Classes: Concepts, Context, & Identification

Daniel M. Berry

Introduction -1

All your acquaintance with classes prior to this course has been as a device to implement information hiding and object orientation.

You have looked at them as documentation of code written in C++, Java, and possibly other languages.

Introduction -2

You may have even looked at class diagrams as an expression of the architecture of a program either to be built or that has been built out of classes.

You may have looked at class diagrams as a notation in which to play with the architecture of a program in order to arrive at the best architecture.

Introduction -3

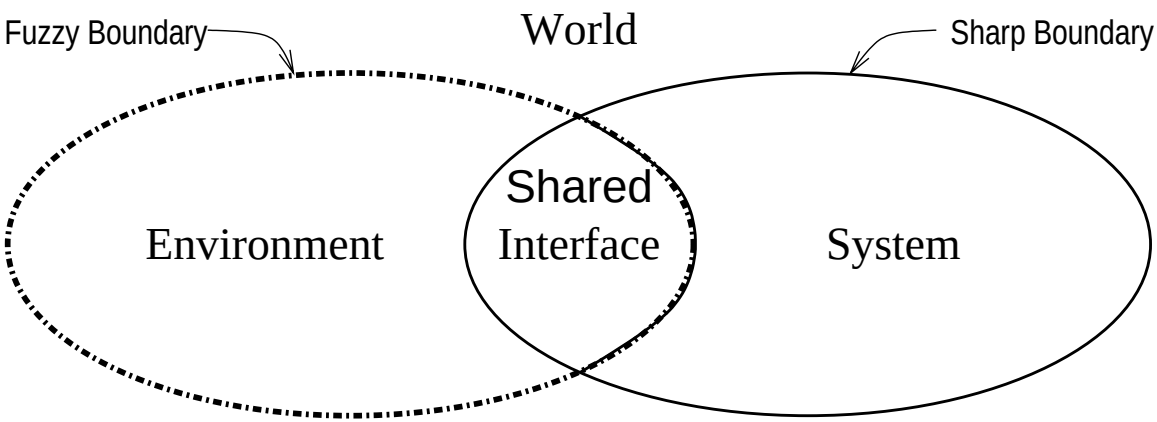
However, the question remains, “How are class diagrams, and, indeed how are all of UML, used in requirements engineering to help arrive at a specification of requirements?”

This lecture tries to answer this question by considering some examples of deriving a class diagram from a problem description.

Example 1: Turnstile

The city of Waterloo has decided to raise funds by instituting users fees for public parks. We need to implement a complete system of money collection, security, etc.

Dividing the World



The Environment is the part of the World that is affected by the System.

Turnstile Requirements

***Informal requirements*:Collect $1 fee from each human park user on entry to park (no fee to leave).**

- **Ensure that no one may enter park without paying.**
- **Ensure that anyone who has paid may enter park.**

Possible Solutions

***Solution #1*: Employ human fee collectors. Enforce security by instituting the Waterloo Park Militia, armed guards who make certain no one uses a park without paying a user fee.**

***Solution #2*: Use chain link fences for security, use turnstiles with automated coin collection. After some research, we find appropriate turnstile hardware, but it's brand new technology so we must create the embedded software system....**

The Park World -1

There is a barrier to enter a park. A person inserts a coin, the barrier unlocks, allowing the person to push the barrier and enter the park.

The Park World -2a

environment	**visitorS**
	does insert of coinS to CoinSlot
	detects unlocking of Barrier
	does push of Barrier
	coinS
	fence
	p~~ersonS~~

The Park World -2b

shared phenomena	**coinSlot**	**receives insert(denom)**
		receives inserted?()
		does addCoin(denom) to TurnstileSystem
	Barrier	**receives push()**
		receives unlock()
		receives inRotation?()
		receives lock() or locked?()
		does addVisitor() to TurnstileSystem

The Park World -2c

software system	**TurnstileSystem**

does inserted?() to CoinSlot
does inRotation?() to Barrier
does unlock() to Barrier
does lock() or locked?() to Barrier
receives addCoin(denom)
receives addVisitor()

The Park World -3

The software system and the environment interact via the shared phenomena, which may be both sensed and controlled by both the software system and the environment:

The Park World -4

- **The environment controls insertions of coins into coin slots.**
- **The software system senses coin insertion and then reacts by unlocking the barrier.**

The Park World -5

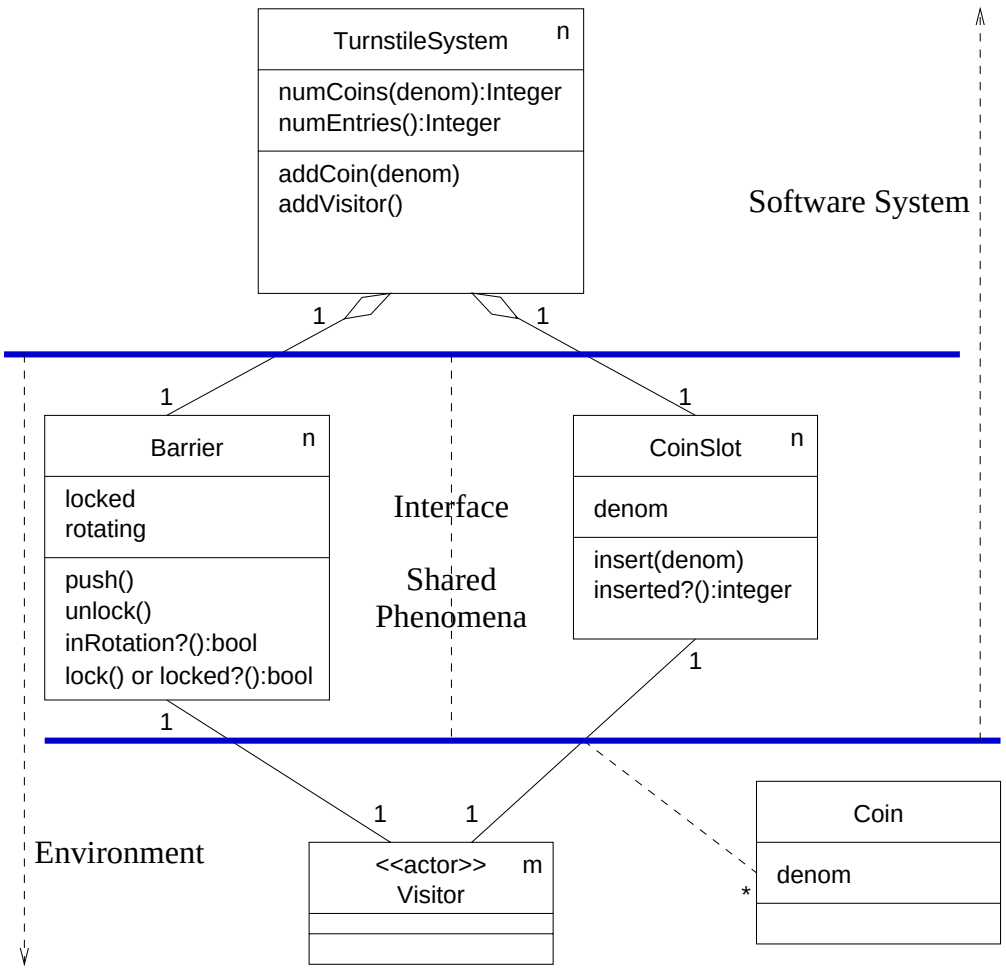
- **The environment senses that the barrier is unlocked and a person can rotate the barrier (to enter the park).**
- **The software system senses barrier rotation and eventually either locks the barrier, senses that the barrier has locked, or assumes that the barrier locks itself after rotation.**

The Park World -6

What locks and unlocks and what is pushed and is rotated is the barrier, and this barrier together with the coin slot form the turnstile. The barrier and the coin slot together are the phenomena shared between the turnstile software system and the visitor in the environment.

Accordingly, we may construct a class diagram.

Turnstile Class Diagram



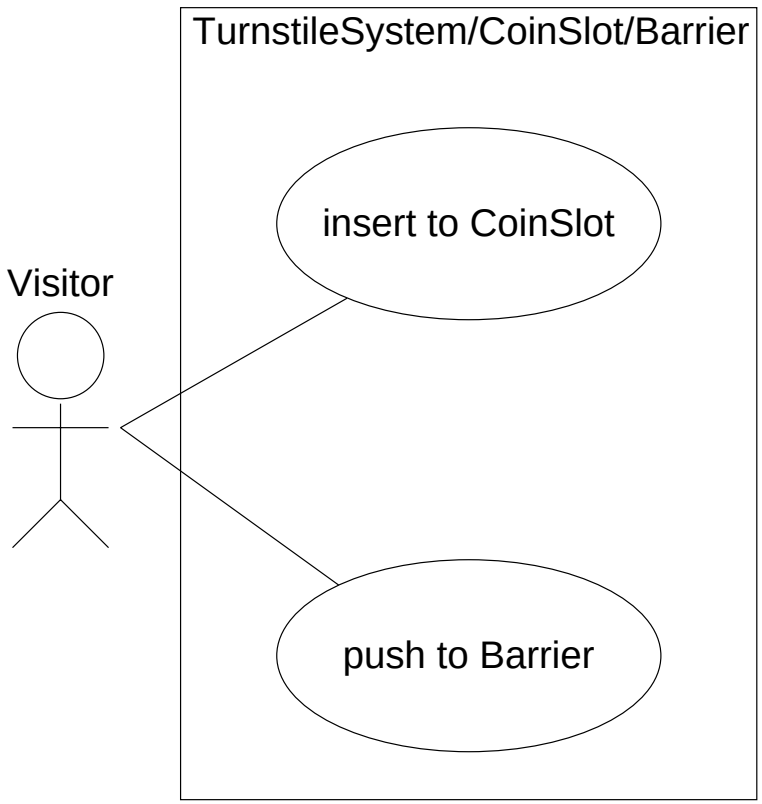
The Park World -7

We now identify the use cases as the operations in the Interface classes that are both

- **accessible to and**
- ***done* by**

the actors.

Turnstile Use Cases



Role of the REng -1

Please understand the role of the requirements engineer (REng).

E is often called in to work with problems that are totally new to em. The problem description uses vocabulary unfamiliar to em.

Role of the REng -2

It is the job of the requirements engineer to begin to form a model of the described problem so that E can use the model to identify what E does not understand and to ask questions of the client.

Role of the REng -3

Very often this initial model is formed in ignorance. The requirements engineer identifies the nouns, verbs, adjectives, and adverbs of the problem description and uses them as the names of classes, operations, attributes, and nonfunctional requirements in the model formed in ignorance.

Encoding Ignorance

The thingamajig snarkles the doodad.

The thingamajig snarkles the doodad.

Nouns:

Verbs:

Questions:

Exceptions:

The thingamajig snarkles the doodad.

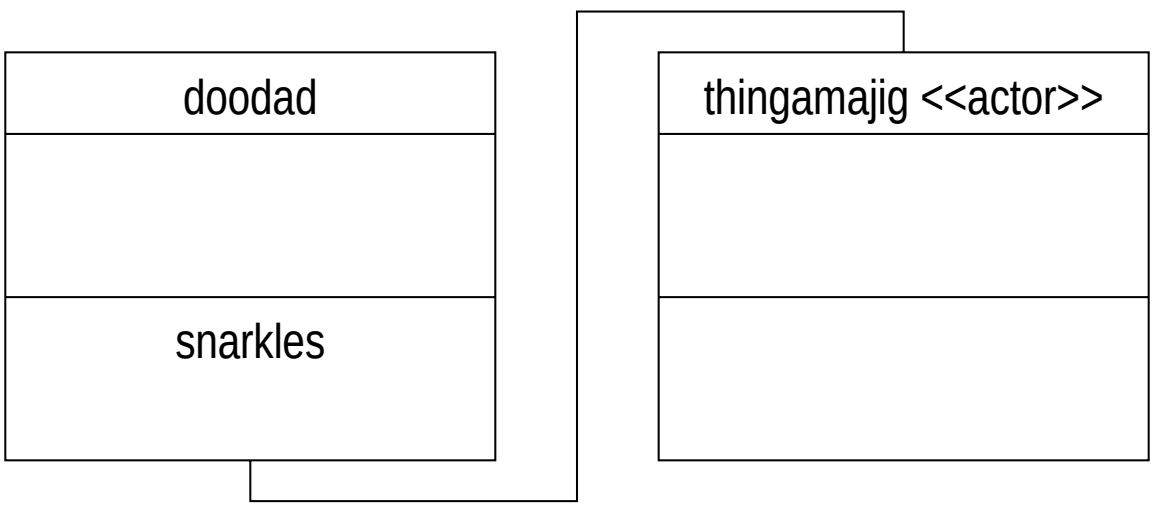
Nouns: thingamajig
doodad

Verbs: snarkle

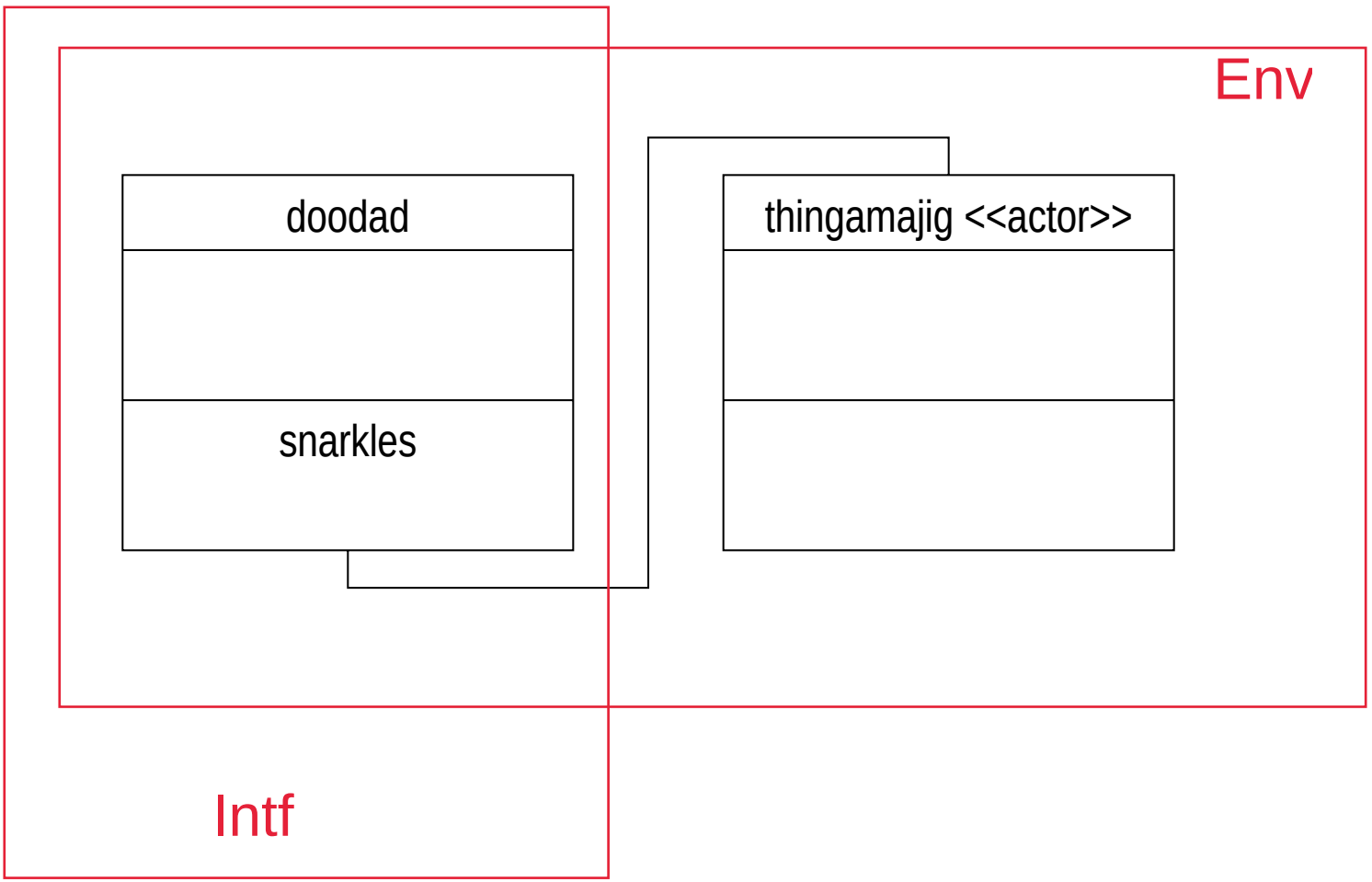
Questions: What is a thingamajig?
What is a doodad?
What is snarkling?

Exceptions: Can snarkling ever fail in any reason that we can check for?

The thingamajig snarkles the doodad.

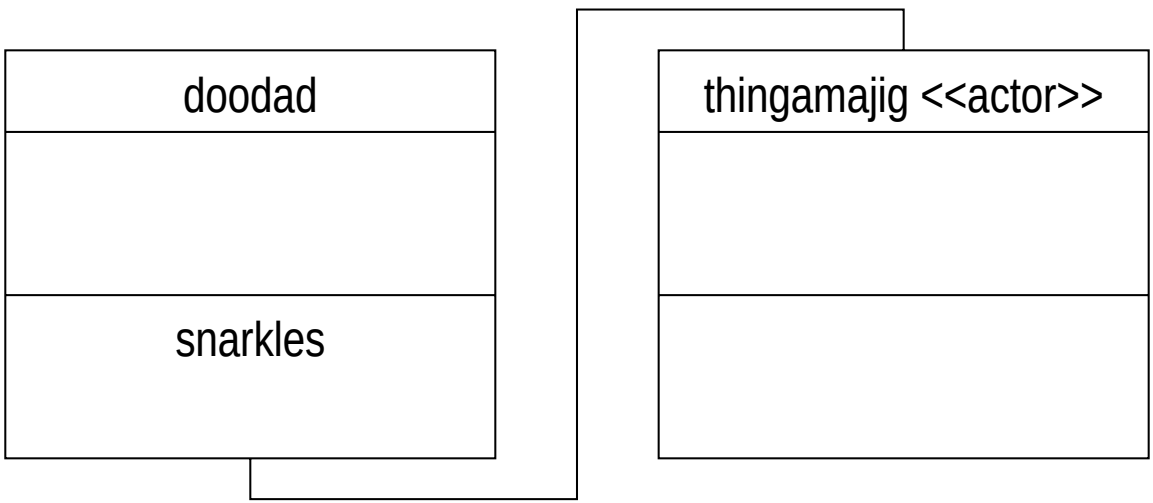


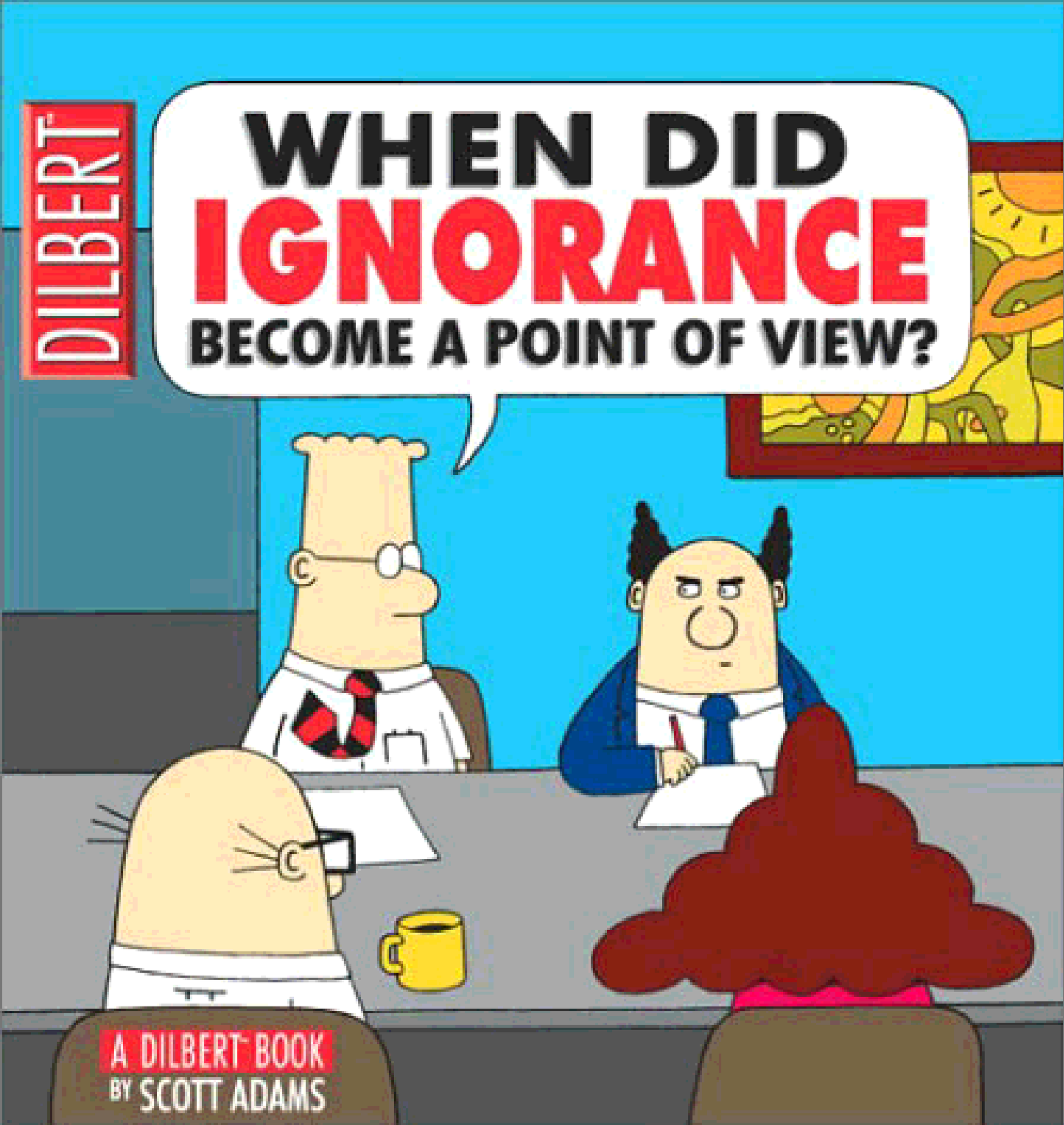
The thingamajig snarkles the doodad.



Encoding Ignorance

The thingamajig snarkles the doodad.




DILBERT
WHEN DID
IGNORANCE
BECOME A POINT OF VIEW?
A DILBERT BOOK
BY SCOTT ADAMS

Role of the REng -4

That is, the requirements engineer is skilled enough in modeling that E can take the words of the problem description and put them in the right places in the model, so as to end up with an intelligible model, even though E does not understand the words.

Real World Example

Now we consider how to build a necessarily incomplete model from the poor information you get from clients.

Example 2: Classroom Podium

We now build a model of the podium system in the typical classroom at the University of Waterloo.

First, the nouns of the problem and the multiplicities, that can become classes in the model.

Classroom Podium -1

- **1 Nexus Client Computer**
- **1 Podium Control Panel**
- **1 Input Selector**
- **1 Output Selector**
- *** Jack**
- **1 Loud Speaker**
- **1 Podium Computer**
  - **1 Keyboard**
  - **1 Mouse**
  - **1 Screen**
  - **1 Trackpad**

Classroom Podium -1

1 Internet
*** Departmental Nexus Server**

1 Laptop
1 Sound Source
1 Video Source

Classroom Podium -2

* **Data Projector**
* **Video Screen**
* **Projection Screen**
* **Projection Screen Motor**
  - **1 On Button**
  - **1 Off Button**
  - **1 Up Button**
  - **1 Down Button**

Classroom Podium -3

1 Lecturer
*** Listener**

Build the Model

Build the model on the board with the class's help!

Build the Model

As often happens during the building of any model, a new noun was discovered.

To handle the fact that all Video Screens and Data Projectors were to receive and display the same image, a

1 Video Controller

is needed to receive the image that is to be shared with all Video Screens and Data Projectors.