CS5127/6027: Requirements Engineering

(Fall 2024)

Prof. Nan Niu (nan.niu@uc.edu)

Office Hours: 10am-11am, Mondays, Rhodes 832

© 2024, Nan Niu

Today's Menu

Last Lecture (Friday 10/25):
Req.s Analysis

This Lecture (Monday 10/28):
Req.s-Based Testing

Next Lecture (Friday 11/1):
starting at *9:30am*
ASN3 release

© 2024, Nan Niu

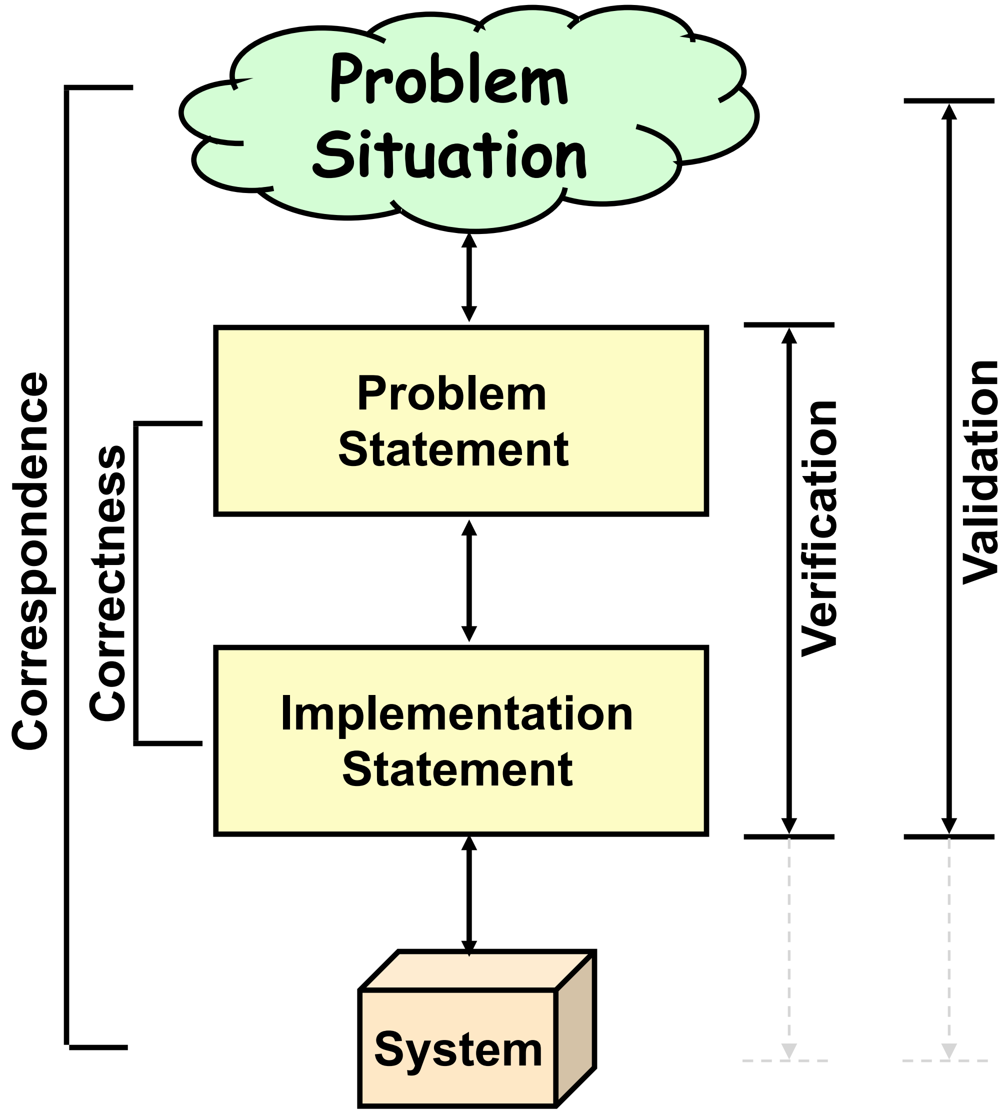
Updates

→ASN2 graded

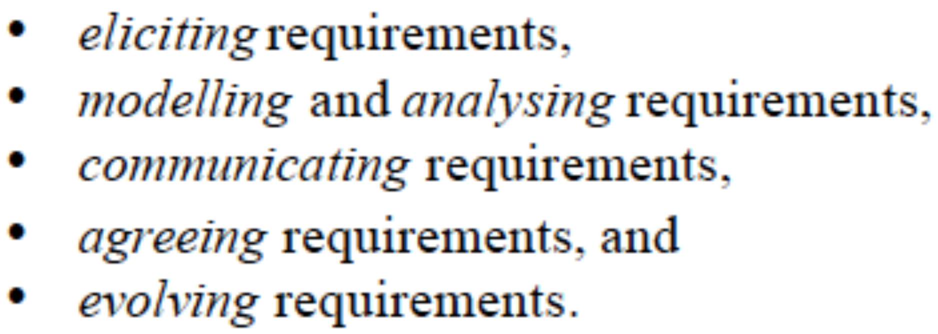
→Quiz9 due: this Wednesday (Oct 30)

Req.s Testing? Req.s-Based Testing?

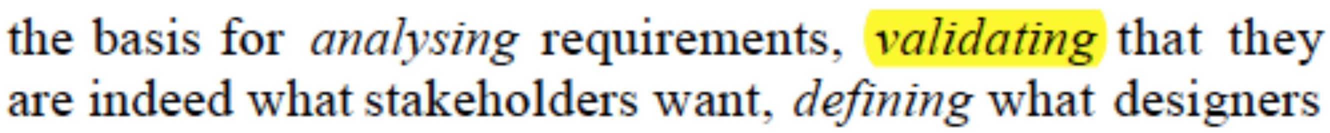
- Verification
  - "Are we building the system *right*?"
- Validation
  - "Are we building the *right* system?"



From slide #5 of Aug 30, 2024



- *eliciting* requirements,
- *modelling* and *analysing* requirements,
- *communicating* requirements,
- *agreeing* requirements, and
- *evolving* requirements.



the basis for *analysing* requirements, *validating* that they are indeed what stakeholders want, *defining* what designers

Therefore, we're **NOT** talking about "req.s testing", or "testing/QA-ing req.s" *per se*.

© 2024, Nan Niu

Req.s-Based Testing

- In black-box testing, we are interested in creating a suite of tests from requirements that adequately exercise the behavior of a software system without regard to the internal structure of the implementation *[Whalen et al. ISSTA'06]*.
  - the other ends: white-box testing, structure of the code …

Designing test cases (TCs)

- →From where?
- →Based on what?
- →How many should be there?

```
int proc1 (int a, int b) {
    if (a<0 or b<0) exit;
    tag = 0;
    int result = a;
    if (a<b) {
        tag = 1;
        result = b;
    }
    if (tag) {
        for (int i=0, i<a, i++) {
            result++;
        }
    } else {
        for (int j=0, j<b, j++) {
            result++;
        }
    return result;
}
```

```
int proc2 (int a, int b) {
    if (a<0 or b<0) exit;
    int result = a;
    for (int k=0, k<b, k++) {
        result++;
    }
    return result;
}
```

```
int proc3 (int a, int b) {
    //TBD
}
```

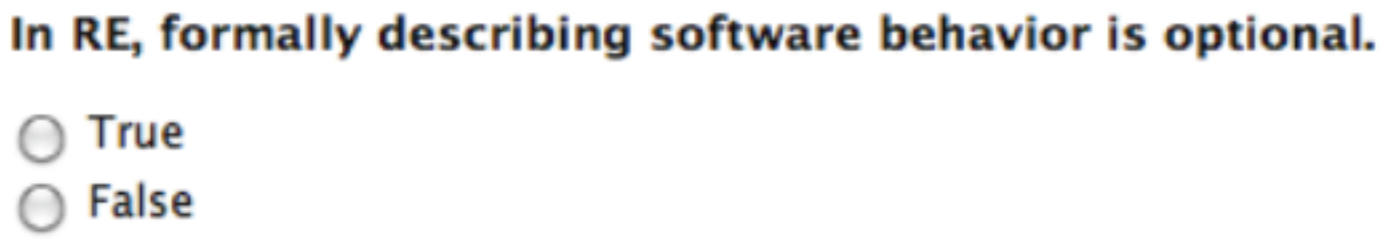
Req.s-based testing ➔ Model-based testing

- Requirements engineers are modelers
  - Goal modeling: actors, dependencies, rationalities, alternatives …
  - Structure modeling: entities, relationships, attributes, cardinalities …
  - Behavioral modeling: states, transitions, conditions, guards, …
  - …

© 2024, Nan Niu

Modeling Behavior

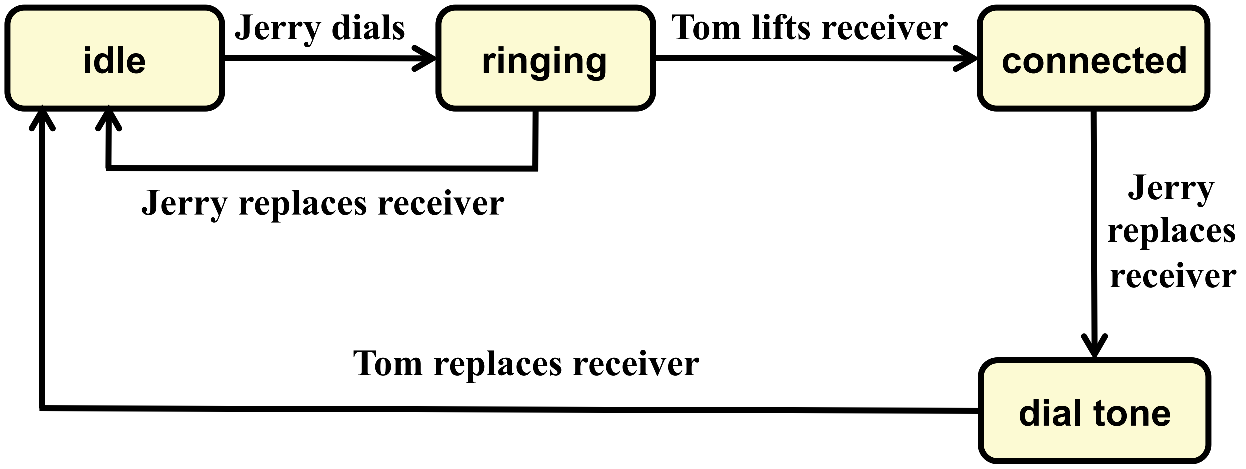
- Behavior of "what"?
  - The intended software
- Why worrying about "behavior"?
  - Software is invisible, so arguably the most significant way to "see" software is to interact with it
  - Software is experienced ➔ the behavior of software defines what it is
- Why worrying about behavior in "RE"?



In RE, formally describing software behavior is optional.

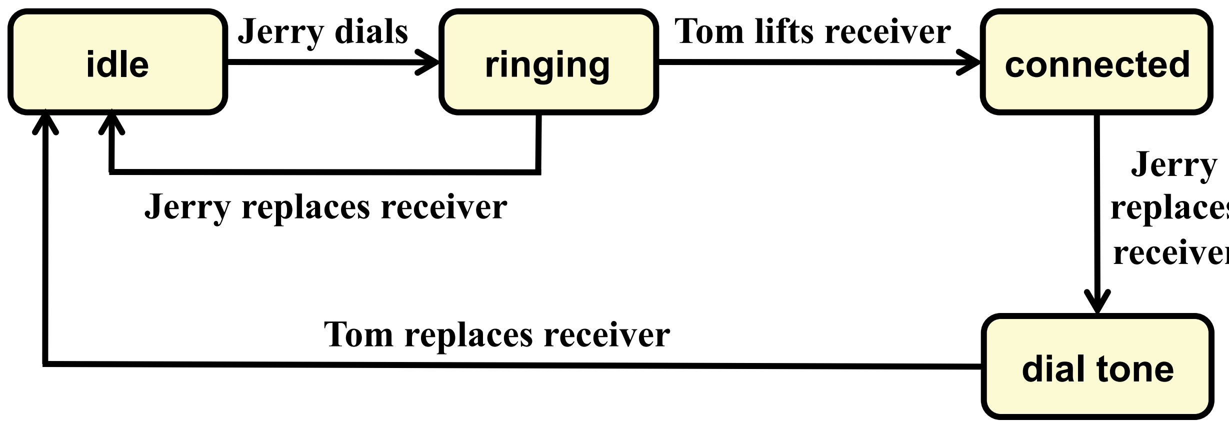
◯ True
◯ False

Are smart phones too smart?

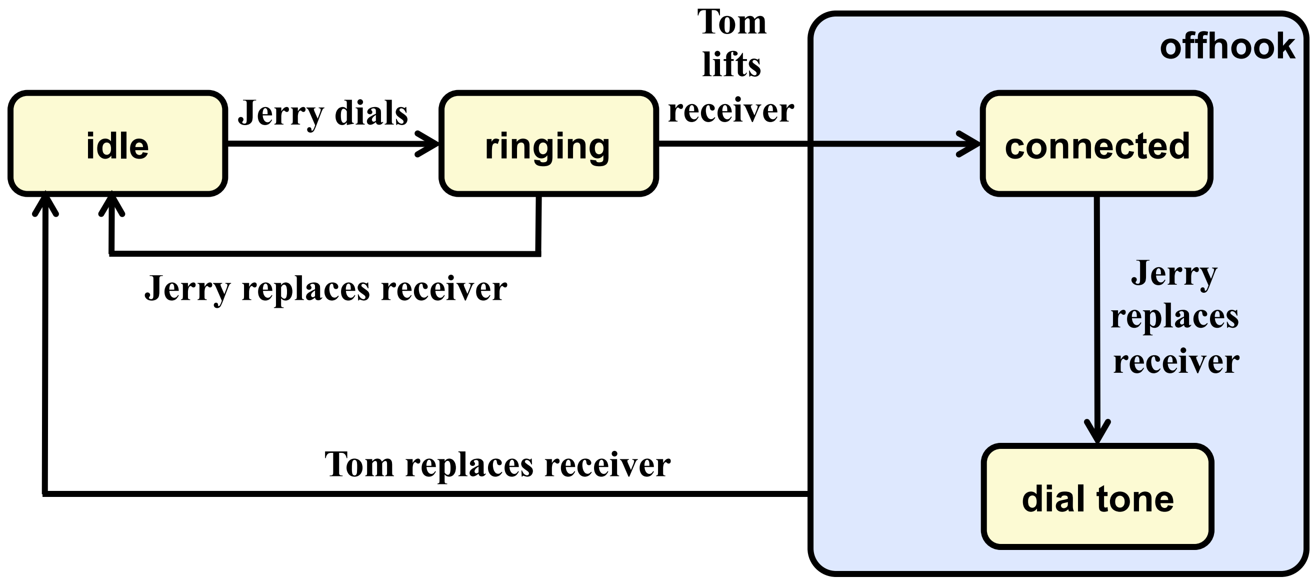


Behavioral mismatches ➔ bugs

Ana's model



Bob's model



Today's Take-Aways

- Requirements-based testing is about deriving a suite of tests (or test cases) from req.s that adequately exercise the software behavior without regard to the internal structure of the implementation
  - Model-based testing is commonly practiced
- To-do
  - Review today's slides
  - Quiz9 is due: this Wednesday (Oct 30)
  - Friday's class (Nov 1) will *start at* ***9:30am*** and we'll release ASN3

© 2024, Nan Niu