# UML

Unified Modeling Language

www.omg.org

# UML

UML, why?

- Need to model software
- Visually expressive language
- Process Independent.  UML is a language, not a methodology
- Supports higher level practices such as collaborations, frameworks, and patterns

# UML - History

- During the peak of the OO revolution in the early 90’s, many methods sprung up for modeling software design.
- Three of the more popular methodologists, Booch (OOD), Rumbaugh (OMT), and Jacobson (OOSE) were eventually assimilated by Rational.
- Rational proposed UML 0.8 as the standard in ‘95.
- Industry did not trust Rational so OMG (Object Management Group) formed a task force to aid in the creation of this new software standard.
- OMG Combined several modeling language proposals and eventually birthed UML 1.1 in ’97.
- UML updates are now handled by a Revision Task Force (RTF) of the OMG. Current version is 1.4. Version 2.0 is due out in the very near future.

# UML - Demo

## Requirements Analysis and Design

- Design a call tracking application
  - Application is to be designed for tracking telephone usage by roommates thus simplifying payment of Telco bills each month.

## UML - Demo

Basic steps

- Start Analysis with Use Cases
- Realize Use Cases to find classes
- Design Classes

Steps will vary based on the methodology

# UML - Demo

- Use Case
  - Use case is a short sequence of events.  The events are written from the perspective of the users (actors).
  - Communicate with end users using use cases.

# UML - Demo

**Call Tracker**

Finding Use Cases

Basic System Functions

1. Make a normal call
2. Make an emergency call
3. Print Monthly Statement

# UML - Demo

## Use Cases

1. Make a normal call
   1. Caller picks up phone
   2. Caller enters their pin number
   3. Caller waits for confirmation tone
   4. Caller enters telephone number to dial
   5. Caller talks on phone
   6. Caller hangs-up phone

# UML - Demo

## Use Case Finding

1. Make a normal call
    1. Caller picks up phone
    2. Caller enters their pin number
    3. Caller waits for confirmation tone
    4. Caller enters telephone number to dial
    5. Caller talks on phone
    6. Caller hangs-up phone

FIND THE ACTORS AND DEVELOP USE CASES

# UML - Demo

## Use Case Finding

1. Make a normal call
   1. Caller picks up phone
   2. Caller enters their pin number
   3. Caller waits for confirmation tone
   4. Caller enters telephone number to dial
   5. Caller talks on phone
   6. Caller hangs-up phone

Use Cases:

Pickup Phone, Enter Pin, Start Call, End Call

Actors:

Caller, Telephone, Database

# UML - Demo

## Use Case “Enter PIN” or Verify PIN

1 – User enters PIN

2 – If PIN valid user hears a confirmation tone

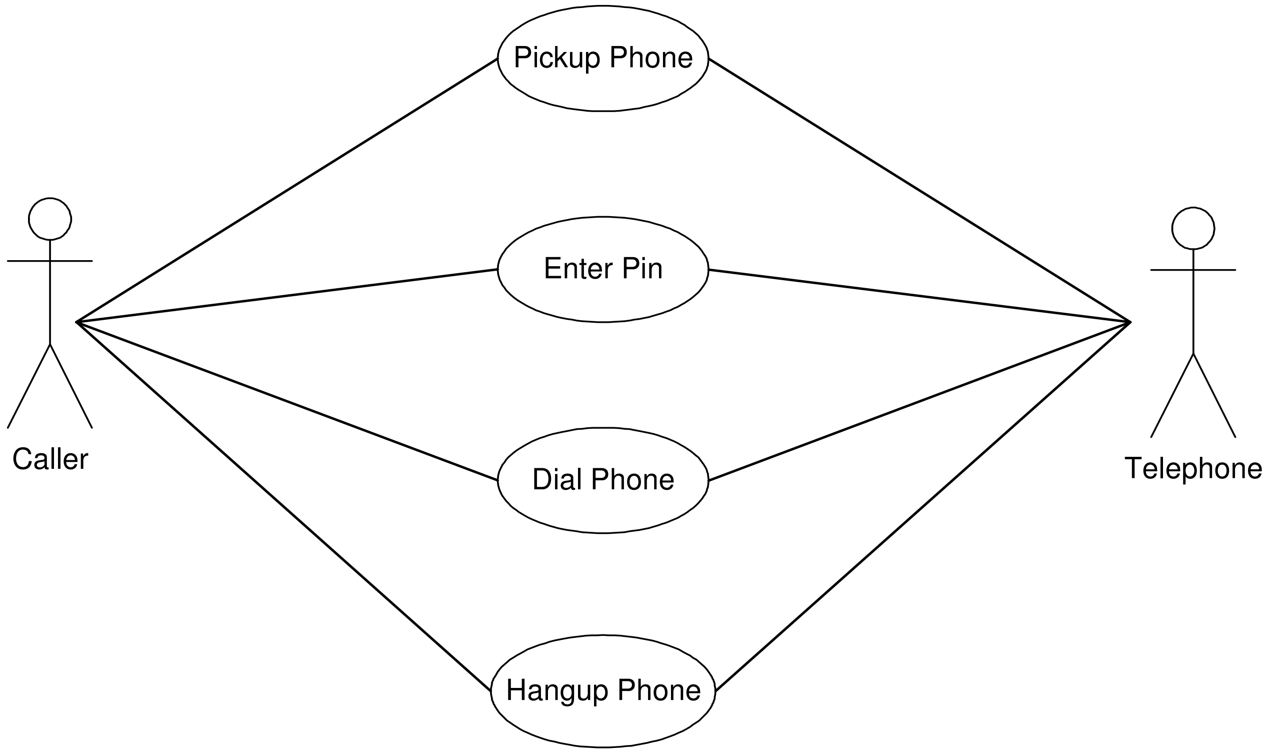
2 (alt) – If PIN invalid use case ends

3 – User hears dial tone indicating phone number can now be dialed

# UML - Demo

## Use Cases

1. Make a normal call – Use Case composite

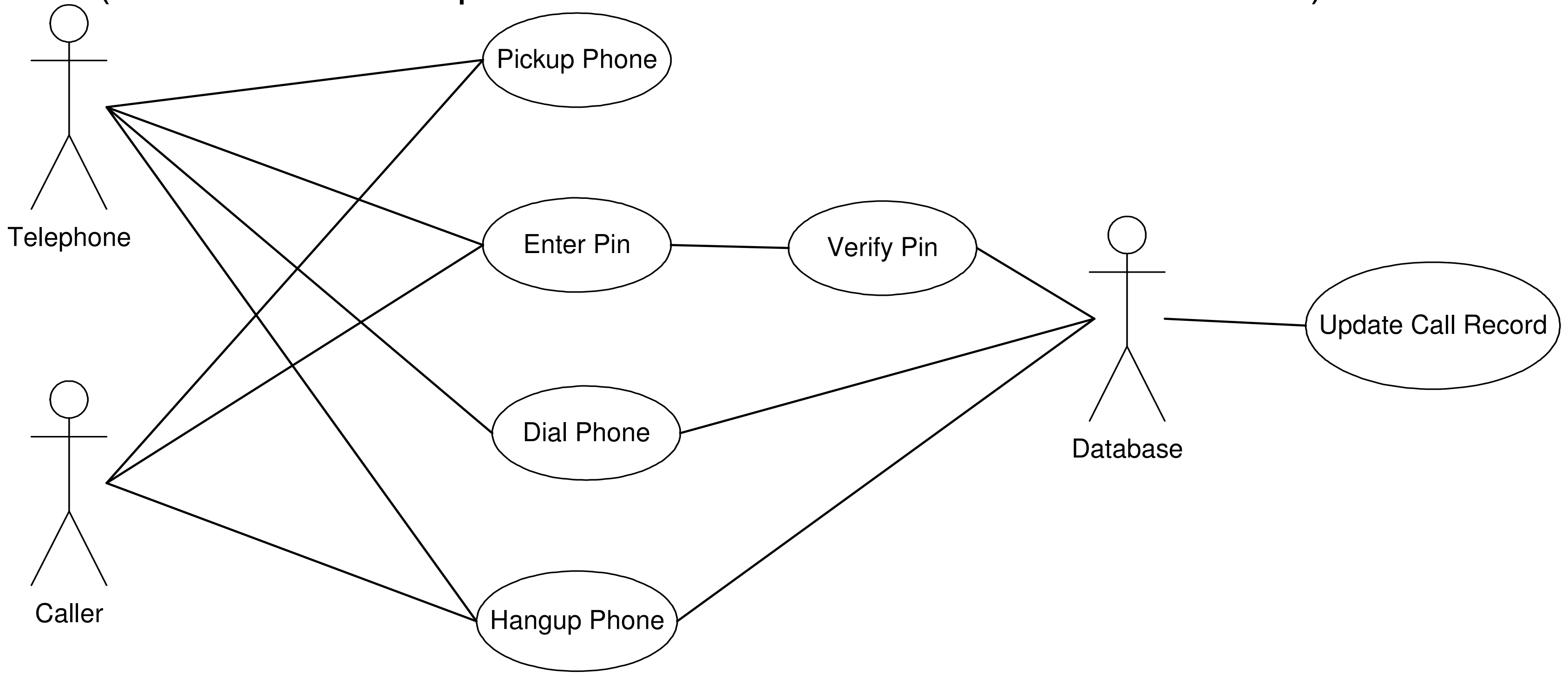


# UML - Demo

## Use Cases

1. Make a normal call

(note that the Telephone actor and Caller actor are redundant)



## UML - Demo

Use Cases - realizing

1. Make a normal call

Realizing a use case:  Takes the use case from the user’s point of view and change it to the systems point of view.

This is usually done through interaction diagrams such as sequence diagrams and collaboration diagrams.

## UML - Demo

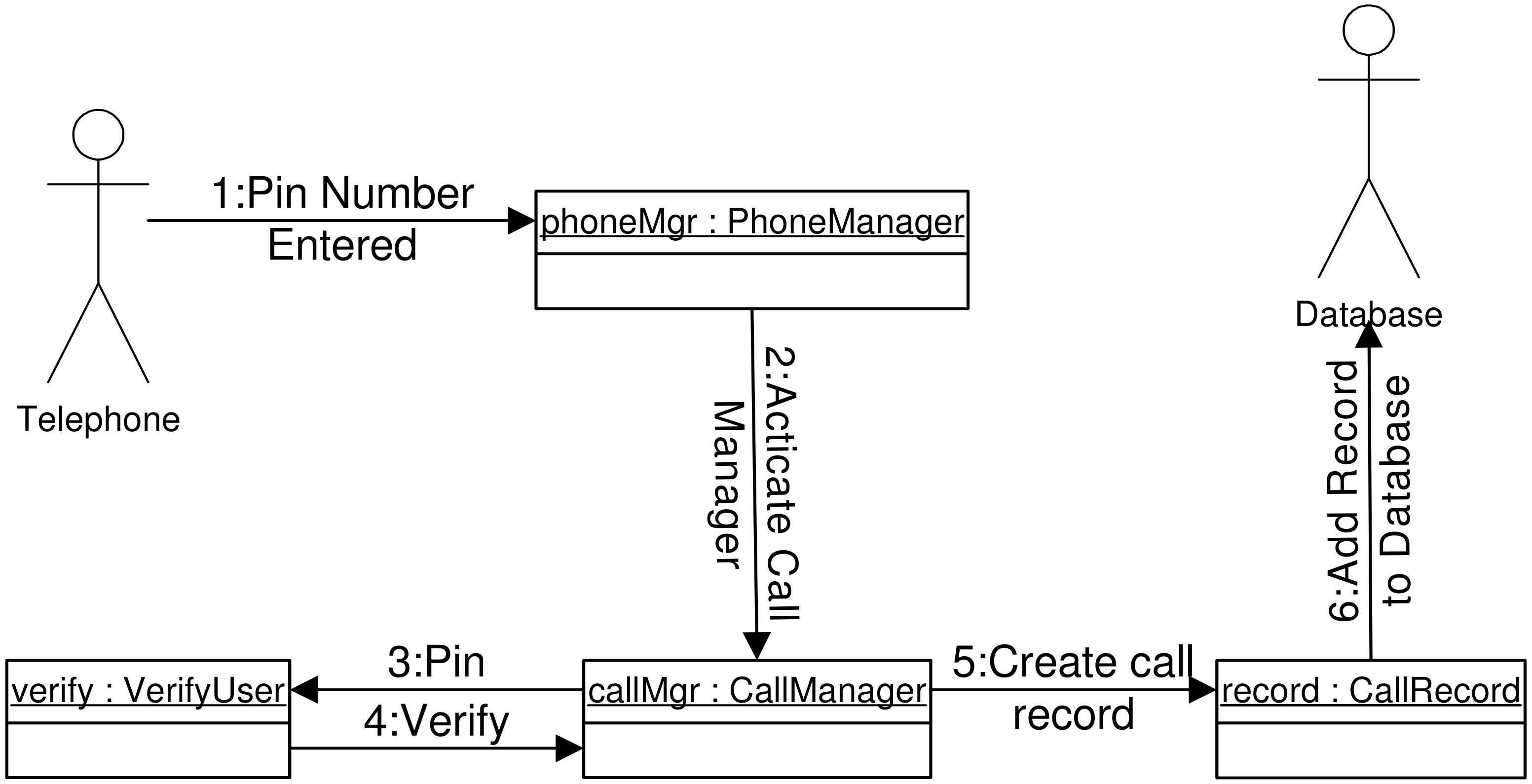
Use Cases – realizing

Collaboration Diagram

- Capture the behavior of a single use case
- When designing complex systems, this can help you to determine what classes you shall need
- When realizing other use cases of your system, these classes can be re-used
- For simple systems, you can often develop a class diagram first
- Interaction diagrams model objects, instances of classes

# UML - Demo

## Collaboration Diagram – Verify Pin Number Use Case



# UML - Demo

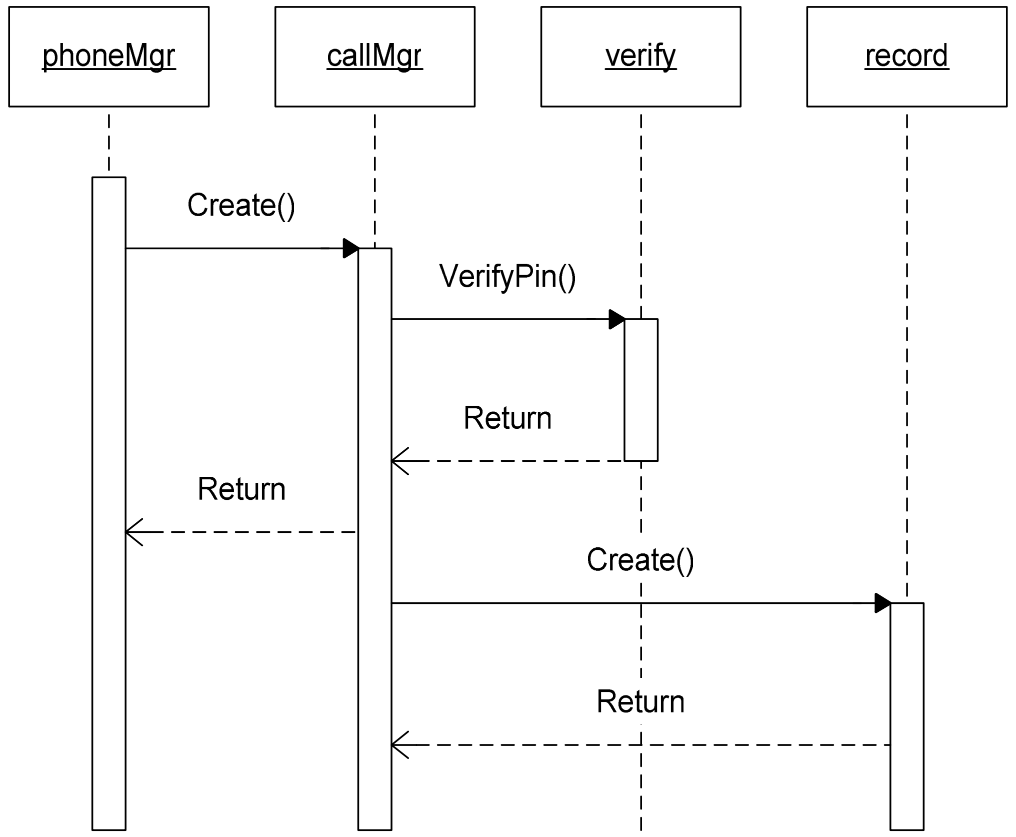
Collaboration Diagram -> Sequence Diagram

A collaboration diagram and a sequence diagram contain the same basic information.

Many CASE tools have functions that will convert between these diagrams.

# UML - Demo

## Sequence Diagram



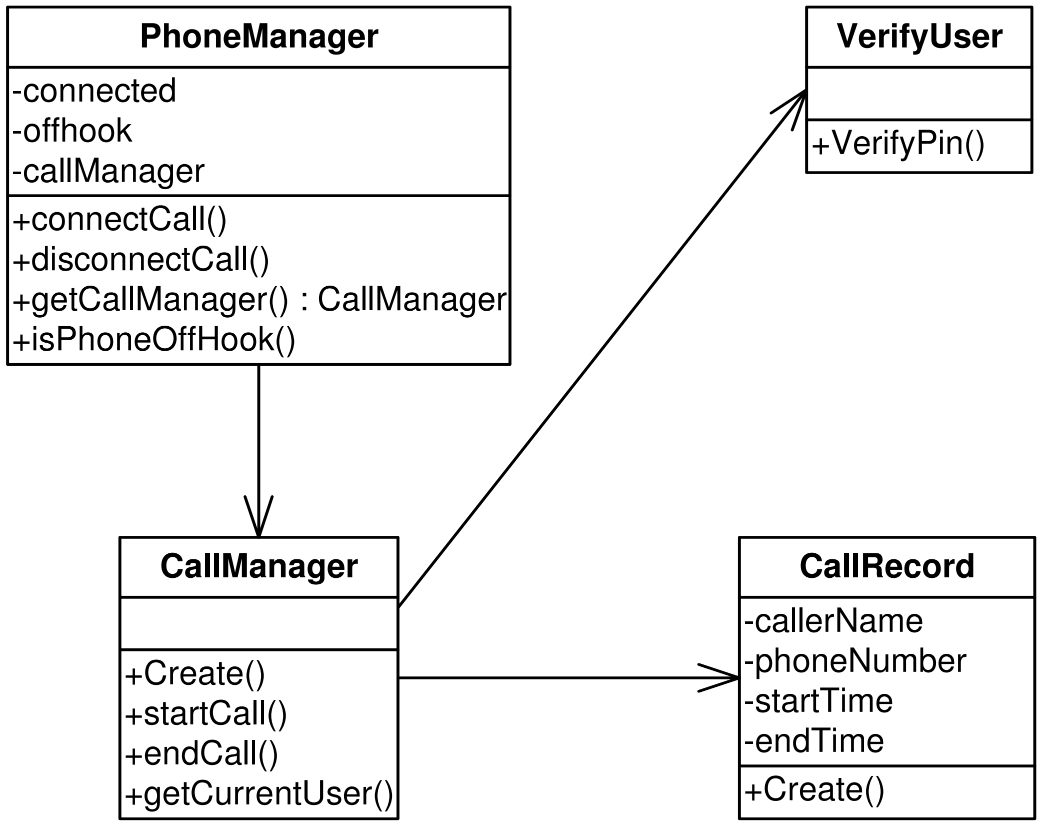
# UML - Demo

Class Diagram

From the interaction diagram, you can determine what major behaviors and attributes your class will need.

# UML - Demo

## Class Diagram

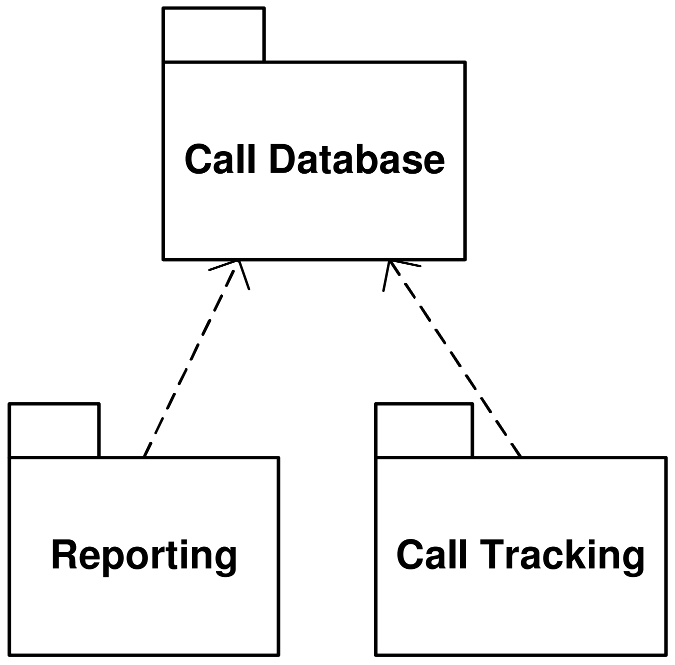


# UML - Demo

## Packages

Package contains a logical grouping of classes

Shows dependencies between classes



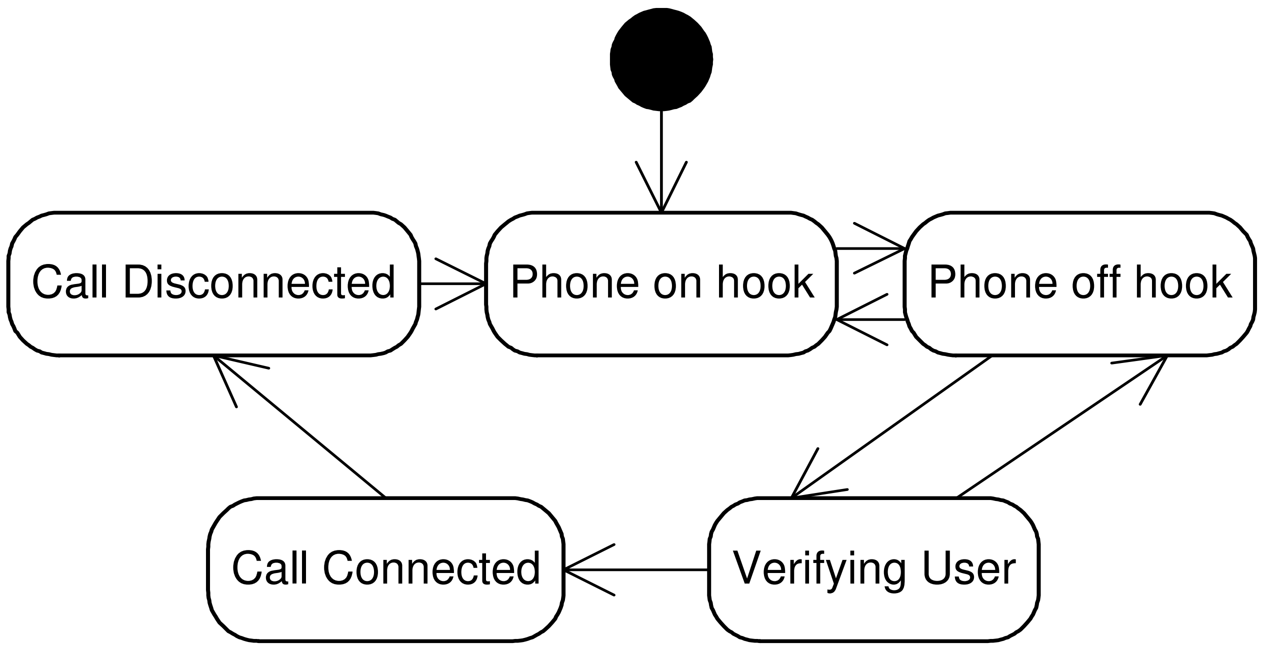
# UML - Demo

## State Diagrams

- State diagrams describe the behavior of a system
- Shows all the possible states an object can be in
- State diagram drawn for a single class

# UML - Demo

## State Diagrams - Phone Manager



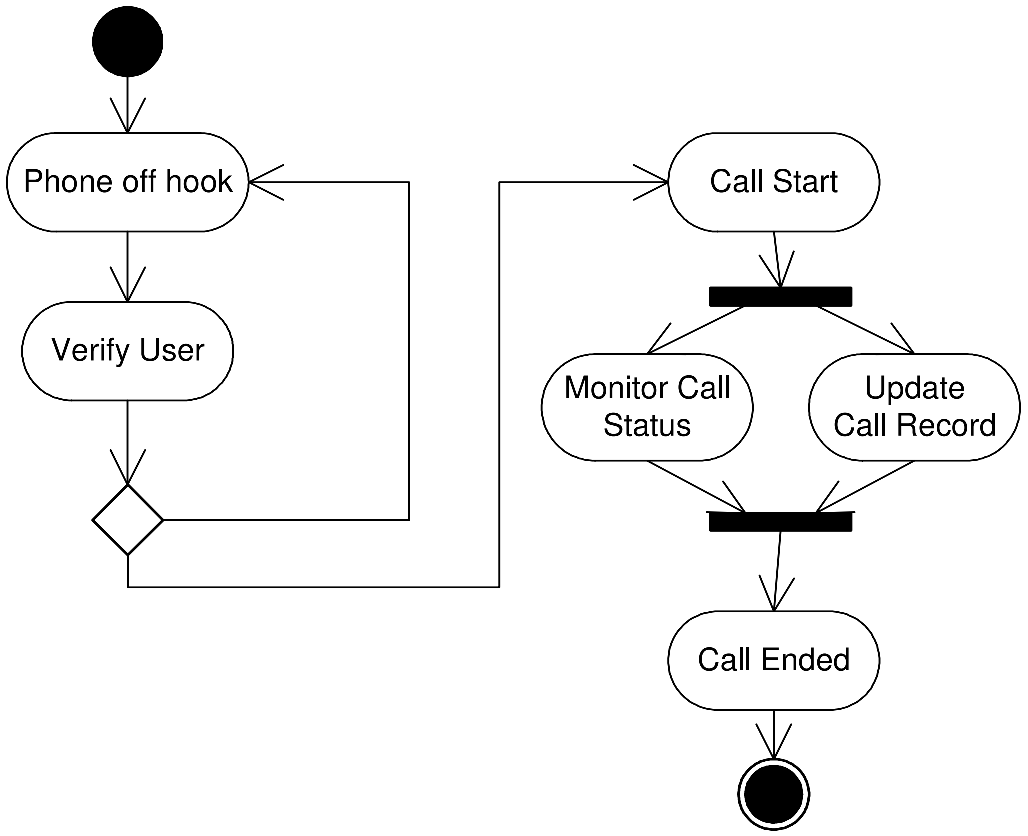
# UML - Demo

## Activity Diagrams

- Activity diagrams describe a sequence of activities
- Shows parallel states
- Shows conditional states

# UML - Demo

## Activity Diagrams



# UML

Other UML diagrams include

Deployment,

Object,

Physical

See UML specification for more details:

http://www.omg.org/technology/documents/formal/uml.htm

# UML CASE Tools

CASE Tools

Computer Aided Software Engineering

There are a variety of CASE tools on the market, most support UML.

These include:

- Microsoft Visio
- Rational Rose
- Pencil and Paper

Some IDEs also include CASE tools:

- JDeveloper
- Microsoft Visual Studio

# UML Methodologies

To effectively use UML, you should follow a methodology

A methodology describes how to analyze, design, and even implement your system

The Call System demo followed the Unified Process

There are many methodologies, often the company you are working has their own process

# UML Methodologies

UML and how to use it; major Methodologies

Rational Unified Process – RUP

www.rational.com

http://www.iconprocess.com/iconProcess/phases.php

Agile Modeling – As an enhancement to
Extreme Programming (XP)

www.agilemodeling.com

## UML Future

*"UML will become a programming language.  There is no technical barrier – only a political barrier,"  -*

- Ivar Jacobson, June 11 2001, UML World Conference.