Recall the waterfall model
Requirements / Design / Implementation / Testing / Maintenance

Requirements analysis refers to the art of precisely specifying what a project should do, without
a. assuming anything about the customer.
b. describing how to do it.

Thus requirements analysis is mostly about effective communication:
a. Between developer and client about what the product should do.
b. Between the requirements team and the design team, about the client's needs.
Some simple rules:

a. Don't assume anything.

b. When in doubt, ask.

c. Avoid making design decisions before the design phase.

d. Employ accepted elicitation methods.

e. Seek consensus.
a. Don't assume anything because invalid assumptions lead to increased development cost.
b. When in doubt, ask, because cost of development is minimal when the actual requirements are documented at the earliest possible time.
c. Avoid making design decisions, because the design team needs freedom to choose the best possible path.
d. Employ accepted requirements elicitation methods, so that the design team will not have to learn a new method for expressing requirements!
e. Seek consensus because stakeholders will otherwise not use the product.
Requirements versus design

Requirements describe what to do.
Design describes how to do it.
Most common software engineering mistake: mix up design with requirements.
Why is this a bad thing?
Several basic requirements analysis and elicitation techniques:

- **Dictionaries**: listing all of the entities involved and their precise definitions.
- **Use cases**: creating a "script" of one or more specific kinds of use.
- **Entity/relationship analysis**: documents how entities interact.
- **State diagrams**: document how entity states can change, and why.
- **Time/space diagrams**: document how entities interact over time.
- **Control flow charts**: describe the actions of a single entity.
- **Data flow charts**: describe how data is exchanged between entities.
A simple example: task management

Wednesday, September 16, 2009
12:25 PM

What are the requirements for a task management system?

Inspiration: www.rememberthemilk.com
A user browses to the website. Website says "login". User enters username and password. Website shows tasks to be done today. Buttons on website allow viewing of all tasks and entering new tasks. User presses the button to see all tasks. User enters a new task. Task appears in list of all tasks, in alphabetical order. User selects the category "Inbox". Website shows all tasks in that category. New task appears in the category "inbox". User clicks on the task. Task is checked in the list. User types "c". Task is marked as completed.
Entities: nouns in the system
Interactions: verbs
Arrows: transitive verbs
States of a task, with user input that causes each state change.
Control flow charts:

- Process
- Sequence
- Decision
- Initiation/termination
Data D flows from A to B
Data flow example

Monday, September 10, 2012
3:56 PM
Obviously, not all methods work on all problems

Some problems don't have data flow.
Some problems aren't easily expressible as state transitions.
The requirements document is a bridge between the client and the designers.
- high level of abstraction
- in language understandable to the client
- but precise enough to empower design.
Some more general survival mechanisms:

- **domain analysis**: understanding similar problems
- **modeling**: keeping requirements high-level and specific to the problem domain.
- **stepwise refinement**: from general to specific
Domain analysis

fancy name for **figuring out what is known about your problem**
what related applications are there?
what do they do differently?
why?
Modeling

- raises the level of abstraction
- generates more understandable depictions of state
- documents one aspect at a time
Kinds of models:

- **Scenario-based** models: describe a product's use in context.
- **Behavioral** models: describe time-varying behavior.
- **Data** models: describe input and output structure.
- **Class** models: describe entities and interactions without considering time (document `interface` instead).
- **Flow** models: describe movement ("flow") of data.
Scenerio-based models

 easiest example: use case
 more subtle example: user survey
 Both indirectly describe what a product should do
Behavioral models

easiest example: state machines/flow charts
more subtle example: protocol diagram
Role: say what should happen.
Class models:
easiest example: taxonomy of kinds of tasks in the task manager.
more subtle example: object-oriented modeling. **Watch out:** object-oriented modeling (OOM) describes behavior, while object-oriented design (OOD) describes structure. These are different!
Most effective weapon in requirements analysis: **stepwise refinement**

- Simply stated: start with the big picture, evolve details.
- Largely a matter of "opening the box" when appropriate.
- For every requirements analysis methodology, there is a different kind of refinement.
Simplest model of a task: two states

But active tasks can have several states too:
The Jackson Duality principle:
"Software that processes data tends to follow the form of the data that is processed."
Jackson diagrams:
Model the form of data and/or the software that processes it.
A is comprised of B, plus perhaps more.

Sequence: A is comprised of B followed by C, plus perhaps more

Conditionals: A is comprised of 0 or 1 B's, plus perhaps more details other than B

Iteration: A is comprised of 0 or more repeats of B, plus perhaps more details.

Some other useful forms:

Iteration with lower bound: A is comprised of 1 or more repeats of B
Simple example: flowchart and Jackson diagram.
What Jackson analysis gives us

Ability to express choices without details.
An easy metric for program complexity
(McCabe's metric)

McCabe's metric:
Complexity = # if statements + 1 = total count of 0's and *'s + 1.

McCabe's claim:
If a piece of software exceeds a metric of 10, it cannot be tested by a human.
The power of Jackson diagrams

Can model data or actions.
Duality principle: model for data is model for actions (in some sense).
No assumption that model is complete.
Naturally indicates the complexity of the entity
(which determines cost of implementation)
Jackson diagrams versus flow charts

<table>
<thead>
<tr>
<th>Flow chart</th>
<th>Jackson diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expresses all activity</td>
<td>Expresses part of activity</td>
</tr>
<tr>
<td>Difficult to refine</td>
<td>Easy to refine</td>
</tr>
<tr>
<td>Does not show hierarchical structure</td>
<td>Shows hierarchical structure in a natural way</td>
</tr>
</tbody>
</table>
An accounts payable client record includes the name and address of a person, and an optional email address and phone number. It also indicates a payment amount and date to pay. The accounts payable database is a set of these records.