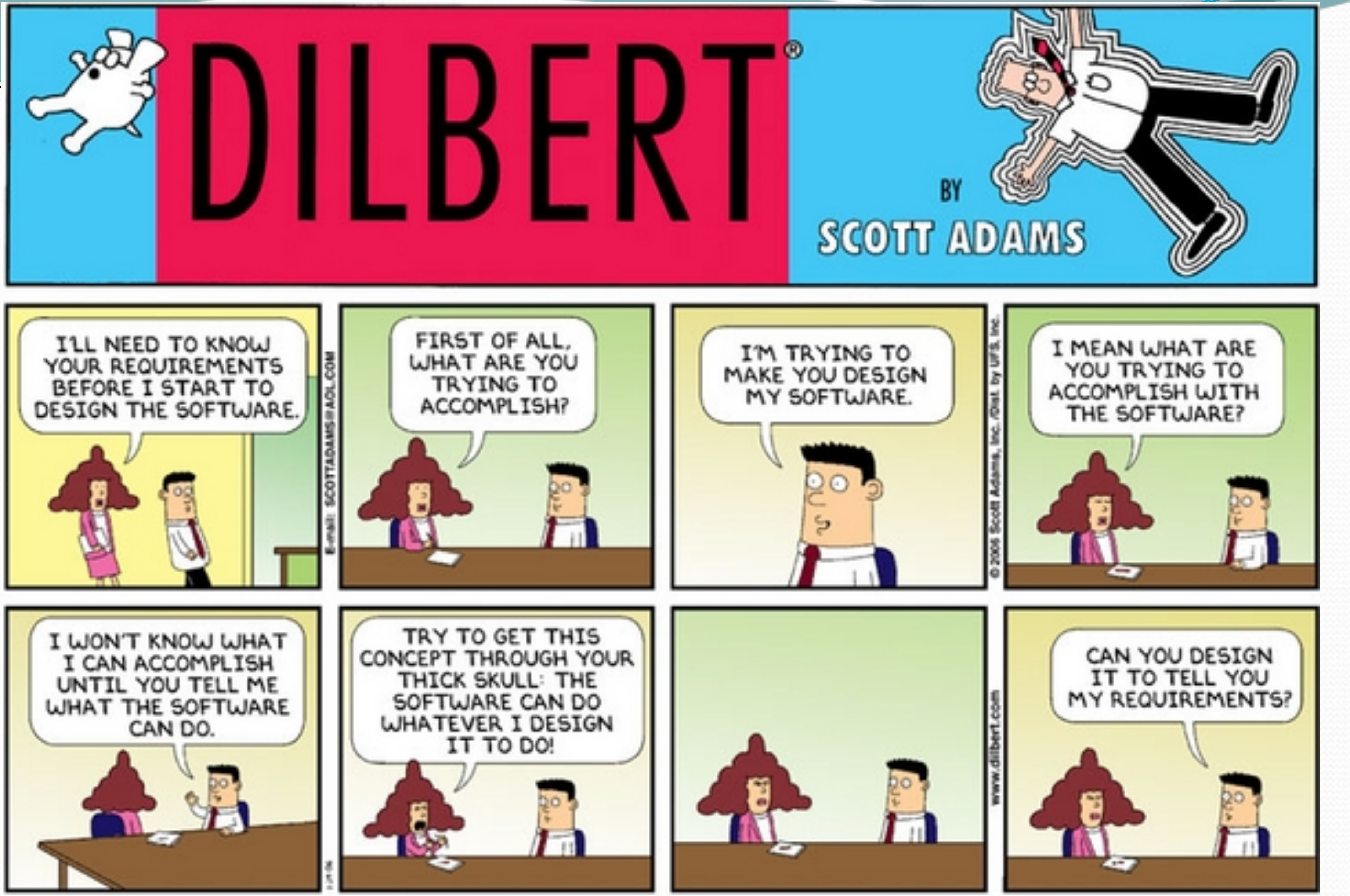

DILBERT®
BY SCOTT ADAMS
I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc. /Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

Requirements Engineering

Lots of jobs in this area

Lecture Content
Requirements engineering

- What is a User requirement
- Gathering User requirements
- Functional User requirements
  - We will discuss non-functional requirements in the next lecture
- Example

(User) Requirements

- Analysing what the customer **wants**
- In terms that the customer **understands**
- **What** not How
- Acceptance **tests** (Criteria) written

Requirement Engineering

- about producing a conceptual model, an explicit description, of what the system is about.

- Requirements may be determined
  - **<u>In detail at the beginning</u>** so that requirements engineering and design are totally separate, (waterfall method),
  - or **only very high level requirements** at the start.
    - These will be developed, becoming more detailed, as the systems is developed.
    - 1 feature / requirement at a time. (Agile development)

Requirements engineering is difficult:

- Stakeholders may have difficulty **articulating** what they want
- They may have implicit, **domain** knowledge and express requirements in their terms
- There may be **conflicting** requirements
- The importance of requirements may **change**, different stakeholders will have different priorities
- Some may be incompatible, others impractical.
- Constraints may mean that a selection has to be made from the list of requirements.
- **New** requirements may arise

Requirements Elicitation:

Working with customers, end users and other stake holders (indirect users) to find out about:

- **the application domain**; stakeholders will use domain language and have implicit knowledge, as the developer you must gain an understanding of the application domain and any existing systems.
- **services the system should provide;**
- **constraints**, (standards, Legal issues, Hardware, Software)

Sources of requirements :

- **Ask** the users;
  - Interviews (closed & open), Questionnaires, brainstorming
- **Observation**;
  - See what actually happens rather than what people say happens or the procedures that are described in documents, shows requirements based on cooperation and awareness of other peoples activity. Can help to determine implicit requirements.
- **Documents**;
  - Can provide information about the domain. Forms tell you about data in the domain (e.g. a bank paying in slip), manuals of procedures tell you how things are supposed to be done, (but remember may not be kept up to date).

Sources of requirements :

- **Scenarios;** -
  - Integral part of agile methods, see later in module
  - descriptions of example interaction sessions,
  - easier to relate to a 'real-life' example so users can understand and critique a scenario.
  - Can be text based.

NOTE - Use-case is a scenario based technique, it can be used to identify individual interactions with the system and supported with descriptive text or with other **UML models** which provide more detail.

- **Prototypes**. – Can be used to help clarify some requirements.

Requirements Validation

- The **requirements** specification needs to be **validated** to ensure that stated requirements are the ones that the different **stakeholders actually want**
- Requirements should be reviewed to see if they are:
  - **Valid** - expresses the real needs of the stakeholders
  - **Unambiguous** - not be open to misinterpretation
  - **Complete** - no relevant aspect has been left out
  - **Consistent** - no contradictions or conflicts with other requirements
  - **Feasible** - possible to satisfy requirements within known constraints
  - **Testable/verifiable** - measurable where possible - avoid vague terms

Requirements Validation

- The requirements specification should be used to develop test strategy for system
- Acceptance tests should be defined
  - Developing verifiable test criteria for each requirement helps reduce ambiguity in validating the final product

Specifying User Requirements

- Two types of Requirements
  - **Functional** -- What the customer needs the system to enable then to do.
  - **Non Functional** – What the customer needs the system to be. (non functional requirements can lead to functional system requirements.)

Functional Requirements

- Can be written in Natural language
  - Prone to ambiguity, therefore acceptance criteria

- Models – UML – Use Cases (Diagram and Descriptions)

  Structured natural language - Used to document **functions** provided by the software

    - OOA - Description of a use case might include - use case ID, use case name, actors, pre-conditions, description of the main steps in the scenario, related use cases, post-conditions (return to later)

A Functional requirement is

a **specific thing** a stake holder wants to be able to do using the system.

- Statements about what the stake holder should be able to do
- Literally the functions ‘someone acting on the system’ can carry out.

Prioritise

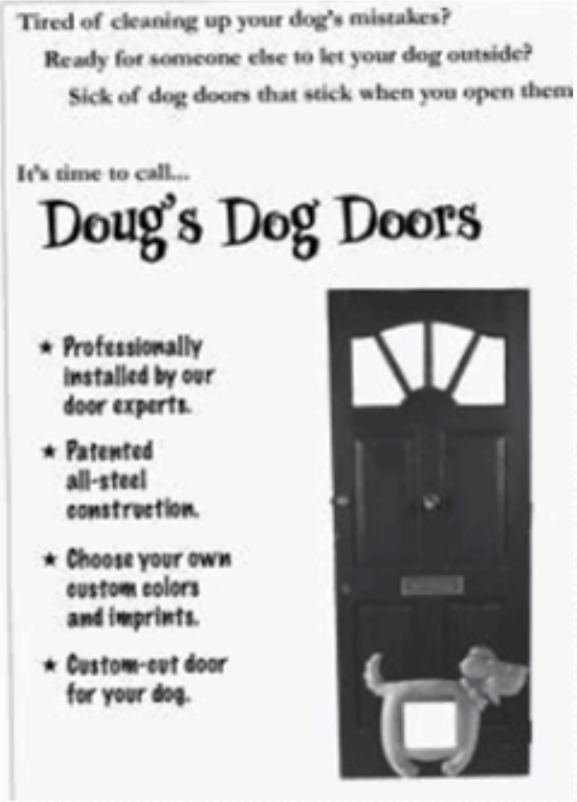
- in order of importance
  - **MoSCoW** –
    - **M**ust have,
    - **S**hould have,
    - **C**ould have,
    - **W**ont have

Help: reveal hidden assumptions, project planning and rescheduling if this becomes necessary.

What the customer wants

Todd and Gina want more than a normal doggie door. They has everything from the TV to the music system to the garage door operating off remote controls. They want a dog door operating off a remote control.

They have contacted Doug dog doors for a door and now Doug want us to build them the system to operate the dog door of Todd and Gina's dreams



Tired of cleaning up your dog's mistakes?

Ready for someone else to let your dog outside?

Sick of dog doors that stick when you open them

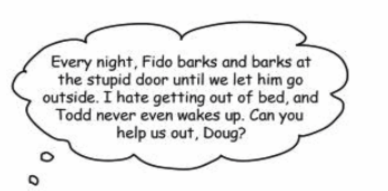
It's time to call...

Doug's Dog Doors

- Professionally installed by our door experts.
- Patented all-steel construction.
- Choose your own custom colors and imprints.
- Custom-cut door for your dog.

From words to wants

Todd and Gina:
Todd and gina want more than a normal doggie door. Todd has everything from his TV to his music system to his garage door operating off remote controls. Todd wants his dog door operating off a remote control. They have contacted Doug dog doors for a door and now Doug want us to build them the system to operate the dog door of Todd and Gina's dreams


Every night, Fido barks and barks at the stupid door until we let him go outside. I hate getting out of bed, and Todd never even wakes up. Can you help us out, Doug?

Todd and Gina
Your dog door requirements list:
A
B
C will use Mongo DB
D

NO
This is a HOW not a WHAT

Use a Scenario

Todd and Gina's Dog Door

What the Door does

1. Fido barks to be let out.
2. Todd or Gina hears Fido barking.
3. Todd or Gina use the remote control.
4. The dog door opens.
5. Fido goes outside.
6. Fido does his business.
   6.1 The door shuts automatically.
   6.2 Fido barks to be let back inside.
   6.3 Todd or Gina hears Fido barking (again)
   6.4 Todd or Gina use the remote control.
   6.5 The dog door opens (again).

7 Fido goes back inside.
8 The door shuts automatically.

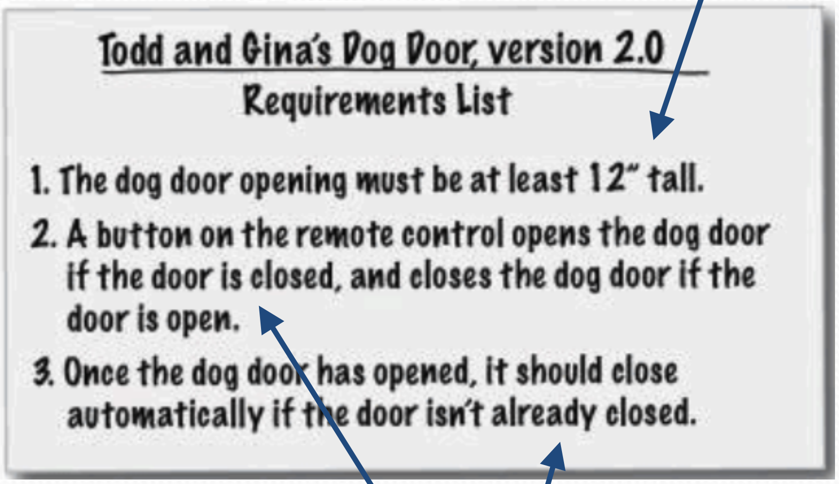
What the customer wants

Todd and Gina's Dog Door

What the Door does

1. Fido barks to be let out.
2. Todd or Gina hears Fido barking.
3. Todd or Gina use the remote control.
4. The dog door opens.
5. Fido goes outside.
6. Fido does his business.
   6.1 The door shuts automatically.
   6.2 Fido barks to be let back inside.
   6.3 Todd or Gina hears Fido barking (again)
   6.4 Todd or Gina use the remote control.
   6.5 The dog door opens (again).
7 Fido goes back inside.
8 The door shuts automatically.

Non functional. Not what your system does



Todd and Gina's Dog Door, version 2.0

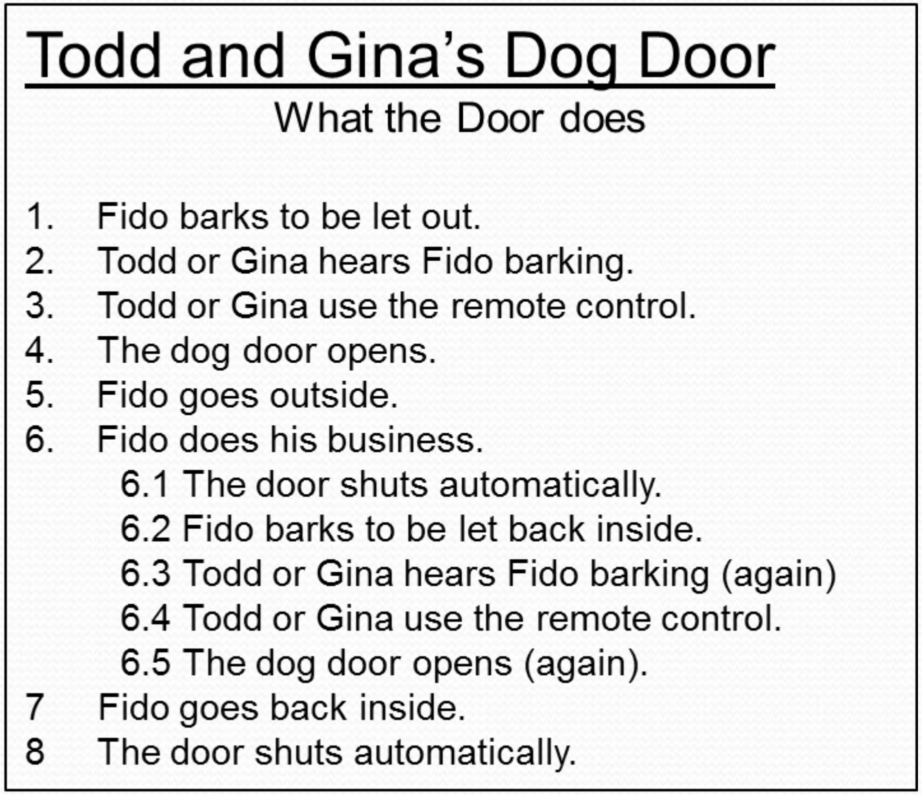
Requirements List

1. The dog door opening must be at least 12" tall.
2. A button on the remote control opens the dog door if the door is closed, and closes the dog door if the door is open.
3. Once the dog door has opened, it should close automatically if the door isn't already closed.

Functional

Draft set of requirements

What the customer wants



Todd and Gina's Dog Door

What the Door does

1. Fido barks to be let out.
2. Todd or Gina hears Fido barking.
3. Todd or Gina use the remote control.
4. The dog door opens.
5. Fido goes outside.
6. Fido does his business.
   6.1 The door shuts automatically.
   6.2 Fido barks to be let back inside.
   6.3 Todd or Gina hears Fido barking (again)
   6.4 Todd or Gina use the remote control.
   6.5 The dog door opens (again).

7 Fido goes back inside.

8 The door shuts automatically.

Todd and Gina's Dog Door

Requirements List

????

???

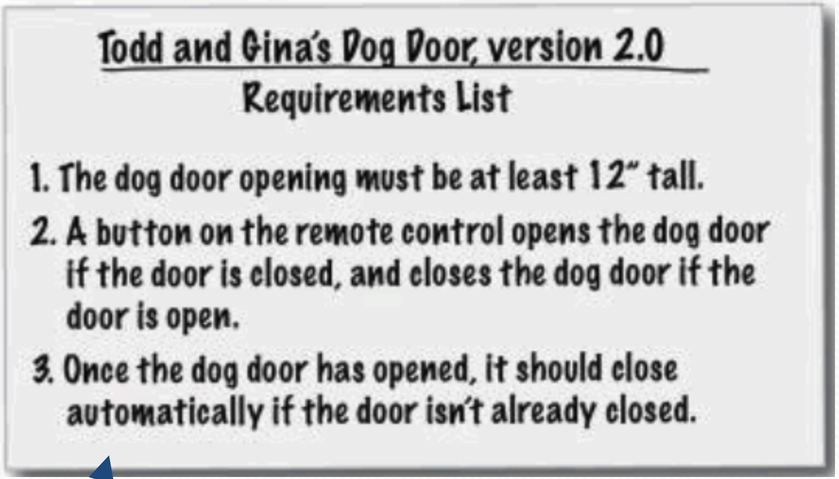
Example Requirement

- **Dog Door system**
- Requirement 1. (Must)
  - **Requirement : Todd and Gina MUST be able to open the Dog Door flap via a remote device.**
  - Acceptance Criteria.
    - Demonstration of
      - On using the remote device to select to open the dog door, the system should open the Dog Door flap.
      - The Door should remain open for 5 Minutes
      - The Door should them close and lock.

Ambiguous and conflicting

Example: Requirements 2 and 3 are ambiguous and conflicting

Requirements should be written so they are not open to miss interpretation. Acceptance criteria helps this.



Todd and Gina's Dog Door, version 2.0

Requirements List

1. The dog door opening must be at least 12" tall.
2. A button on the remote control opens the dog door if the door is closed, and closes the dog door if the door is open.
3. Once the dog door has opened, it should close automatically if the door isn't already closed.

A lot of the detail is now in the acceptance criteria

Example Requirement

- **Dog Door system**
- Requirement 11. (Could)
  - Requirement : Todd / Gina COULD be able to alter the time between the door opening and then closing.
  - Acceptance Criteria.
    - Demonstration of
      - Todd / Gina selects to change the timing.
      - System allows Todd / Gina to enter new time delay – number of minutes between 1 and 60.
      - Todd / Gina asked to confirm new time.
      - System confirms that timing has been altered.

Example Requirement

- **Dog Door system**
- Requirement 15. (Won’t)
  - Requirement : The system will not (Won’t) respond to the bark of a dog as a signal to open the door.

Are these functional requirements?

A. Allow Library member to borrow a book.
B. Store the book in a linked list
C. Search to find out if a book is out on loan
D. Access loan list from a webpage
E. Should be reliable

These are functional requirements.

A. Allow Library member to borrow a book.

C. Search to find out if a book is out on loan