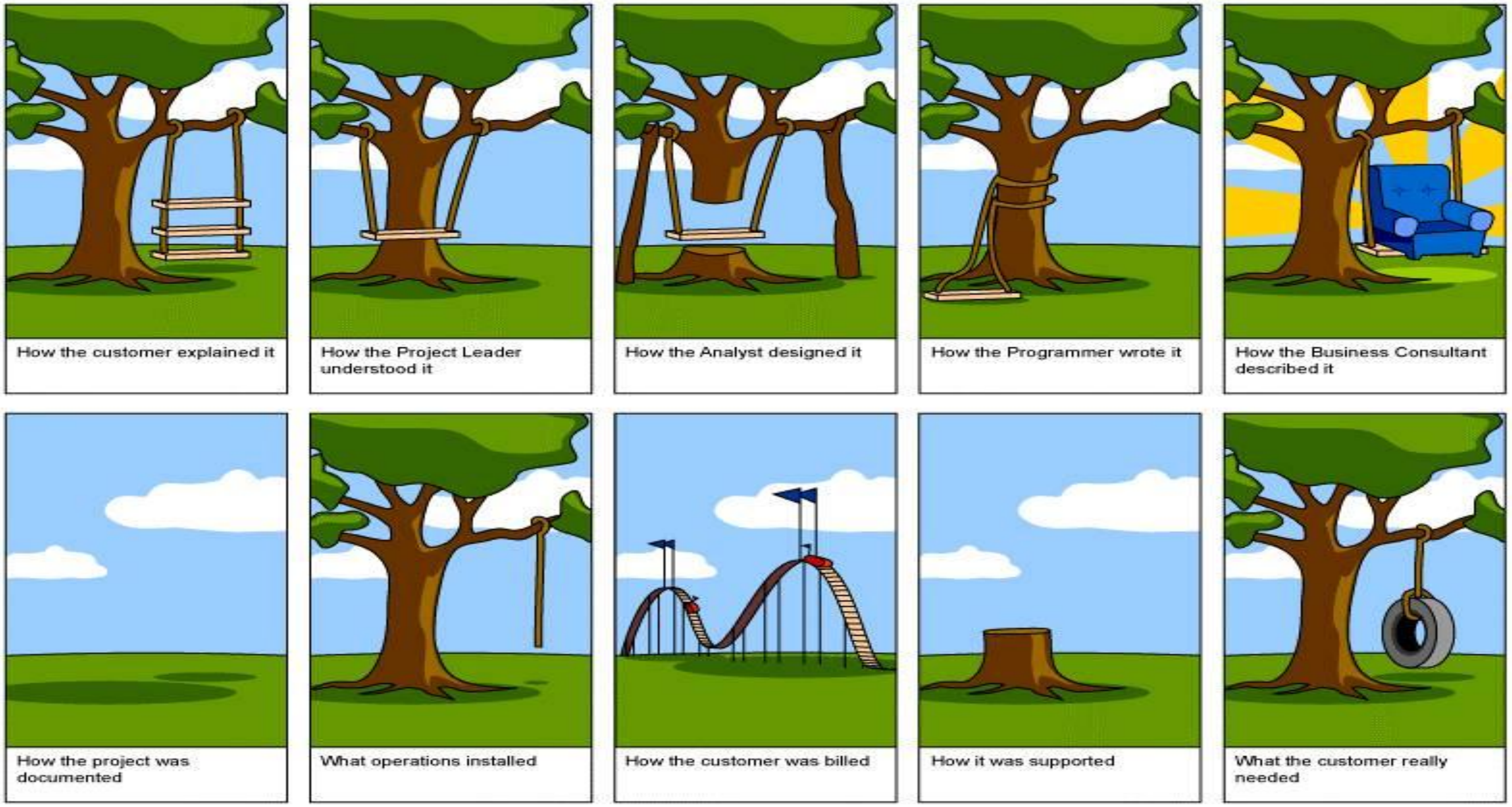
Requirements Analysis

Session 12

INFM 603

Different Perspectives on Design


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Thanks to Satish Mishra

The System Life Cycle

- Systems analysis
  - How do we know what kind of system to build?
- User-centered design
  - How do we discern and satisfy user needs?
- Implementation
  - How do we build it?
- Management
  - How do we use it?

Systems Analysis

- First steps:
  - Understand the task
    - Limitations of existing approaches
  - Understand the environment
    - Structure of the industry, feasibility study
- Then identify the information flows
  - e.g., Serials use impacts cancellation policy
- Then design a solution
  - And test it against the real need

What are Requirements?

- Attributes
  - Appearance
  - Concepts (represented by data)

- Behavior
  - What it does
  - How you control it
  - How you observe the results

Types of Requirements

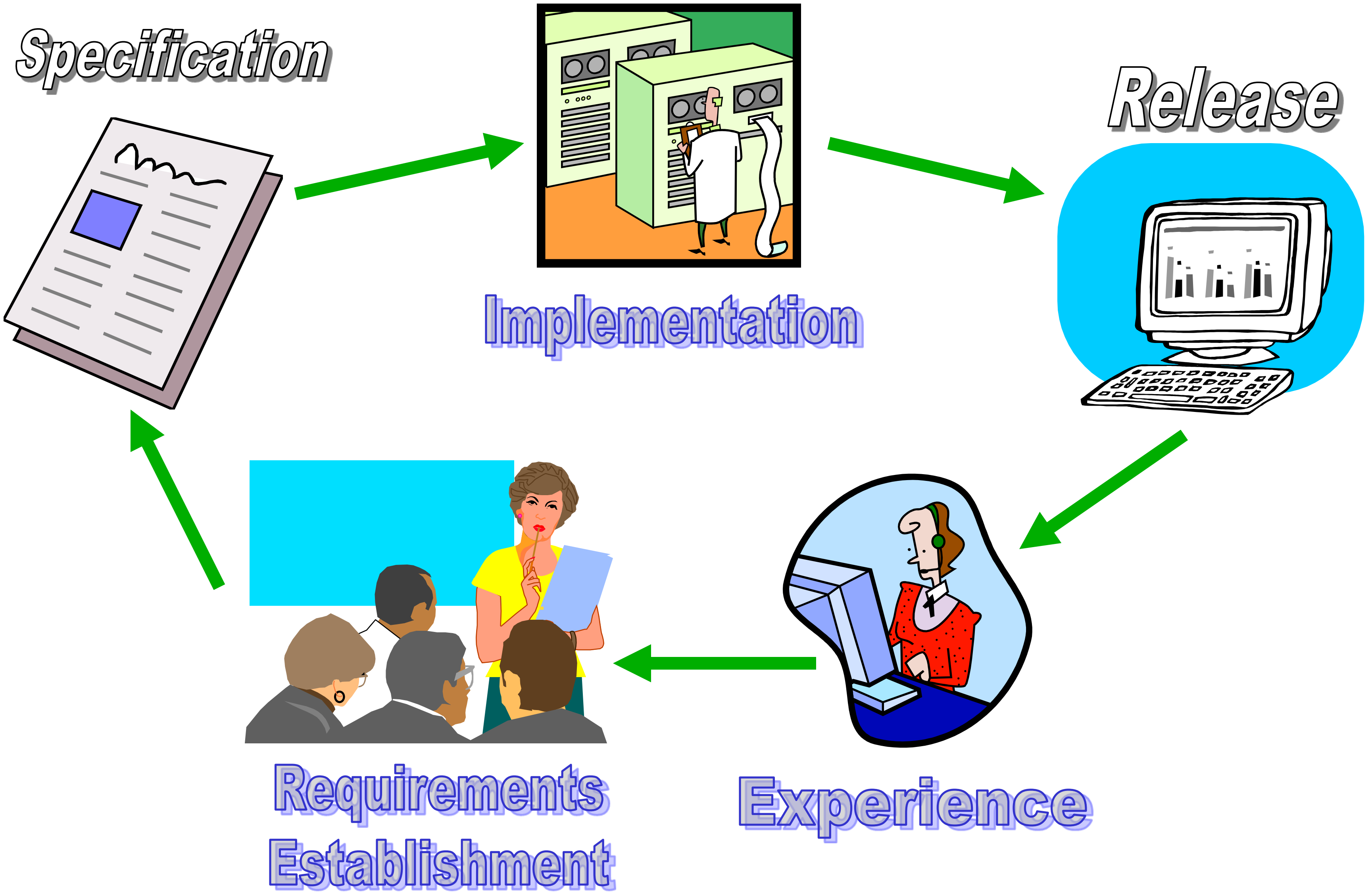
- User-centered
  - Functionality
- System-centered
  - Availability
    - Mean Time Between Failures (MTBF)
    - Mean Time To Repair (MTTR)
  - Capacity
    - Number of users for each application
    - Response time
  - Flexibility
    - Upgrade path

Who Sets the Requirements?

- People who need the task done (customers)
- People that will operate the system (users)
- People who use the system’s outputs
- People who provide the system’s inputs
- Whoever pays for it (sponsor)

The Waterfall Model

Agile Methods



The Requirements Interview

- Focus the discussion on the <u>task</u>
  - Look for <u>objects</u> that are mentioned
- Discuss the system's most important <u>effects</u>
  - Displays, reports, data storage, device control, …
- Learn where the system's <u>inputs</u> come from
  - People, stored data, devices, …
- Note any data that is mentioned
  - Try to understand the <u>structure</u> of the data
- Shoot for the big picture, not every detail

Analyze the Information Flows

- Where does information originate?
  - Might come from multiple sources
  - Feedback loops may have no identifiable source
- Which parts should be automated?
  - Some things are easier to do without computers
- Which automated parts should be integrated?
- What existing systems are involved?
  - What information do they contain?
  - Which systems should be retained?
  - What data will require “retrospective conversion”?

Interaction Modality Choices

- Interactive
  - Do it while the user is present
- Batch processing
  - Save it up and do it all at once

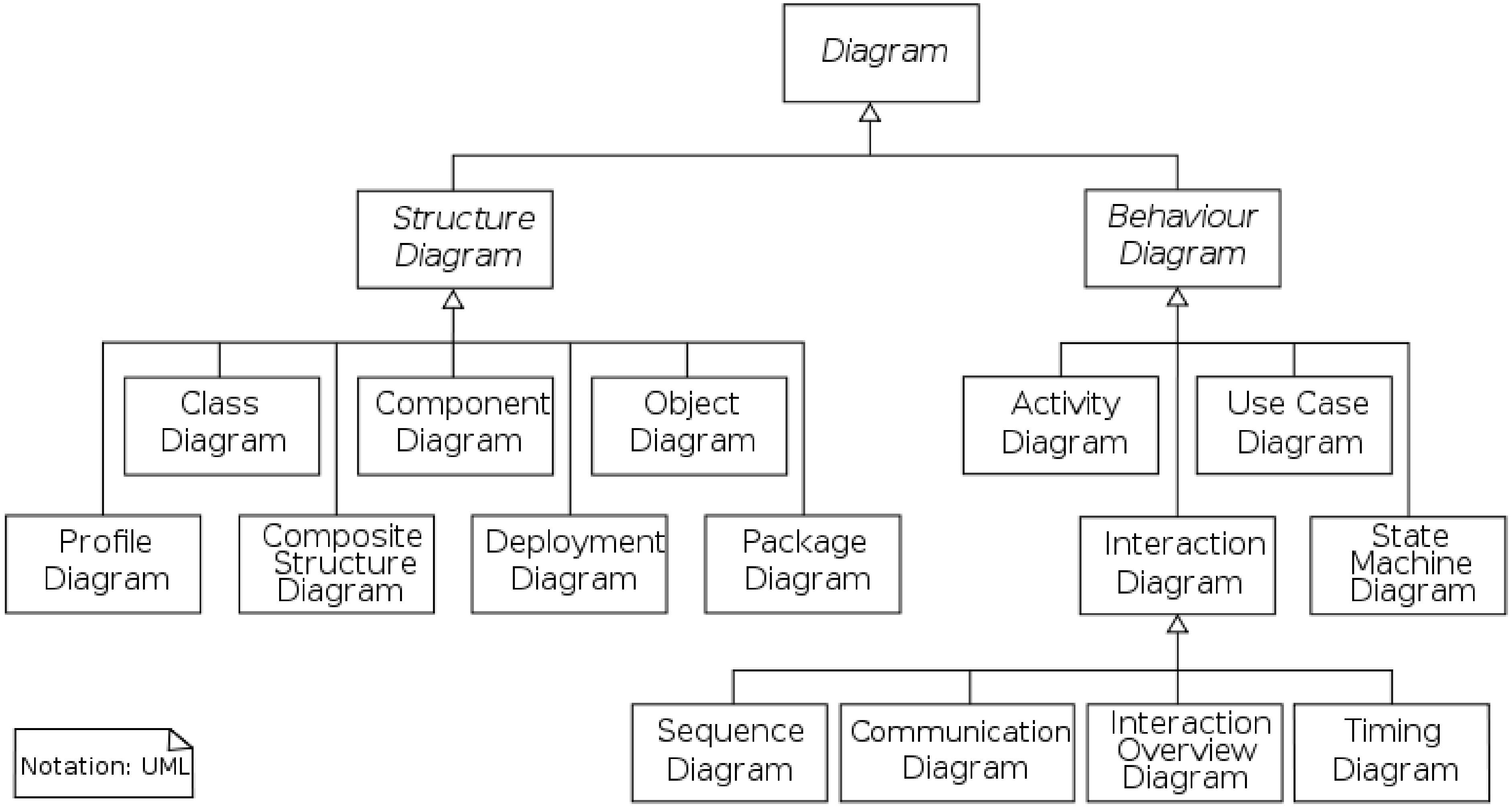
Unified Modeling Language

- Real systems are more complex than anyone can comprehend

- Key idea: Progressive refinement
  - Carve the problem into pieces
  - Carve each piece into smaller pieces
  - When the pieces are small enough, code them

- UML provides a <u>formalism</u> for doing this
  - But it does not provide the <u>process</u>

Unified Modeling Language



Specifying Structure

- Capturing the big picture
  - Use case diagram (interactions with the world)
  - Narrative
  - Scenarios (examples to provoke thinking)

- Designing the object structure
  - Class diagram (“entity-relationship” diagram)
  - Object diagram (used to show examples)

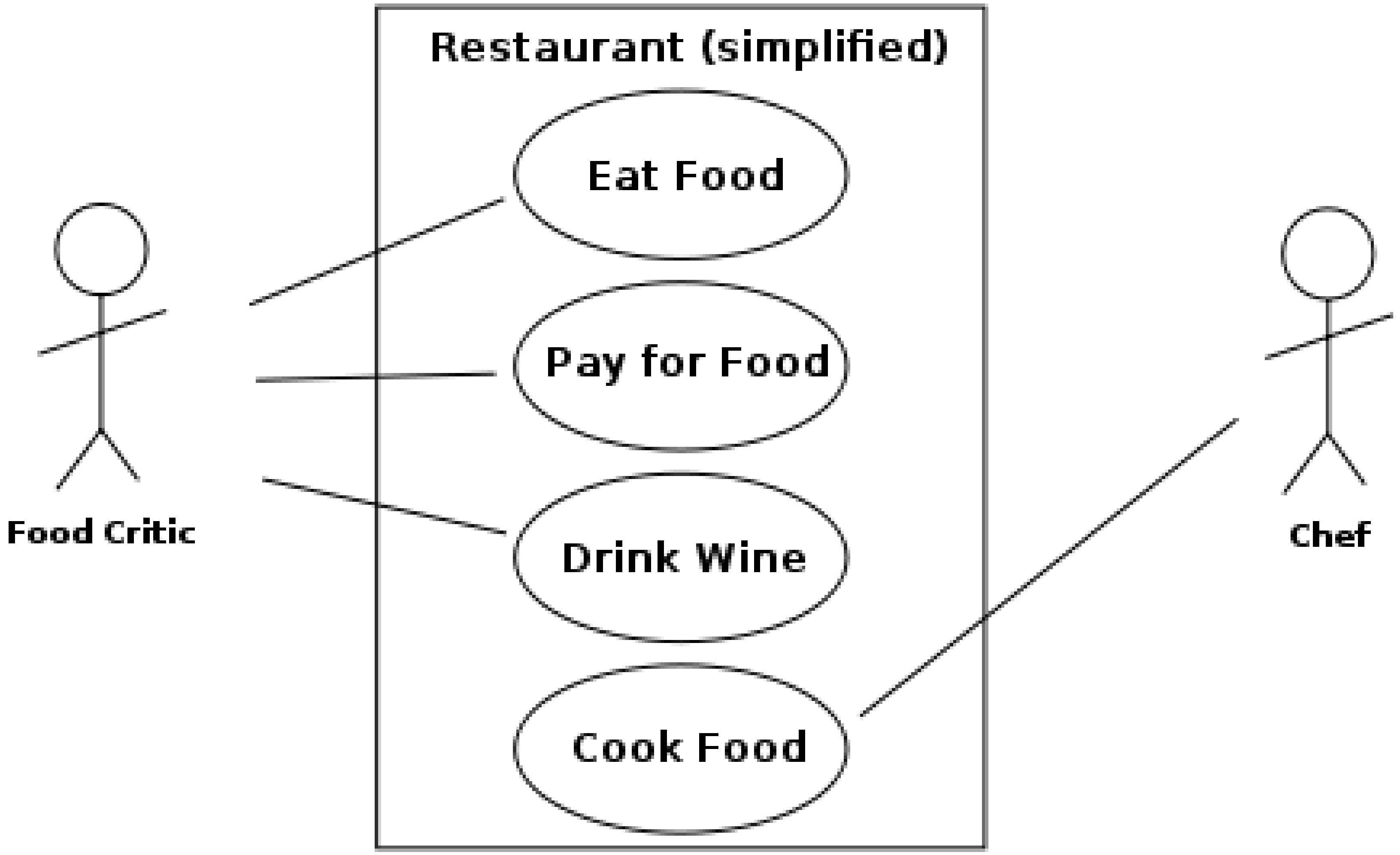
Specifying Behavior

- Represent a candidate workflow
  - Activity diagram (a “flowchart”)
- Represent object interactions for a scenario
  - Collaboration diagram (object-based depiction)
  - Sequence diagram (time-based depiction)
- Represent event-object interactions
  - Statechart diagram (a “finite state machine”)

Use Case Design

- Use Case Diagram
  - Input-output behavior
- Use Case Narrative
  - Explains each use case
- Use Case Scenario
  - Activity diagram shows how the use cases are used together

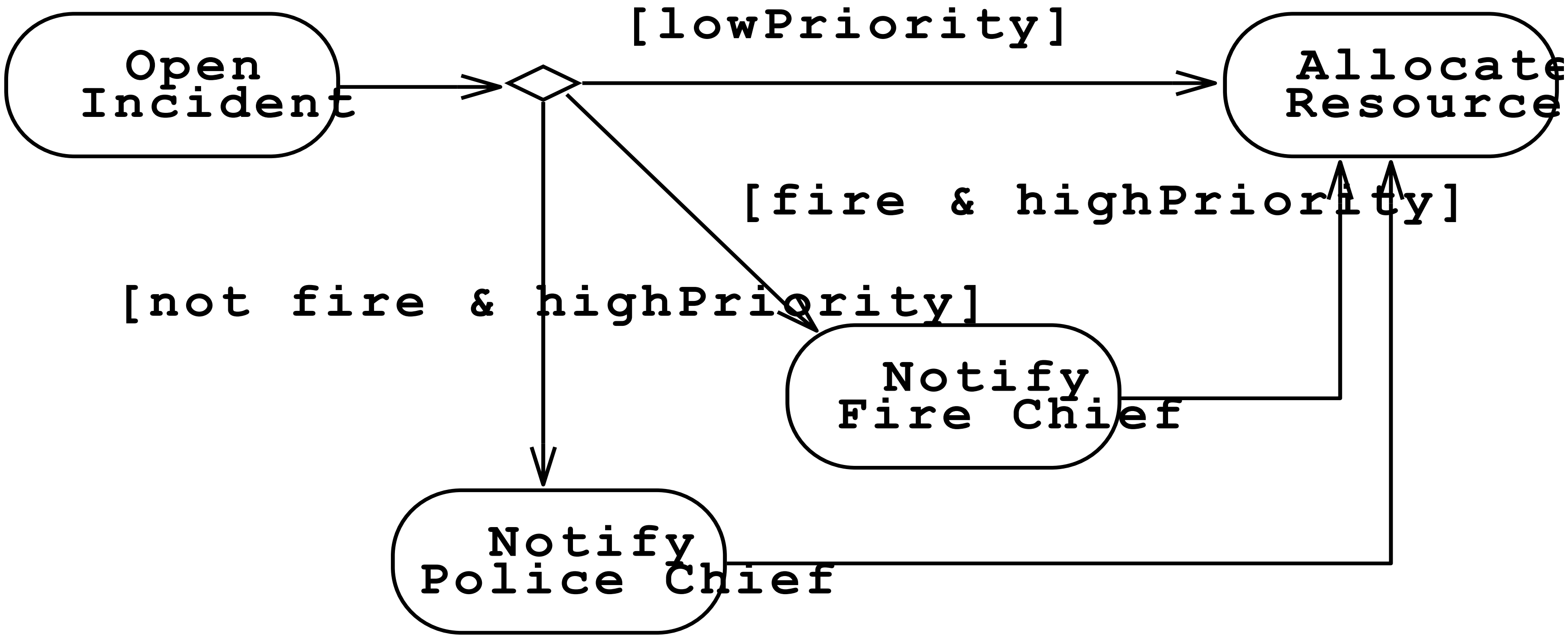
Use Case Diagram



Use Case Diagram

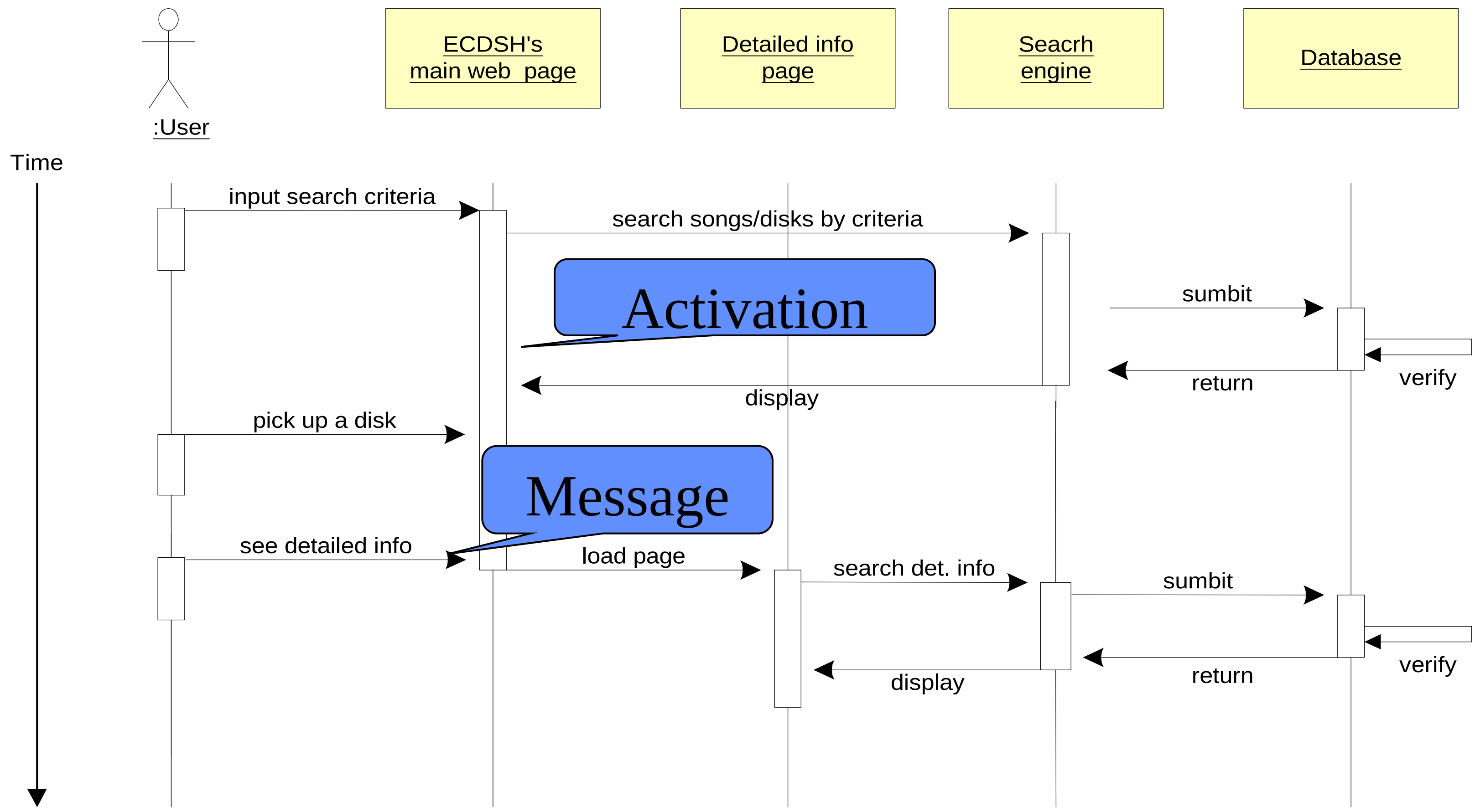
- External “actors”
  - Roles of people
  - Types of systems
- Use cases
  - Top-level functions (solid arrows to/from actors)
- Relationships among use cases
  - Always-depends-on (dashed <<include>>)
  - Sometimes-is-depended-on (dashed <<extend>>)
  - Inherits-from (solid triangle-arrow)

Activity Diagram: Modeling Decisions



Thanks to Satish Mishra

Sequence Diagram



Thanks to Satish Mishra

Good Uses for UML

- Focusing your attention
  - Design from the outside in
- Representing partial understanding
  - Says what you know, silent otherwise
- Validate that understanding
  - Structuring communication with stakeholders

Avoiding UML Pitfalls

- Don’t sweat the notation too much
  - The key is to be clear about what you mean!
- Don’t try to make massive conceptual leaps
  - Leverage abstraction encapsulation
- Don’t get to attached to your first design
  - Goal is to <u>find</u> weaknesses in your understanding

Total Cost of Ownership

- Planning
- Installation
  - Facilities, hardware, software, integration, migration, disruption
- Training
  - System staff, operations staff, end users
- Operations
  - System staff, support contracts, outages, recovery, …

Management Issues

- Policy
  - Privacy, access control, appropriate use, …
- Training
  - System staff, organization staff, “end users”
- Operations
  - Fault detection and response
  - Backup and disaster recovery
  - Audit
  - Cost control (system staff, periodic upgrades, …)
- Planning
  - Capacity assessment, predictive reliability, …

Strategic Choices

- Acquisition
  - Proprietary (“COTS”)
  - Open source
- Implementation
  - Integrate “Best-of-breed” systems
  - “One-off” custom solution

Open Source "Pros"

- More eyes ⇒ fewer bugs
- Iterative releases ⇒ rapid bug fixes
- Rich community ⇒ more ideas
  - Coders, testers, debuggers, users
- Distributed by developers ⇒ truth in advertising
- Open data formats ⇒ Easier integration
- Standardized licenses

Open Source “Cons”

- Communities require incentives
  - Much open source development is underwritten
- Developers are calling the shots
  - Can result in feature explosion
- Proliferation of “orphans”
- Diffused accountability
  - Who would you sue?
- Fragmentation
  - “Forking” may lead to <u>competing</u> versions
- Little control over schedule

Open Source Business Models

- Support Sellers

  Sell distribution, branding, and after-sale services.

- Loss Leader

  Give away the software to make a market for proprietary software.

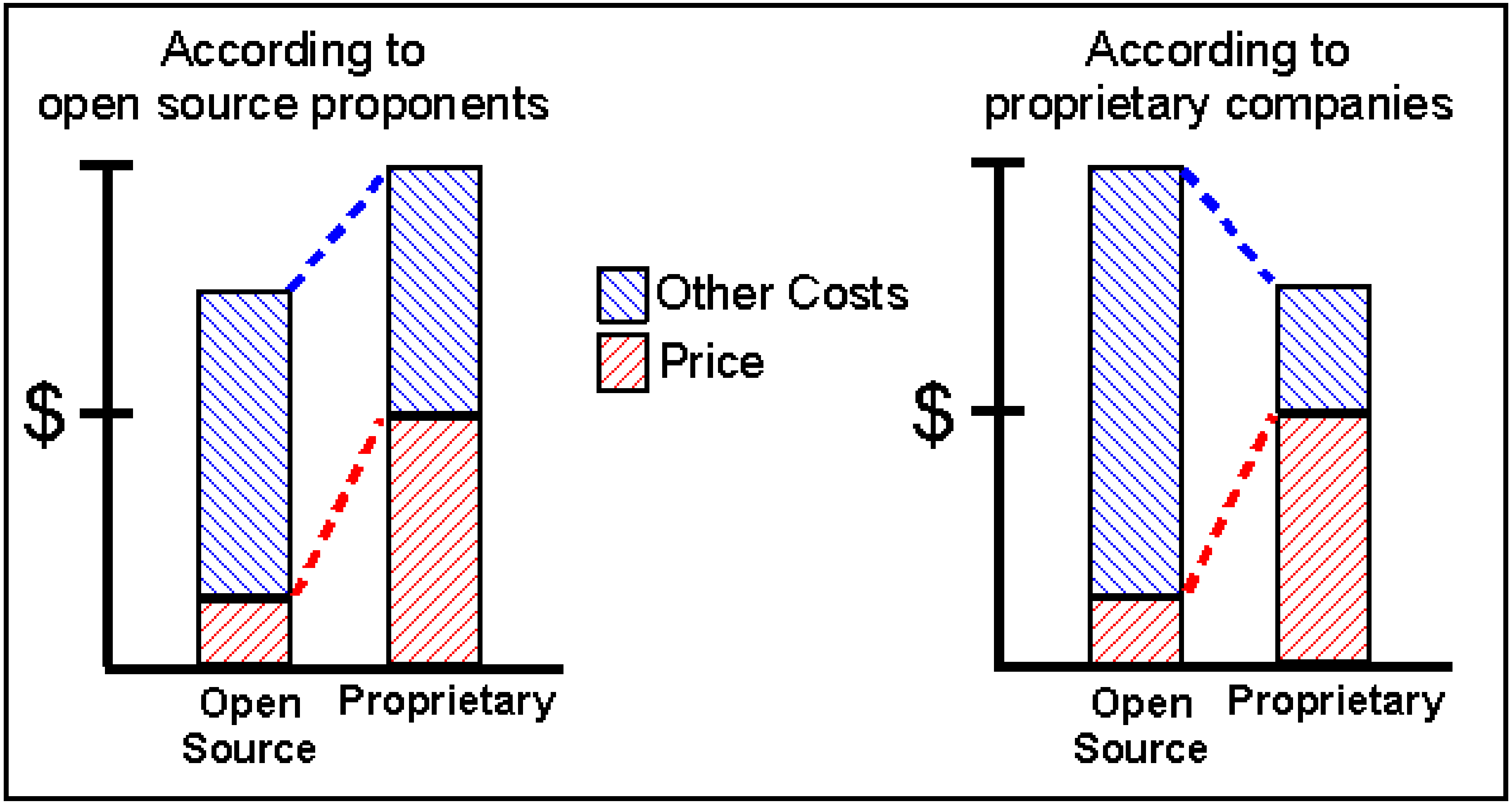
- Widget Frosting

  If you’re in the hardware business, giving away software doesn’t hurt.

- Accessorizing

  Sell accessories:
  books, compatible hardware, complete systems with pre-installed software

Total Cost of Ownership



Summary

- Systems analysis
  - Required for complex multi-person tasks
- User-centered design
  - Multiple stakeholders complicate the process
- Implementation
  - Architecture, open standards, …
- Management
  - Typically the biggest cost driver