CSE503: Software Engineering

David Notkin
University of Washington
Department of Computer Science & Engineering
Winter 2002

1

Success of a system

- A system is judged not by properties of the program, but by the effects of the machine in the world
- You don't care how Caller ID works, just *that* it works
- TCAS is a collision-avoidance system for commercial aircraft
  - Pilots love it (on the whole) because it helps them fly more safely and easily — not because it has great data structures or a fascinating specification

2

Requirements & specification

- More software systems fail because they don't meet the needs of their users than because they aren't implemented properly
- Boehm:
  - Verification: Did we build the system right?
  - *Validation: Did we build the right system?*
  - *Validation: Did we build the right system?*

3

Three challenges

- To figure out the desired effects (requirements) of the machine in the world
- To figure out how to write this down in an effective way
- To figure out how to make sure that the machine (program) you build satisfies the requirements

4

Challenge #1

- Determining the "right" requirements
  - Requirements analysis, requirements discovery, requirements elicitation, requirements engineering, etc.
- This is *extremely* hard and we won't address it this quarter
  - Any experience among you in doing this?

5

Challenge #2: Writing it down

- Even if you know what you want to say, how do you write it down?
- Why does this matter?
  - It will help clarify what you think
  - It is necessary to communicate with your customers
  - It is necessary to communicate with your team members
  - It could form the basis for a contractual relationship
- Not unrelated to Challenge #1, since the process is really iterative

6

How to write it down?

- Choice #1: natural language
- Choice #2: structured natural language
- Choice #3: formal language(s)

7

Natural language

- Inherently ambiguous
  - If you don’t believe it, make sure to teach or TA an undergraduate course sometime!
- Also complex
  - (You’ll have your own favorites along these lines; this is from one of Jackson’s books)
  - In an airport at the foot of an escalator are two signs
    - “Shoes must be worn.”
    - “Dogs must be carried.”

8

In logic it’s clear

- forall x • (OnEscalator(x) =>
  there-exists y • (PairOfShoes(y) and IsWearing(x,y))
- forall x • ((OnEscalator(x) and IsDog(x)) => IsCarried(x)

9

Or is it?

- Do dogs have to wear shoes?
  - Is this a question of the types of x and y?
- What are “shoes”? What are “dogs”? What does it mean to “wear shoes”?
- Why do the formalizations say “dogs are carried” and “shoes are worn” while the signs say “must be”?
- As Jackson said in the video (with a different example)
  - The formalizations are in the indicative mood: statements of fact
  - The signs are in the optative mood: statements of desire
  - This kind of “mood mixing” increases confusion

10

“dog” (noun)

- OED has 15 definitions
  - 11K words in the full definition
- Webster’s 11 definitions include
  - a highly variable domestic mammal (*Canis familiaris*) closely related to the common wolf
  - a worthless person
  - any of various usu. simple mechanical devices for holding, gripping, or fastening that consist of a spike, rod, or bar
  - FEET
  - an investment ... not worth its price
  - an unattractive girl or woman

11

“shoe” (noun, Webster’s): six definitions including

- an outer covering for the human foot usu. made of leather with a thick or stiff sole and an attached heel
- another's place, function, or viewpoint
- a device that retards, stops, or controls the motion of an object
- a device (as a clip or track) on a camera that permits attachment of accessory items
- a dealing box designed to hold several decks of playing cards

12

Optative vs. indicative mood: reprise

- Indicative: describes how things in the world are regardless of the behavior of the system
  - “Each seat is located in one and only one theater.”
- Optative: describes what you want the system to achieve
  - “Better seats should be allocated before worse seats at the same price.”

13

Principle of uniform mood

- Indicative and optative properties should be entirely separated in a document
  - Reduces confusion of both the authors and the readers
  - Increases chances of finding problems
- If the software works right, both sets of properties will hold as facts

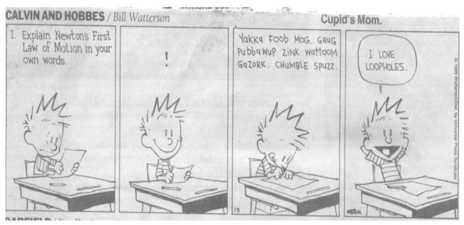
14

Mood mixing: example

- The lift never goes from the nth to the n+2nd floor without passing the n+1st floor.
- The lift never passes a floor for which the floor selection light inside the life is illuminated without stooped at that floor.
- If the motor polarity is set to up and the motor switch setting is changed from off to on, the lift starts to rise within 250 msecs.
- If the upwards arrow indicator at a floor is not illuminated when the lift stops at the floor, it will not leave in the upwards direction.
- The doors are never open at a floor unless the lift is stationary at that floor.
- When the lift arrives at a floor, the lift-present sensor at the floor is set to on.
- If an up call button at a floor is pressed when the corresponding light is off, the light comes on and remains on until the call is serviced by the lift stopping at that floor and leaving in the upwards direction.

15

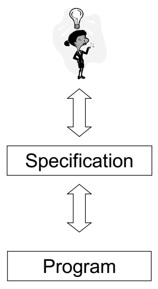
A students’ view


CALVIN AND HOBBES / Bill Watterson
Cupid's Mom.
1. Explain Newton's First Law of Motion in your own words.
!
Yakka Foob Mog. Grug Pubbawup Zink Wattoom Gazork. Chumble Spuzz.
I LOVE LOOPHOLES.
© 1995 Watterson/Dist. by Universal Press Syndicate
WATTERSON

16

Another high-level issue

- Specification languages that are “closer” to the user decrease the change of building the wrong system
  - But increase the chance of building the system wrong
- And specification languages that are “closer” to the program do the opposite
  - Increasing the chance of building the wrong system



17

Informal approaches

- Running plain text requirements specifications are increasingly less common
- Uniform mood, designations/descriptions, etc. don’t argue for a particular style of presentation, but rather for properties that requirements specifications should have

18

“Will” and “Shall”

- Some government groups write requirements with specified meanings for “will” and “shall” and “may” and such
  - “shall” is a requirement
  - “may” is an optional requirement
  - “will” describes something not under control of the system
- Almost always unclear
  - Related to mood mixing

19

Structured natural language

- I
  - I.A
    - I.A.ii
      - I.A.ii.3
        - » I.A.ii.3.q
- Although not ideal, it is frequently better than unstructured natural language
  - Unless the structure is used as an excuse to avoid content

20

Formal languages

- Write down your requirements in some form of mathematics
  - The precision is greater, thus less ambiguity
  - Tools for reasoning about the properties of the system are then feasible
  - Support automatic generation of all or part of the system
- Great ideas — but they seem to be hard to use in practical situations

21

Non-functional requirements

- We’re simply going to ignore non-functional requirements
  - Performance, ease of change, etc.
- I’m not proud of this, but there is relatively little known about this issue
- Worthwhile concrete discussion: should an interface’s specification (documentation) specify the performance of the operations?
  - Pro: Sure, it’s a key property (and people will find it out anyway)
  - Con: No way, since I’m supposed to be able to change an implementation as long as it behaves the same

22

Next Lectures: three specification approaches

- Model-based specification: Z
- Algebraic specification
- State-machine specification

23