CSE584: Software Engineering

Lecture 6 (November 10, 1998)

David Notkin
Dept. of Computer Science & Engineering
University of Washington
www.cs.washington.edu/education/courses/584/CurrentQtr/

This week and next

- Requirements and specification
  - Perhaps more accurately, although confusing, it should be "requirements specification" and "specification"
- At the highest level, this topic focuses on
  - how to determine what a software system is supposed to do and
  - how to write that down
    - Recall Michael Jackson's comments on "what" vs. "how"

Today you've already seen

- A lot of Michael Jackson
  - In particular, a video of his 1995 keynote at the 17th International Conference on Software Engineering
- The focus was: how does a software system (machine) fit into the world

Caveat

- There has been broad agreement (within the students in this class, as well as others) that a common problem in software engineering is ill-defined requirements (and instable ones, too)
- The approach Jackson uses to attack this question isn't going to be easy for some of you to swallow
  - It's more focused on contract-style systems and less focused on desktop applications (he would strongly deny this, probably rightfully so)
  - I believe the ideas are broadly applicable and worth seeing
  - His results are intellectual, not technology results

John Gall (*Systemantics*)

- From Jackson (of course)
- "To those Within a System, The Outside Reality Tends to Pale and Disappear."

Comments on video?

A few more examples

- Illustrating some of Jackson's points
  - Primarily taken from Jackson's book, *Software Requirements & Specifications: a lexicon of practice, principles and prejudices*

Ambiguity

- (You'll have your own favorites along these lines)
- In an airport at the foot of an escalator
  - Must I carry a dog?
  - What about the shoes I just bought that are still in my shopping bag?

Shoes Must Be Worn

Dogs Must Be Carried

$\forall x \bullet (\text{OnEscalator}(x) \rightarrow \exists y \bullet (\text{PairOfShoes}(y) \wedge \text{IsWearing}(x,y))$

$\forall x \bullet ((\text{OnEscalator}(x) \wedge \text{IsDog}(x)) \rightarrow \text{IsCarried}(x)$

But we're not done

- The formalization still leaves unanswered questions
  - Do dogs have to wear shoes?
  - What are shoes? 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"?
    - The formalizations are in indicative mood
    - The signs are in optative mood

"dog" (noun)

- OED has 15 definitions (11K words in 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

"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

Clarity is hard and crucial

- We all know this, but Jackson helps us think about it more clearly
  - Moods
  - Designations vs. definitions
    - Designations are the atomic phenomena
      - e.g., genetic mother
    - Definitions define terms in terms of designations and other previously defined descriptions
      - e.g., genetic child of
    - Refutable descriptions can in principle be disproven
      - $\forall m, x \bullet \text{Mother}(x,m) \rightarrow \neg \text{Mother}(m,x)$
      - Can't do this with definitions

Mood mixing

- 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 without 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

Principle of uniform mood

- Indicative properties 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

Informal approaches

- Running plain text requirements specifications are increasingly less common
- There are a number of approaches between this and formal specifications that give varying degrees of leverage
- Note that Jackson didn't argue for any particular style or notation
  - But rather for properties that requirements specifications should have

"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
- Not always too clear
  - Related to mood mixing

Structured requirements

- I
  - I.A
    - I.A.ii
      - I.A.ii.3
        - » I.A.ii.3.q
- Say no more!
  - It didn't work for me in the assigned work description; it often doesn't work in practice
    - Although it is usually better than unstructured natural language

Formal methods

- The original use of formalism in software engineering was for proving the equivalence between a specification and an implementation
  - This had a number of problems
- But there has been a resurgence of interest in formal methods
  - Mostly due to potential usefulness in specification
  - And a few success stories

Potential benefits

- Increased clarity
- Ability to check for internal consistency
- Ability to prove properties about the specification (related to Jackson's refutable descriptions)
- Provides basis for falsification (perhaps more useful than verification)
- But not always worth the effort

Styles of specifications

- Model-oriented (e.g. Z, VDM)
- Algebraic (e.g. OBJ, Larch)
- Process Model (e.g. CCS, CSP)
- Finite state-based (e.g. Statecharts, RSML)
- Logical, constructive, multi-paradigm, broad spectrum, ...

Model oriented

- Model a system by describing its state together with operations over that state
  - An operation is a function that maps a value of the state together with values of parameters to the operation onto a new state value
- A model oriented language typically describes mathematical objects (e.g. data structures or functions) that are structurally similar to the required computer software

Aside: Jackson

- From a specification of a small telephone system
  - "…a subscriber is a sequence of digits. Let **Subs** be the set of all subscribers … ...certain digit sequences correspond to unobtainable numbers, and some are neither subscribers, nor are they unobtainable."
- "Only a mathematician could treat the real world with such audacious disdain." —Jackson

Algebraic specifications

- Represent structures as algebras
  - Represent results as compositions of operations, not as explicit state
  - Closely related to ADTs
- Algebraic methods tend to provide less implementation bias than some other methods

Process based specifications

- For describing concurrent systems
- Also algebraic in nature, but focus on processes that can be composed over a variety of operators (such as run in parallel)

Finite state specifications

- Represent system as a finite state machine
- Transitions fired by external (and maybe internal) events
- Often useful in describing aspects of embedded systems
  - Inputs from sensors, outputs to actuators
- For the Leveson et al. paper I handed out you might think about how it fits into the Jackson material

Next week

- Next week I'll pick a few of these mthods and show "how they work"
- Note that little attention has been paid to "non-functional" requirements
  - These have, however, been very high on your radar during class discussions