Software Quality

Thursday, November 29

Software Quality - why it matters

vs.

Measurements

Software is measured by quality of the implementation

Measurements

Software is measured by quality of the implementation

Sufficiency

Measurements

Software is measured by quality of the implementation

Sufficiency how well a component satisfies design specifications

Measurements

Software is measured by quality of the implementation

Sufficiency how well a component satisfies design specifications

Robustness

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	

Measurements

Software is measured by quality of the implementation

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

Measurements

Software is measured by quality of the implementation

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

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements
Scalability	

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements
Scalability	measure of the ability to use the component as scope increases

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements
Scalability	measure of the ability to use the component as scope increases
Reusability	

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements
Scalability	measure of the ability to use the component as scope increases
Reusability	how usable a component is in related applications without modification

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements
Scalability	measure of the ability to use the component as scope increases
Reusability	how usable a component is in related applications without modification
Security	

Measurements

Software is measured by quality of the implementation

Sufficiency	how well a component satisfies design specifications
Robustness	how well the component will recover from anomalous events
Reliability	the average amount of time between failures
Flexibility	how adaptable to ‘reasonable’ changes a component is
Efficiency	how well a component satisfies speed or storage requirements
Scalability	measure of the ability to use the component as scope increases
Reusability	how usable a component is in related applications without modification
Security	how resilient a component is to an 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

Achieving Dependability

Testing!

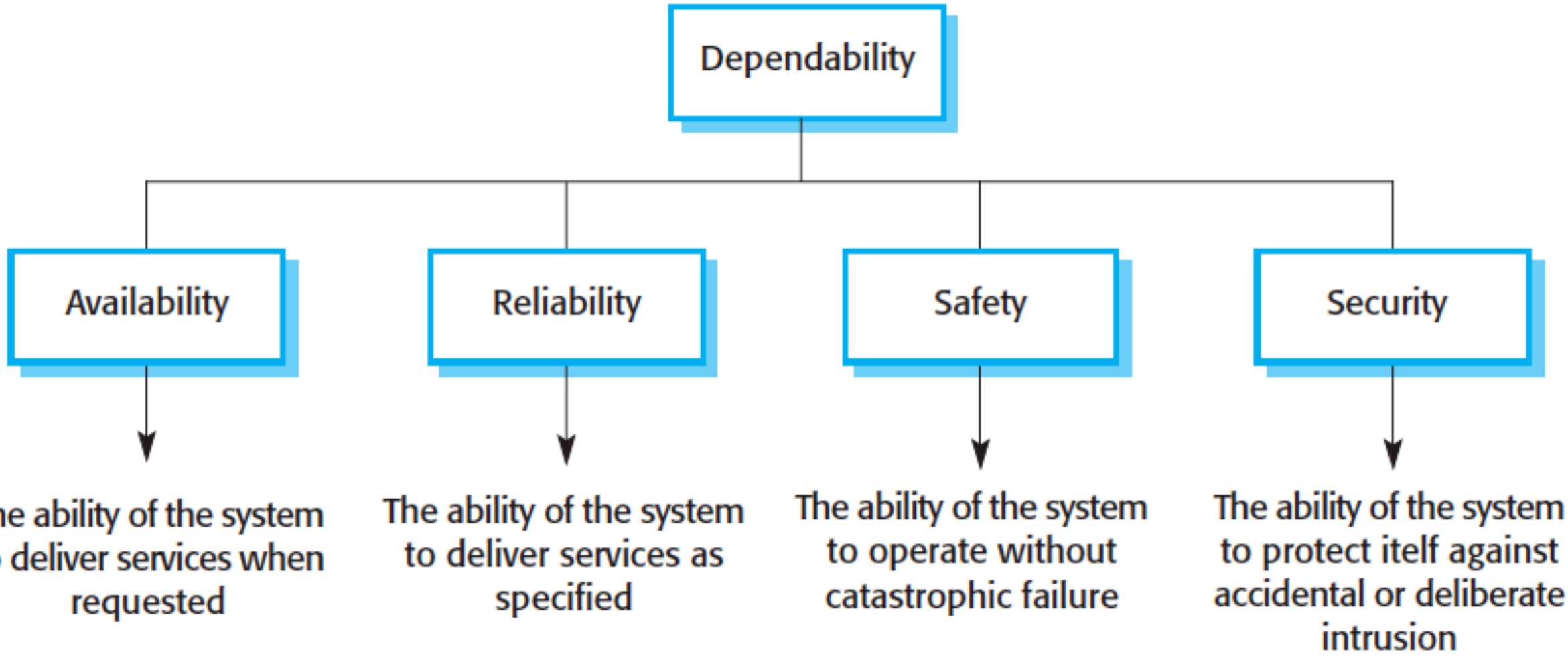
Write Unit Tests for each feature

Run all the tests after each change

Code reviews!

*"Given enough eyeballs, all **bugs** are **shallow."***

Linus' Law



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

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?

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?

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

Safety / Reliability

Can a reliable system be unsafe?

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

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

system specification can fail to account for specific situations that lead to serious errors in an otherwise reliable system

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

system specification can fail to account for specific situations that lead to serious errors in an otherwise reliable system

hardware failure or degradation can create anomalous states that software can interpret incorrectly

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

system specification can fail to account for specific situations that lead to serious errors in an otherwise reliable system

hardware failure or degradation can create anomalous states that software can interpret incorrectly

users can generate inputs that individually are correct but when combined with state from other errors introduce anomalous data states

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

system specification can fail to account for specific situations that lead to serious errors in an otherwise reliable system

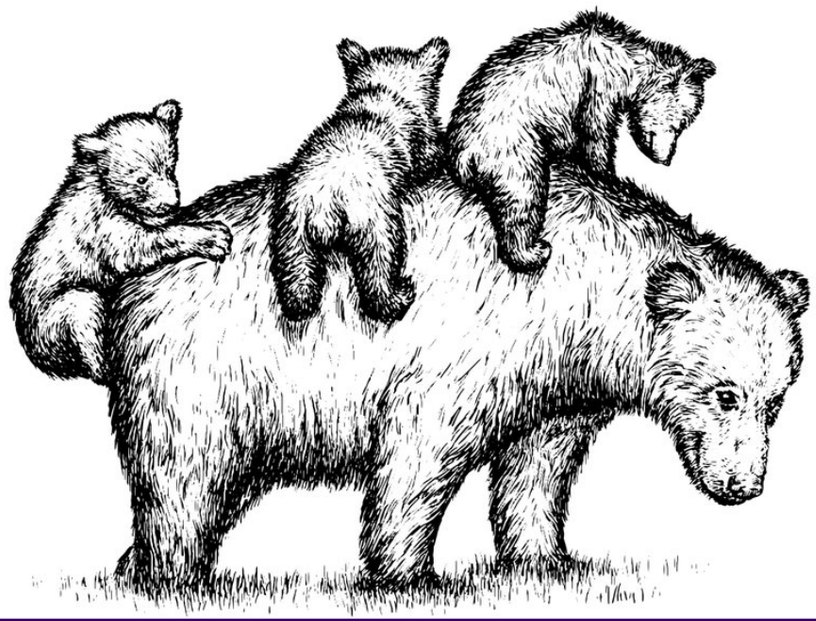
hardware failure or degradation can create anomalous states that software can interpret incorrectly

users can generate inputs that individually are correct but when omalous data

Designing safe software requires significant verification effort

Scalability

Getting the wrong idea from that conference talk you attended



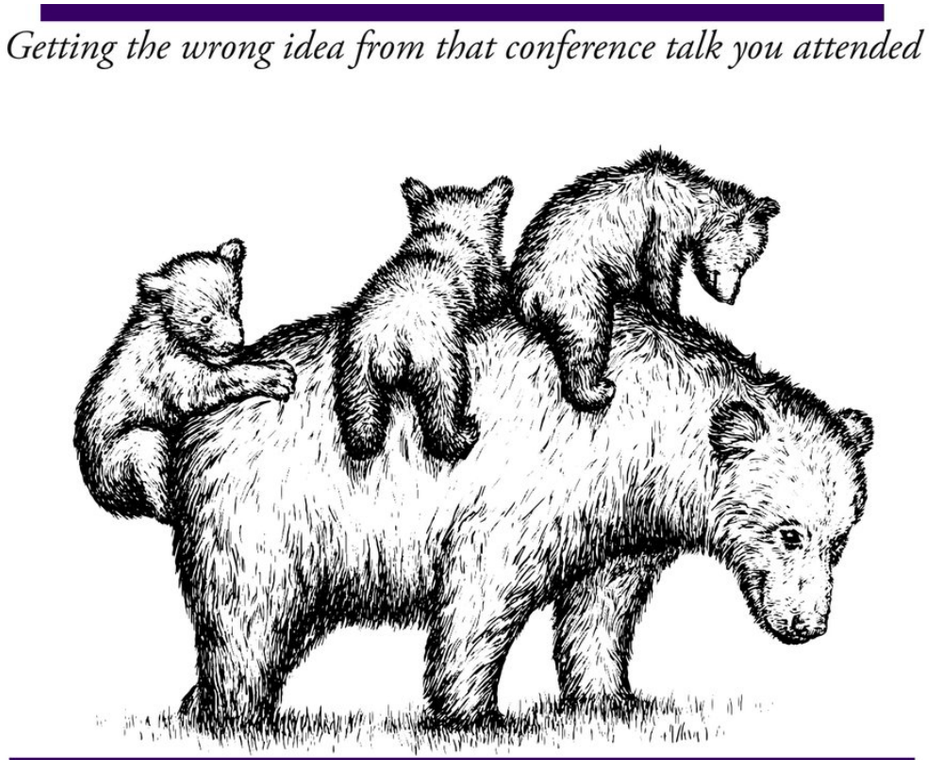
Solving Imaginary Scaling Issues

At Scale

O RLY?

@ThePracticalDev

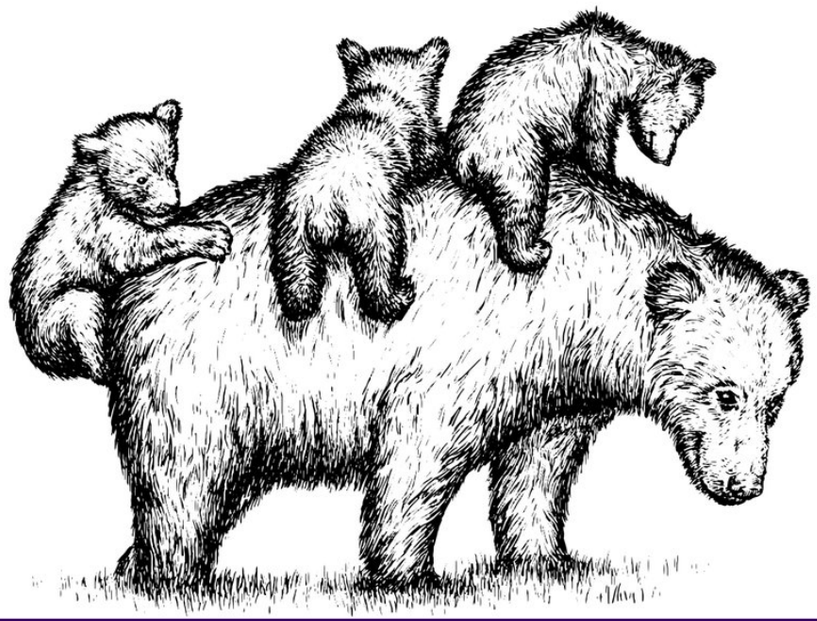
Scalability



Does it scale?

Scalability

Getting the wrong idea from that conference talk you attended



Solving Imaginary Scaling Issues

At Scale

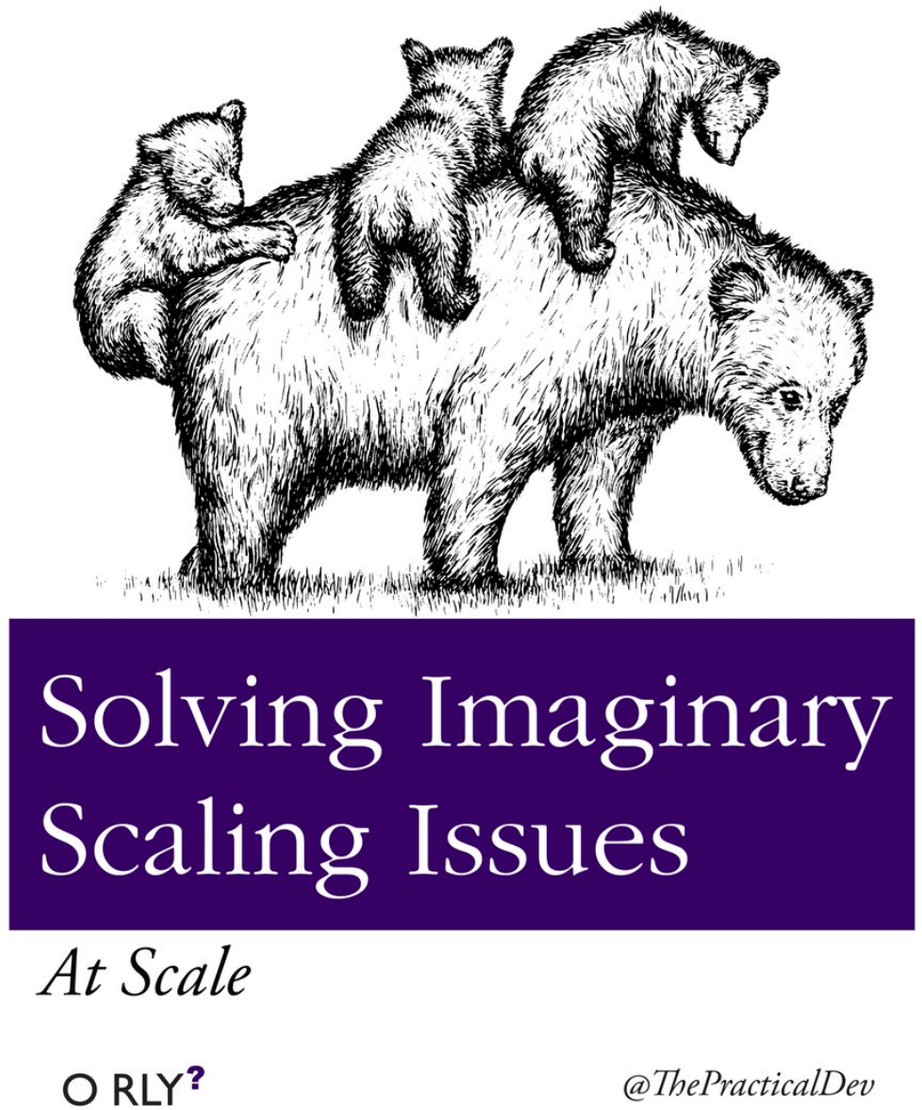
O RLY?

@ThePracticalDev

Does it scale?

Does it matter?

Scalability



Does it scale?

Does it matter?

Uneven loads

Scalability

Does it scale?

Does it matter?

Uneven loads

Verified trend-line

Security

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

More commonly considered than safety

Web-based or networked systems are more vulnerable due to the exposure of the system to many users;

Security

Three mechanisms

threats to **confidentiality** of data

threats to the **integrity** of data

threats to the **availability** of the system

Security

Three mechanisms

threats to **confidentiality** of data

threats to the **integrity** of data

threats to the **availability** of the system

Design and limit how the system exposes data and maintains state

Security Terms

Asset - something of “value” that needs to be protected. Can be software or data;

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.

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

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

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

...
```

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

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

...
```

Asset

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

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

...
```

Asset the grade database and its data

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

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

...
```

Asset the grade database and its data

Exposure

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

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

...
```

Asset the grade database and its data

Exposure data could be obtained or manipulated by an unauthorized user

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

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

...
```

Asset the grade database and its data

Exposure data could be obtained or manipulated by an unauthorized user

Vulnerability

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

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

...
```

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,

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

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

...
```

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	

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

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

...
```

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

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

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

...
```

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	

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

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

...
```

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

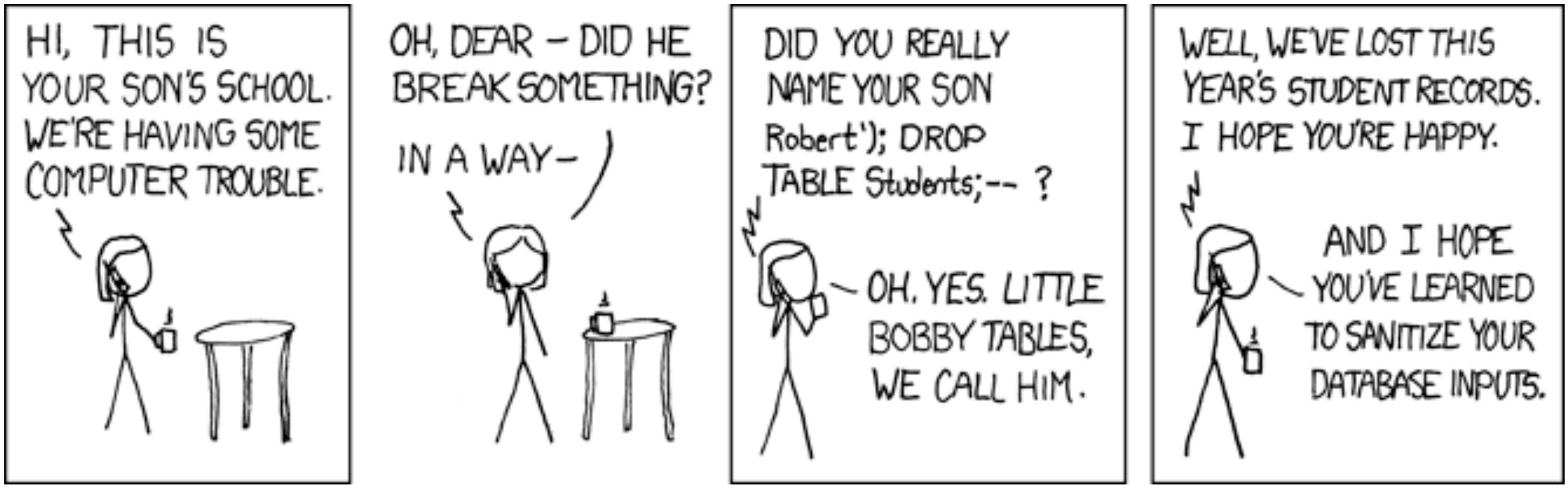
```
String sql_select = “select * from Grades where student_id = “+
request.getParameter(“student_id”);

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

...
```

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
Control	

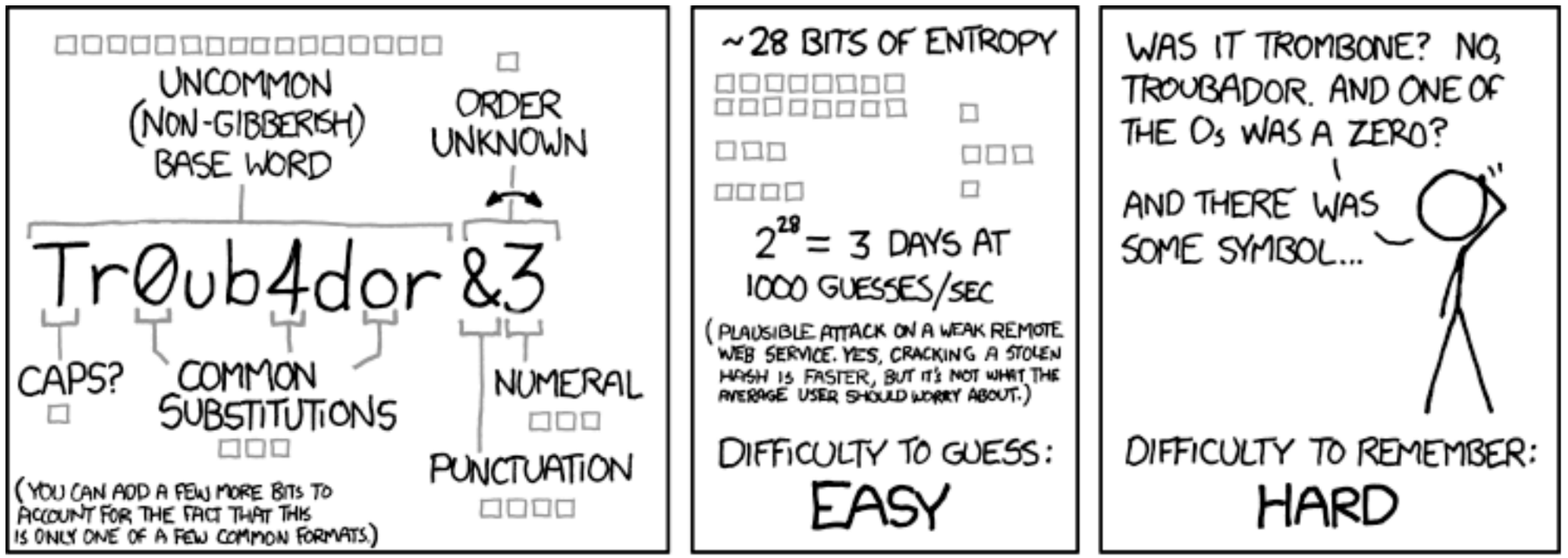
```
String sql_select = “select * from Grades where student_id = “+
request.getParameter(“student_id”);

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

...
```

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
Control	check for values before accepting the query or returning results

Sanitize your inputs!



The weakest link ?



What we covered

Java & OO

OO Design Basics

Design Patterns

Software Architecture

Unit tests

Code smells & Refactorings

What we covered

UI Design & Evaluation

Paper Prototypes

CRC Cards

UML Diagrams

SLDC

Project Management

We used

Java

Continuous Integration

Unit tests

HTML & CSS

Javascript