

An Introduction to
**Object-Oriented
Analysis**

Objects in plain English.

Chapter 4: Data-Oriented Models.

Based on the book by
David W. Brown

John Wiley & Sons, ISBN 0471110280

1

Copyright

Copyright 1997 Flying Kiwi Productions
All rights reserved.

This slide presentation is based on "An Introduction to Object-Oriented Analysis; Objects in Plain English," by David W. Brown, Wiley, ISBN 0471110280, "The Book."

Permission is hereby granted to copy, modify or excerpt all or any part of this slide presentation, provided it is solely for use with courses, seminars or other presentations or productions where a copy of The Book is purchased by or for each and every participant or recipient.

An instructor guide is available from the publisher for such presentations.

2

Chapter 4 Data-Oriented Models

- 4.1. Data Models: The ERD
- 4.2. Paradigm Shift
- 4.3. Entity-Relationship Modeling
- 4.4. Object-Oriented Models
- 4.5. An Introduction to Objects
- 4.6. Object-Oriented vs.... Object-Based
- 4.7. Conceptual, Logical & Physical Models
- 4.8. Models as Communication Tools

3

4.1. Data Models: The ERD

Mid-1970s: Databases were just coming into use

Research to find ways to model data
To give template for database design
Many notations were tried
Chen (1976) proposed the

***Entity-Relationship Diagram
(ERD)***

4

The ERD was the first systems analysis tool to focus on

DATA

and

***How it is Linked
and Organized.***

It was from this basis that objects then evolved.

5

The ERD was the first systems analysis tool to focus on

DATA

and

***How it is Linked and
Organized.***

It was from this basis that objects then evolved.

6

4.1. Data Models: The ERD

Chen made two contributions:

The Entity-Relationship Principle, and
The Entity-Relationship Notation.

These further issues were also
investigated by Chen and others:

E-R models for database design
The **significance** of the data of an enterprise
Data as a **corporate asset**.
The **stability** of the data of an enterprise

7

4.2. Paradigm Shift

A **paradigm shift** is a major change in
the way we think about something.

True paradigm shifts are rare.
(Brooks, 1976)

8

4.2. Paradigm Shift

Physics: Newton ~~E~~instein.
Chemistry: Alchemy ~~A~~atoms.
Astronomy: "Bible" ~~G~~Galileo.
"Bible" = Geocentric, as interpreted by Inquisition
Biology: Creation ~~D~~arwin.

**Each of these was a
major change in how
we viewed our world.**

9

4.2. Paradigm Shift

A **paradigm** is

a *pattern* or mode of *thinking and believing*
that serves to organize
the way we approach
knowledge, learning and understanding.

A **paradigm shift** occurs

when we adopt
a *radically different way* of organizing
all or part of our worldview.

10

4.2. Paradigm Shift

The change to data-oriented analysis
methods i.e., to ERDs was a
true paradigm shift.

While Objects are an evolution from
Entities, the change is so dramatic and
far-reaching. . .

that it also qualifies as a true
paradigm shift.

11

4.3. Entity-Relationship Modeling

**As we saw earlier, Chen
made two main
contributions, the
ERD Principle and
ERD Notation, as well as
investigating several
other issues.**

12

The following six points all relate to both E-R and Object modeling. Now let's examine them in this order:

- ❶ The Entity-Relationship *Principle*.
- ❷ E-R models for database design.
- ❸ The stability of data.
- ❹ The significance of data.
- ❺ Data as a corporate asset.
- ❻ Entity-Relationship Notation.

13

❶ The Entity-Relationship *Principle*.

To manage a sales operation, we need to keep data about :

Sales
Customers
Products
Sales
Sales clerks

14

❶ The Entity-Relationship *Principle*.

To run a phone company, we would need to keep data about:

Phone
Customers
Numbers
Lines
Calls
Services

15

❶ The Entity-Relationship *Principle*.

To run a booking agency we must record information about:

Booking
Venues
Artists
Agents
Concerts
Performances
Customers
Seats

16

❶ The Entity-Relationship *Principle*.

To run a booking agency we must record information about:

Booking
Venues
Artists
Agents
Concerts
Performances
Customers
Seats

17

❶ The Entity-Relationship *Principle*.

A date is data that describes a Concert
A date can describe many things
The key word here is

Things

Notice there is no Date in this list.

Why Not?

18

① The Entity-Relationship *Principle*.

Each of these lists is a list of *things*

Sales	Phone	Booking
Customers	Customers	Venues
Products	Numbers	Artists
Sales clerks	Lines	Agents
	Calls	Concerts
	Services	Performances
		Customers
		Seats

① The Entity-Relationship *Principle*.

The E-R principle focuses on the things we need to keep data *about*.

Earlier methods emphasized what the users need to know to do their job. Here we emphasize what things the users need to know *about*. The word Entity means a thing. *Later* we worry about what items they need to know *about* each entity.

① The Entity-Relationship *Principle*.

Webster: An entity is “something that has separate and distinct existence”

You can see it is something we need to know about (i.e., keep data about) to do our job

So we define it. . .

① The Entity-Relationship *Principle*.

A Data Entity is some *thing* that has separate and distinct existence *in the world of the users* and is of interest to the users in that they need to keep data *about* it in order to do their job.

① The Entity-Relationship *Principle*.

So these are the **entity lists** for these businesses:

Sales	Phone	Booking
Customers	Customers	Venues
Products	Numbers	Artists
Sales clerks	Lines	Agents
	Calls	Concerts
	Services	Performances
		Customers
		Seats

① The Entity-Relationship *Principle*.

Grammatically, the term Entity refers to a single specific *Customer* or *Product* or *Sale* or *Call* or *Artist*.

What we need to talk about mostly is the class of all Customers, or in other words the **type** of thing called a Customer.

① The Entity-Relationship *Principle*.

So we define:

An **Entity Type** is a class or category expressing the common properties that allow a number of entities to be treated similarly.

25

① The Entity-Relationship *Principle*.

Entity types are always singular.

So our entity lists look like this:

Sales	Phone	Booking
Customer	Customer	Venue
Product	Number	Artist
Sales clerk	Line	Agent
	Call	Concert
	Service	Performance
		Customer
		Seat

All Singular

26

① The Entity-Relationship *Principle*.

An individual *Customer* or *Product* or *Sale* or *Call* or *Artist* is then an **Occurrence** of the Entity Type.

27

① The Entity-Relationship *Principle*.

I am an occurrence of the entity type “Employee”

You are an occurrence of the entity type “Student”

We each live in an occurrence of the entity type “Dwelling”

The last time you caught an occurrence of the entity type “Bus,” it followed an occurrence of the entity type “Route” while executing an occurrence of the entity type “Trip.”

28

① The Entity-Relationship *Principle*.

Attributes are the data elements carried by an entity that describe it and record its state.

Attributes are the things we need to know about an Entity.

29

① The Entity-Relationship *Principle*.

So how did the Country occurrence *Canada* happen to get the string value “Canada” as the value for its *Name* attribute?

30

❶ The Entity-Relationship *Principle*.

On the day of Canada's confederation, the Founding Fathers placed 26 slips of paper in a hat with the 26 letters of the alphabet on them. "Just draw 3 letters," they said, "We want a short, crisp name for our beautiful new country." So the one with the hat did as directed, and announced:

"C, *eh?*"

"N, *eh?*"

"D, *eh?*"

31

❶ The Entity-Relationship Principle.
Relationships:

So we have:

Hundreds of *Customers*

Thousands of *Products*

Dozens of *Sales Clerks*

Do we have a business?

NOT YET!

32

❶ The Entity-Relationship Principle.
Relationships:

Not until:

A Customer buys a Product

A Sales Clerk sells a Product

A Sales Clerk sells to a Customer

These are the *interactions*
that occur between entities.

Note each involves a verb.

33

❶ The Entity-Relationship Principle.
Relationships:

We define:

A **Relationship** is the interaction
of two entities and is represented
by a verb.

34

❶ The Entity-Relationship Principle.
Relationships:

The purpose is to provide access
paths to the data.

When a sale occurs, we will link:

A Customer

To a Product

To a Sales Clerk.

Using a *verb* to describe each link.

35

❶ The Entity-Relationship Principle.
Relationships:

Note we have not made any mention of
inputs or outputs,

And we have touched on business
processes, but

only as they concern the data.

36

❶ The Entity-Relationship Principle.
Relationships:

In this way we are able to diagram the structure of our user's business data,

independent of any way that the data may be used, either now or in the future.

37

❶ The Entity-Relationship Principle
SUMMARY

An Entity is a thing the users need to know (i.e. record) something about.

An Entity Type is a group or class of entities that are all the same kind of thing.

An Occurrence of an Entity Type is a specific individual thing of the kind the Entity Type describes.

Attributes are the things we need to know about an Entity.

A Relationship is the interaction of two Entities and is represented by a verb.

These interactions among the Entities (i.e., these *Relationships*) show us the pathways we need to follow through the database to access the data.

38

❷ E-R models for database design.

There is a very significant reason why Entities, Attributes and Relationships are so fundamentally important for systems development.

39

❷ E-R models for database design.

To start with, the search for data modeling methods was for *database design*.

Many notations were developed and some actually tried out.

Chen's ERD turned out best and took over.

It also caught on as a tool for systems analysis, as well as for data analysis.

40

❷ E-R models for database design.

For database design:

Each ***Entity*** becomes a file or table.

Each ***Attribute*** becomes a field (i.e., a column)

Each ***Relationship*** becomes an access pathway (i.e., a foreign key)

41

❸ The **Stability** of Data.

This year, our favorite Transit Company is concerned with entity types:

Bus, Driver, Route, Trip, Bus Stop and Passenger.

Ten years from now, there will be many changes in how the company is run and in how the people do their jobs.

But they will *still* have to deal with

Buses, Drivers, Routes, Trips, Bus Stops and Passengers!

Not only that, but ten years ago they worked with Buses, Drivers, Routes, Trips, Bus Stops and Passengers!

42

③ The Stability of Data.

And a college registration system must deal with entity types

Student, Course, Subject Area, Room,
Instructor and Enrolment

now,

in the future

and in the past,

“As it was in the beginning, is now, and
forever shall be, data without end. . .”

43

③ The Stability of Data.

This tells us that

**Data Entities Are Stable
Over Time.**

44

③ The Stability of Data.

When we organize our data around the entities from the real world of the user,

Those entities are stable.

45

③ The Stability of Data.

However, entities are neither static nor stagnant;

There may be an occasional new entity,

There will often be new attributes,

But as long as we stay in the

same business

there will be very

few new entities.

46

③ The Stability of Data.

**This stability contributes
greatly to reducing the costs
and delays in system
maintenance.**

47

③ The Stability of Data.

However, ***beware new business!***

Mergers, acquisitions, takeovers and diversifications can all put your company into

new businesses

with a host of

new entity types.

48

④ The **Significance** of Data.

The Data modeler's Creed:

Data
is the
Center of the Universe

49

④ The **Significance** of Data.

Out of the universe of all data
Our project is concerned only with
the subset
that belongs in the business area we
are studying.
The ERD is the tool we use to study,
document and understand this data.

50

④ The **Significance** of Data.

The structure of this data drives the
design of the databases that must store it.
The databases in turn support the design
of the programs,
That support the users' operation by
supplying exactly the data need, when
they need it, in a form they can use,
And by accepting the data they generate
as they do their jobs.

51

④ The **Significance** of Data.

All of this must be done in a transparent,
user-friendly way,

So that the software enhances the *users'*
performance.

**This, after all, is the only
reason why the software
should exist at all!**

52

⑤ Data as a **Corporate Asset.**

**Since the users both need and
generate data as they work, it
can be said that their
operation rests upon this
pool of data
that must exist for them to
function.**

53

⑤ Data as a **Corporate Asset.**

Since data is something they *must have*
in order to work,

It can rightly be regarded as a **Business
Asset or Resource**

Just like

Finance
Human Resources
Plant and Equipment
Vehicles
Inventories
Etc.

54

⑥ Data as a Corporate Asset.

Like any corporate asset, there are some basic functions that must be done as part of managing the asset:

- Acquisition
- Organizing, storing and safekeeping
- Deployment, on time, to the people who need it
- Disposal when no longer needed

55

⑥ Data as a Corporate Asset.

This has led to two new functions in the information industry,

- Data Administration (DA) and
- Data Resource Management (DRM).

DA is a middle-management function concerned with finding and documenting every data element the company uses.

DRM is a high-level management function, involving long-term strategic planning. DRM reports to the CIO (Chief Information Officer; in a well-organized company the CIO is a vice-president).

56

⑥ Data as a Corporate Asset.

The focus of DA and DRM is to manage data as we would manage any other business resource.

Up to now, the ERD has been the tool used to understand, document and administer data.

As of now, the Object Model is taking over this function.

57

⑥ The Entity-Relationship Notation.

There are many E-R notations. All of them work, any will do the job.

Use whichever one you or your boss or your professor or your client prefers.

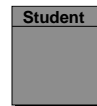
In this course we shall use a simplified version of Chen's original notation, for our Entities and our Objects.

58

⑥ The Entity-Relationship Notation. Entities

Represent an Entity with a square or rectangular box, divided by a line near the top.

Above the line place the name, singular.



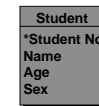
59

⑥ The Entity-Relationship Notation. Attributes

If there are not too many, Attributes may go in the bottom part of the box.

Primary key first, marked with an asterisk.

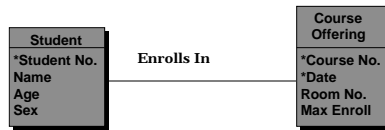
If not enough room, show only the key, list the others in the accompanying write-up.



60

6 The Entity-Relationship Notation.
Relationships

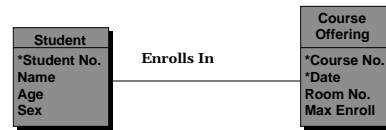
Draw a line joining two entity boxes to show a relationship exists
Write the verb against the line.



61

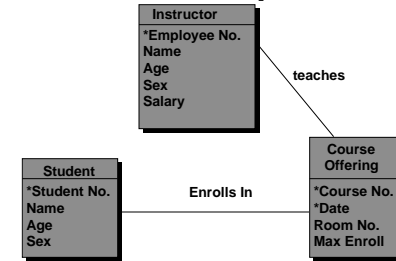
6 The Entity-Relationship Notation.
Relationships

Place the verb above or below the line so that it makes a sentence when you read it clockwise around the line:
"Student *enrolls in* course"



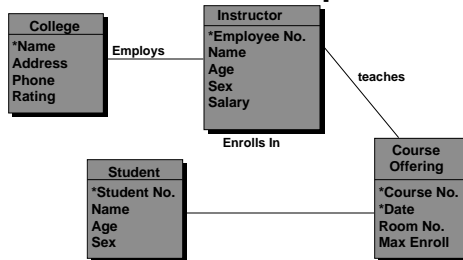
62

6 The Entity-Relationship Notation.
Relationships



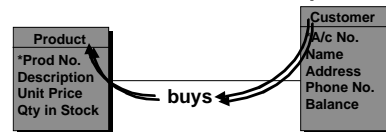
63

6 The Entity-Relationship Notation.
Relationships



64

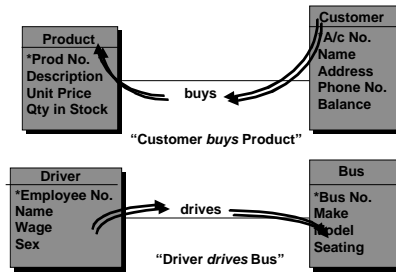
6 The Entity-Relationship Notation.
More Examples of Relationships



Read the sentence clockwise around the line:
"Customer *buys* Product."

65

6 The Entity-Relationship Notation.
More Examples of Relationships



Read the sentence clockwise around the line.

66

⑥ The Entity-Relationship Notation.
Cardinality

Q. When a driver *drives* a bus, **how many** does she drive?

A. One, usually. But over time?
Why, a whole bunch!

Q. When a Customer *buys* a Product, how many does he buy?

A. One per transaction, perhaps, but over time we hope he'll buy a great many.

67

⑥ The Entity-Relationship Notation.
Cardinality

The critical thing we need to know is

Is it **One**, or
Is it **Many**?

This is the *Cardinality* of the relationship.

68

⑥ The Entity-Relationship Notation.
Cardinality

Webster defines **Cardinality** as “The number of members in a set.”

For us, **Cardinality** is the *number of occurrences* of each entity type that can be involved in a relationship.

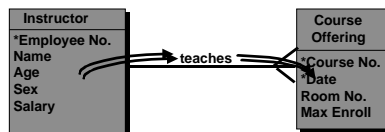
69

⑥ The Entity-Relationship Notation.
Cardinality

We diagram this by adding a symbol to the relationship line whenever it should show “many.”

Read this one as

“Instructor *teaches* **many** course offerings”

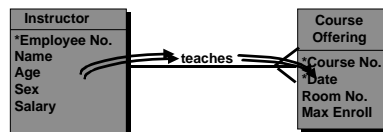


70

⑥ The Entity-Relationship Notation.
Cardinality

We refer to this as a

“One-to-Many” Relationship,
or 1:M



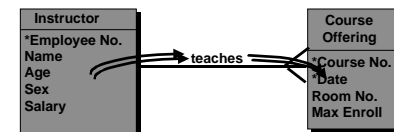
71

⑥ The Entity-Relationship Notation.
Cardinality

Read the sentence from left to right above the line

i.e., *clockwise* around the line.

Read the “crow’s foot” (↯) symbol as the word “many.”

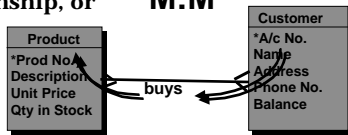


72

⑥ The Entity-Relationship Notation.
Cardinality

And, since we wish many Customers to buy many Products,

This one is a **Many-to-Many** relationship, or **M:M**



⑥ The Entity-Relationship Notation.
Cardinality

There are numerous other symbols also used for the “many” end of a relationship, including single and double arrows, etc.

The “crow’s foot” is easy to draw by hand.

Other notations have their own ways to show “many” and what direction to read it.

⑥ The Entity-Relationship Notation.
Summary

There are many ERD notations. We are using a simplified Chen notation.

An entity is a square box, with a singular name above a horizontal line.

The primary key attribute is below the line, with an asterisk (*)

Other attributes are listed below or on another page.

ctd. . .

⑥ The Entity-Relationship Notation.
Summary

A relationship is a line joining two entities. Write the verb beside the line.

Reading clockwise around the line,

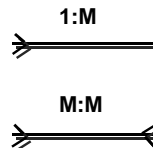
entity-verb-entity

should make a simple sentence.

ctd. . .

⑥ The Entity-Relationship Notation.
Summary

Cardinality is *one* (straight line) or *many* (crow’s foot) at each end of the line.



There is one major side benefit to both ERDs and Object modeling.

Both serve these four important functions:

Allow us to understand the users' business quickly and accurately

Document the information needs of the users' business

Give the DA the tool to understand, document and ultimately *control* the data

Give a *template* or *pattern* for database design.

And they are **more effective**, and give us **better insights and understanding**, than any of the earlier methods.

79

4.4. Object-Oriented Models

Object-Oriented Programming (OOP) is now the state of the art.

Pool of graphics and engineering programmers

Pool of C programmers ←C++

Object-Oriented Analysis and Design (OOA&D) are still very much the leading edge.

Object-Oriented Databases (OODBMS) are powerful and rapidly catching up.

80

4.4. Object-Oriented Models

Object-Oriented Development Environments so far are mostly *Object-Oriented Front Ends* that link to a relational database.

They are truly O-O only in the GUI.

They are evolving toward true O-O,

And so are the relational databases!

ORACLE 8 is O-O!

81

4.5. An Introduction to Objects

Like an entity, an object is described by a noun,

And it is “Some *thing* in the user's world that has a separate and distinct existence, and is of interest in that they need to keep data *about* it.”

82

4.5. An Introduction to Objects

But an object is more than an entity.

In addition to the data, the object carries program code,

That *uses or changes that data*.

83

Now here is the
fundamental principle
of **Objects:**

84

4.5. An Introduction to Objects

The **only** way that a program can read or change the data carried by an object, or access the data in any way at all is by invoking one of the defined pieces of program code that the object carries within itself.

85

4.5. An Introduction to Objects

I'll say that again. . .

86

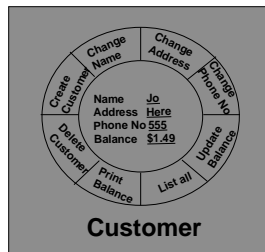
4.5. An Introduction to Objects

The **only** way that a program can read or change the data carried by an object, or access the data in any way at all is by invoking one of the defined pieces of program code that the object carries within itself.

87

4.5. An Introduction to Objects

This can be illustrated with a Taylor Donut Diagram:



88

4.5. An Introduction to Objects

The “behaviors” around the outside are subroutines or functions.

They physically are pieces of compiled program code.

In the real world they correspond to things this object can do or have done to it.



4.5. An Introduction to Objects

These are called variously:

- Functions
- Behaviors
- Operations
- Services
- Responsibilities
- Methods (esp. at the OOP level)

The data in the center is surrounded and protected by them.



4.5. An Introduction to Objects

Thus, as Coad and Yourdon put it,
An object encapsulates both “the data and the exclusive functions on that data.”

By “exclusive” we mean that no other code can access or manipulate this data.



4.5. An Introduction to Objects

So our tasks as Analysts is to investigate the users' business to find:

- The objects and classes
- Their attributes and relationships
- The operations that can be performed on, to, with or by these objects.

Then we check how each operation will use or affect the data attributes.

In the Design phase we produce specs for each operation.

In Coding we use these specs to write them as methods in whatever OOPL we chose.

4.5. An Introduction to Objects

OOPLs and OODBMSs actually store the program code for the methods in the database along with the data.

We write small pieces of code, each to do a single operation.

A Customer object will need:

An operation called UpdateAddress

A piece of code to carry it out

This code is then attached to the object in the database.



92

4.5. An Introduction to Objects

When an object is first created, it appears as a collection of fields (attributes) *in RAM*.

Some are deleted after a short lifetime.

Some disappear when the machine is turned off.

These are all **Transient Objects**, which **DO NOT** survive beyond the current session.



4.5. An Introduction to Objects

Some objects we need to keep around longer

- Customers
- Employees
- Products, etc.

i.e., all the things we normally would expect to put in a database.

These we create as

Persistent Objects,

that **DO** survive beyond the current session.



4.5. An Introduction to Objects

Summary

Objects are entities (*things* that carry data) with behavior (operations) added.

These operations exclusively access the data it carries - no other code can touch it.

Operations are coded in an OOPL or OODBMS as *methods* (functions, subroutines).

Code for the methods is stored in the database along with the data for the object.

Any program that wants data from this object can get it only by calling one of the methods defined on this object.

96

4.5. An Introduction to Objects Summary

Transient Objects are created by OOPLs in the RAM of the computer and

DO NOT Survive

the current session.

Persistent Objects are stored by a database (OODBMS) and

DO Survive

beyond the current session.

97

4.6. Object-Oriented vs.... Object-Based

Some programming languages have objects but do not qualify as Object-Oriented.

e.g. ADA 85, Clipper

To be truly O-O, they must have two important features:

Inheritance, and

Polymorphism

(These are defined and fully explained in Chapter 8)

These two features account for a great deal of the power and benefits of O-O methods.

98

4.7. Conceptual, Logical & Physical Models

There are a variety of modeling paradigms available: DFDs, ERDs, Objects, etc.

Each can be done at three levels .

These are three different levels of abstraction:

Conceptual

Logical, and

Physical.

99

4.7. Conceptual, Logical & Physical Models

A physical model is one that shows in full detail:

Who will do each process

Physically where the data will be stored

What communications medium data will be transferred on

100

4.7. Conceptual, Logical & Physical Models

A ***physical model*** is the final design document showing how things will be written, done or built, and depicting all hardware and software platform details, including data storage and data transmission media.

101

4.7. Conceptual, Logical & Physical Models

At the **logical** level, we build a model that shows everything that must be included, and everything that the system must do, ***without*** specifying ***how***.

It makes no reference to hardware, software or media.

102

4.7. Conceptual, Logical & Physical Models

A **logical model** shows what a system must do or have, without regard for how it is to be done, built or represented.

103

4.7. Conceptual, Logical & Physical Models

A **conceptual model** is like a logical one, except that it shows the **users' concept** of their operation.

It will miss out a lot of behind-the-scenes activity and structure

Things that must be there, i.e., must exist or must happen, even though the users are mostly *unaware* of them.

These things are added when we later make a logical model.

It includes many-to-many relationships.

104

4.7. Conceptual, Logical & Physical Models

A **Conceptual Model** is a representation of the users' business in terms of **their conception** of how it operates.

For ERDs and Object models, this means it includes M:M relationships.

105

4.7. Conceptual, Logical & Physical Models

We begin by working with the users to produce a **conceptual model**.

Then we expand that into a **logical model** by adding all the features that were not apparent to the users.

Finally we make all our design decisions and document them on the **physical model**.

106

4.8. Models as Communication Tools

Users always accuse us "technoids" of speaking in a foreign tongue.

They complain that:

"Those computer people never listen to us,"

or "They never do what we were asking for"

or "They gave us way more than we needed"

or "They can never explain in English how to make it work."

107

4.8. Models as Communication Tools

The secret to solving all of these is **Communication**, but

"The biggest problem with communication is the *illusion* that it has taken place!"

108

4.8. Models as Communication Tools

To build an accurate model, it is essential that you are **user-driven** in your modeling. The difference between a *good* systems analyst and a *great* one is in **people skills**, especially **listening skills**.
“God gave us two ears and one mouth!”
Remember, we are using our object-oriented techniques to understand **their** business.

109

4.8. Models as Communication Tools

Once the users learn the notation, they quickly take to using the model to discuss and solve problems with the analysts.
All players on the team and sometimes outside it can make use of these models.

110

End of Chapter Four.

111