Software Quality

Attribution: some slides from Ian Cottingham, RAIK 383H

Interview - Ciera Jaspan

Google

Oregon State University

Software Quality - why it matters

vs.

Oregon State University

Software Quality Measurements

Software is measured by *quality* of the implementation

1. Sufficiency -
2. Robustness -
3. Reliability -
4. Flexibility -
5. Efficiency -
6. Scalability -
7. Reusability -
8. Security -

Oregon State University

Software Quality Measurements

Software is measured by the *quality* of the implementation

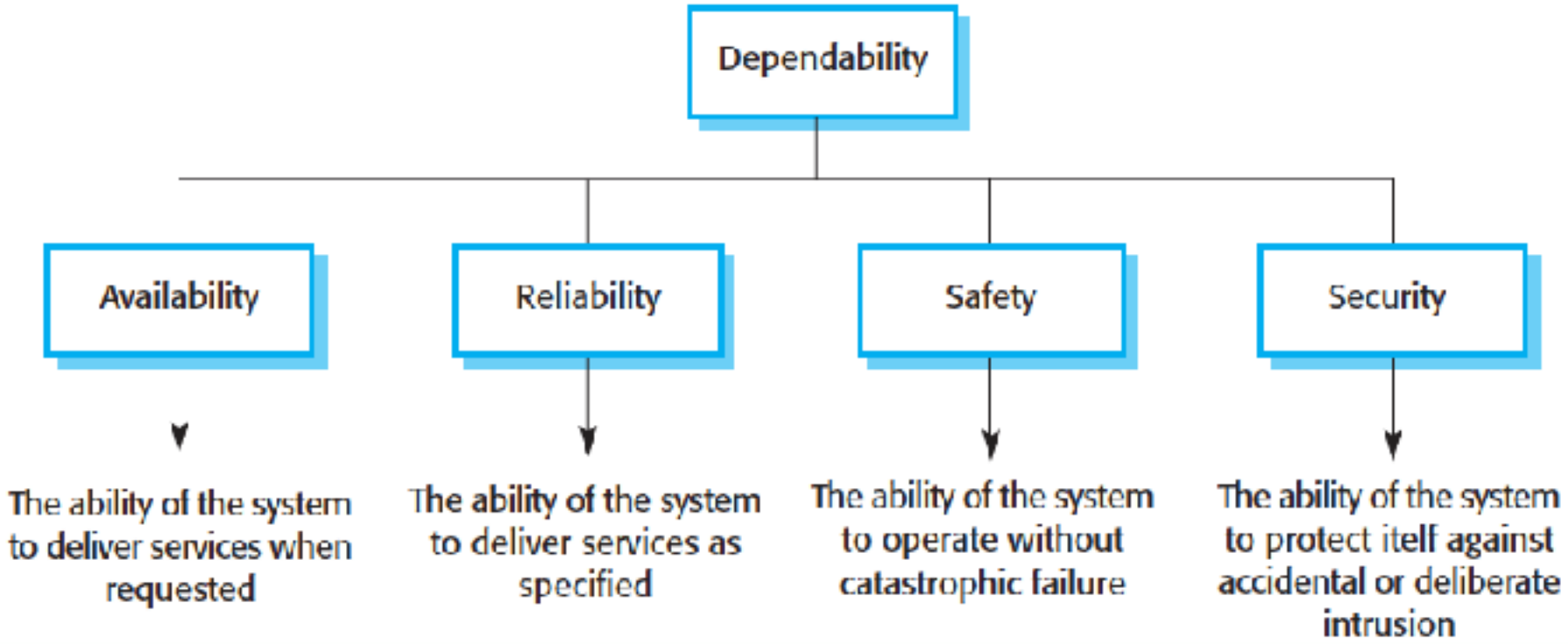
- Sufficiency - measure of how well a component satisfies design specifications
- Flexibility - measure of how adaptable to 'reasonable' changes a component is
- Robustness - measure of how well the component will recover from anomalous events
- Reliability - measure of the average amount of time between failures

Software Quality Measurements

- Efficiency - measure of how well a component satisfies speed or storage requirements within a specified margin
- Scalability - measure of the ability to use the component as scope increases
- Reusability - measure of how usable a component is in related applications without modification
- Security - measure of how resilient a component is to attack

Achieving Dependability

- Avoid the introduction of accidental errors when developing the system
- Design Verification and Validation processes that are effective at discovering residual defects in the system
- Configure the system correctly for its operating environment:
  - Include recovery mechanisms to assist in restoring normal operation after a failure.
- Develop process to support implementation quality



Oregon State University

Availability

Availability - the probability that a system at a point in time will be operational

- Availability is measured in terms of “9s”:
  - 90% availability (“one nine”) - 36.5 days of down time per year
  - 99% availability (“two nines”) - 3.65 days of down time per year
  - 99.9% availability (“three nines”) - 8.76 hours of down time per year
  - 99.99% availability (“four nines”) - 52.56 minutes of down time per year
  - 99.999% availability (“five nines”) - 5.25 minutes of down time per year
  - 99.9999% availability (“six nines” - 31.5 seconds of downtime per year

Oregon State University

Reliability

- The probability of failure free operation over a specified time period, in a given environment, for a given purpose.
- Measured as a rate of failure per some number of inputs:
  - 2 errors for every 1,000 inputs = a system that is 99.8% reliable (or has a failure rate of 0.002).

- Do all faults affect reliability?
- What does it mean for you – when writing test cases?

Oregon State University

Availability/ Reliability

- As availability or reliability requirements increases so does the cost; the curve grows exponentially
- Important to consider both properties
  - A system that is always on, but does not have sufficient (correct) results
  - A system that is up half the times, but always has correct results
- Evaluate your design, requirements, tests, and know the potential faults
- What about your project?

Oregon State University

Safety

- Safety critical: essential that the operation of the system is always safe
- Examples: control system for a nuclear reactor, navigation systems in planes, monitoring sensors for security systems, heart monitors, etc.

Safety / Reliability

Can a reliable system be unsafe?

- faults can be *hidden for long periods* of time and have catastrophic results even low occurrence rate (e.g., O-ring failure on rocket engines due to excessive cold),
- system specification can fail to *account for specific situations* that lead to serious errors in an otherwise reliable system (e.g., tsunami wipes out secondary power controls on a nuclear power plant),
- hardware failure or degradation can create anomalous states that *software can interpret incorrectly* (e.g, Ariane 5)
- users can generate inputs that individually are correct but when combined with state from other errors *introduce anomalous data states* (e.g., Therac 25)

Designing safe software requires significant verification effort

Oregon State University

Security

Ability of a system to protect itself from intrusion or attack leading to loss of data or services

- more common to consider than safety
- web-based or networked systems due to the exposure of the system to many users;

- Three mechanisms
  - threats to confidentiality of data,
  - threats to the integrity of data
  - threats to the availability of the system

Design and limit how system exposes data and maintains state

Oregon State University

Security Terms

- Asset - something of “value” that needs to be protected. Can be software or data;
  - value is relative - embarrassing pictures on Facebook
- Exposure - possible loss or harm realized from a security breach;
- Vulnerability - a weakness in software than can be exploited to cause loss or harm;
- Threat - a circumstance that has the potential to cause loss or harm;
- Attack - exploiting a vulnerability in a system;
- Control - a protective measure that reduces a vulnerability.

Oregon State University

Example

```
String sql_select = “select * from Grades where student_id = “+
request.getParameter(“student_id”);

ResultSet rs = conn.createStatement().executeQuery(sql_select);

...
```

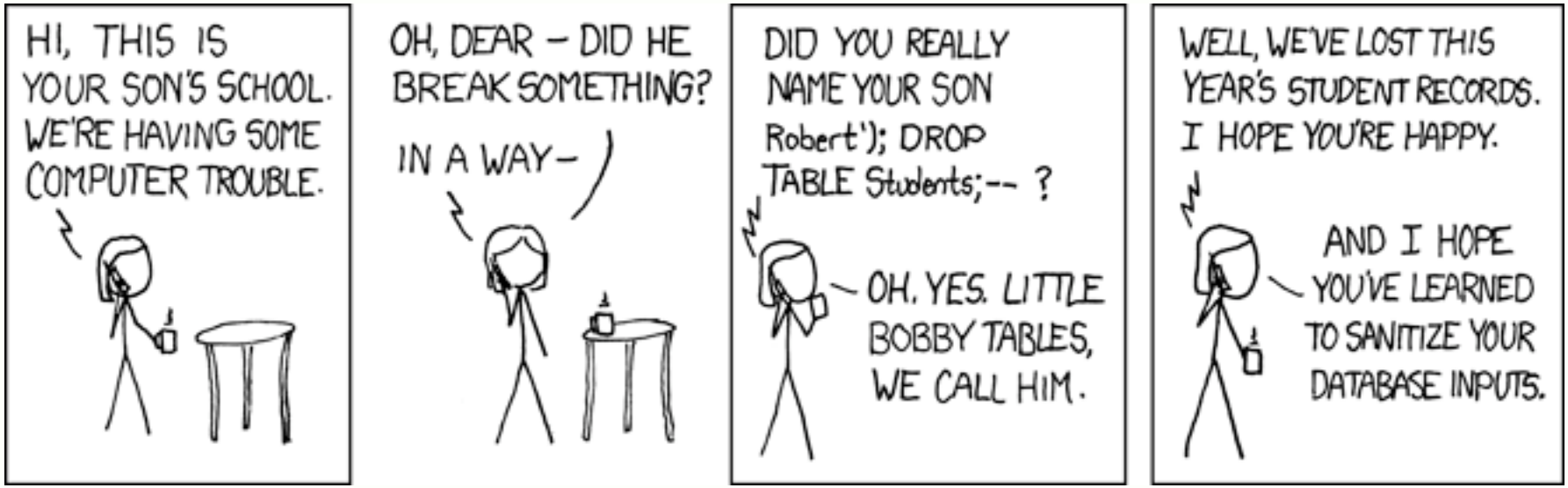
Identify the *assets, exposures, vulnerabilities, and possible attacks, threats, and controls*

Oregon State University

Example

- Asset - the grade database and its data,
- Exposure - data could be obtained or manipulated by an unauthorized user,
- Vulnerability - user input is passed unchecked to the database,
- Attack - the user could append sql strings to their input;
- Threat - the student_id parameter is “002323; select * from Grades” then the
  - second SQL statement could be executed, returning all grades.
  - any other student ID could be provided, etc. etc.
- Control – check for values before accepting the query or returning results

Sanitize your inputs!



Oregon State University

The weakest link ?

Quiz - Refactoring

Oregon State University