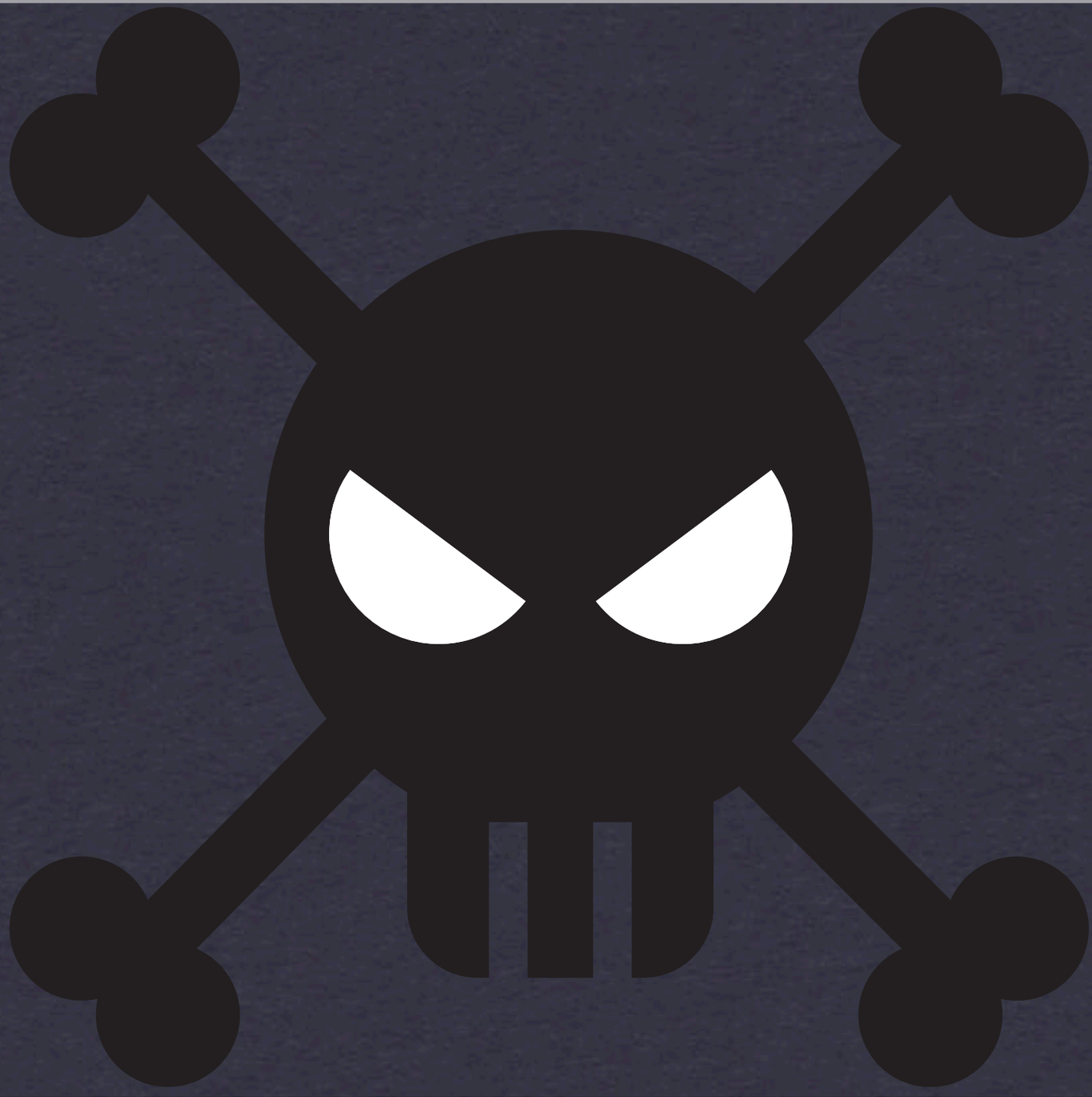


PROJECT

# SOFTWARE SPECIFICATIONS

## A USE CASE CRASH COURSE (V1.1)

DATE JAN 03, 2008

PRESENTER JEREMY FISHER, AWT WEB DEV MGR

# Agenda

- Jeremy idealizes (process)
- Jeremy talks (use cases)
- Hand-out: Use Case Diagram Cheat Sheet
- Hand-out: Written Use Case Cheat Sheet
- Use Case Walk Thru
- Hand-out: “Add a Lead” Sample Use Case
- Practical Exercise

# Warnings

- I was sober when authoring this presentation, and so can make no assurances as to the quality of my content.
- This presentation could also be called "Jeremy has fun with Key Note."

# One More Thing

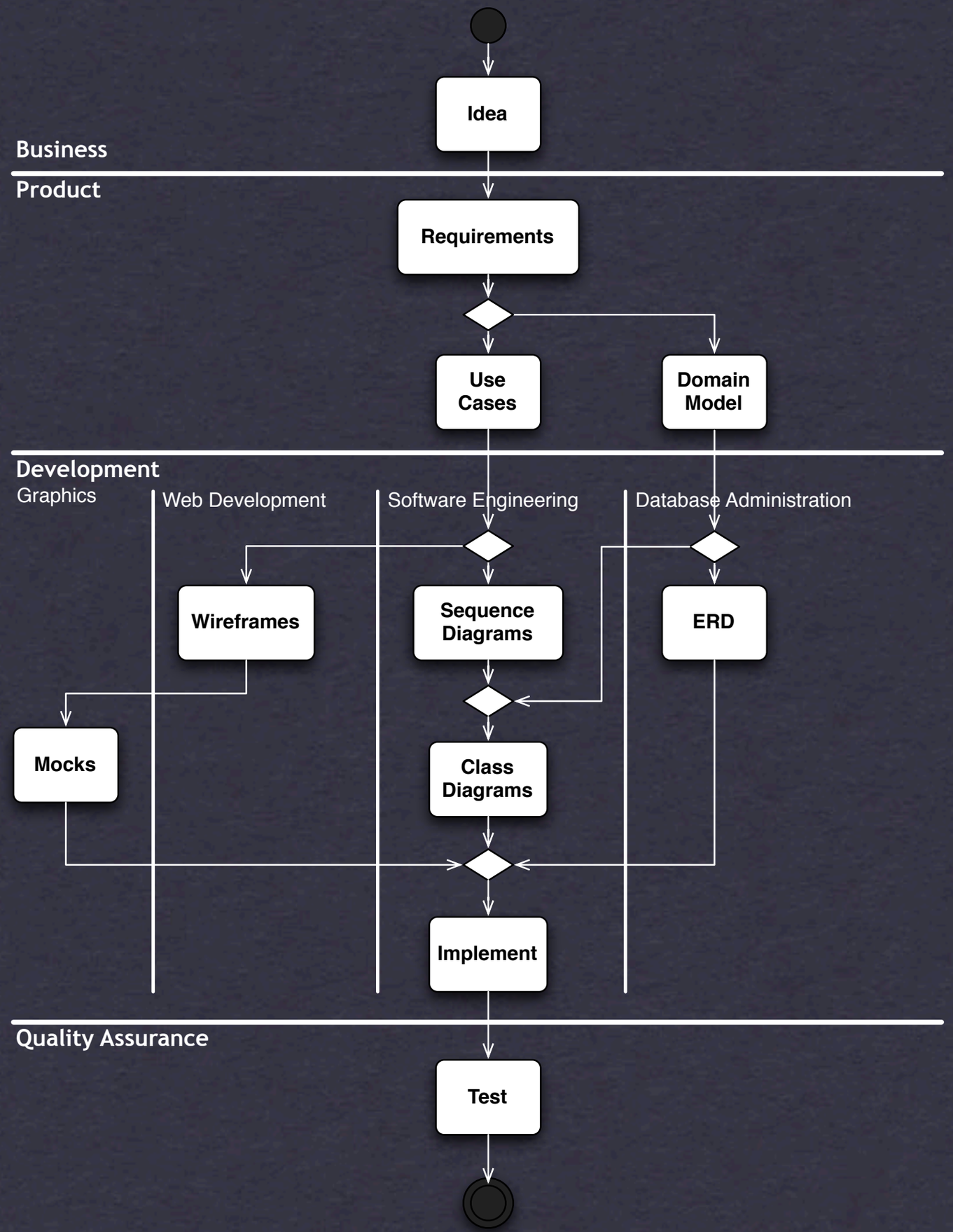
Please reserve questions & input for Q&A slides

## First...

# A few words on process and the big picture

# Process

- Idea
- Requirements
- Use Cases
- Domain Model
- Wireframes
- Sequence Diagrams
- ERD
- Mocks
- Class Diagrams
- Implementation
- QA



# Process

You must find the right balance of process for your project and people.

# Process

"Seriously though, 90% of the jobs I've had, I've had that job because:

1. Not enough process and/or management: Teams are disjointed, don't know what the hell each other is doing, don't care, or are full of inept people.
2. Too much process: The capable people are hamstrung by miles of red-tape, in a process that guarantees nothing ever really gets finished, because business needs are *never* concrete."

-Bryan Varner, Software Consultant
(And close friend of the author)

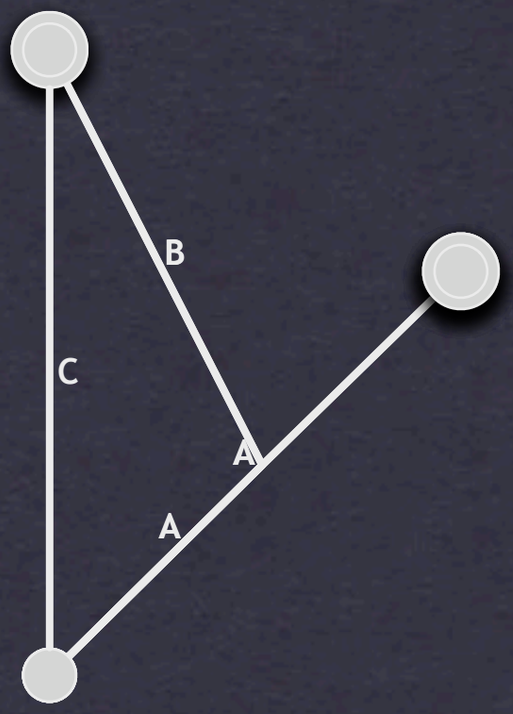
# Process: Agile Development


- Development Iterations
- Collaboration/Communication
- Working Software

# Why Bother with Specifications?

"Measure twice, cut once."
- English Proverb

# Why Bother with Specifications?




A + B > C

# Roles - The Hats We Wear

- Skill sets in action.
- Not positions.
- One person often fills several roles.

# Roles

## It takes a lot of roles to build a web app...

- Product developer
- Systems analyst
- Software engineer
- Web Developer
- User Interface Designer
- Usability Expert
- Manager (Engineering, Product, QA, Whatever)
- Information Architect
- Release Engineer
- Quality Assurance Engineer
- Graphic Designer
- Database Administrator
- Systems Administrator
- Chief Mugwump
- Pirate
- Ninja
- Storm Trooper
- etc.

**But all of these may be filled by one person.**

**Or a huge team.**

**Or anything in-between.**

# Job Encapsulation

"You could be outsourced because your job is so richly defined that it can be documented and explained to any reasonable professional on the planet who might work for much less than you. Sorry. Go Capitalism!"

- Micheal Lopp

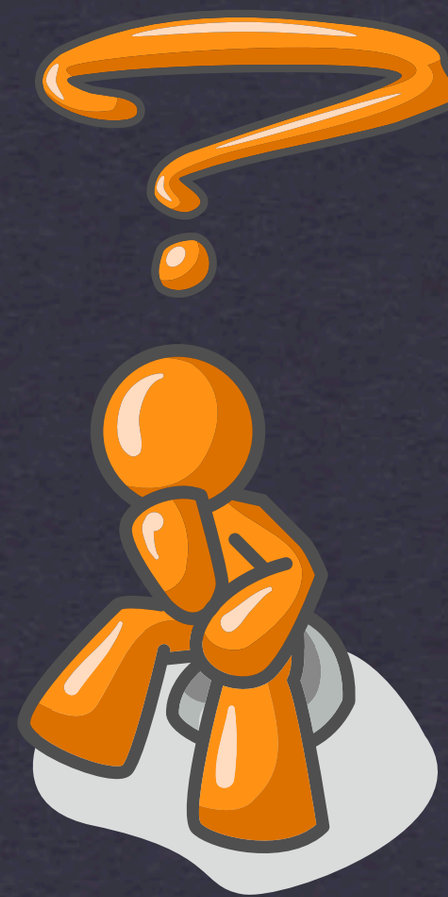
# Our Process...?

"When one has no form, one can be all forms; when one has no style, he can fit in with any style."

- Bruce Lee

# Questions?

(OR ANSWERS?)



## And now...

# Developing Software Specifications

# The Philosophy of Happiness

"Happiness is that state of consciousness which proceeds from the achievement of one's values."

- Ayn Rand

# The Happiness of Software

"Software happiness is that state of consciousness which proceeds from software having achieved the values -- erm, requirements -- of it's business."

- Jeremy Fisher

# The System

We are working on (not fighting) "The System".

The term is so pervasive that it got it's own slide.

# High-Level Requirements

- Requirements are things the system will (or shall) do.
- Requirements are the *What*, not the *How* of software.
- May be notes from stakeholder meetings.
- Are usually presented in a list, such as this:
  - The system shall...
  - The system must be able to...
  - Thou shalt not...

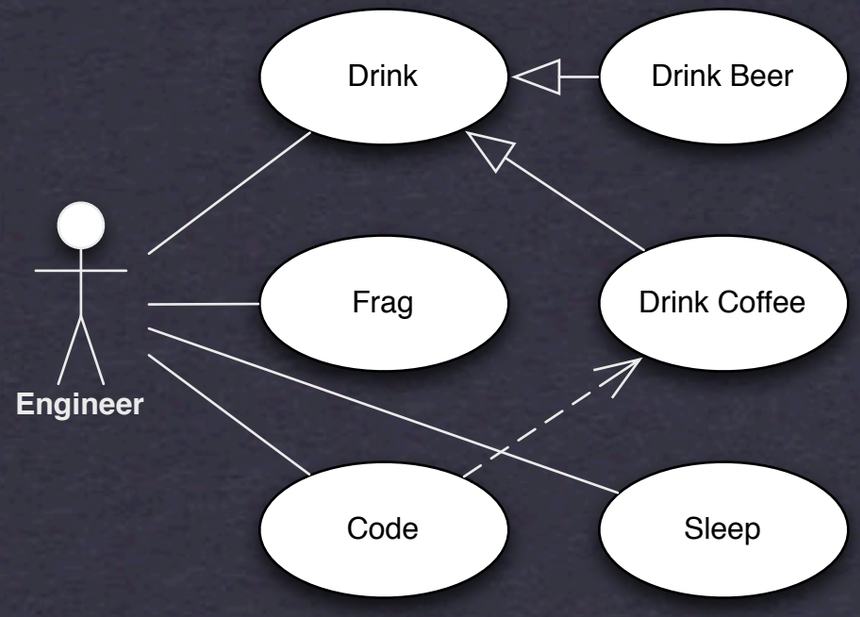
# High-Level Requirements

1. The system shall allow leasing consultants and automation systems the ability to add a lead to the database
2. The system shall not permit any request to add a lead that is already in the database.
   - A lead is only unique if no other leads exist with the same first name, last name, and either e-mail address or phone number.
3. Only leasing consultants, leasing managers, and qualified automation systems may add leads to the database.

**Note: It helps to provide a means of referencing requirements. That's what the numbers are for. You'll thank me later.**

# Use Cases

- Something that provides a measurable result to the user or some external system
- A scenario, or related collection of scenarios, in the use of the system
- The actors involved in an interaction and the names and types of interactions
- How the system is used and by whom
- Generalizations of system behavior
- May be general (a few sentences) or detailed (what we'll be discussing today)



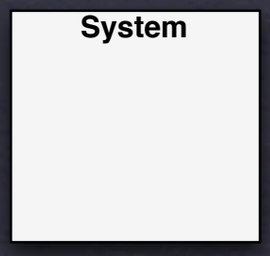
# Use Cases: A Quick Reference

## Use Case Diagram Cheat Sheet
## (Hand-out time)

### Entities

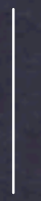

**Actor**
External systems or people that interact with the system


**Use Case**
Types of interactions with and within the system



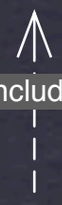

**System Boundary**
Shows boundaries of the system by containing use cases

### Relationships



**Communication**
Shows communication between actors and use cases

**Generalization**
Shows one use case or actor extending another

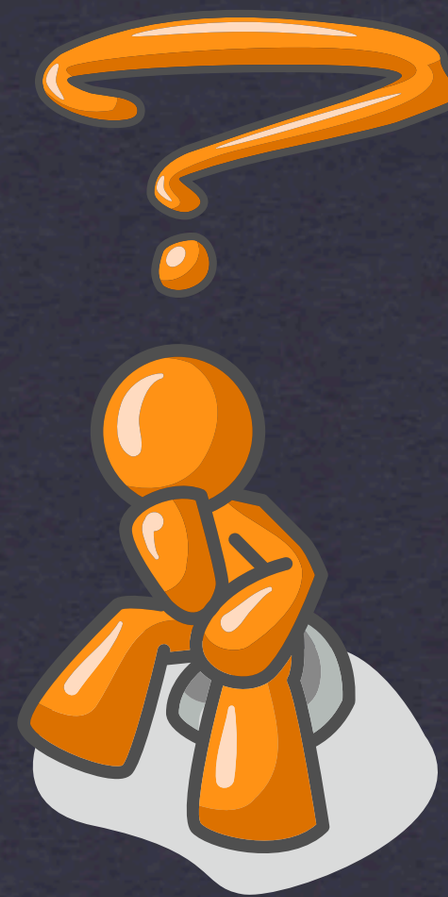



**Inclusion**
Shows a use case including another

http://dev.residentsource.com/trac/wiki/EngineeringUseCaseDiagrams

# Questions?

(OR ANSWERS?)



## Finally...

# A Use Case Walk Thru

# Use Cases: Find the Actors

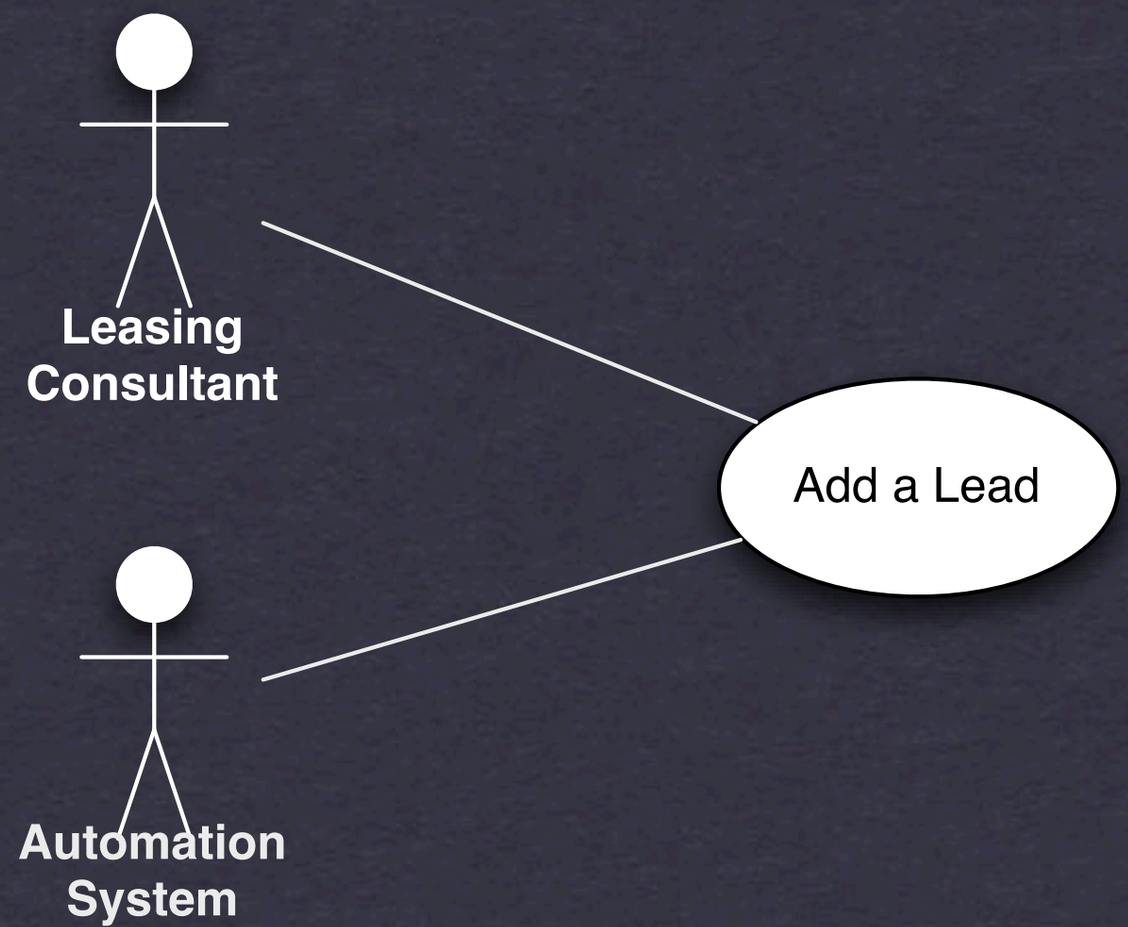
1. The system shall allow **LEASING CONSULTANTS** and **AUTOMATION SYSTEMS** the ability to add a lead to the database
2. The system shall not permit any request to add a lead that is already in the database.
   - A lead is only unique if no other leads exist with the same first name, last name, and either e-mail address or phone number.
3. Only **LEASING CONSULTANTS** and qualified **AUTOMATION SYSTEMS** may add leads to the database.

# Use Cases: Diagram Actors


# Use Cases: Find the Interactions

1. The system shall allow leasing consultants and automation systems the ability to **ADD A LEAD** to the database
2. The system shall not permit any request to add a lead that is already in the database.
   - A lead is only unique if no other leads exist with the same first name, last name, and either e-mail address or phone number.
3. Only leasing consultants, leasing managers, and qualified automation systems may add leads to the database.

# Use Cases: Diagram Interactions



# Use Cases: Writing

## A template helps. I like this one from O'Reilly's *Learning UML 2.0*

| Title | |
|---|---|
| Related Requirements | |
| Goal in Context | |
| Preconditions | |
| Successful End Result | |
| Failure End Result | |
| Primary Actors | |
| Secondary Actors | |
| Trigger | |
| Included Cases | |
| Main Flow | |
| Extensions | |

# Use Cases: Writing

## But what do these fields mean?

| Title | |
|---|---|
| **Related Requirements** | References to high-level requirements |
| **Goal in Context** | Why does this matter? |
| **Preconditions** | Stuff we need before this use case |
| **Successful End Result** | What happens when the use case completes successfully? |
| **Failure End Result** | What happens on failure? |
| **Primary Actors** | Who is primarily involved? (Usually diagrammed on the left) |
| **Secondary Actors** | Who else is involved? (Usually diagrammed on the right) |
| **Trigger** | What happened to invoke this use case? A user action? A cron event? |
| **Included Cases** | What additional use cases does this one invoke? |
| **Main Flow** | The meat and potatoes |
| **Extensions** | Branching actions go here |

# Use Cases: Add a Lead

## Hmmm... what do we know so far?

| Add a Lead | |
|---|---|
| **Related Requirements** | 1, 2, 3 |
| **Goal in Context** | A lead contacts an apartment community, and their information is entered into the system for future reference |
| **Preconditions** | The user has proof of identity, lead detail data |
| **Successful End Result** | A new lead is entered into the database |
| **Failure End Result** | The request is rejected and the user notified that their request failed; the database is unchanged |
| **Primary Actors** | Leasing consultant, automation system |
| **Secondary Actors** | |
| **Trigger** | A leasing consultant or automation system asks the system to add a lead |
| **Included Cases** | |
| **Main Flow** | |
| **Extensions** | |

# Use Cases: Main Flow

This is where we explain the use case step-by-step.

1. The user asks the system to add a lead
2. The user submits the lead's detailed information
3. The lead is tested for uniqueness within the current community
4. The lead is added to the database

# Use Cases: Main Flow

Notice that the text is technology-agnostic.

1. The user **ASKS** the system to add a lead
2. The user **SUBMITS** the lead's detailed information

Do you see the words "click" or "button" anywhere? I didn't think so.

# Use Cases: Main Flow

## But wait... what about requirement #3?

3. Only leasing consultants, leasing managers, and qualified automation systems may add leads to the database.

# Use Cases: Main Flow

1. The user asks the system to add a lead

include::Verify Access

2. The user submits the lead's detailed information
3. The lead is tested for uniqueness within the current community
4. The lead is added to the database

# Use Cases: Main Flow

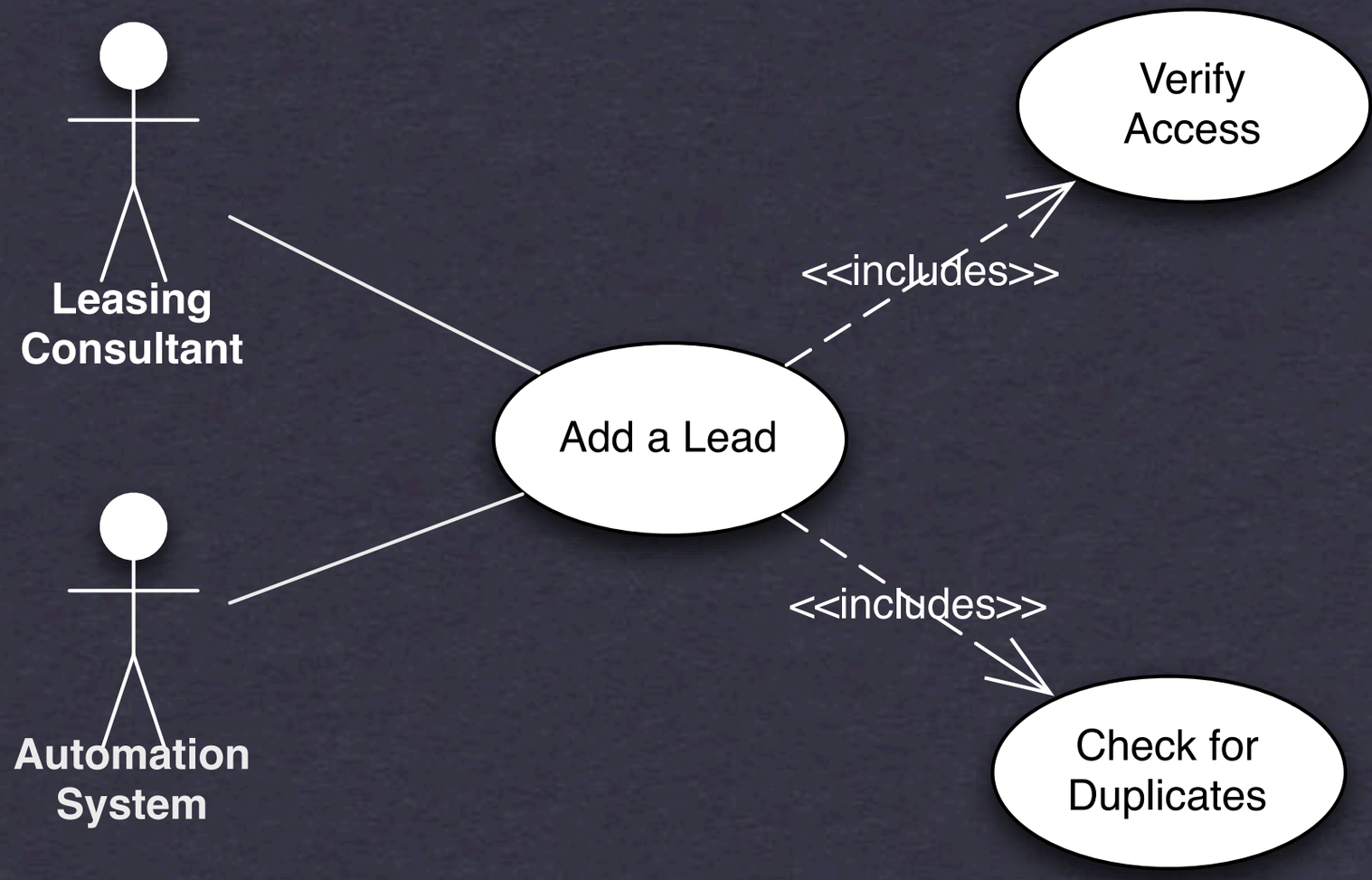
## And #2 sounds complicated. Maybe that should be it's own use case?

2. The system shall not permit any request to add a lead that is already in the database.
   - A lead is only unique if no other leads exist with the same first name, last name, and either e-mail address or phone number.

# Use Cases: Main Flow

1. The user asks the system to add a lead
include::Verify Access
2. The user submits the lead's detailed information
3. The lead is tested for uniqueness within the current community
include::Check for Duplicates
4. The lead is added to the database

# Use Cases: Update the Diagram



# Use Cases: Extensions

Branching actions show special cases that are not part of the main flow.

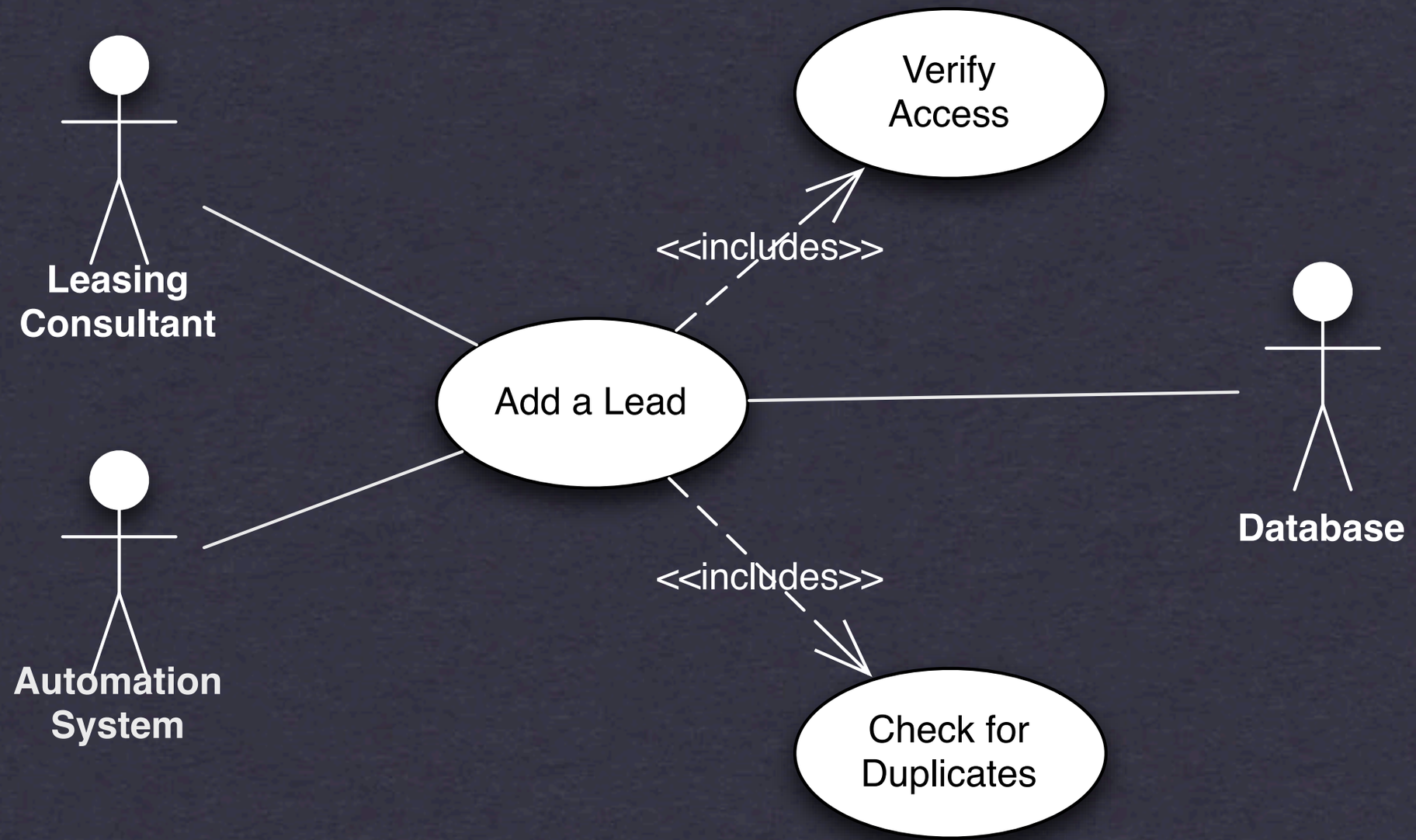
1.1 The user is not allowed to add a lead (per Verify Access)
1.2 The request to add a lead is rejected

3.1 The lead is not unique (per Check for Duplicates)
3.2 The request to add a lead is rejected

# Use Cases: Secondary Actors

1. The system shall allow leasing consultants and automation systems the ability to add a lead to the **DATABASE**
2. The system shall not permit any request to add a lead that is already in the **DATABASE**.
   - A lead is only unique if no other leads exist with the same first name, last name, and either e-mail address or phone number.
3. Only leasing consultants and qualified automation systems may add leads to the **DATABASE**.

# Use Cases: Update the Diagram



# Use Cases: Secondary Actors

Note: The database is usually part of "The System" by implication. I included it to show you:

- Secondary actors
- How to model an external system as an actor

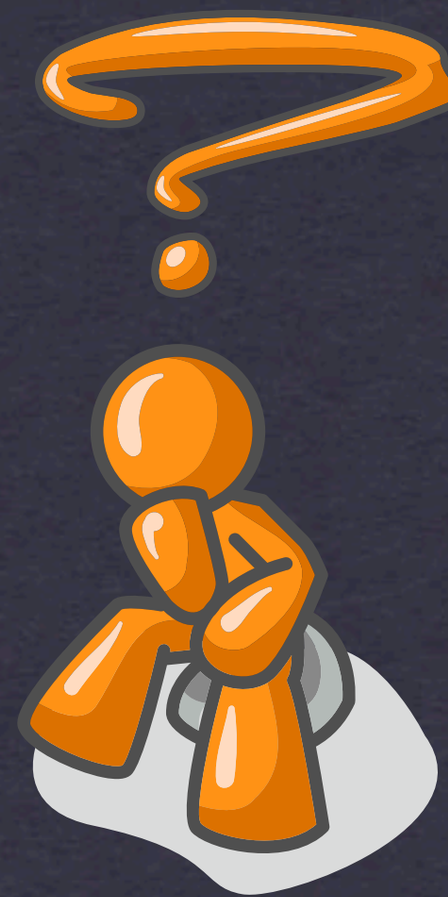
I also included it to confuse you.

# Use Cases

For the completed Use Case document, see the "Add a Lead" sample use case hand-out.

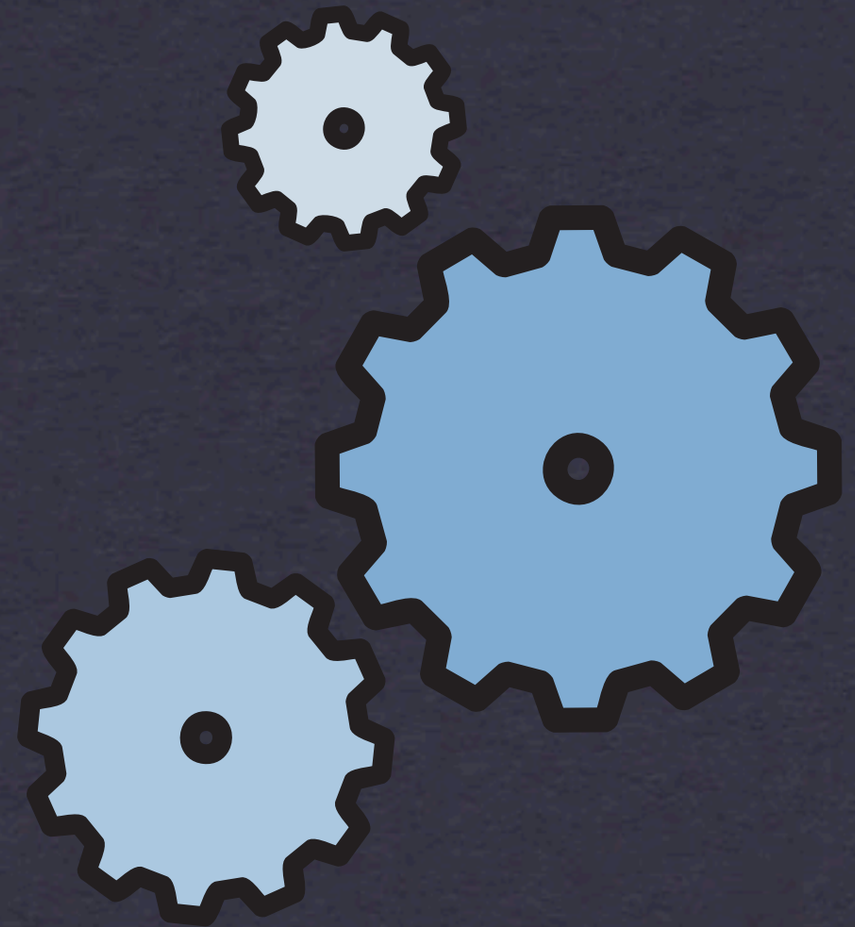
# Questions?

(OR ANSWERS?)

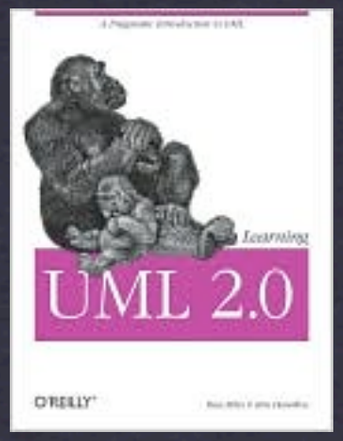


# Practical Exercise

1. Find actors & use cases
2. Diagram use cases
3. Choose one use case and detail the "main flow" section



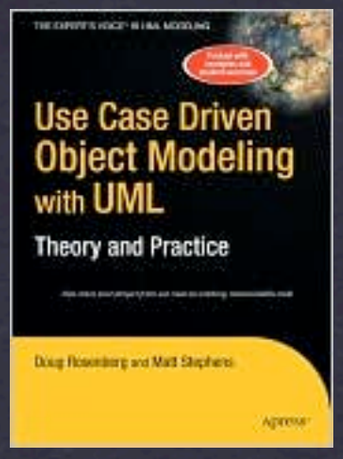
# Sources & Props




## Learning UML 2.0

Authors: Russ Miles, Kim Hamilton, Dawn Griffiths

Publisher: O'Reilly




## Use Case Driven Object Modeling with UML : Theory and Practice

Authors: Doug Rosenberg, Matt Stephens

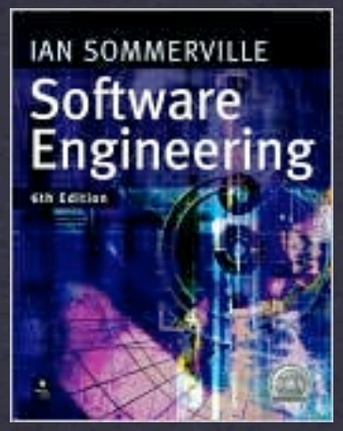
Publisher: Apress


## Managing Humans

Author: Michael Lopp

Publisher: South-Western

# Sources & Props




## Software Engineering

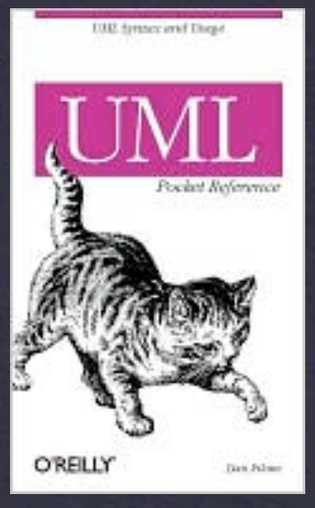
Author: Ian Sommerville

Publisher: O'Reilly


## UML in a Nutshell

Author: Sinan Si Alhir

Publisher: O'Reilly




## UML Pocket Reference

Author: Dan Pilone

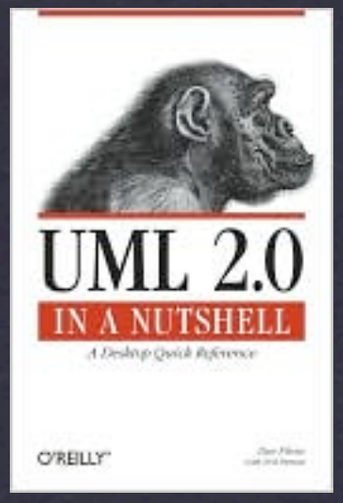
Publisher: O'Reilly

# Sources & Props: Newer Stuff


## Software Engineering

Author: Ian Sommerville

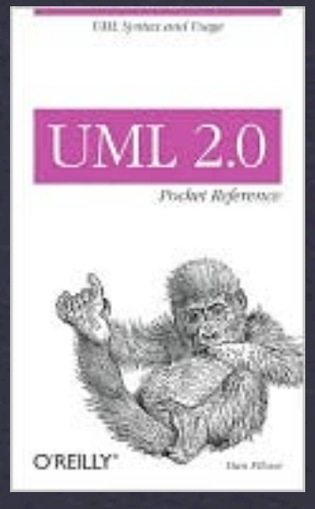
Publisher: O'Reilly




## UML 2.0 in a Nutshell

Author: Dan Pilone, Neil Pitman

Publisher: O'Reilly




## UML 2.0 Pocket Reference

Author: Dan Pilone

Publisher: O'Reilly

# EOF

exit 0;