6. Initial Requirements and Use Caes

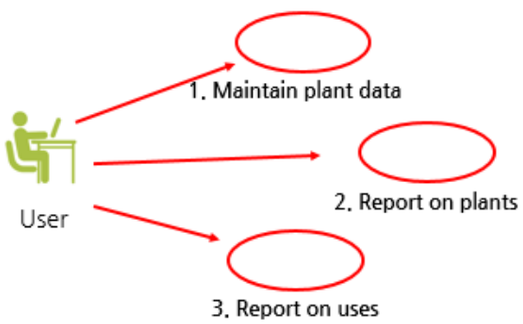
Fall 2018

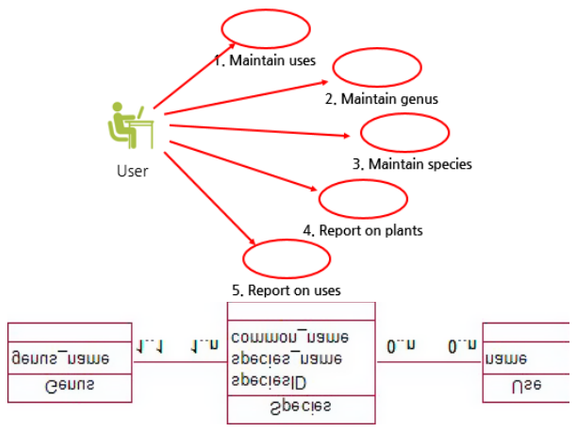
- We consider part of the first step from real-world problem to eventual real-world solution.
- We need to make sure we really understand the problem.
- There are two things we need to do:
  - Understand what tasks need to be carried out by all the people who will use the system, and
  - Figure out what data needs to be stored to support them.

analysis →

Real world | **Abstract world**

Problem





- We have to *fully* understand the real problem.

  The analyst needs to immerse himself in the problem domain so deeply that he begins to discover nuances that even those who live with air traffic control every day have not fully considered.

- Experts seldom need to think in an abstract way about the details.

- Answers such as “No, not really,” or “Hardly ever,” or “Umm, no, I don’t think so, umm, well maybe,” are a sign that a complication exists that needs to be understood before any design of a database should proceed further.

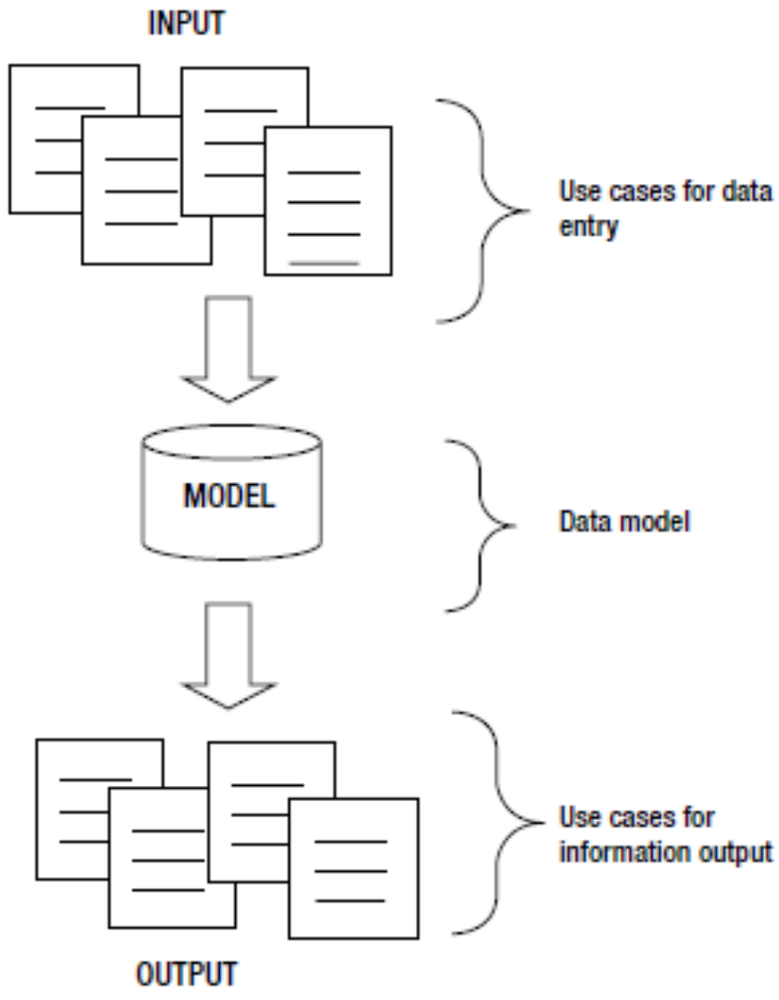
- There are two views of the problem. One is the concrete, real-world view from the **eventual user** (client), and the other view is the more abstract model from the **person who is designing and possibly developing the system** (analyst). If you are designing your own database, then wear two hats and swap them as necessary.

Today's Lecture

1. Real and Abstract Views of a Problem
2. What Does the User Do?
3. What Data Are Involved?
4. What Is the Objective of the System?
5. What Data are Required to Satisfy the Objective?
6. What are the Input Use Cases?
7. What is the First Model?
8. What Are the Output Use Cases?
9. More About Use Cases
10. Finding Out More About the Problem
11. What Have We Postponed?
12. Excercises
13. Summary

Real and Abstract Views of a Problem

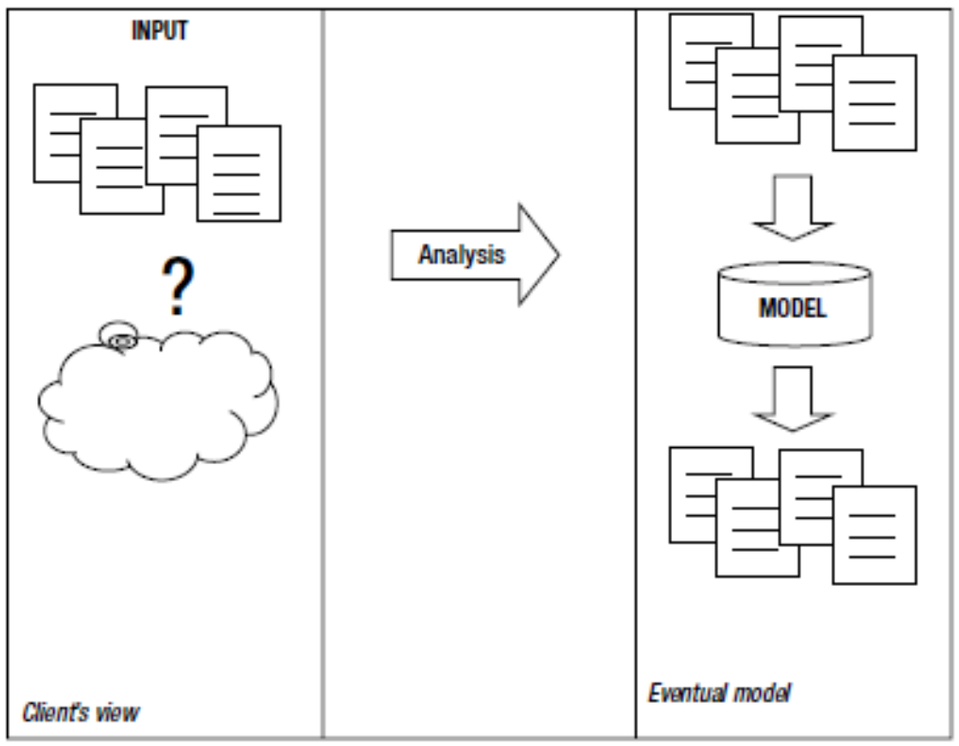
- Processing can mostly be separated into:
  - Entering, editing, or otherwise maintaining data.
  - Extracting information from the database based on some criteria.



- An analyst is to understand the client's problem in sufficient detail to help determine the input and output requirements (both immediate and potential).
- These can be expressed in use cases.
- The analyst then needs to develop a data model that will support those requirements.
- The data model provides considerable insight into the details of a system, so the use cases and data model are often developed in tandem.
- Establishing the use cases is not a simple problem. Users or clients seldom have a clear idea of the whole process.

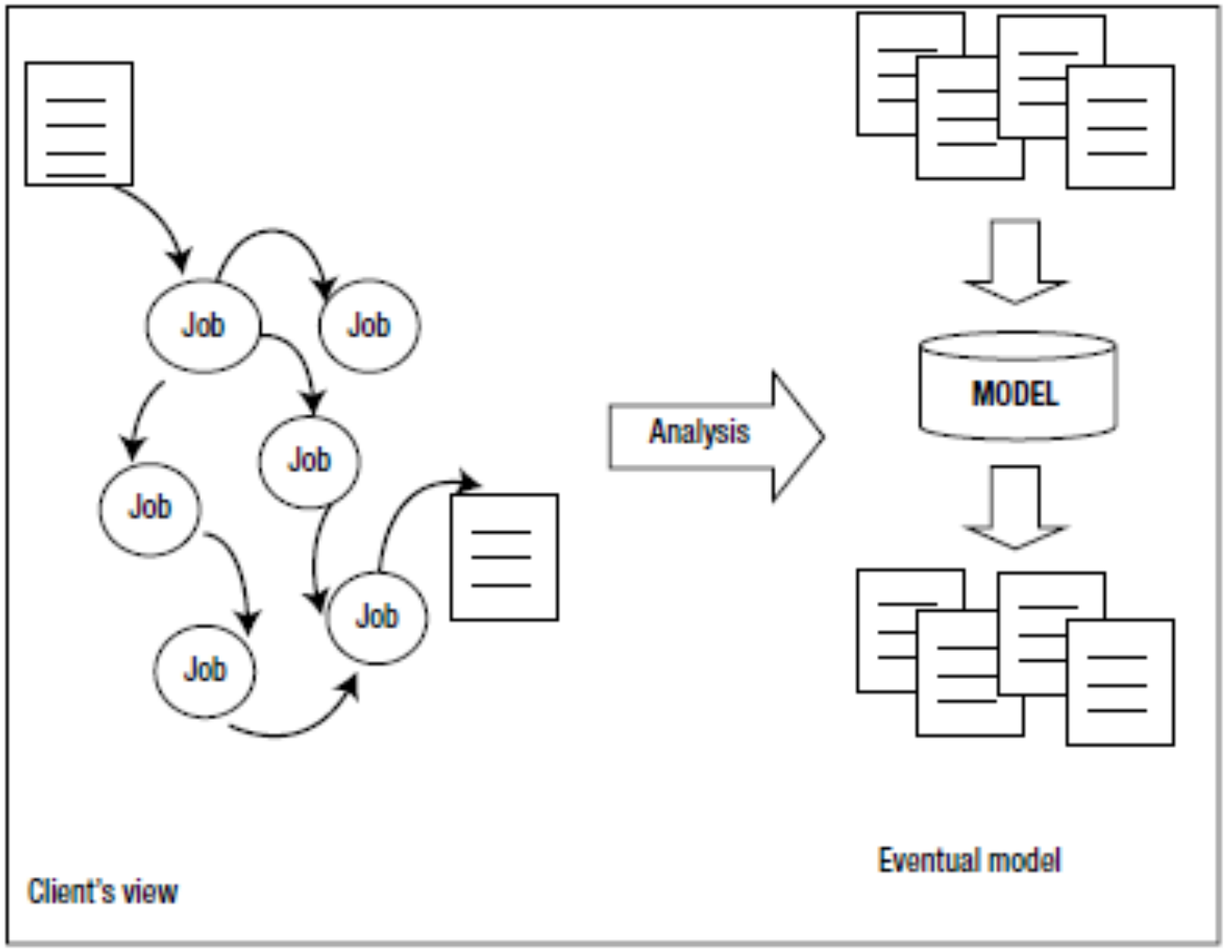
Data Mining

- The analyst's responsibility is to think ahead and ask questions about how else the data might be used, and store it in such a way as to allow for the immediate and possible future requirements.
- A careful analysis helps prevent the very common and infuriating situation of knowing the data is "in there" but not being able to "get it out" conveniently.
- Predicting the potential output requirements is one of the most difficult aspects of storing data.



Task Automation

- Many projects involve a client with a job that needs to be automated.
- These clients usually have a clear idea of what they do.
- The analyst's job here is to separate what the client *does* from what needs to be *recorded* and *reported*, and recast the problem



A task automation problem at a local school

When parents call up to say that children are sick, we have to let their classroom teachers know, and if it's sports day and the child is on a school team, the sports teacher might have to sort out substitutes. Then we need to count up all the days missed to put on the child's report. The Department of Education needs the totals each term, too.

A good start is to determine answers to the following questions:

- What does the user do?
- What data are involved?
- What is the main objective of the system?
- What data are needed to satisfy this objective?
- What are the input use cases?
- What is the first data model?
- What are the output use cases?

Example 3-1: Meal Deliveries

Visitors to the city staying in local motel or hotel rooms are offered a service that will deliver to them a variety of fast food or takeaway meals (pizzas, burgers, Indian takeout, and so on). A visitor phones the company and places an order for some meals. A driver is selected and dispatched to pick up the meals from the appropriate fast-food outlets. The driver delivers the meals to the customer, receives the payment, and informs the depot. He also fills in a time sheet, which he returns to the depot later.

One of the reasons given for wanting to automate this currently manual process is to be able to produce statistics about the numbers of orders taken and about the time taken to complete orders.

What Does the User Do?

- A question particularly relevant to task automation problems.
- Listing the jobs that the user regularly undertakes.
  - Receptionist records details of order (address, phone number, meals, total price).
  - Receptionist selects a driver and gives him the information about the order.
  - Driver picks up meal(s) from fast food outlet(s).
  - Driver delivers meal(s) and informs the depot.
  - Driver hands in time sheet at the end of his shift.
  - Receptionist or manager produces weekly and monthly statistics.

What Data Are Involved?

- Stated from the users' point of view and are what physically take place.
- Needed to step back a bit, put on our analysts' hats, and think about what data, if any, need to be recorded or retrieved at each step.

Physical User Tasks and Related Data

Task	Physical Jobs	Data That Could Be Recorded
1	Take order.	Order number, address, phone, name, meals, price, time.
2	Dispatch driver.	Driver's name (or ID?), order number, time, outlets to visit.
3	Pick up meals.	Order number, time of picking up each meal.
4	Deliver meals.	Order number, time of delivery.
5	Enter time sheet.	Anything other than what we already have for each order? Sign-in time, sign-out time?

Take order:

Recording the information about an order seems fairly straightforward. We need to be able to identify an order easily. We could refer to the customer and the time of placing the order, but generally assigning an order number will make it easier to track the order through its various stages. The information about the customer is fairly obvious. We need to at least record where the meals are to be delivered and how to get in touch with the customer. What about the meals that have been requested? How do we record this information? Presumably the customer is choosing from some list of available meals. Should the system be able to somehow provide that list of meals to the receptionist so that a selection can be made? What about price? If we have data about the meals, we will know the price. Is there some other cost that needs to be entered? Is there a mileage charge perhaps?

Dispatch driver:

First up, we need to think about how we know which driver is going to deliver the order. Does the system need to keep track of the whereabouts of drivers and determine which driver is the most appropriate? Does the receptionist choose from a list of drivers on duty? Does the system need to keep track of which drivers are available or which are currently on a delivery? If all the drivers are busy, what happens?

Having decided on a driver, we then need to tell him about the order (two curries, two pizzas). Do we also tell him where to go to get them (e.g., are there several pizza outlets from which to choose)? Does the system need to record which outlets provided the meal for this order? If the outlets for pizzas and curries are far apart, might two drivers be involved?

Pick up meals:

What do we want to record about a driver picking up a meal? Do we want the system to be able to tell us the current stage of an order (e.g., “Curries were picked up at 8:40, pizzas have not been collected yet”)? Do the eventual statistics need to be separated into times that meals were picked up and times that meals were delivered, or will overall times do?

Deliver meals:

If statistics on time are important, recording the time the meals were delivered will be essential.

Enter time sheets:

Assuming that time sheets are currently managed manually, looking at an existing time sheet will be very helpful. It is possible that the manual time sheet will contain some of the information we have already discussed. Is there any data that we have not recorded yet? Does the system need to record information about pay rates and payments made to the drivers? We discuss looking at existing manual forms again.

What is the Objective of the System?

- A system to record meal deliveries could be quite small or very large depending on how much of the information we decide to record. With our analysts' hats on, we need to sort out the main objectives and provide pragmatic solutions (as opposed to all-encompassing ones).
- One common problem is that when you ask questions, your clients may become quite enthusiastic about broadening the scope of the system to include more and more.
- It is important not to see everything that *could* be automated as something that *should* be automated.
- It is best to keep the scope of the problem as small and tightly defined as possible in the early stages of the analysis. Satisfy the most pressing requirements first.

- The initial incentive for developing the database was to provide summary information about the orders and the times involved. Information about orders in a summary might include the total number of orders and/or their combined value, probably within some timeframe (weekly or monthly). This information might allow the company to identify some trends and adapt its business accordingly.

- A question such as “What statistics do you want about time?” may not elicit adequate detail from a client.
- You might try to think of what could be achieved and try some more specific questions.
  - Do you need to have statistics to back up statements such as “Our meals are delivered within 40 minutes” or “Our average delivery time is 15 minutes”?
  - Do you need to be able to break down the delivery time to see where the delays are? For example: How long does an order typically have to wait before a driver becomes available? What proportion of the time is spent waiting for the meals to be prepared? What is the average time taken to deliver a meal from outlet to customer?
  - Do you need to be able to break these statistics down by driver? For example, to find out if any drivers are regularly slower than others?
  - Do you need to be able to break these statistics down by outlet? For example, do you need to see the average waiting times for each outlet to determine whether any are significantly slower?

- Let's assume that the main objective is just to get some idea of the overall times from phone call to delivery. Asking the other questions may (or may not) lead the client to become too ambitious: "I never thought of that. What a good idea. Throw that in as well."
- It is essential to consider how realistic it is to obtain data sufficiently reliable to fulfill these extra ideas.
- The main objective of overall delivery times isn't too difficult. It requires the time of the call to be logged, as well as the time of final delivery.
- Any more detail than that comes at significant cost.
- It is agreed that only the total delivery time is required. We can now restate the main objectives of the project:

To record orders for meals so that summaries of the number, value, and overall time taken to process orders can be retrieved for different time periods.

What Data are Required to Satisfy the Objective?

Take order:

If we are to provide statistics by month or week, we will need to record a date. The client has confirmed that there is a price list of different meals, and it would be useful for the receptionist to be able to make selections from this list. We will therefore need an additional task: to enter and maintain information about meals and their prices. The client confirms that the cost of the order is just the total cost of all the meals.

Dispatch driver:

We need to know how a driver is chosen and determine what we need to record.

Let's assume we discover that the drivers are assigned to be on duty for various time units. Obviously, being able to maintain and print out duty rosters would be useful. However, automating rosters doesn't directly contribute to our main objective. It is agreed to leave the rosters outside the scope of the system for now. The receptionist will use information available independently of the database (probably a list of names pinned to a notice board) to determine who should be assigned to deliver an order.

Even though the receptionist will assign the driver manually, we still need to consider what the system will need to record. How important is the accuracy of the driver information?

Where does the driver go to pick up the pizzas? Is it part of the system to suggest or record the outlet?

Pick up meals:

We agreed with the client that only the overall time from initial contact to final delivery of an order is required. This means we do not need to record the times at every stage of the process.

Deliver meals:

If we want to have statistics on overall delivery times, we clearly need to record the time that each meal is delivered. We don’t need to be concerned at this stage how that information gets into the database.

When the order is delivered, the receptionist also needs to know that the driver is free to take another order. We decided in the section about dispatching drivers that for now these decisions would be independent of the database

Enter time sheets:

We already have the driver's name, information about the order, and delivery times recorded. Is there anything else we need to record at this step? Let's say that a look at the current manual time sheets confirms that we already have all the information we need.

Example 3-2: Restatement of Meal Deliveries

The system will record and provide information about meals and their current prices. It will maintain data about orders including the date, the meals requested, and contact information for the customer and the driver assigned to the delivery. It will also maintain the time the order was placed and the time it was finally delivered. Given this, the system will be able to provide summary information about the number and value of orders within particular time periods and also summaries of the time taken for total processing of orders. The system will not maintain any additional information about drivers nor about which drivers were associated with a particular order. The system will not maintain any information about outlets nor which were used for any particular order.

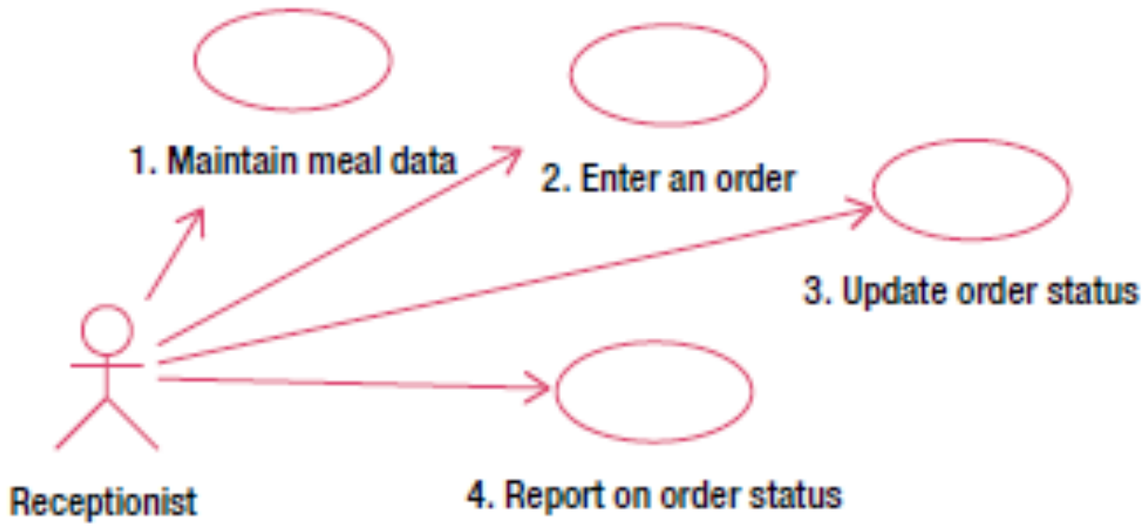
What are Input Use Cases?

- The most useful level for our purposes of trying to understand and describe a database system is the user task level.
- small enough ”that a user could do in less than about twenty minutes and then go off and have a coffee.”
- “significant enough so that if a user did several of the tasks in a day he could use it as evidence for a raise.”
- “manage the orders for the business” / “look up driver’s phone number”

Task	Physical Job	Interaction with System
0	Record available meals.	Enter and maintain data about each item that can be ordered (ID, description, current price).
1	Take order.	Enter order data (order number, time, address, phone) and the ID of each meal required (assume for now that prices don't change).
2	Dispatch driver.	Record driver's contact number with appropriate order.
3	Pick up meals.	Nothing.
4	Deliver meals.	Record delivery time for the appropriate order (here or possibly at the next step).
5	Enter time sheet.	Nothing.

- How big should each use case be? Should we combine some tasks or split others into more than one use case?
- At the first pass, about five to ten use cases is enough (and not too many) to give a clear view of the components of a small problem.
- Tasks are all quite separate, performed at different times, and possibly by different people.
- We can (if we feel like it) combine these into one use case called, for example, “Update Order Status.”
- Thinking about updating the status of an existing order leads us to ponder how the user will be able to locate a particular order. It might be useful to provide lists of orders yet to be assigned a driver or yet to be delivered.

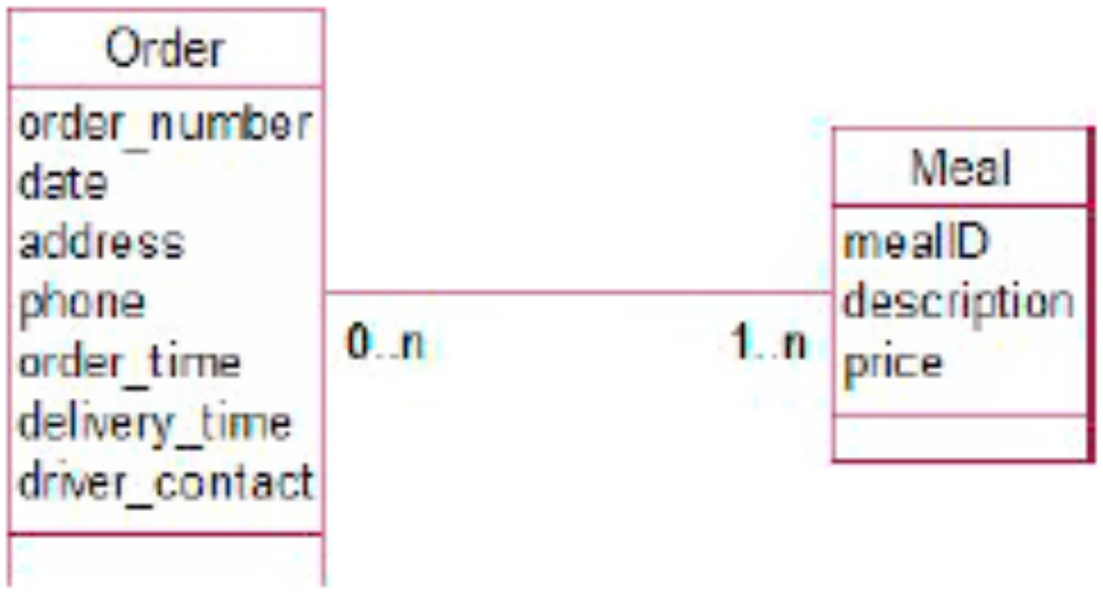
Example 3-3: Initial Use Cases of Meal Deliveries



- **Use case 1**: maintain meal data. Enter and update data on meals (Id, description, current price).
- **Use case 2**: Enter an order. Enter initial order information (order number, date, address, phone) and for each meal record the Id. (This assumes prices do not change.) Each meal must be one that is already in the system.
- **Use case 3**: update order status. For a particular order already in the system, add driver contact number or delivery time.
- **Use case 4**: Report on order status. Retrieve all orders satisfying required status (e.g., no driver contact number or no delivery time).

What is the First Data Model?

- We clearly have data about at least two separate things, orders and the types of meals that can be supplied, and so have two classes



What Are the Output Use Cases?

- Let's think about the statistics on orders and delivery times that are part of our main objective.
- The statistics on orders can be found by considering the Order objects. We can find the value of each order by summing the prices of each meal associated with that order, given (for now) that prices remain constant.
- We can also determine the time taken for each order by subtracting the order_time from the delivery_time.
- By selecting those order objects that are in the date period of interest, we can determine different statistics about the times (e.g., averages or totals) during a particular week or month or whatever is required.
- We have enough information stored in our data model to satisfy the requirements of our main objective.

- It is useful at this point to look at the data we are storing and see what other information can be deduced. What other statistics could we supply?
  - How about grouping all the orders for a particular type of meal?
  - The clients like to know how much gross income came from pizzas, or
  - how many people ordered curries, or
  - if orders containing particular types of meals took longer to deliver.

- Do we have the information in a form that would make this type of report readily available?

- We have information about particular meals (e.g., a chicken vindaloo or a lamb korma) but it is not easy to find out about different categories of meals (pizzas versus curries).
- Maybe it would be useful to introduce a new attribute or class, Category. Each meal could then be assigned a particular category.
- We will look more closely at whether something like a category should be an attribute or a class.
- This is only a small extension to the problem and may provide considerable additional information for little extra effort or cost.

Example 3-4: Statistical Reporting Use Case for Meal Deliveries

Use case: Summary reports on orders. (This assumes constant prices.)

For each completed order with a date *in the required time period:*

- Find all the associated meals and retrieve their prices,
- If required calculate the time of the order by subtracting order_time from delivery_time.
- If required, group orders by smaller time periods (day, week, etc.).
- Average and/or total prices/times.

More Use Cases?

- We are using use cases as a way to clarify and learn more about the proposed project, its scope, and its complexities.
- There are no hard-and-fast rules about what use cases should include or how they should be presented.
- The overriding consideration is that they should be readable and provide a clear and complete description of what each task involves.

Actors

- We use an *actor* as a representation of a user of our database. In order to take into account all the different ways our users might interact with the database, it is useful to consider all the different types of people our users may encompass.
- We distinguish two actors: receptionist and manager. It is not necessary to become too concerned about which people are associated with particular use cases.
- What is important is to consider the different roles of people likely to interact with the system and see the problem from the perspective of each.

Clerical/data entry operators: Users in this role deal with entering or updating raw data (e.g., entering order details or finding an order to enter a delivery time).

Supervisors: Users in this role deal with day-to-day details. They may require lists of transactions, rosters, and so on. For our meal delivery database, these users would probably deal with things such as a list of which orders have not yet been delivered or details of specific orders to follow up on problems.

Managers: Managers are more likely to be interested in summaries rather than day-to-day details. They may also require very general summaries that show trends and which can be used for forecasting and strategic management decisions.

Exceptions and Extensions

- The textual description of each use case is the place to include any exceptions or problems that might occur.
  - What to do about orders that run past midnight so as to get the elapsed time correct.
  - What happens if an order is not completed for some reason.

    We need to differentiate orders that have been cancelled from those that have not yet been delivered so our report on the status of current orders is correct. Every time we ask for those orders not yet delivered, we don't want to include all the discontinued orders from the beginning of time.

    Here are two possibilities: cancelled or terminated orders could be **deleted** from the system, or we could **add a new attribute, status**, to the Order class that could have values such as ordered, delivered, cancelled, and so on.

    The second option is more advisable in that it seems wasteful to delete information that is already in the system, and

    it is quite probable that a manager would be very interested to know what percentage of orders were cancelled (and very possibly why-but that introduces yet another level of complexity).

    Any additions such as keeping track of cancelled orders would have to be reflected in the use cases and data model.

Use Cases for Maintaining Data

- Maintaining data includes three activities: storing new values, altering existing values, and deleting data. The three updating tasks can be combined into one use case (e.g., maintain meal data).
- A user could not really use the fact that she had corrected many misspellings of a meal description as evidence for a raise. Database software provides facilities to carry out data maintenance activities.
- For many classes, it is quite reasonable to include these maintenance activities in one use case and leave the particulars for when we design a user interface at some later point.
- It may be sensible to separate out different aspects of maintaining a particular class of data.
- Considering the entering and updating tasks separately encouraged us to think about how a receptionist might conveniently find the appropriate order to update its status, and so led us to provide reports on the status of current orders.

Use Cases for Reporting Information

- Reporting tasks are probably the most significant part of the database system. We need to be able to extract objects that meet some criteria and then do something with them: display them on a screen or web page, write them out in a report, group them together, count them, or average or total some attribute value(s).
- We considered grouping orders by the type of meal and quickly realized that a broader definition of meal category might prove useful. Asking detailed questions about reports early on is a good investment because it will have an impact on the classes that will be required.
- How many use cases do you need for reports? If we were to include other quite different reports (rosters, invoices, and so on) then each should have its own use case.
- All that matters at this stage is that the data are stored in such a way as to make the reports possible.

Finding Out More About the Problem?

- A great deal of information is also available from other sources. The existing forms and reports that the client (business, researcher, club, etc.) is using are an excellent way to get an overview of a project. Having a close look at input forms and reports right at the start can improve the understanding of the problem and form a great basis for a line of detailed questioning.
- It is important to realize that you are looking at the forms and reports to find out about the problem.
- Existing reports also give you a guide as to what information is currently accessible to the client. But bear in mind that this project has possibly been commissioned because the existing reports are unsatisfactory in some respects.

What Have We Postponed?

- We need a bit more expertise with data modeling to represent some of the complexities.

Changing Prices

- The Meal class has an attribute that we have called price. This is the current price of a meal, and clearly it will change over time.
- When a new order is placed, we need to know the current price that is recorded with the meal information. If the prices change and we run a report about old orders we will have a problem.
- The only prices we are storing are the current prices, so we will not necessarily find the total cost of particular orders when they were placed, but instead will find how much those same orders would cost at today's current prices.
- The simplest would be to include another attribute in the Order class to contain the total value of the order at the time of ordering

Changing Prices

- The Meal class has an attribute that we have called price. This is the current price of a meal, and clearly it will change over time.
- When a new order is placed, we need to know the current price that is recorded with the meal information. If the prices change and we run a report about old orders we will have a problem.
- The only prices we are storing are the current prices, so we will not necessarily find the total cost of particular orders when they were placed, but instead will find how much those same orders would cost at today's current prices.
- The simplest would be to include another attribute in the Order class to contain the total value of the order at the time of ordering

Meals That Are Discontinued

- Another thing that is certain to change over time are the meals being offered.
- Adding new meals doesn't raise any problems; however, removing a meal is trickier.
- We have to consider what happens to old orders in the system that are associated with that meal. We probably want to retain this historical data, so we may choose never to remove any meals that are associated with orders.
- The set of meals includes some that should not be associated with new orders. One way to deal with this is to add an attribute, *available*, to the Meal class that indicates whether the meal can be ordered at the present time.
- We would need to alter our use case for entering an order to say that only meals that are available can be included. Our reporting use cases, however, would probably include all meals that were ordered during the reporting period.

Quantities of Particular Meals

- What if our customer orders two chicken vindaloos?
- We can associate the Order object with the Meal object, but where do we keep the information about how many of this particular meal is to be delivered for this order?
- This is a very serious oversight, and to fix it requires a new class between the Order and Meal classes.

Exercise 6-1

When parents call to say that children are sick, we have to let their classroom teachers know, and if it's sports day and the child is on a school team, the sports teacher might have to sort out substitutes. Then we need to count up all the days missed to put on the child's report. The Department of Education needs the totals each term, too.

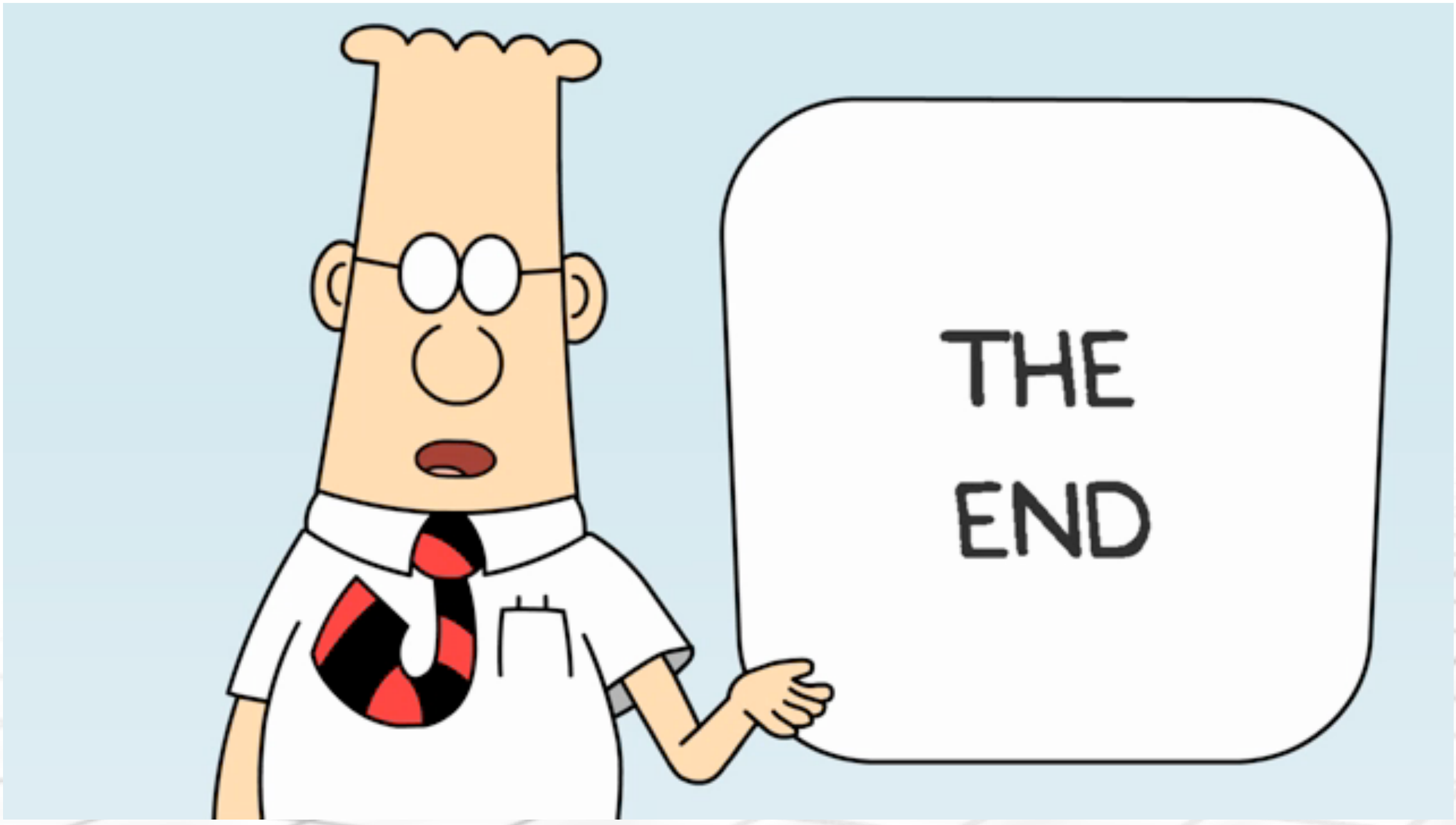
Run through the steps in the followings and sketch some use cases and an initial data model. Assume that the main objectives are to record the absences for the classroom teacher, for school reports, and for statistics given to the Department of Education.

- Determine the main objective of the system.
- Determine the jobs different users do in an average day.
- Brainstorm the data that could be associated with each job.
- Agree on the scope of the project and decide on the relevant data.
- Sketch data input use cases, consider exceptions, and check existing forms.
- Sketch a first data model.
- Brainstorm the possible outputs given the data being collected.
- Sketch information output use cases.
- Check that the data model can readily provide the output information

Summary

- Understand the main objectives and the scope of the project.
- Get inside the heads of all the different types of people
  - who will use the system to understand
  - what they require now and
  - what they are likely to need in the future.

- Determine the main objective of the system.
- Determine the jobs different users do in an average day.
- Brainstorm the data that could be associated with each job.
- Agree on the scope of the project and decide on the relevant data.
- Sketch data input use cases, consider exceptions, and check existing forms.
- Sketch a first data model.
- Brainstorm the possible outputs given the data being collected.
- Sketch information output use cases.
- Check that the data model can readily provide the output information



출처: metachannels.com

Thank you!