15-413

Requirements Elicitation

Bernd Bruegge
Carnegie Mellon University
School of Computer Science

3 September 1998

Defining the System Boundary: What do you see?

Odds and Ends

- Nondisclosure Form:
  - **Read it and return it signed after class, or latest,on Tuesday**
  - **You cannot participate unless you have signed the NDA**

System Identification

- Development of a system is not just done by taking a snapshot of a scene (domain)
- How can we identify the system?
- Definition of the system boundary
  - **What is inside? (*Subsystems, objects)***
  - **What is outside? *(Actors)***
  - **What does it do*? (Scenarios, use cases)***
- Requirements Process:
  - **Requirements Elicitation: Definition of the system in terms understood by the customer**
  - **Requirements Analysis: Definition of the system in terms understood by the developer.**

Requirements Elicitation vs Requirements Analysis

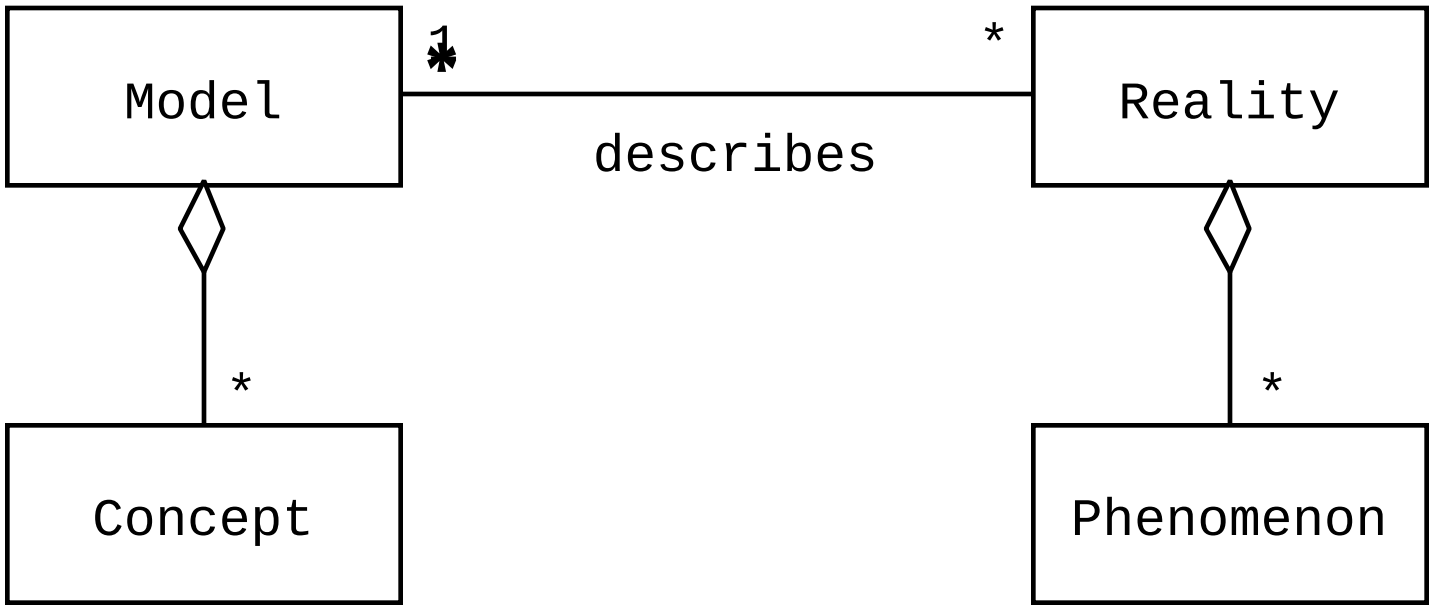
- Requirements Elicitation
  - **Focuses on describing what the system should be.**
  - **The client, developers and users identify a problem area and describe a system that would address the problem**
  - **The description is called the system specification**
    - **The customer understands this document**
    - **The problem statement is the first iteration of the system specification**
- Requirements Analysis
  - **Structures and formalizes a system specification and produces a requirements analysis model**
  - **The requirements analysis model is described in the requirements analysis document (RAD)**
    - **The developer understands this document**

Modeling

- Requirements elicitation and requirements analysis are modeling activities.
- The developer constructs a model describing the reality as seen from a user's point of view.
- Modeling consists of identifying and classifying real world phenomena (e.g., aspects of the system under construction) into concepts
- Model properties
  - **Completeness: All relevant phenomena are represented by at least one concept**
  - **Consistency: All concepts represent phenomena of the same reality**
    - **If the model is inconsistent it represents aspects of two different realities (one too many!)**

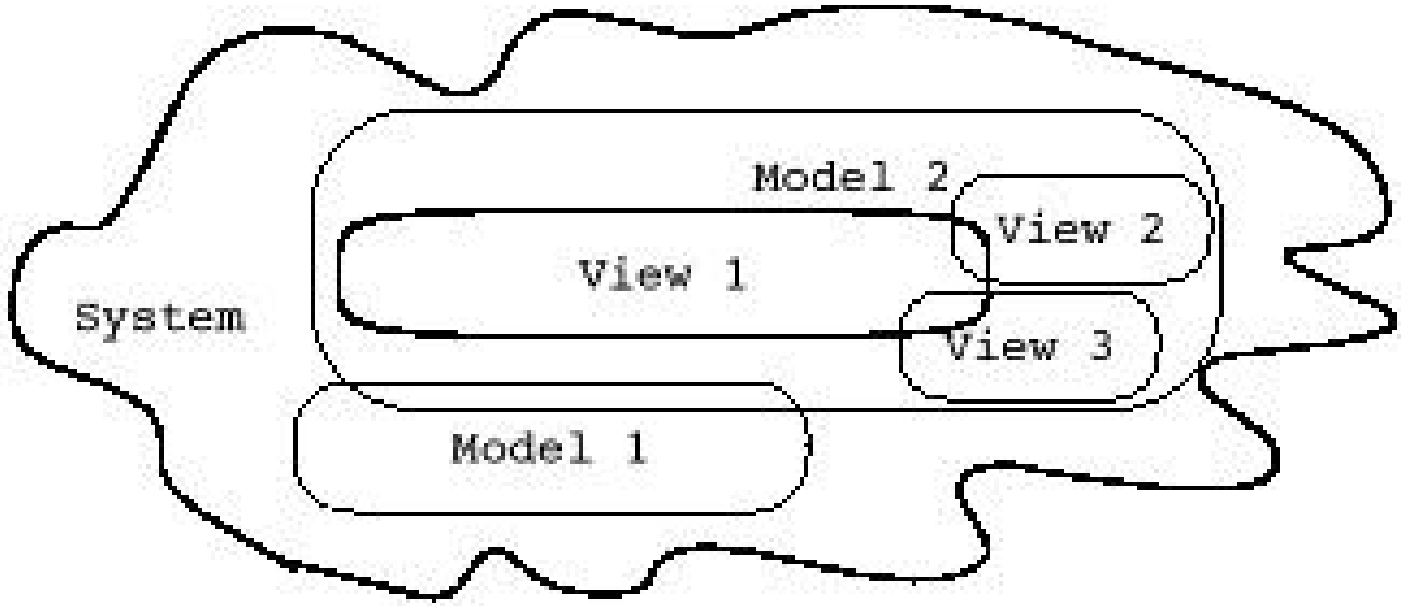
Phenomena and Concepts

- Reality is a collection of phenomena
- A model is a collection of concepts that represent the phenomena. Many models can represent different aspects (views) of the same Reality.



Modeling is an Abstration

- Model describe reality, which can be real systems (physical world, human being) or artificial systems (software system).



UML Notation

- A formal notation to describe models
- UML stands for Unified Modeling Language.
  - **It unifies notations from Rumbaugh (OMT),Jacobson (OOSE), Booch (Booch "clouds") and various other methodologists.**
- Based on type theory
  - **UML describes a system as a set of classes**
- A UML class is a type
  - **For now we don't make a distinction between class and type**

UML Diagrams are notations for models

- **A UML diagram consists of a set of nodes and arcs:**
  - **The nodes represent the model elements**
  - **The arcs represent relationships between the model elements**

UML Diagrams used in 15-413

- *1. Use Case Diagrams* (Functional Model)
- *2. Class and Object Diagrams* (Object Model)
- *3. Sequence Diagrams* (Functional Model)
- *4. Activity Diagrams* (Dynamic Model)
- *5. State Diagrams* (Dynamic Model)
- *6. Collaboration Diagrams* (Dynamic Model)
- *7. Implementation Diagrams* (Object Model)
  - **Component Diagrams, Deployment Diagrams**

We introduce the diagrams as we go along

Online Documentation for UML

- UML 1.1, September 1997
  - **http://www.rational.com/uml/index.html**
  - **Http://www.rational.com/uml/documentation.html**
- Series of documents:
  - **UML Summary**
  - **UML Notation Guide**
  - **UML Semantics**
  - **UML Extensions (Lifecycle Process, Business Modeling**
- I. Jacobson, J. Rumbaugh and G. Booch
  - **Unified Modeling Language User Guide, Addison-Wesley, to be published in Oct 1998**

Definition: Requirements

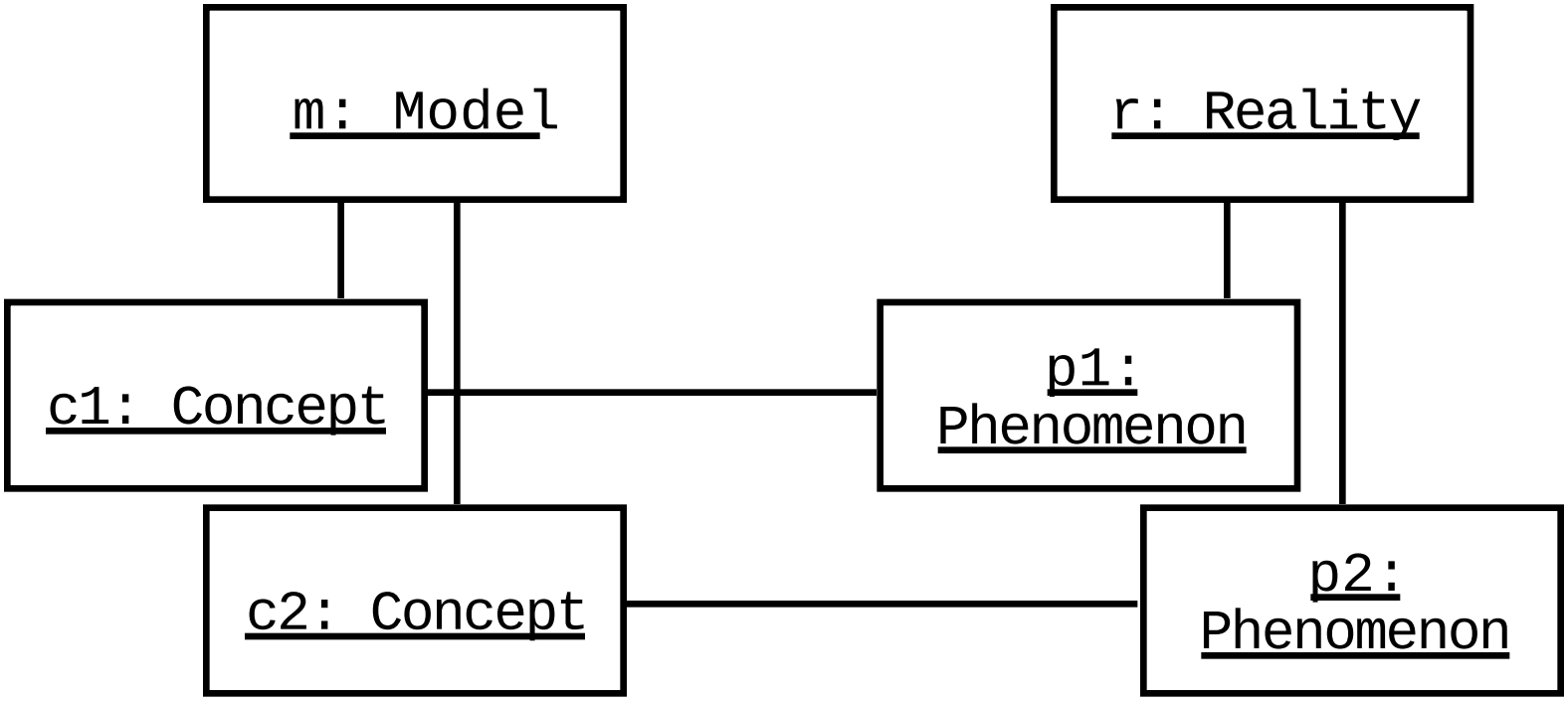
- Functional Requirements
  - **Describe the interactions between the system and its environment independent of its implementation**
  - **The environment includes the user and any other external system which with the system interacts**
  - **Example: Depositing money in bank account**
- Nonfunctional Requirements
  - **Describe user visible aspects of the system that are not directly related to the functional behavior of the system.**
  - **Example: Response time (The system reacts in 2 seconds or less)**
- Pseudo Requirements
  - **Constraints imposed by the client that restricts the implementation of the system**
  - **Example: Programming language must be COBOL**

Requirements Validation

- Validation checks if the requirement specification is correct, complete, consistent, unambiguous and realistic
- *Completeness:* All possible behavior through the system are described, including exceptional behavior by the user or the system
- *Correctness:* The requirements represent the client's view
- *Consistency:* There are functional or nonfunctional requirements that contradict each other
- *Unambiguity:* Exactly one system is specified, it is not possible to interpret the requirements in two or more ways.
- *Realism:* Requirements can be implemented and delivered

Completeness

- Every phenomenon of interest in the reality has a corresponding concept in the model

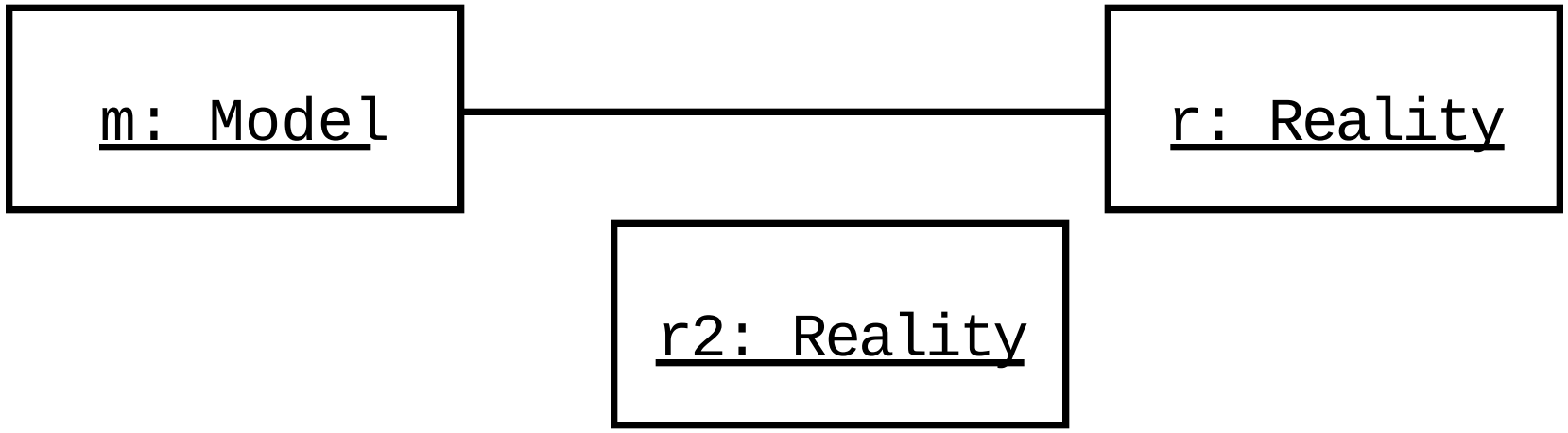


UML Object Diagram

- A collection of elements such as classes and their relationships to each other.
  - **Objects are shown as solid rectangles with 3 compartments for name, attributes and operations**
    - **Object names are of the form Name:Class and are always underlined (Class names are not)**
  - **Associations are shown as solid arrows.**
- Other components:
  - **Multiplicity, Roles, Qualifier, Aggregation, Inheritance**
- More details:
  - **Bruegge & Dutoit, Chapter 2**
  - **Lecture on Object Modeling (Sep 8)**
  - **http://www.rational.com/uml/html/notation/notation5a.html#5.2**

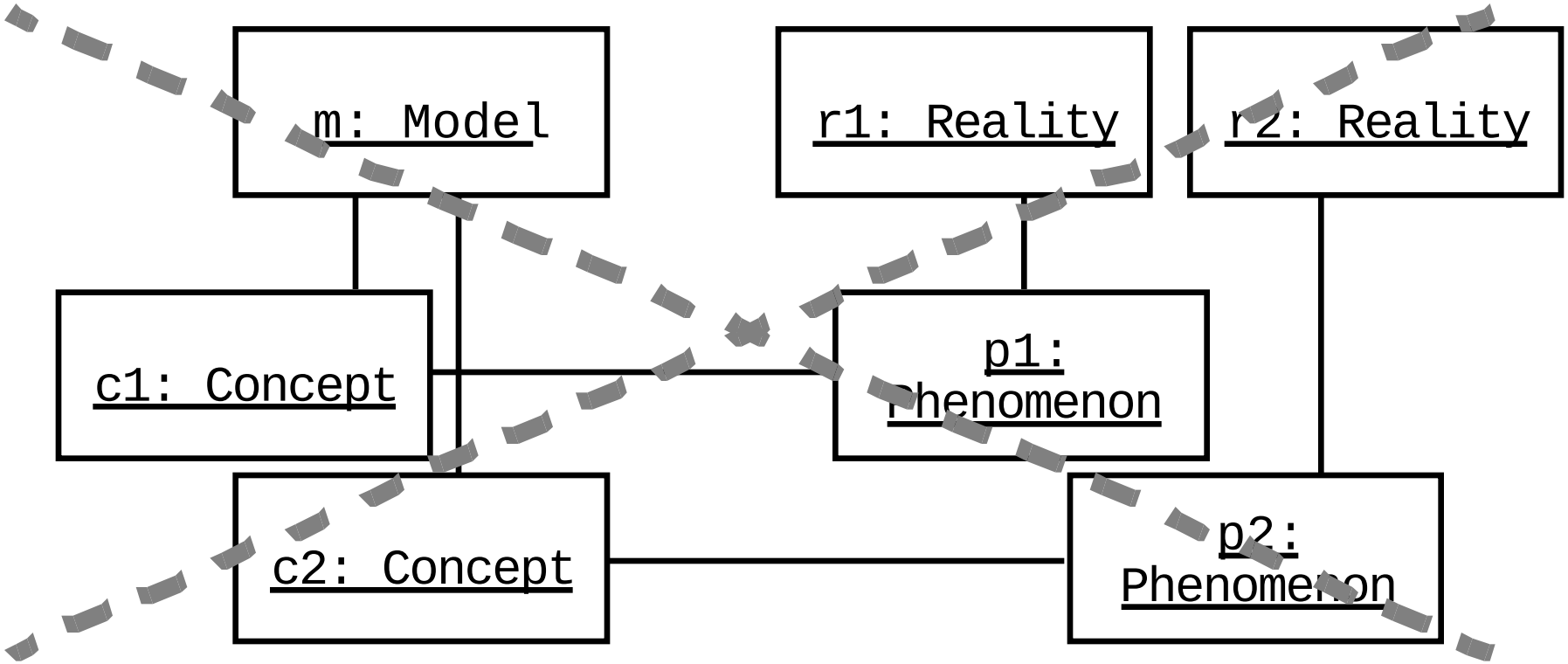
Correctness

- The model (requirements specification) describes the reality of interest to the client, not another reality



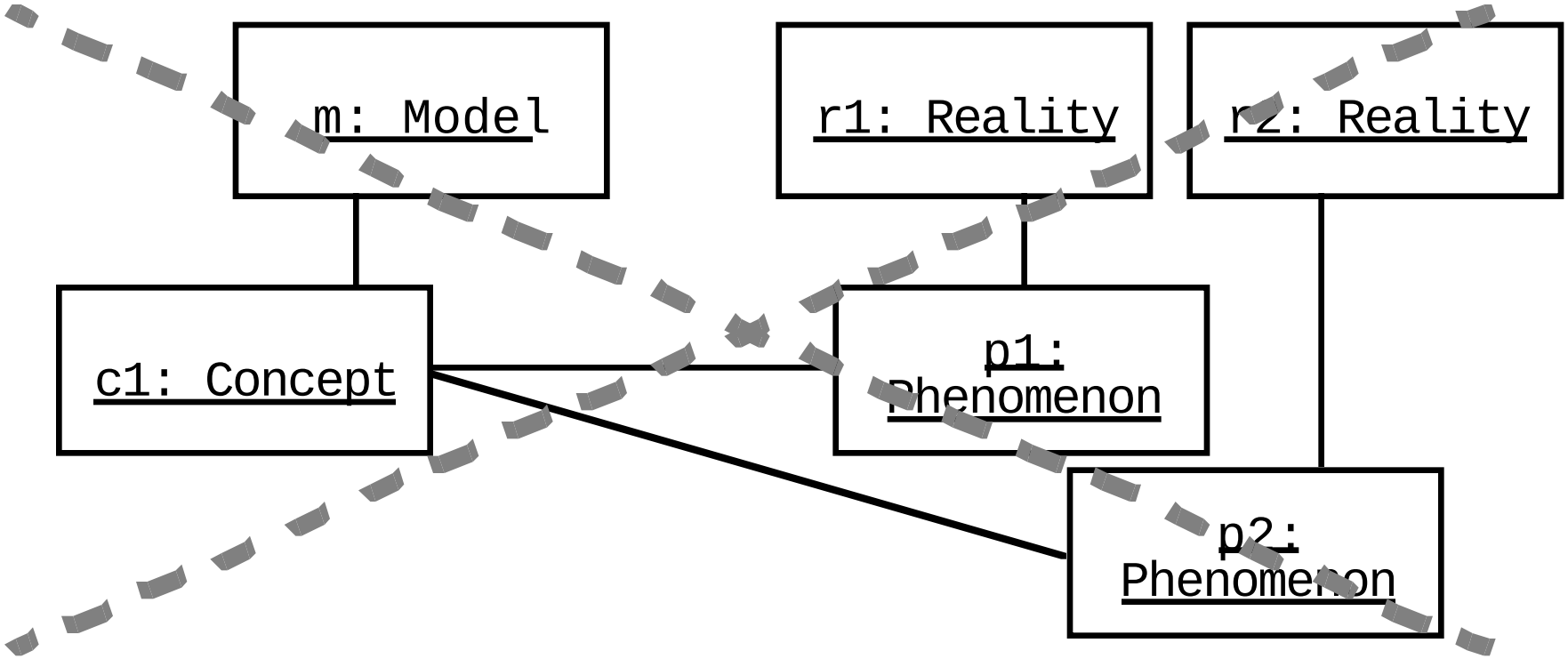
Consistency

- All concepts in the model correspond to phenomena of the same reality

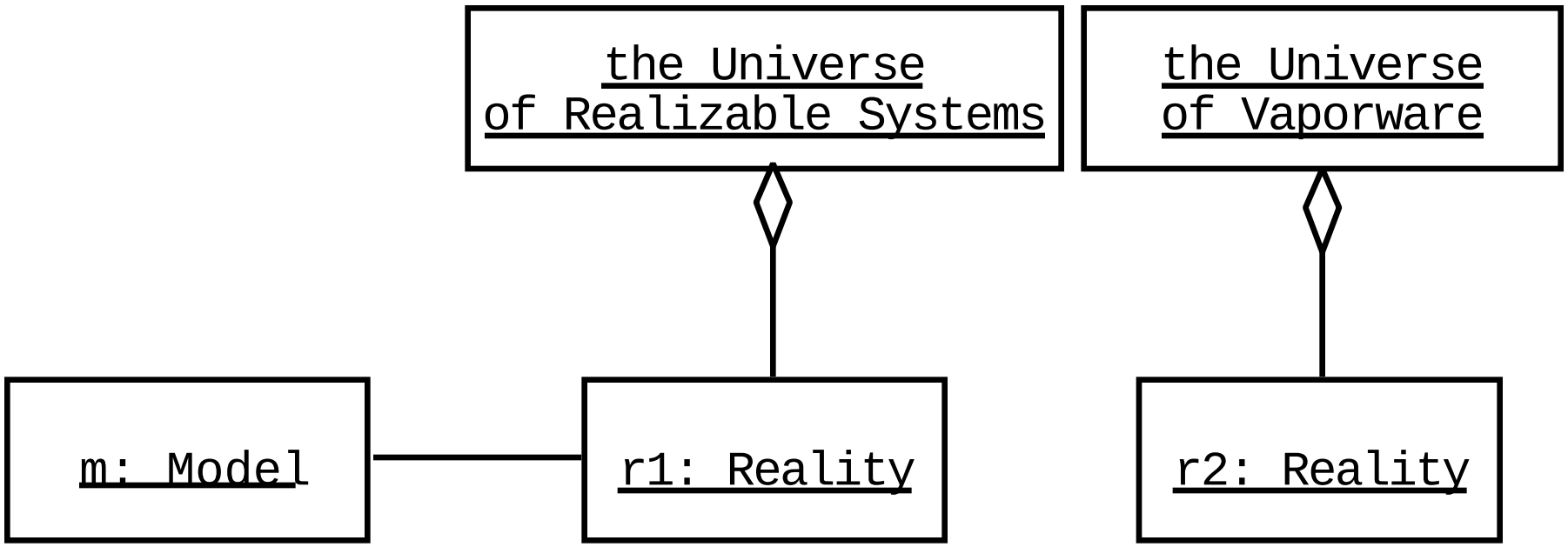


Unambiguity

- All concepts in the model correspond to exactly one phenomenon



Realism


the Universe
of Realizable Systems
the Universe
of Vaporware
m: Model
r1: Reality
r2: Reality

Requirements Validation

- *Traceability:*
  - **Each system function can be traced to a corresponding set of functional requirements**
- Tool:
  - **RequisitPro from Rational**
    - **Stores requirements in a repository**
    - **Multi-user access**
    - **Automatically creates a requirements document from the repository**
    - **Provides traceability and change management throughout the project lifecycle**
    - **http://www.rational.com/products/reqpro/docs/datasheet.html**

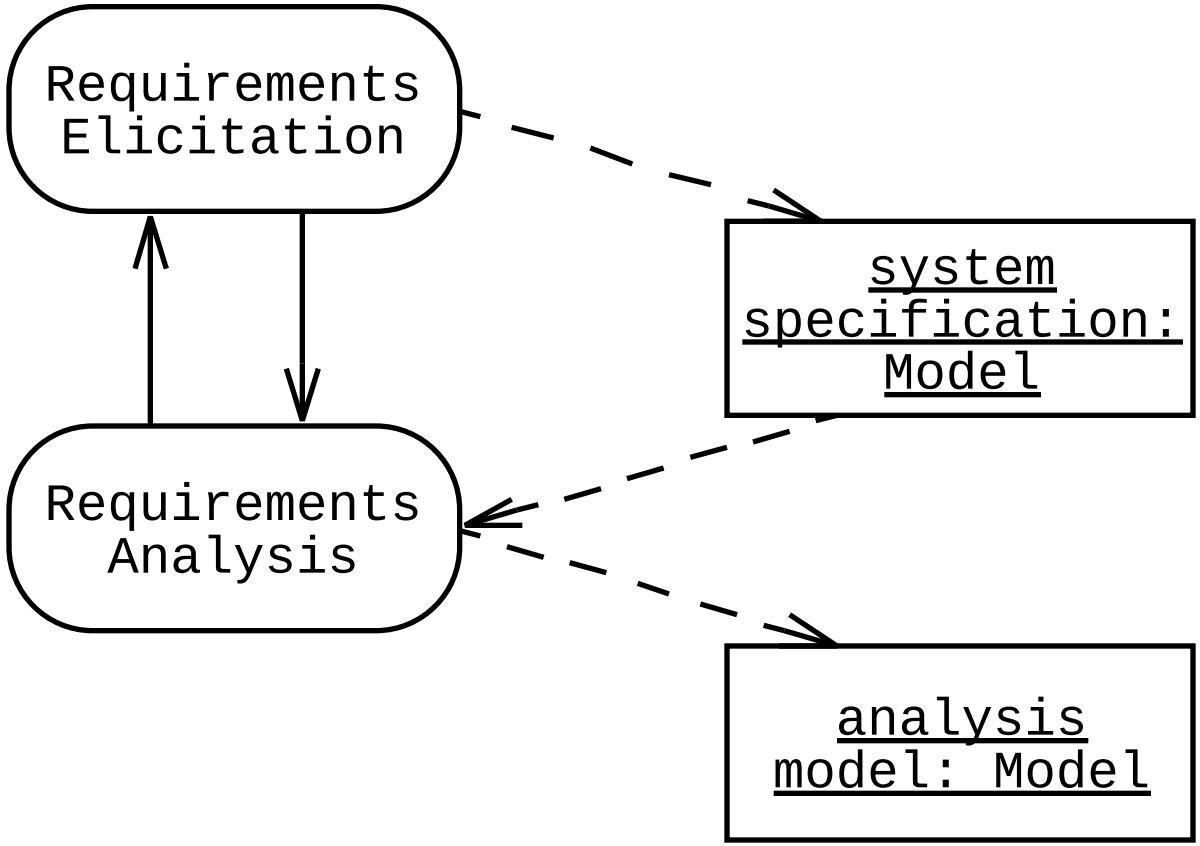
Requirements Elicitation in Different Projects

- Greenfield Engineering
  - **Development starts from scratch, no prior system exists, the requirements are extracted from the end users and the client**
  - **Triggered by user needs**
  - ***Requirements constantly changing***
- Re-engineering
  - **Re-design and/or re-implementation of an existing system using newer technology**
  - **Triggered by technology enabler**
  - ***Functional requirements are constant but changes are tempting, nonfunctional requirements are changing***
- Interface Engineering
  - **Provide the services of an existing system in a new environment**
  - **Triggered by technology enabler or new market needs**
  - ***Requirements are constant, nonfunctional requirements changing***

Requirements Elicitation

- Very challenging activity
- Requires collaboration of people with different backgrounds
  - **User with application domain knowledge**
  - **Developer with implementation domain knowledge**
- Bridges the gap between user and developer
- Scenario-based Requirements Elicitation
  - **Describes the use of the system in terms of a series of interactions with between the user and the system**

Interaction between Requirements Elicitation and Requirements Analysis



UML Activity Diagram

- A variation of a finite state machine
  - **Action State: Activity (shown as oval) representing the performance of operations**
  - **Transitions (shown as solid arrows) are triggered by the completion of the operations**
- Other components of an Activity Diagram:
  - **Objects, Object Flow Relationships, Decisions, Swimlanes**
- Object Flow Relationship: States operate on objects (shown as solid squares) who are responsible for performing the action. Object flow from input object or to output object (shown as dashed arrows) .
- More details:
  - **Bruegge & Dutoit, Chapter 2**
  - **Lecture on Dynamic Modeling (Oct 1)**
  - **http://www.rational.com/uml/html/notation/notation10.html#10.1**

Types of Scenarios

- As-is scenario:
  - **Used in describing a current situation. Usually used during re-engineering. The user describes the system.**
- Visionary scenario:
  - **Used to describe a future system. Usually described in greenfield engineering or reengineering.**
  - **Can often not be done by the user or developer alone**
- Evaluation scenario:
  - **User tasks against which the system is to be evaluated**
- Training scenario:
  - **Step by step instructions designed to guide a novice user through a system**

Example Scenario: Warehouse on Fire

- Bob, driving down main street in his patrol car notices smoke coming out of a warehouse. His partner, Alice, activates the “Report Emergency” function from her laptop.
- Alice enters the address of the building, a brief description of its location (i.e., north west corner), and an emergency level. In addition to a fire unit, she requests paramedic units on the scene. She confirms her input and waits for an acknowledgment.
- John, the Dispatcher, is alerted to the emergency by a beep of his workstation. He reviews the information submitted by Alice and acknowledges the report. He allocates a fire unit and two paramedic units to the Incident site and sends their estimated arrival time (ETA) to Alice.
- Alice receives the acknowledgment and the ETA.

Observations about Warehouse on Fire Scenario

- Concrete scenario
  - **Describes a single instance of reporting a fire incident.**
  - **Does not describe all possible situations in which a fire can be reported.**
- Participating actors
  - **Bob, Alice and John**

Heuristics for Finding Scenarios

- Don’t expect the client to be verbal if the system does not exist (greenfield engineering)
- Don’t wait for information even if the system exists
- Engage in a dialectic approach (evolutionary, incremental)
  - **You help the client to formulate the requirements**
  - **The client helps you to understand the requirements**
  - **The requirements evolve while the scenarios are being developed**

Finding Scenarios

- Insist on *task analysis* if the system already exists (interface engineering or reengineering)

- Ask to observe and speak to the end user, not just to the software contractor
  - **Expect resistance and try to overcome it**

- Do not rely only on questionaires, it is important to observe the behavior of the end user

Task Analysis

- Identify and describe the user tasks that a system has to support
  - **Important field in Human Computer Interaction (HCI)**
- KAT (Knowledge Analysis of Tasks by Johnson):
  1. **Identify objects and actions**
  2. **Identify procedures (pre-condition, set of actions, post-condition)**

     Procedures are identified by task observation, then asking the task performer to select&order cards containing action descriptions)
  3. **Identify goals and subgoals (State to be achieved for the task to be successful)**
  4. **Identify typicality and importance (Each element is rated according to frequency and relevance to accomplishing a goal)**
  5. **Construct model of task and validate it with performer**

Finding Scenarios ctd

- After task observation, ask the following questions:
  - **What are the main actors and primary tasks that the system needs to perform?**
  - **What data will the actors create, store, change, remove or add in the system?**
  - **What external changes does the system need to know about?**
  - **What changes or events will the actor of the system need to be informed about?**
  - **Are there any typical scenarios?**
  - **Are there any problems related to load**
  - **What are typical response times?**

Example: Eliciting the Requirements for an Accident Management System

- What needs to be done to report a “Cat in a Tree” incident?
- What do you need to do if a person reports “Warehouse on Fire?”
- Who is involved in reporting an incident?
- What does the system do if no police cars are available? If the police car has an accident on the way to the “cat in a tree” incident?
- What do you need to do if the “Cat in the Tree” turns into a “Grandma has fallen from the Ladder”?
- Can the system cope with a simultaneous incident report “Warehouse on Fire?”

Specifying a Scenario

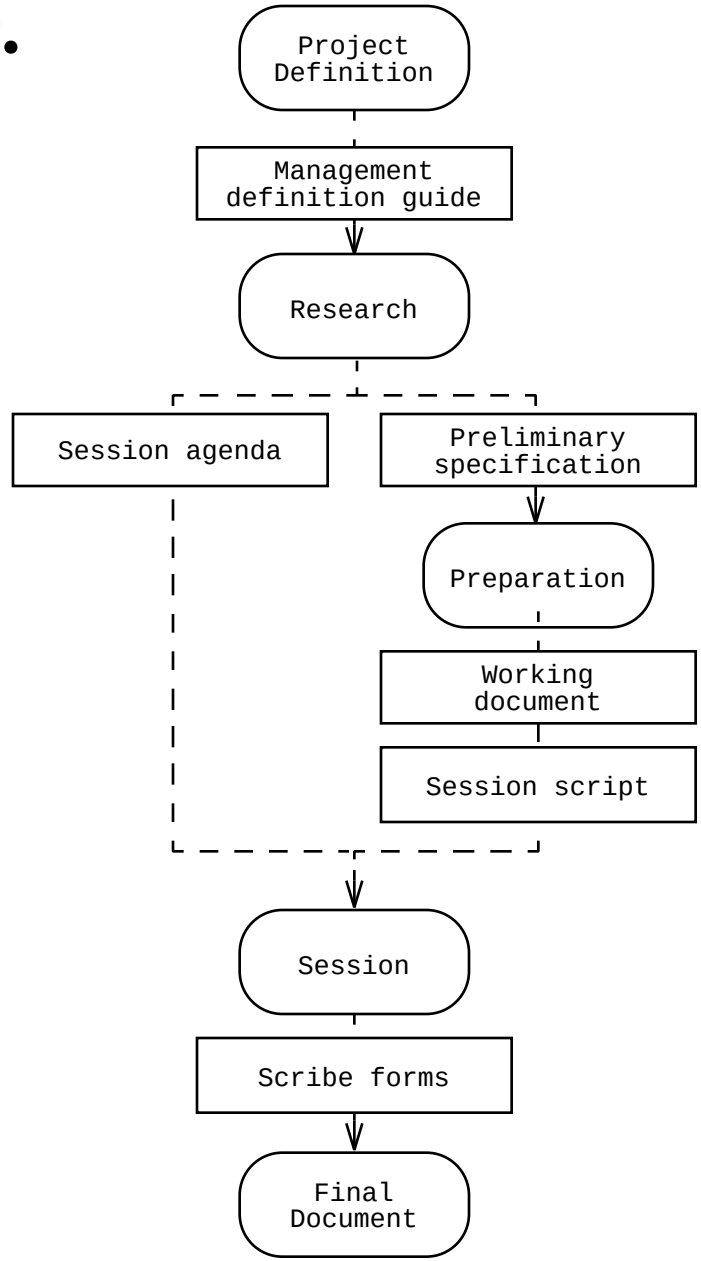
- To communicate with the client use cenarios in problem statement and in requirements specification
- Components of a scenario:
  - **Scenario Name**
  - **Participating actor instances**
  - **Description (Flow of Events)**
  - **Nonfunctional Requirements**

Scenario: Allocate a Resource

- Scenario Name: Send a police car to an incident
- Participating actors:
  - **Carl: Field Supervisor**
  - **Carol: A neighbor**
  - **Mary: Dispatcher**
  - **Bob: Resource Allocator**
  - **Alice: Field officer**
- Description:
  - **After the the patrol car P11 comes back from the repair shop, Carl adds it to the fleet of available patrol cars at the Bellevue Police Department**
  - **Carol calls the police and reports that the warehouse at Morrison Rd 55 is on fire.**
  - **Mary takes the call and creates an incident #54 describing Carol's call**
  - **Bob looks at the list of available police cars and selects patrol car P11. He then commits the car to the Ware house on fire incident that was reported 30 minutes ago.**
  - **Alice drives P11 to Morrison Rd. It turns out that there is no fire at the warehouse. Alice drives the car back to police headquarters**
  - **Bob to put the car back on the list of available police cars.**

Joint Application Design (JAD).

- Developed by IBM in the 70s using flip charts in a single room involving all stakeholders (users, clients, developers and trained session leader)
- A workshop with 5 activities
  - **Project Definition**
  - **Research**
  - **Preparation**
  - **Session**
  - **Final Document**
- At the end of the workshop the requirements specification is produced



Scenarios lead us to use cases…

- Lecture on Use Case Modeling (Sep 8)

Summary

- Types of Projects
  - **Greenfield Engineering, Reengineering, Interface Engineering**
- Requirements
  - **Functional and Nonfunctional Requirements, Constraints**
- Requirements Validation
  - **Correctness, Completeness, Consistency, Unambiguity, Realism**
- Scenarios:
  - **Great way to establish communication with client**
  - **As-Is Scenarios, Visionary scenarios, Evaluation scenarios Training scenarios**
- Scenarios can be used all the way through a project, from requirements elicitation to system testing
- Methodologies for Requirements Elicitation (JAD, KAT)