dependence & dependability

A Tribute To Michael Jackson
Vancouver · May 19, 2009

Daniel Jackson & Eunsuk Kang, MIT
must dogs wear shoes?
must dogs wear shoes?

focus on the real world
must dogs wear shoes?

focus on the real world

air traffic control, proton therapy, voting
must dogs wear shoes?

focus on the real world

*air traffic control*, *proton therapy*, voting

description before invention
must dogs wear shoes?

focus on the real world

*air traffic control, proton therapy, voting*

description before invention

*famous failures, explained*
must dogs wear shoes?

focus on the real world

air traffic control, proton therapy, voting

description before invention

famous failures, explained

beneficent difficulty
must dogs wear shoes?

focus on the real world
air traffic control, proton therapy, voting
description before invention
famous failures, explained
beneficent difficulty
this project especially
kemper arena, kansas city, 2007
kemper arena, 1979
what happened?

Levy & Salvadori, Why Buildings Fall Down
failure = flawed success story
Therac 25

AECL fault tree analysis (1983)
› did not include software
› \(P(\text{computer selects wrong energy}) = 10^{-11}\)

accidents (1985-87)
› massive overdoses cause death & injury

Leveson & Turner (1993)
› race conditions, lack of interlocks, etc
research goals

devise a notation for
› for analyzing design alternatives
› for justifying dependability
› for explaining failures

desiderata
› simple, intuitive, graphical
› support formal analysis
the notation
There probably isn't a best way to build the system, or even any major part of it; much more important is to avoid choosing a terrible way, and to have a clear division of responsibilities among the parts.

Butler Lampson
*Hints for computer system design* (1983)
key idea

inspired by Problem Frame diagrams

represent explicitly
› properties (requirements)
› components (domains)
› and their relationship
properties & components
a specification is a property
a specification is a property

a component may satisfy >1 property
a specification is a property

a component may satisfy >1 property

components can be justified independently but achieve a common goal
properties & components
properties & components

Property established by component and property of another component

P1 -> C1

P2 -> C2
properties & components

property established by component and property of another component

equivalent diagram, less familiar layout
an example: tracking stocks
track stocks with given set of ticker symbols and display message when move exceeds bound

AAPL: now 12295 prev hi: 12295, prev lo: 12289
IBM: now 10218 prev hi: 10218, prev lo: 10212
INTC: now 1550 prev hi: 1552, prev lo: 1550
public class QuoteApp {
    public static void main(String[] args) throws Exception {
        Timer timer = new Timer();
        for (String ticker: args)
            timer.schedule(new Tracker(ticker), 0, 10000);
    }
}

public class Tracker extends TimerTask {
    String ticker;
    int hi = 0; int lo = Integer.MAX_VALUE;
    int MOVE = 1;

    public Tracker (String t) {ticker = t;}
    public void run () {
        int q = Quoter.getQuote(ticker);
        hi = Math.max(hi, q);
        lo = Math.min(lo, q);
        if (hi - lo > MOVE) {
            System.out.println(ticker + "\t now " + q + " prev hi: " + hi + ", prev lo: " + lo);
            hi = lo = q;
        }
    }
}

public class Quoter {
    public static int getQuote (String ticker) {
        URL url = new URL("http://finance.yahoo.com/d/quotes.csv?s=" + ticker + "&f=l1");
        String p = new BufferedReader(new InputStreamReader(url.openStream())).readLine();
        return (int) (Float.valueOf(p) * 100);
    }
}
uses relation

- Timer
  - QuoteApp
  - TimerTask
    - Tracker
      - Yahoo Server
      - Quoter
        - System.out
          - java.net
dependency diagram

schedule(t,p) results in call to t.run every p secs

make Tracker t for each ticker and call t.run() every k secs

Tracker(s) makes Tracker for ticker s

for all tickers display message if big move within last k secs

run() displays message if big move

http-get for u?s returns last price of s

getQuote(s) returns last price of s

java library specs

println writes to console

Yahoo Server

Quoter

java.net java.io

System.out
finding a property’s support

schedule(t,p) results in call to t.run every p secs

Timer

make Tracker t for each ticker and call t.run() every k secs

QuoteApp

Tracker(s) makes Tracker for ticker s

Tracker

Tracker(s) makes Tracker for ticker s

Tracker

for all tickers display message if big move within last k secs

run() displays message if big move

http-get for u?s returns last price of s

Yahoo Server

getQuote(s) returns last price of s

Quoter

java library specs

java.net java.io

println writes to console

System.out
finding a property’s support

- `schedule(t, p)` results in call to `t.run` every p secs
- `make Tracker t for each ticker and call t.run()` every k secs
- `Tracker(s)` makes Tracker for ticker s
- `run()` displays message if big move within last k secs
- `http-get for u?s returns last price of s`
- `getQuote(s) returns last price of s`
- `java library specs`
- `println writes to console`
- `Yahoo Server`
- `Quoter`
- `System.out`
- `java.net java.io`
finding a component’s impact

- **schedule(t,p)** results in call to t.run every p secs
  - Timer

- **make Tracker t for each ticker and call t.run() every k secs**
  - QuoteApp

- **Tracker(s) makes Tracker for ticker s**
  - Tracker

- **run() displays message if big move within last k secs**
  - Yahoo Server
    - http-get for u?s returns last price of s
  - Quoter
    - getQuote(s) returns last price of s
  - java.library specs
  - println writes to console
    - System.out
    - java.net
    - java.io
finding a component's impact

**schedule(t,p)** results in call to **t.run** every p secs

**make Tracker t for each ticker and call t.run() every k secs**

**Tracker(s) makes Tracker for ticker s**

**run() displays message if big move within last k secs**

- Timer
- QuoteApp
- Tracker

**http-get for u?s returns last price of s**

**getQuote(s) returns last price of s**

- Yahoo Server
- Quoter

**java library specs**

- java.net
- java.io

**println writes to console**

- System.out
explaining a flaw

schedule(t, p) results in call to t.run every p secs

make Tracker t for each ticker and call t.run() every k secs

Tracker(s) makes Tracker for ticker s

run() displays message if big move within last k secs

Timer

QuoteApp

Tracker
explaining a flaw

- **schedule(t,p)** results in call to `t.run()` every p secs
- **make Tracker t for each ticker and call** `t.run()` every k secs
- **Tracker(s)** makes Tracker for ticker s
- run() displays message if big move

**Timer**

**QuoteApp**

**Tracker**

**run completes quickly**
five failures, explained
apple file vault

securing files
  › make secure volume
  › transfer files to it

what happens to old copies?
  › unlinked but not erased!
apple file vault

\[
\text{move}(f) \Rightarrow \text{block}'(f).\text{contents}' \text{ is secure}
\]

\[
\text{block}'(f) = \text{block} \oplus f \rightarrow b
\]

\[
\text{block}'(f).\text{contents}' = \text{enc}(\text{block}(f).\text{contents})
\]

Move ToVault

Unlink

Create Encrypted

wrong property

from Simson Garfinkel, 2004
Move ToVault

move(f) ⇒ block'(f).contents' is secure

block' = block ⊕ f→b

Unlink

block'(f).contents' = enc(block(f).contents)

Create Encrypted

Wrong property

from Simson Garfinkel, 2004
Move ToVault

\[ move(f) \Rightarrow block(f).\text{contents} \text{ is inaccessible} \]

\[ block'(f) = block \oplus f \rightarrow b \]

\[ block'(f).\text{contents}' = \text{enc}(block(f).\text{contents}) \]

Unlink

Create Encrypted

wrong property

from Simson Garfinkel, 2004
apple file vault

move(f) \Rightarrow block(f).contents is inaccessible

\begin{align*}
\text{move(f) } & \Rightarrow b' = \text{block } \oplus f \\
\text{block}'(f).\text{contents}' & = \text{enc(block}(f).\text{contents}) \\
\text{block}(f).\text{contents}' & = 0
\end{align*}

- Unlink
- Create Encrypted
- Delete

wrong property

from Simson Garfinkel, 2004
insecure ATMs

a broken PIN scheme
› hash of PIN stored on card
› ATM just checks entered PIN against it to access another account
› just change account number on card!
insecure ATMs

\[
\begin{align*}
\text{Card} & \quad \text{User} & \quad \text{ATM} \\
\text{card}.h = & \quad \text{hash(PIN(acct(u)))} & \quad \text{u enters PIN p} \\
\quad a \neq \text{acct}(u) \quad \Rightarrow & \quad \text{pin} \neq \text{PIN}(a) & \quad \text{access}(u, \text{card}.a) \leftrightarrow \\
\quad \text{¬access}(u, a) & \quad \text{hash}(p) = \text{card}.h
\end{align*}
\]

problem: bad analysis

from Ross Anderson, Why Cryptosystems Fail, 1994
insecure ATMs

problem: bad analysis

from Ross Anderson, Why Cryptosystems Fail, 1994
insecure ATMs

\[ a \neq \text{acct}(u) \Rightarrow \neg \text{access}(u,a) \]

\[ u \text{ enters PIN } p \]
\[ a \neq \text{acct}(u) \Rightarrow p \neq \text{PIN}(a) \]

\[ \text{access}(u, \text{card.a}) \Leftrightarrow p = \text{hash}(\text{card.a}) \]

Card \hspace{2cm} User \hspace{2cm} ATM

\textit{problem: bad analysis}

from Ross Anderson, Why Cryptosystems Fail, 1994
Airbus A320 (1993)

landing in Warsaw
› overrun runway
› pilot & passenger died

explanation
› aquaplaned, so no wheel rotation
› reverse thrust was disabled for 9s
Airbus A320 (1993)

**Problem:** Incorrect environmental assumption

From Michael Jackson, Peter Ladkin
Airbus A320 (1993)

**problem: incorrect environmental assumption**

from Michael Jackson, Peter Ladkin
Panama City (2001)
radiation treatment planning software overexposes 20, killing at least 9
Panama City (2001)

radiation treatment planning software overexposes 20, killing at least 9

dose = D
Panama City (2001)

radiation treatment planning software overexposes 20, killing at least 9
Panama Radiotherapy, 2001

Problem: component fails to meet spec

From IAEA Investigation, 2001
Panama Radiotherapy, 2001

Problem: component fails to meet spec

\[
\text{dose} = \text{prescribed}
\]

\[
\text{beam} = \text{prescribed} \times \text{cSection}
\]

\[
\text{cSection} = \text{area within outline}
\]

Set Beam Energy

enter outline

Nurse

Beam Assembly

\[
\text{dose} = \text{beam} / \text{cSection}
\]
Given [the input] that was given, our system calculated the correct amount, the correct dose. It was an unexpected result. And, if [the staff in Panama] had checked, they would have found an unexpected result.

Mick Conley, Multidata
AT&T outage (1990)

- Failure in 5ESS switch
  - for 9 hours
  - 148 calls made
  - about 50% dropped

Explanation
  - Bug in recent upgrade
  - Caused knock-on crashes
AT&T outage (1990)

problem: feature interaction

from RISKS Forum
AT&T outage (1990)

- recover from transient errors
  - corrupt state \Rightarrow reboot
    - Monitor

- maintain link state
  - reboot \Rightarrow sendAlive
  - recvAlive \Rightarrow update state
    - Reboot Sequence
    - Track Neighbours

problem: feature interaction

from RISKS Forum
AT&T outage (1990)

**Problem**: feature interaction

- **recovery from transient errors**
  - **corrupt state** \(\Rightarrow\) **reboot**
  - Monitor

- **maintain link state**
  - **reboot** \(\Rightarrow\) **sendAlive**
  - Reboot Sequence

- **recvAlive** \(\Rightarrow\) **update state**
  - Track Neighbours

*from RISKs Forum*
plus ça change...

Phone-company technicians traced the problem to a single ‘failure of logic’ in the computer programs that route calls through the AT&T network.

AT&T Network Outage, 1990

We’ve now determined that message corruption was the cause of the server-to-server communication problems ... a handful of messages ... had a single bit corrupted

Amazon S3 Outage, 2009
formalization
observer in alloy

hard part: expressing invocations
contract SubjectView

    Subject supports [  
        value : Value  
        SetValue(val:Value) ↦ Δvalue {value = val}; Notify()  
        GetValue() : Value ↦ return value  
        Notify() ↦ {|| v : v ∈ Views : v→Update()}  
    ]

    AttachView(v:View) ↦ {v ∈ Views}  
    DetachView(v:View) ↦ {v ∉ Views}

Views : Set(View) where each View supports [  
    Update() ↦ Draw()  
    Draw() ↦ Subject→GetValue() {View reflects Subject.value}  
    SetSubject(s:Subject) ↦ {Subject = s}  
]

invariant

        Subject.SetValue(val) ↦ (∀v : v ∈ Views : v reflects Subject.value}

instantiation

(|| v : v ∈ Views : (Subject→AttachView(v) || v→SetSubject(Subject)) )

end contract
modelling invocation

pred control (invokes: Event -> Event) {
  all u: Update | let pre = u.before |
  all o: u.receiver.observers.pre |
  some n: u.invokes & Notify |
      n.subject = u.receiver and n.receiver = o
  all n: Notify |
     some d: n.invokes & Display |
      d.receiver = n.receiver and d.subject = n.subject
}

explicit events with invocation constraints
related work

axiomatic design  Suh, 2001
  › spec/design parameters

design structure matrix  Steward; Eppinger; Baldwin/Clark; Lattix
  › topological sort of uses

evolvability analysis  Sullivan et al
  › derive DSM from constraints on parameters

behavioral compositions  Helm, Holland & Gangopadhyay, 1990
  › properties due to role in contract pattern
how to think like michael jackson?

an answer to Pamela Zave’s question
how to think like michael jackson?

an answer to Pamela Zave’s question

drink much tea
how to think like michael jackson?

an answer to Pamela Zave’s question

drink much tea

take long baths
how to think like michael jackson?

an answer to Pamela Zave’s question

drink much tea

take long baths

always wear a tie