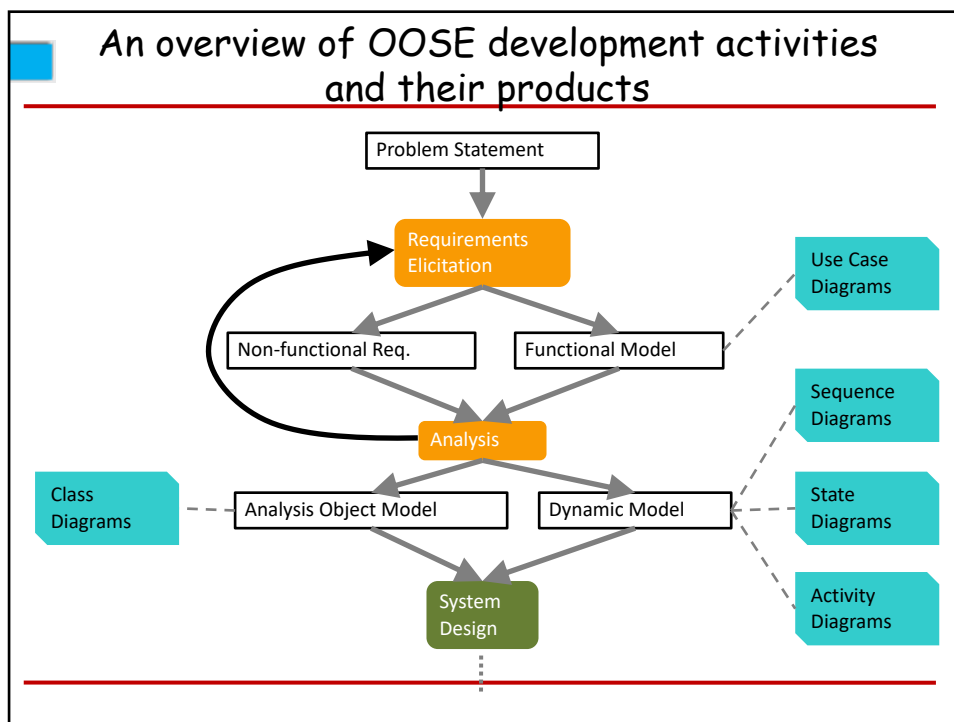DA-IICT

IT 314: Software Engineering
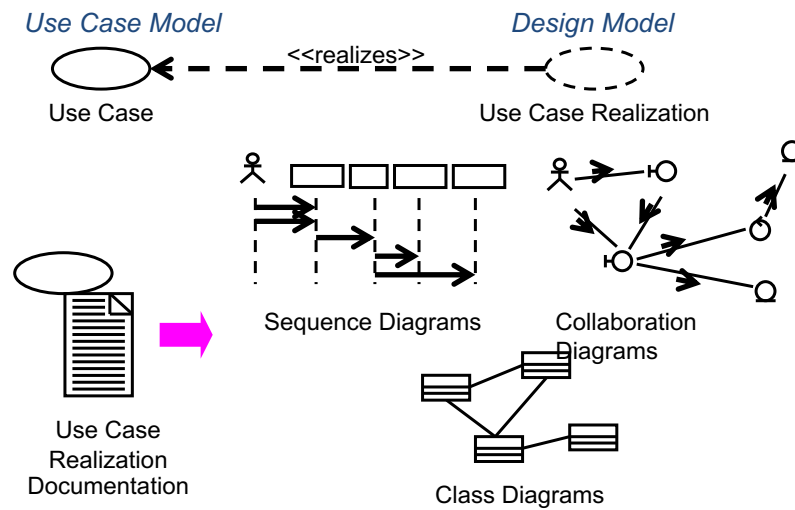
Modeling Objects to Sequence
(Problem Specification to Objects/Class)

Saurabh Tiwari

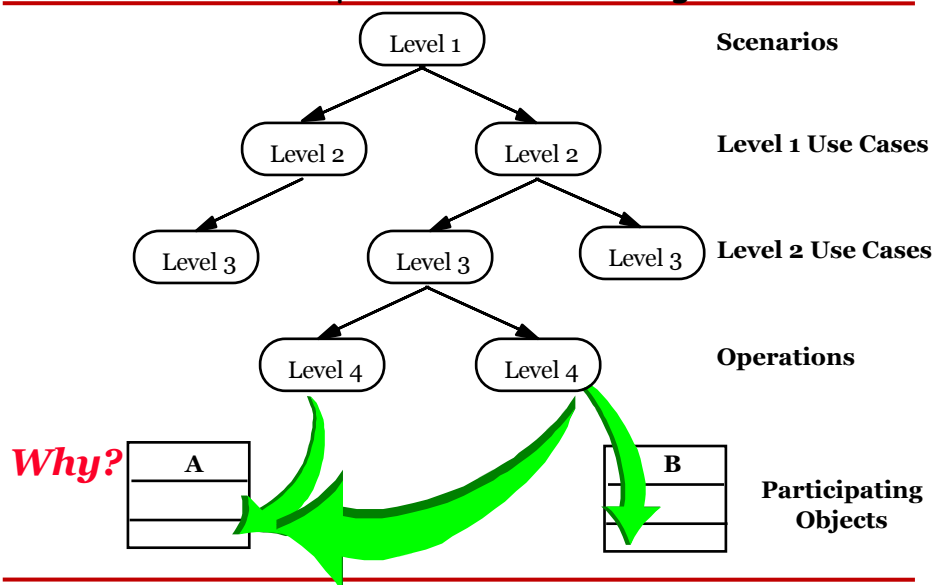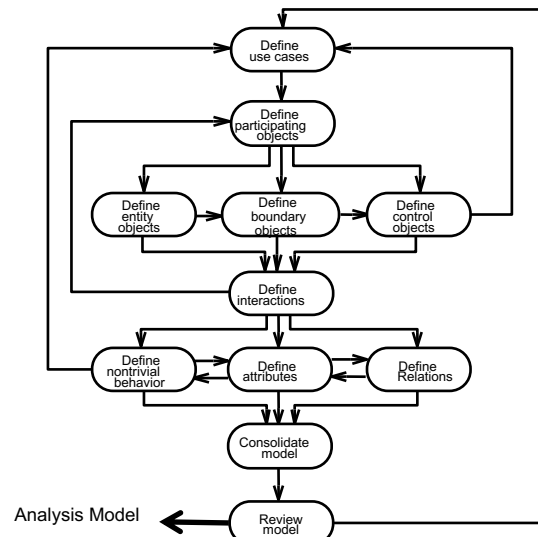# An overview of OOSE development activities and their products

Use Case Realization

Use Case Model   *Design Model*

Use Case   Use Case Realization

Sequence Diagrams   Collaboration Diagrams

Use Case
Realization
Documentation

Class Diagrams

From Use Cases to Objects: Why Functional Decomposition is not Enough

Level 1 — **Scenarios**

Level 2   Level 2 — **Level 1 Use Cases**

Level 3   Level 3   Level 3 — **Level 2 Use Cases**

Level 4   Level 4 — **Operations**

Why?   A   B — **Participating Objects**

Analysis Modeling

Analysis Model

Types of Analysis Classes

- Entity classes:
  - Model persistent data, real world entities, e.g., roles, invoices, databases, file
- Boundary classes:
  - Interaction between the system and its actors, e.g. receiving/presenting data
  - Examples: printer interfaces, terminals, sensors, APIs, forms, GUI items
  - Each boundary class should be related to at least one actor
- Control classes:
  - Classes for coordination, sequencing, processing, transactions, control of other objects etc.
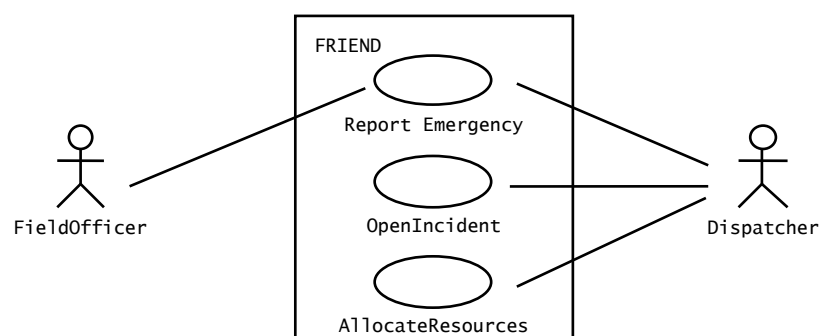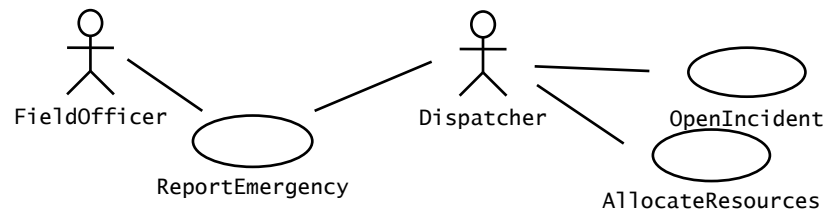
An Example

- FRIEND
  - A distributed information system for accident management. It includes many actors such as *FieldOfficer*, who represent the police, fire officers who respond to accidents, and *Dispatcher*, the polish officer responsible for answering 911 calls and dispatching resources to an accident. FRIEND supports both actors by keeping track of incidents, resources, and task plans. The *FieldOfficer* and the *Dispatcher* interact through different interface – *FieldOfficer* interacts FRIEND through a mobile personal assistant, and *Dispatcher* access FRIEND through a workstation.
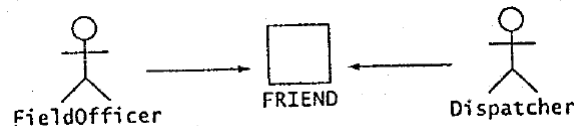
Example: Report Emergency use case in the Accident Management System (FRIEND)

Communication between actors and use cases in FRIEND.

FieldOfficer — ReportEmergency — Dispatcher — OpenIncident — AllocateResources

Use case ReportEmergency

Use case name	**ReportEmergency**
Actors	FieldOfficer, Dispatcher
Entry condition	1. The FieldOfficer activate the "report Emergency" function of her terminal
Flow of event	2. System responds by presenting a form with different details to filled-in
	3. FieldOfficer completes the form. She may also describe possible responses to the situation and request specific resources. She submits the form
	4. The Dispatcher reviews the information and creates an Incident in the DB by invoking **OpenIncident** use case. All the information received from FieldOfficer is then stored in the DB. The Dispatcher selects a response and allocates resources to the Incident (with **AllocateResources** use case) and acknowledge the Emergency Report by sending a FRIENDgram to the FieldOfficer
Exit condition	5. The FieldOfficer receives the acknowledgement and the selected response

Identify Entity Objects

Heuristics
- Terms those are needed to clarify in order to understand the use case
- Recurring nouns
- Real-world entities
- Data sources or sinks

Use case ReportEmergency

- Entity Objects
  - Dispatcher, EmergencyReport, FieldOfficer, and Incident

Dispatcher	Police officer who manage incidents. A Dispatcher opens, documents, and closes Incidents in response to emergency Report. Dispatchers are identified with batch numbers
Emergency Report	Initial incident information report
Field Officer	Police officer on duty and on the spot. She is responsible for reporting the incident. Recognized by her batch number.
Incident	All relevant information of the incident

Identifying Boundary Objects

- Represent system interface with the actors of the system
- Each actor must interact with at least one boundary object
- Need not to be much elaborated here

Identifying Boundary Objects

Heuristics

- Identify user interfaces for actor-system interactions
- Identify forms in which data is to be filled in
- Identify communication/messages between actor and systems
- When multiple actors are involved in a use case, identify actor terminals
- Do not model visual aspects of the interface
- Use user/customer language for describing interfaces

Use case Report Emergency

- Boundary Objects
  - AcknowledgementNotice, DispatcherStation, ReportEmergencyButton, EmergencyReportForm, FieldOfficerStation, IncidentForm

Acknowledgement Notice	
DispatcherStation	Computer used by the Dispatcher
ReportEmergency Button	
EmergencyReport Form	
FieldOfficerStation	Mobile computer used by the FieldOfficer
IncidentForm	This form is presented to the Dispatcher on the DispatcherStation when the EmergencyReport is received

Identify Control Objects

Heuristics
- Identify one control object per use case
- Identify one control object per actor in the use case
- Life span of a control object should cover the extent of the use case or extent of a user session
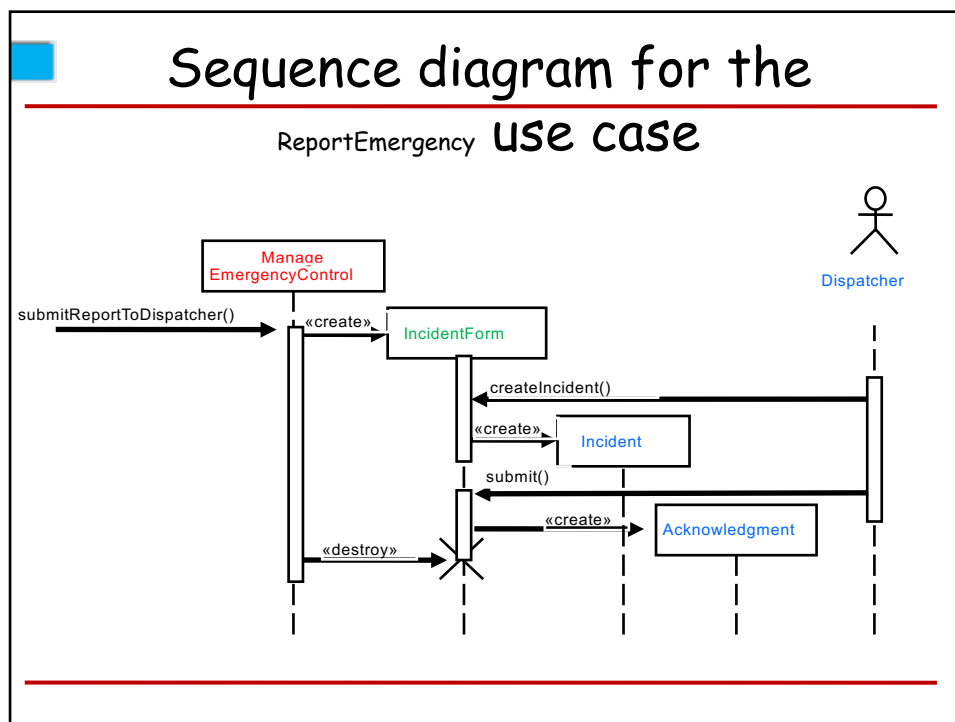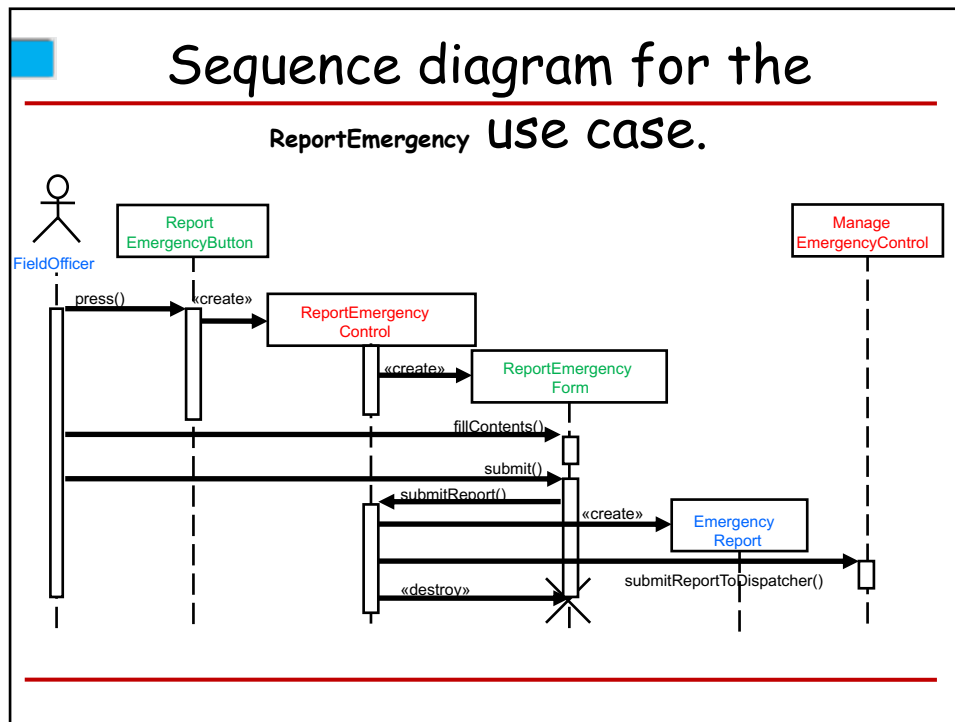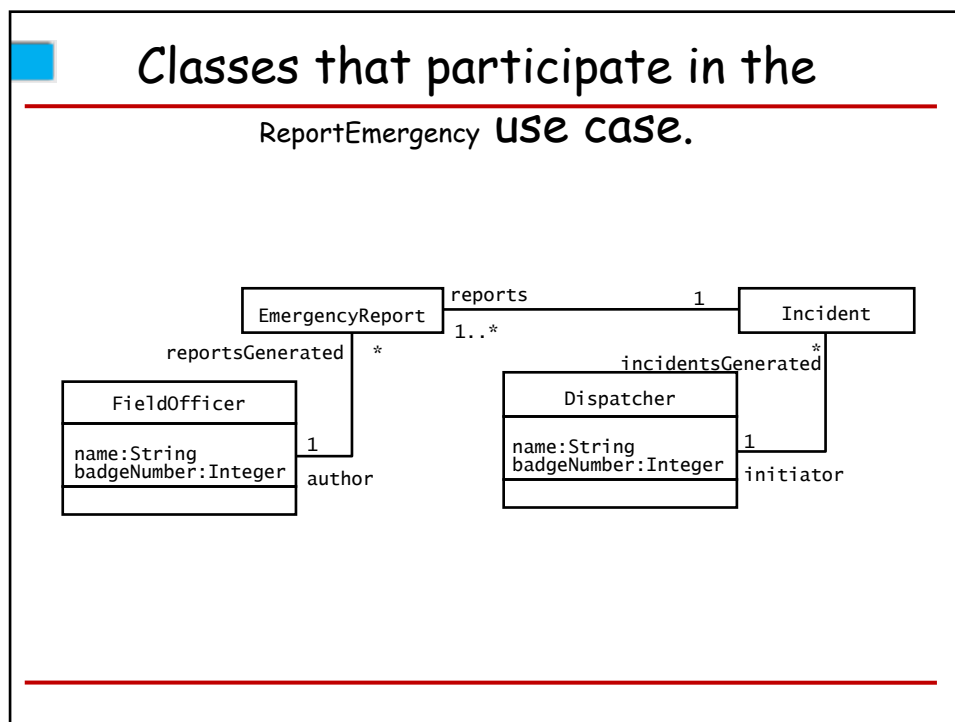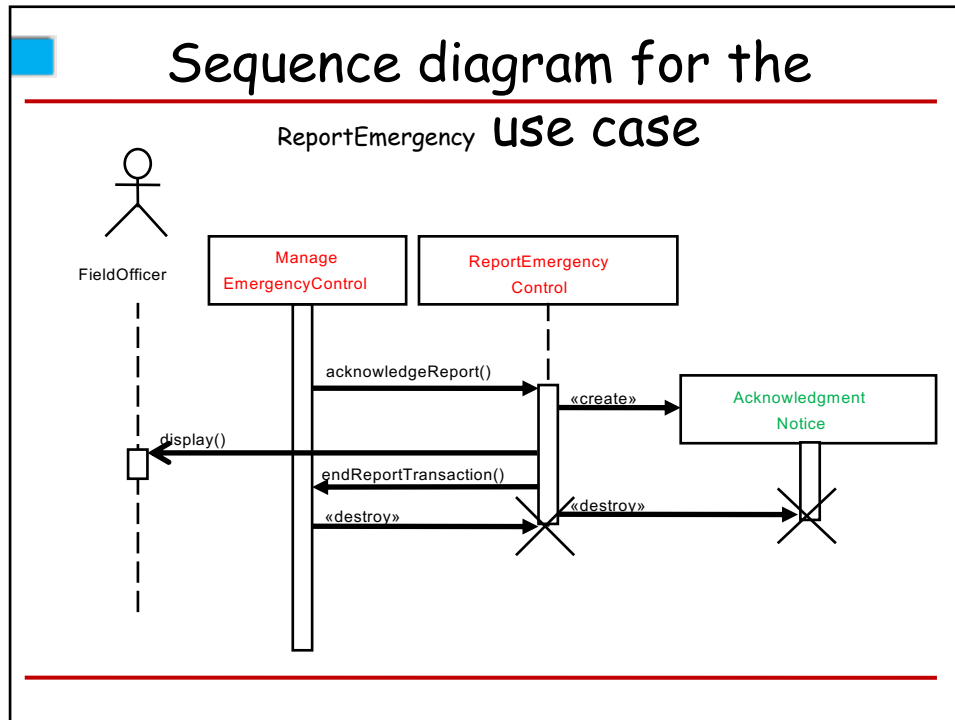
Use case Report Emergency

- Control Objects
  - ReportEmergencyControl, ManageEmergencyControl

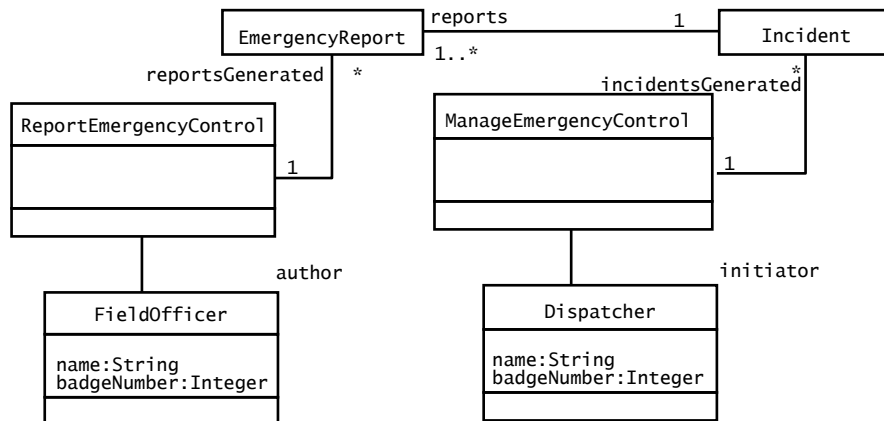| ReportEmergency Control	The object is created when the FieldOfficer selects the "Report Emergency" button. It then creates an EmergencyReportForm and presents it to the FieldOfficer. After getting all the information, it forwards this information to the DispatcherStation. It then waits for an acknowledgement. When received, it creates an acknowledgementNotice and displays it to the fieldOfficer
ManageEmergency Control	This object is created when an EmergencyReport is received. It then creates an IncidentForm and displays it to the Dispatcher. Once the Dispatcher has created an incident, allocated resources, and submitted an acknowledgement, it forwards the acknowledgement to the FieldOfficerStattion

Mapping Use Cases to Object Interactions (Sequence Diagrams)

- To ensure the completeness of our model
- Sequence diagrams tie use cases with objects and their interaction
- Not good at the user level but is a step in transformation
- Allows to find missing objects or grey areas in requirement specification

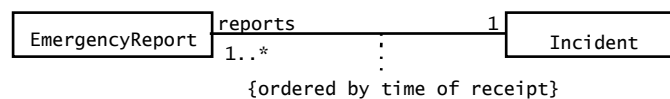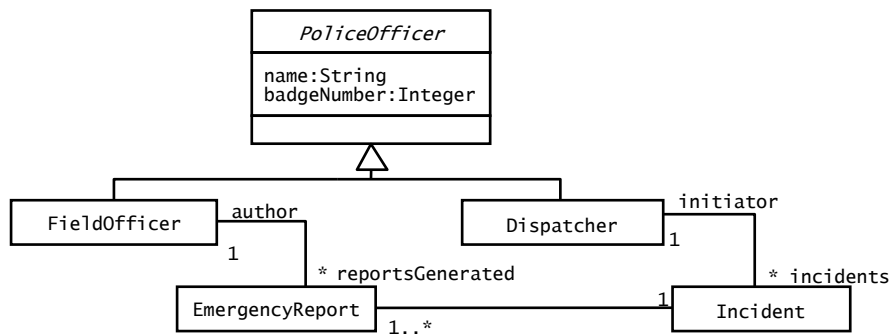# Sequence diagram for the
ReportEmergency use case.

Sequence diagram for the
ReportEmergency use case

Sequence diagram for the ReportEmergency use case

Classes that participate in the ReportEmergency use case.

Classes that participate in the

ReportEmergency use case.

```
                    EmergencyReport  reports          1    Incident
                                     1..*
reportsGenerated   *                    incidentsGenerated   *

ReportEmergencyControl              ManageEmergencyControl

                   1                                     1

              author                              initiator

   FieldOfficer                      Dispatcher

name:String                       name:String
badgeNumber:Integer               badgeNumber:Integer
```

An example of constraint.

```
EmergencyReport  reports          1    Incident
                 1..*          :
              {ordered by time of receipt}
```

An example of a generalization

Another example of a generalization
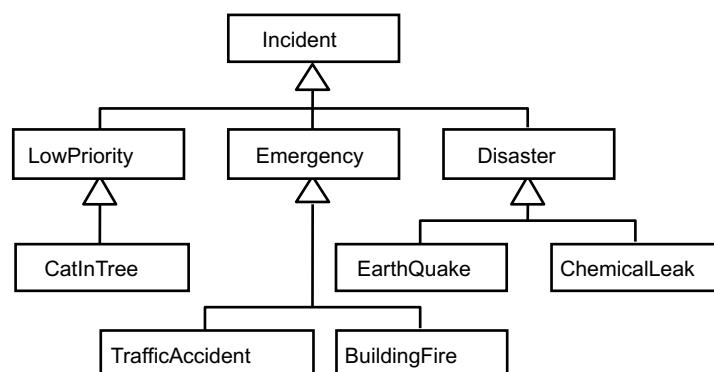
Summary: Requirements Analysis

1. What are the transformations?   → Functional Modeling
   – Create *use case diagram and scenarios*
     • Talk to client, observe, get historical records, do thought experiments

2. What is the structure of the system?   → Object Modeling
   Create *class diagrams*
     Identify objects.
     What are the associations between them? What is their multiplicity?
     What are the attributes of the objects?
     What operations are defined on the objects?

3. What is its behavior?   → Dynamic Modeling
   Create *sequence diagrams*
     Identify senders and receivers
     Show sequence of events exchanged between objects. Identify event
     dependencies and event concurrency.
   Create *state diagrams*
     Only for the dynamically interesting objects.