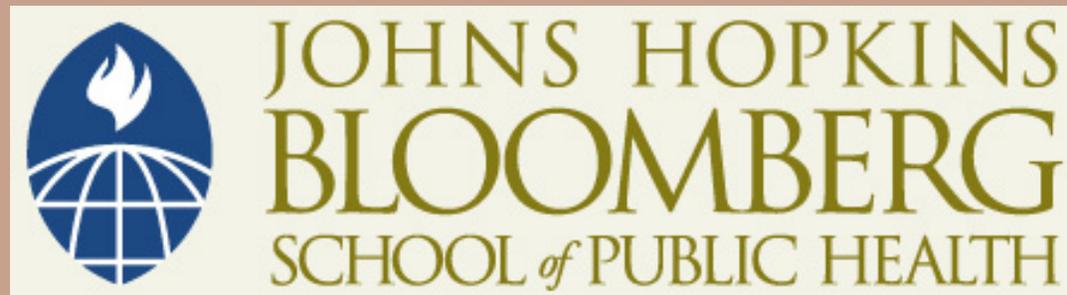


This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike License](https://creativecommons.org/licenses/by-nc-sa/4.0/). Your use of this material constitutes acceptance of that license and the conditions of use of materials on this site.



Copyright 2011, The Johns Hopkins University and Anna Orlova. All rights reserved. Use of these materials permitted only in accordance with license rights granted. Materials provided "AS IS"; no representations or warranties provided. User assumes all responsibility for use, and all liability related thereto, and must independently review all materials for accuracy and efficacy. May contain materials owned by others. User is responsible for obtaining permissions for use from third parties as needed.

JOHNS HOPKINS  
UNIVERSITY

## Section C

---

Requirements Elicitation

# Requirements Elicitation Includes:

- Specifying goals
- *Specifying actors (business and technical)*
- Specifying functional and non-functional requirements
- Specifying use cases
- Developing models/diagrams
  - Use case, workflow, and dataflow
- Specifying high-level system architecture
- Specifying hardware and software requirements
- Specifying system evaluation plan
- Specifying project timeline and documentation

# Specifying Actors

- Actors represent the external entities that interact with the system
- Business actors: stakeholders and entities
  - For example, providers, patients, public health practitioners, public health agencies, hospitals, etc.
- Technical actors: information systems
  - For example, EHR-S, PHR, LIMS, etc.

## Example of Immunization Registries: Goals and Actors

- The goal of the immunization registry is to help prevent spread of infectious diseases by timely administering vaccination to children within a geographic area
- Business actors
  - Patient
  - Physician
  - Nurse
  - Immunization registry staff
- Technical actors
  - Provider EHR system
  - Immunization registry

# Requirements Elicitation Includes:

- Specifying goals
- Specifying actors (business and technical)
- *Specifying functional* and non-functional *requirements*
- Specifying use cases
- Developing models/diagrams
  - Use case, workflow, and dataflow
- Specifying high-level system architecture
- Specifying hardware and software requirements
- Specifying system evaluation plan
- Specifying project timeline and documentation

# Specifying Functions

- Function is the description of the interaction between the information system and its environment
- The environment includes the users (business actor) and external systems (technical actor) with which the system interact

# Specifying Functions

- Generic functions of an information system include:
  - Collect data (input)
  - Manage data (verify, store, upload, etc.)
  - Integrate data
  - Analyze data (SAS, SPSS, STRATA, GIS)
  - Generate reports (output, e.g., summary, reminders, notification, alert, update, etc.)

## Example: Immunization Registries

- The goal of the immunization registry is to help prevent spread of infectious diseases by timely administering vaccination to children within a geographic area
- To achieve this goal, the immunization registry system will support the following *functions*:
  - *Consolidate (collect, manage, integrate)* vaccination records from multiple health care providers within jurisdiction,
  - *Generate reminder and recall notifications (analyze data and generate reports)* to providers and patients,
  - *Assess clinic performance (analyze data and generate reports)* in vaccination coverage

# Specifying Functions

- A **nonfunctional requirement** is a constraint on the operation of the system that is not related directly to a function of the system
- Non-functional requirements have as much impact on the system as functional requirements

# Requirements Elicitation Includes:

- Specifying goals
- Specifying actors (business and technical)
- *Specifying* functional and *non-functional requirements*
- Specifying use cases
- Developing models/diagrams
  - Use case, workflow, and dataflow
- Specifying high-level system architecture
- Specifying hardware and software requirements
- Specifying system evaluation plan
- Specifying project timeline and documentation

# Specifying Functions

- Nonfunctional requirements falls into two categories:
  1. Quality requirements
  2. Constraints or pseudo requirements

# Quality Requirements

- Usability
- Reliability, dependability, robustness, safety
- Performance (response time, throughput, availability, accuracy)
- Supportability, adaptability, maintainability, portability
- Implementation

# Constraints or Pseudo Requirements

- Implementation requirements
- Interface requirements
- Operation requirements
- System security requirements
- Packaging requirements
- Legal requirements

# Requirements Elicitation Includes:

- Specifying goals
- Specifying actors (business and technical)
- Specifying functional and non-functional requirements
- *Specifying use cases*
- Developing models/diagrams
  - Use case, workflow, and dataflow
- Specifying high-level system architecture
- Specifying hardware and software requirements
- Specifying system evaluation plan
- Specifying project timeline and documentation

# Specifying Use Cases

- **Use cases** are general sequences of events that describe all the possible actions between an actor and the system for a given piece of functionality

# Example: Immunization Registry

Use case name	Immunization
Actors	Patient, physician, nurse, immunization registry (IR) staff, EHR system, immunization registry
Flow of events	<ol style="list-style-type: none"><li>1. Patient comes to physician for a general check-up, and he/she is due for immunization</li><li>2. Physician orders an immunization</li><li>3. Nurse administers an immunization</li><li>4. Nurse enters data on the immunization in the Electronic Health Record (EHR) system</li><li>5. Nurse electronically sends immunization data to the local immunization registry</li><li>6. Immunization registry staff receives data</li></ol>
Entry conditions	EHR system
Exit conditions	Immunization registry
Quality	Daily updates

# Use Case and Actors

- Identification of actors and use cases within an application domain results in the definition of the boundary of the system
  - That is, in differentiating the tasks accomplished by the system
- The *actors* are outside the boundary of the system, whereas the *use cases* are inside the boundary of the system

# Requirements Elicitation Includes:

- Specifying goals
- Specifying actors (business and technical)
- Specifying functional and non-functional requirements
- Specifying use cases
- *Developing models/diagrams*
  - *Use case, workflow, and dataflow*
- Specifying high-level system architecture
- Specifying hardware and software requirements
- Specifying system evaluation plan
- Specifying project timeline and documentation

# Modeling

- **Modeling** is one of the basic methods of science
- A model is an abstract representation of a domain (a field, a problem) that enables us to answer questions about the domain
- Models allow to visualize the domain

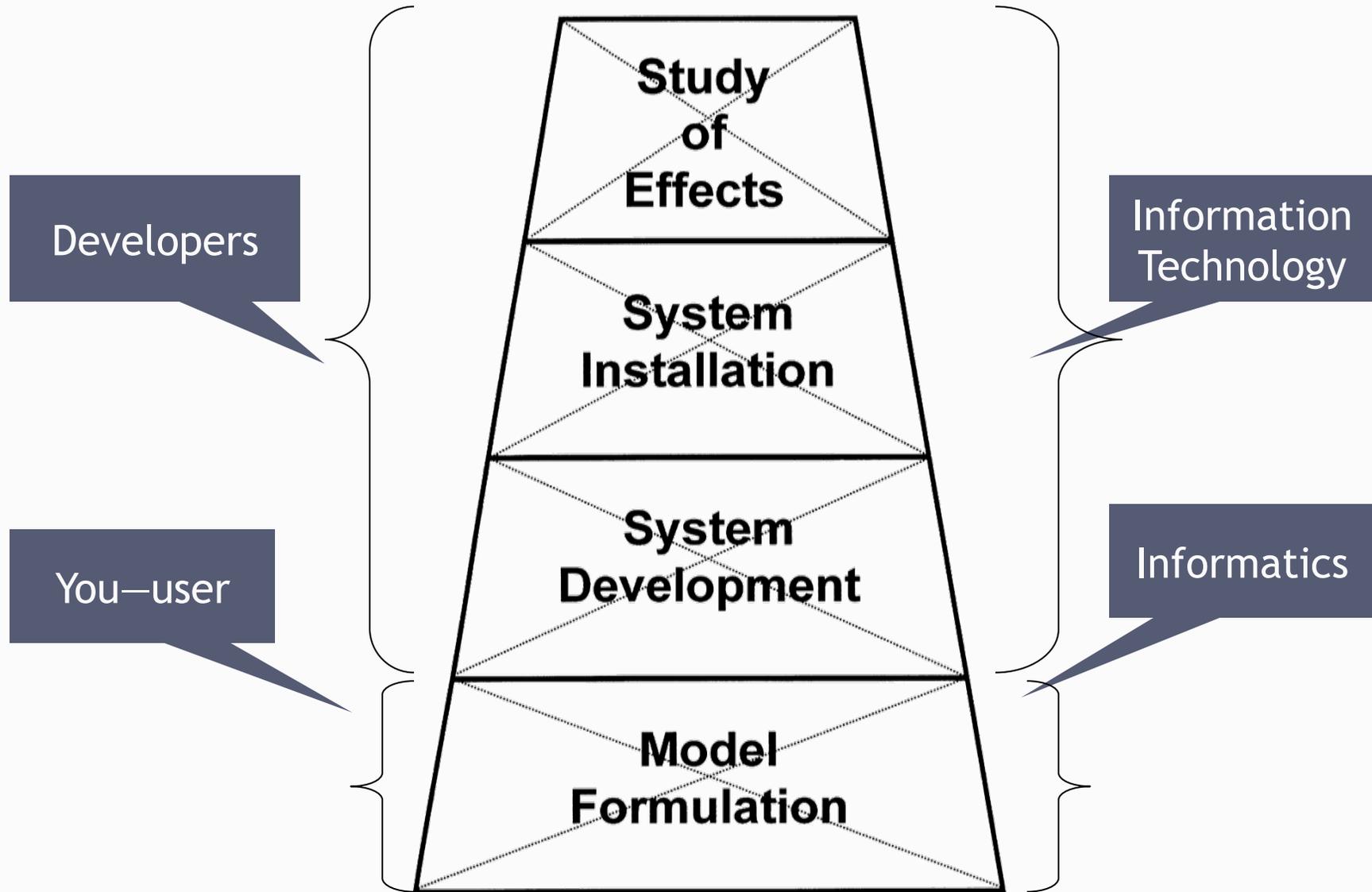
# Modeling

- Software engineering is a field that deals with building software products; it includes:
  - A *modeling activity* to understand an organization of the application domain
  - A *problem-solving activity* to search for acceptable solution.
  - A *knowledge-acquisition activity* that collects data, organize it into information and formalize it into knowledge.
  - A *rationale-driven activity* that puts the found solution in the context in which it will be used in a decision-making process and rationale behind these decisions

# Modeling

- To communicate your needs to developers, you need to learn developer's language
- This language is **modeling**

# Foundation of Successful IT Application: Modeling



# Modeling in Software Engineering

- Software engineering uses modeling of the domain to:
  - Capture domain knowledge
  - Precisely specify requirements
    - ▶ That is, describe a solution in a format of a structured document (specification) so that all stakeholders (actors-users) may understand and agree on them
  - Guide the thought process
  - Generate potential configuration of the system describing its generic structure and meaning
  - Abstract specifications of the essential structure of the system
  - Tell what something does (functional specification) as well as how the function is accomplished (implementation)

# Developer's Language: Graphical Symbols

- Unified modeling language (UML) is a standard used by developers to model the application domain
- UML is a language of graphical symbols that are used in depicting and describing:
  - a. A system in general system architecture
  - b. System components
  - c. The interaction of the system components
- Developer's language is the notation for representing object-oriented models

# Developer's Language: Notation

- A **notation** is a graphical or textual set of rules for representing a model
  - E.g., the Roman alphabet is a notation for representing words
- To enable accurate communication a notation must:
  - Come with a well-defined semantics
  - Be well suited for representing a given aspect of a system
  - Be well understood among project participants
- In the latter lies the strength of standards and conventions: when a notation used by a large number of participants, there is little room for misinterpretation and ambiguity

# The Goal of UML

- The goal of UML is to:
  - Provide a *standard notation* that can be used by all object-oriented methods
  - Select and integrate the best elements of precursor notations

# UML Models

- The **functional model**, represented in UML with use case diagrams, describes the functionality of the system from the user's point of view
- The **object model**, represented in UML with class diagram, describes the structure of the system in terms of objects, attributes, associations, and operations
  - E.g., HL7 RIM
- The **dynamic model**, represented in UML with interaction diagrams, statecharts diagram, and activity diagrams, describes the internal behavior of the system
  - Interaction diagrams describe behavior as a sequence of messages exchanged among a set of objects
  - Statechart diagrams describe behavior in terms of states of an individual object

# Five UML Notations

1. Use case diagrams
2. Class diagrams
3. Interaction diagrams
4. Statechart diagrams
5. Activity diagrams

# Five UML Notations

1. **Use case diagrams**

2. Class diagrams

3. Interaction diagrams

4. Statechart diagrams

5. **Activity diagrams**



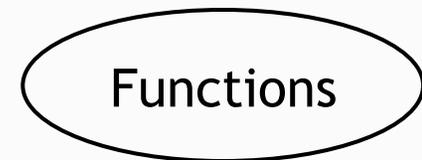
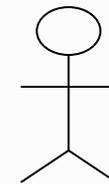
*We will focus only on  
use case diagrams and  
activity diagrams*

# Use Case Diagrams

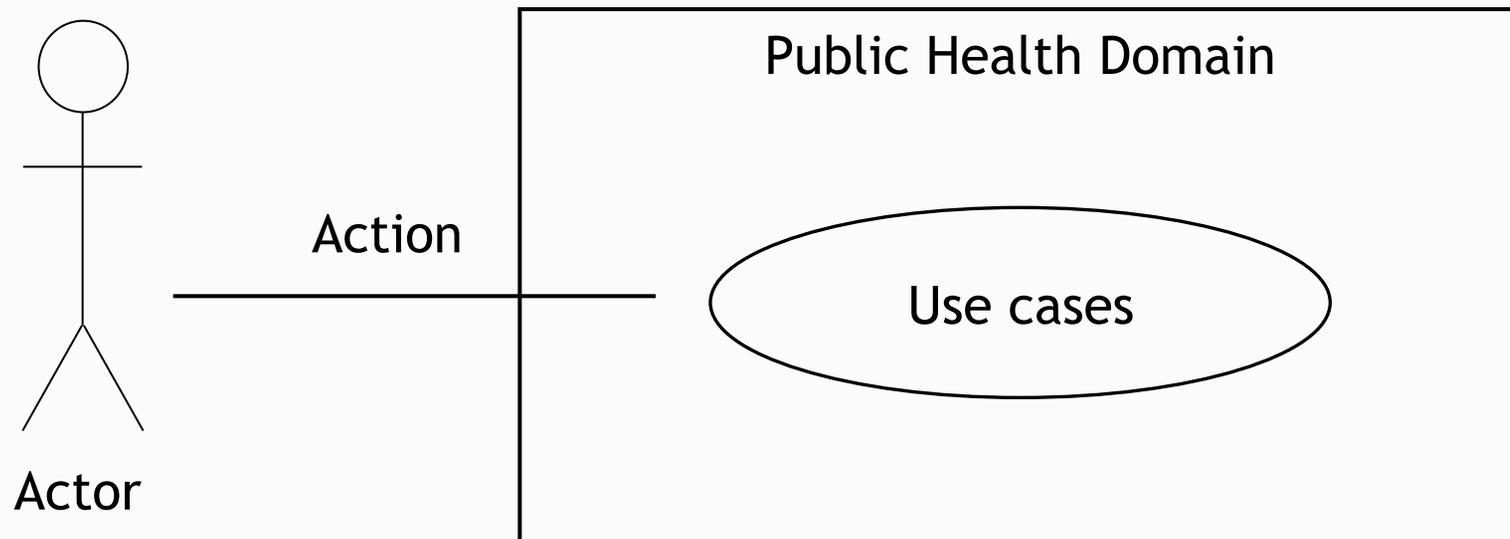
- **Use case diagram** describes the use case in a graphical way
  - That is, what a system does from the standpoint of an external observer (actor)
  - The emphasis is on *what* a system (functions) does rather than *how*

# Use Case Diagram Symbols

- *Rectangular box*: represents the Application Domain described in the Use Case
  - For example, immunization
- *Actor*: any entity (object) that interacts with the system (e.g., end user, another system, etc.)
- *Oval*: depicts the function to be provided by the system that yields a visible result for an actor
- *Line*: indicates action
  - Connects the actor to the function in which he participates in the system
  - Highlights the relationship between objects in the system



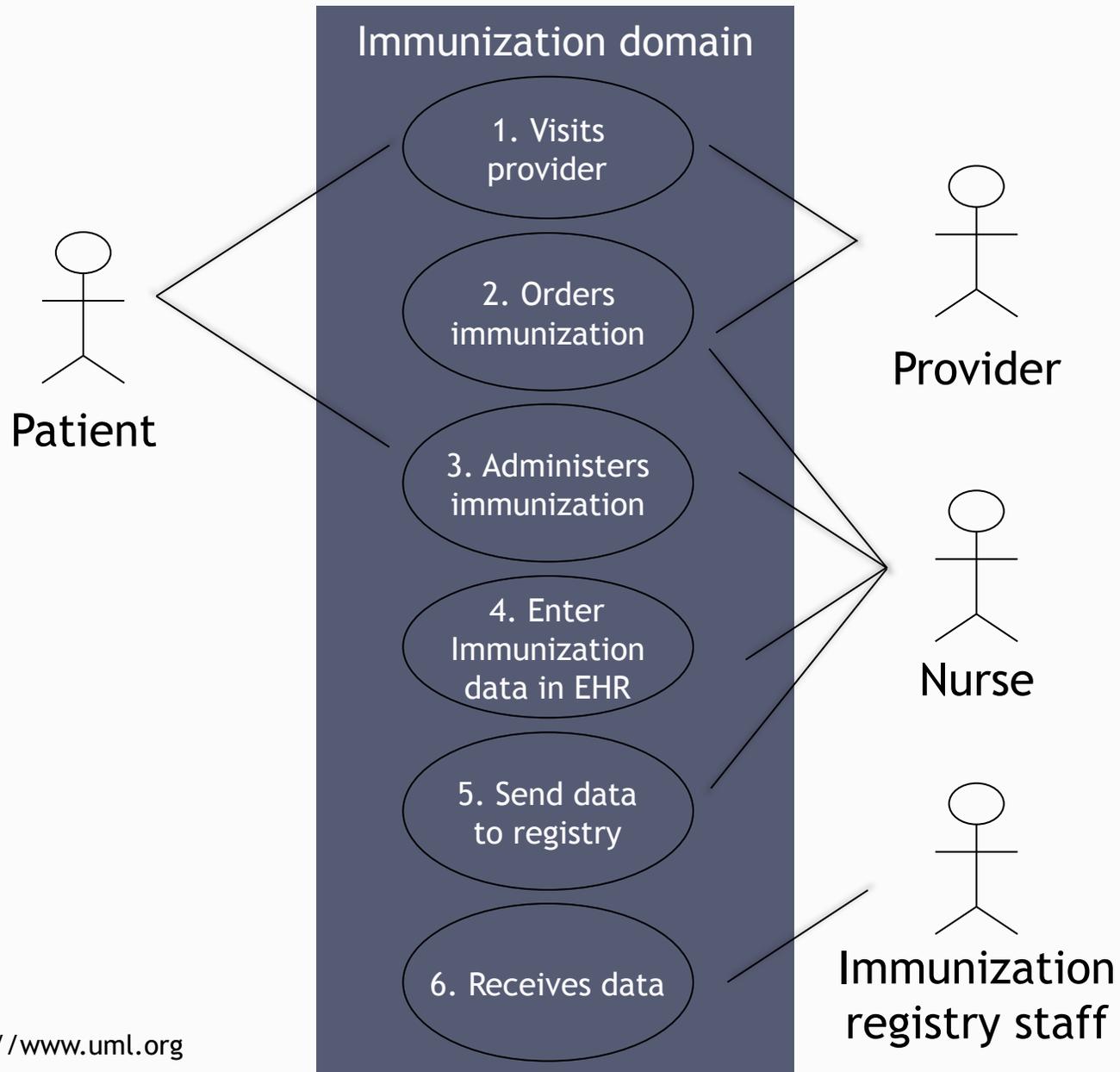
# Use Case Diagram Symbols



# Specifying Use Case—Use Case Format: Immunization

Use case name	Immunization
Actors	Patient, physician, nurse, immunization registry (IR) staff, EHR system, immunization registry
Flow of events	<ol style="list-style-type: none"><li>1. Patient comes to physician for a general check-up, and he/she is due for immunization</li><li>2. Physician orders an immunization</li><li>3. Nurse administers an immunization</li><li>4. Nurse enters data on the immunization in the Electronic Health Record (EHR) system</li><li>5. Nurse electronically sends immunization data to the local immunization registry</li><li>6. Immunization registry staff receives data</li></ol>
Entry conditions	EHR system
Exit conditions	Immunization registry
Quality	Daily updates

# UML Use Case Diagram: Immunization

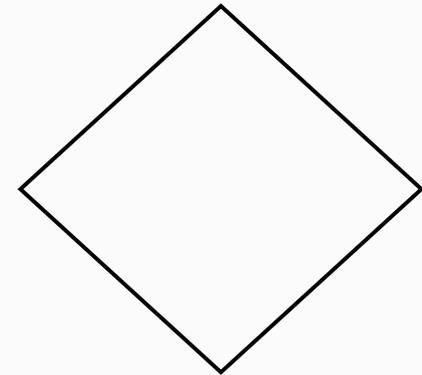


# Work Flow and Data Flow Diagrams

- **Work flow diagrams:** notation for representing user participation in the system
- **Data flow diagrams:** notation for representing systems in term of data sources and data transformation

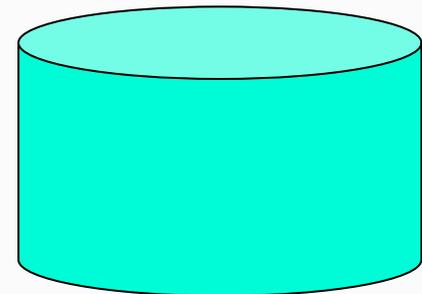
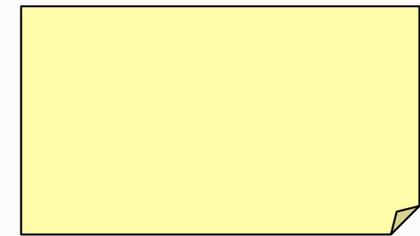
# Work Flow and Data Flow Diagram Symbols

- *Rectangular box*: represents Business Actor = entity (object) involved in the system
  - For example, patient, provider, public health agency, etc.
- *Diamond*: represents a decision point in the process
  - Typically, it requires a *Yes/No* response that triggers further activities within the system—for example, provider orders immunization
- *Rounded box*: represents the event that happens automatically
  - For example, nurse administers immunization as ordered by provider



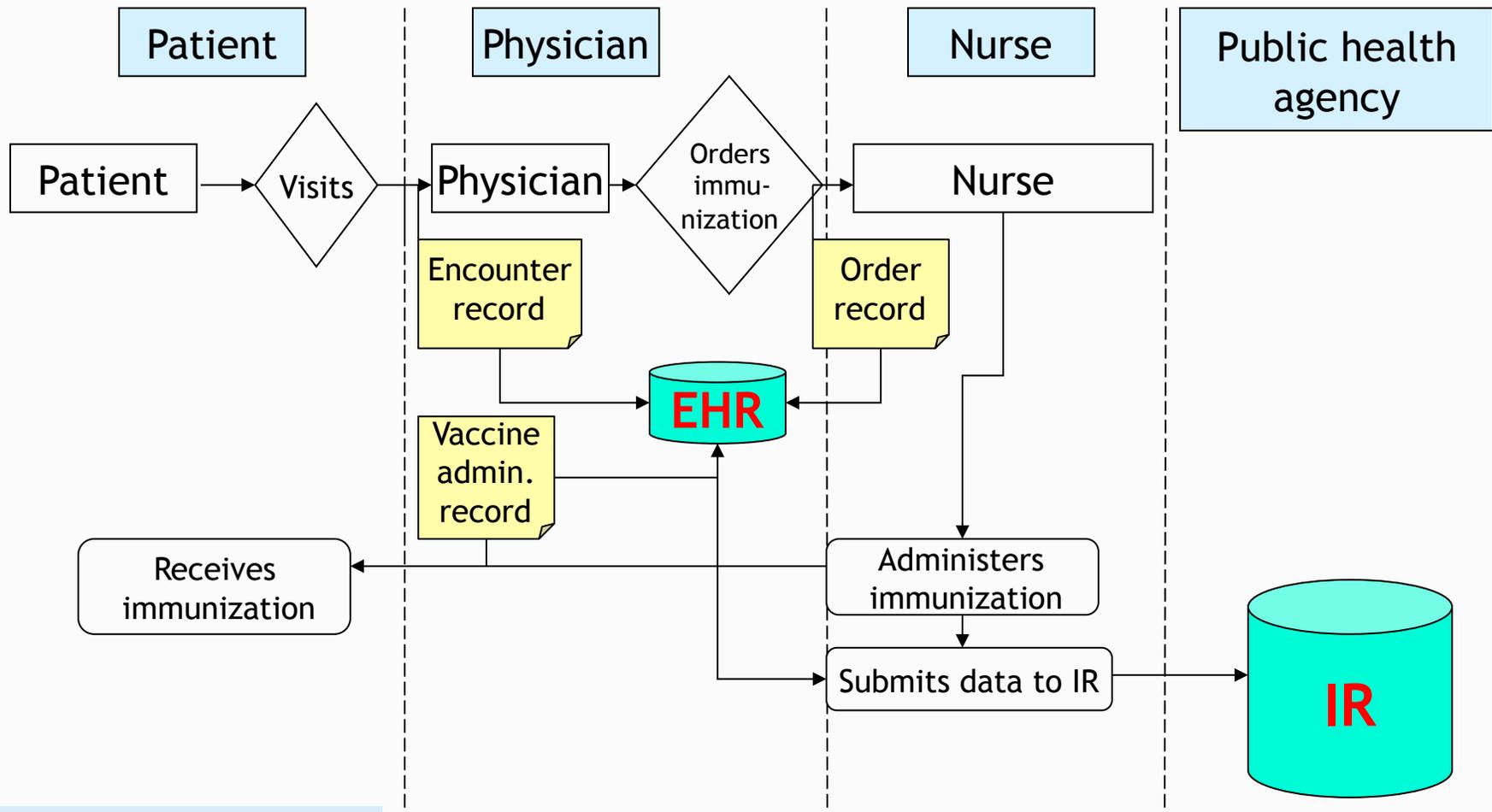
# Work Flow and Data Flow Diagram Symbols

- *Line with arrow*: shows the order/direction of steps/activities (flow of steps) within the system
- *Paper sheet or e-encounter record symbol*: represents data recording points within the system
- *Technical Actor* = information system
  - EHR, IR, etc.



# Work Flow and Data Flow

## Immunization Electronic Data Submission



EPR: electronic patient record  
IR: immunization registry

## Other Use Cases

- There may be situations when an immunization will *not* be administered at the time of the patient visit when immunization might otherwise be due—for example:
  - Provider does not have the immunization history of the patient to know if immunization is due
  - Patient is due for immunization but has contraindications—for example, a fever—so immunization has to be postponed
  - Patient is due for immunization but refuses to be immunized because of beliefs
  - Etc.
- These various scenarios described as Use Cases are the lower levels of knowledge representation of the immunization domain

# Requirements Elicitation Includes:

- Specifying goals
- Specifying actors (business and technical)
- Specifying functional and non-functional requirements
- Specifying use cases
- Developing models/diagrams
  - Use case, workflow, and dataflow
- *Specifying high-level system architecture*
- Specifying hardware and software requirements
- Specifying system evaluation plan
- Specifying project timeline and documentation