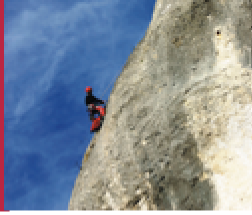
why is software so hard? and what can we do about it?

Daniel Jackson
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA

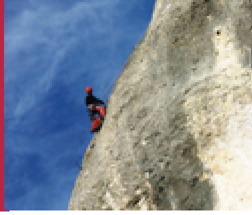
Accenture · India Delivery Center · November 29, 2007

how’s our personal software?

software warranties, 1987

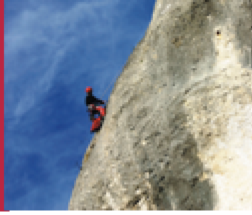


software warranties, 1987



"Cosmotronic Software Unlimited Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error-free. However, Cosmotronic Software Unlimited Inc. warrants the diskette(s) on which the program is furnished to be of black color and square shape under normal use for a period of ninety (90) days from the date of purchase."

software warranties, 1987

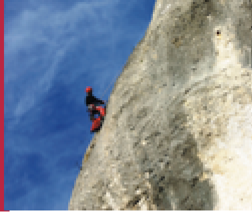


“Cosmotronic Software Unlimited Inc. does not warrant that the functions contained in the program will meet your requirements or that the operation of the program will be uninterrupted or error-free. However, Cosmotronic Software Unlimited Inc. warrants the diskette(s) on which the program is furnished to be of black color and square shape under normal use for a period of ninety (90) days from the date of purchase.”

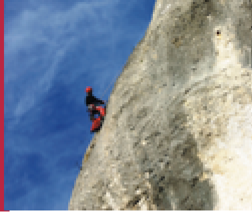
“We don't claim Interactive EasyFlow is good for anything ... if you think it is, great, but it's up to you to decide. If Interactive EasyFlow doesn't work: tough. If you lose a million because Interactive EasyFlow messes up, it's you that's out of the million, not us. If you don't like this disclaimer: tough. We reserve the right to do the absolute minimum provided by law, up to and including nothing. This is basically the same disclaimer that comes with all software packages, but ours is in plain English and theirs is in legalese.”

ACM Software Engineering Notes, Vol. 12, No. 3, 1987

software warranties, 2007



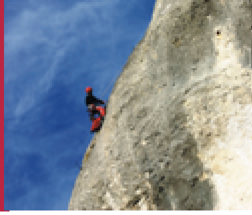
software warranties, 2007



Apple

"Except for the limited warranty on media ... software is provided "as is", with all faults and without warranty of any kind..."

software warranties, 2007



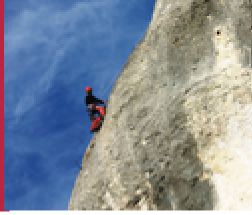
Apple

"Except for the limited warranty on media ... software is provided "as is", with all faults and without warranty of any kind..."

Google

"as is, with no warranties whatsoever"

software warranties, 2007



Apple

"Except for the limited warranty on media ... software is provided "as is", with all faults and without warranty of any kind..."

Google

"as is, with no warranties whatsoever"

Microsoft

"substantially in accordance with the accompanying materials, for a period of 90 days..."

is your PC secure?

typical patch size

- 100MB

typical time to download

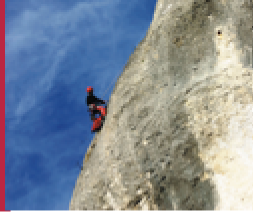
- 10 minutes

average time to infection*

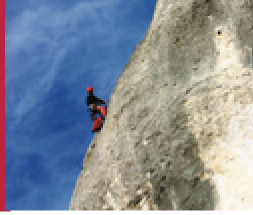
- 4 minutes

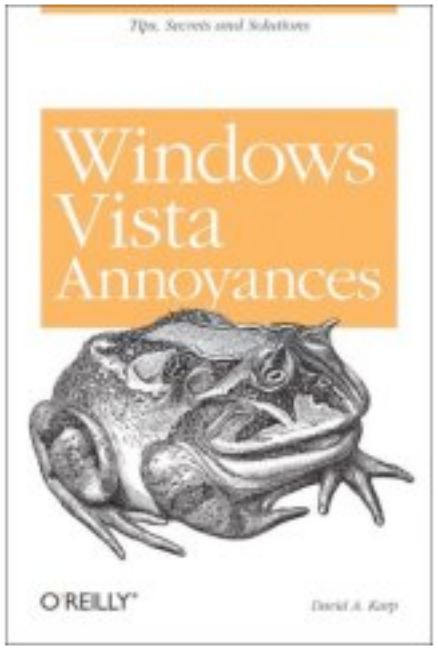
* [Windows XP, default firewall settings] Unprotected PCs Fall To Hacker Bots In Just Four Minutes
Gregg Keizer; Nov 30, 2004; http://www.techweb.com/wire/security/54201306
From: Security Absurdity: The Complete, Unquestionable, And Total Failure of Information Security
Noam Eppel; http://securityabsurdity.com

we love our operating systems

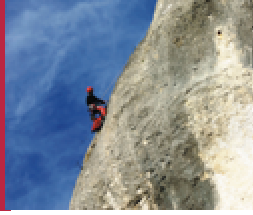


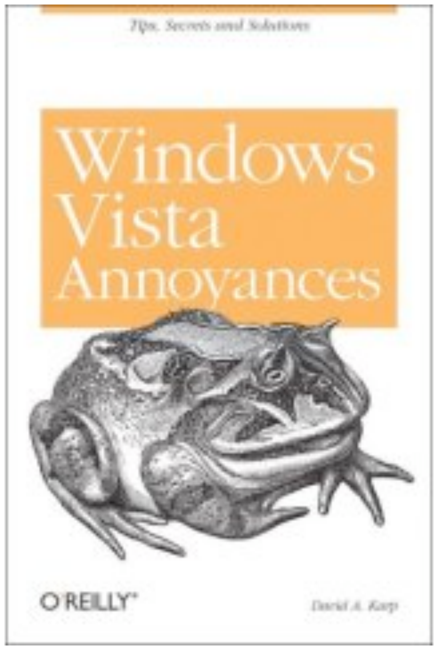
we love our operating systems

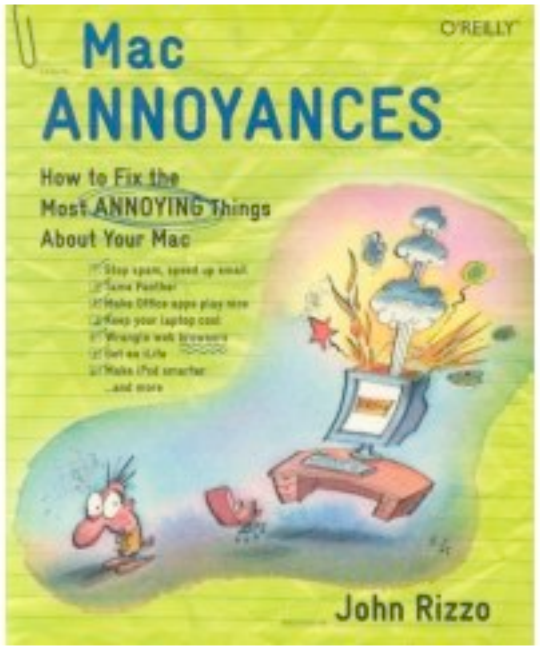




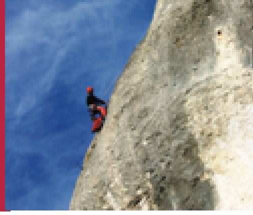
we love our operating systems

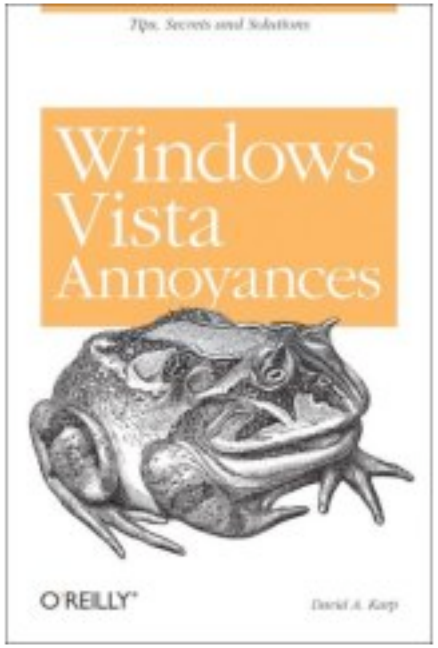


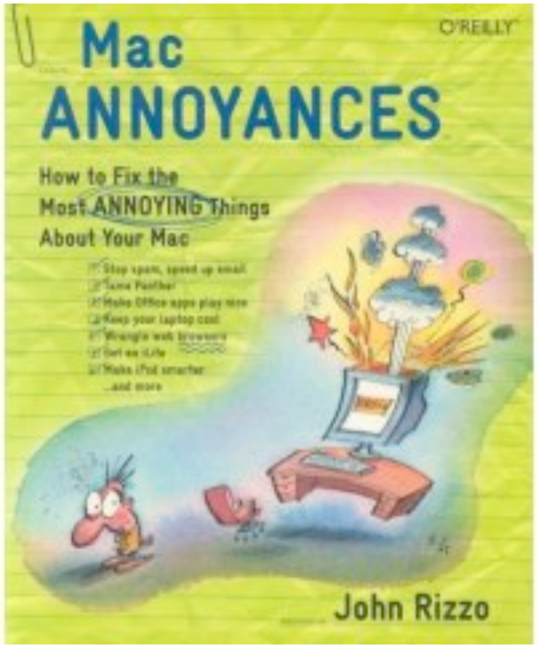


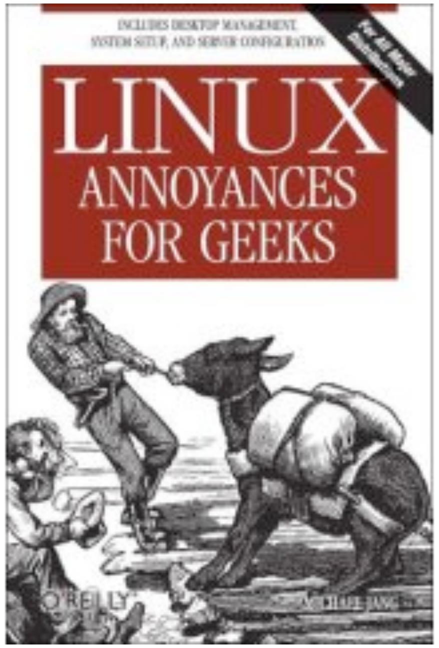


we love our operating systems



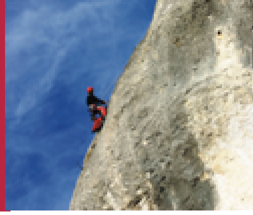


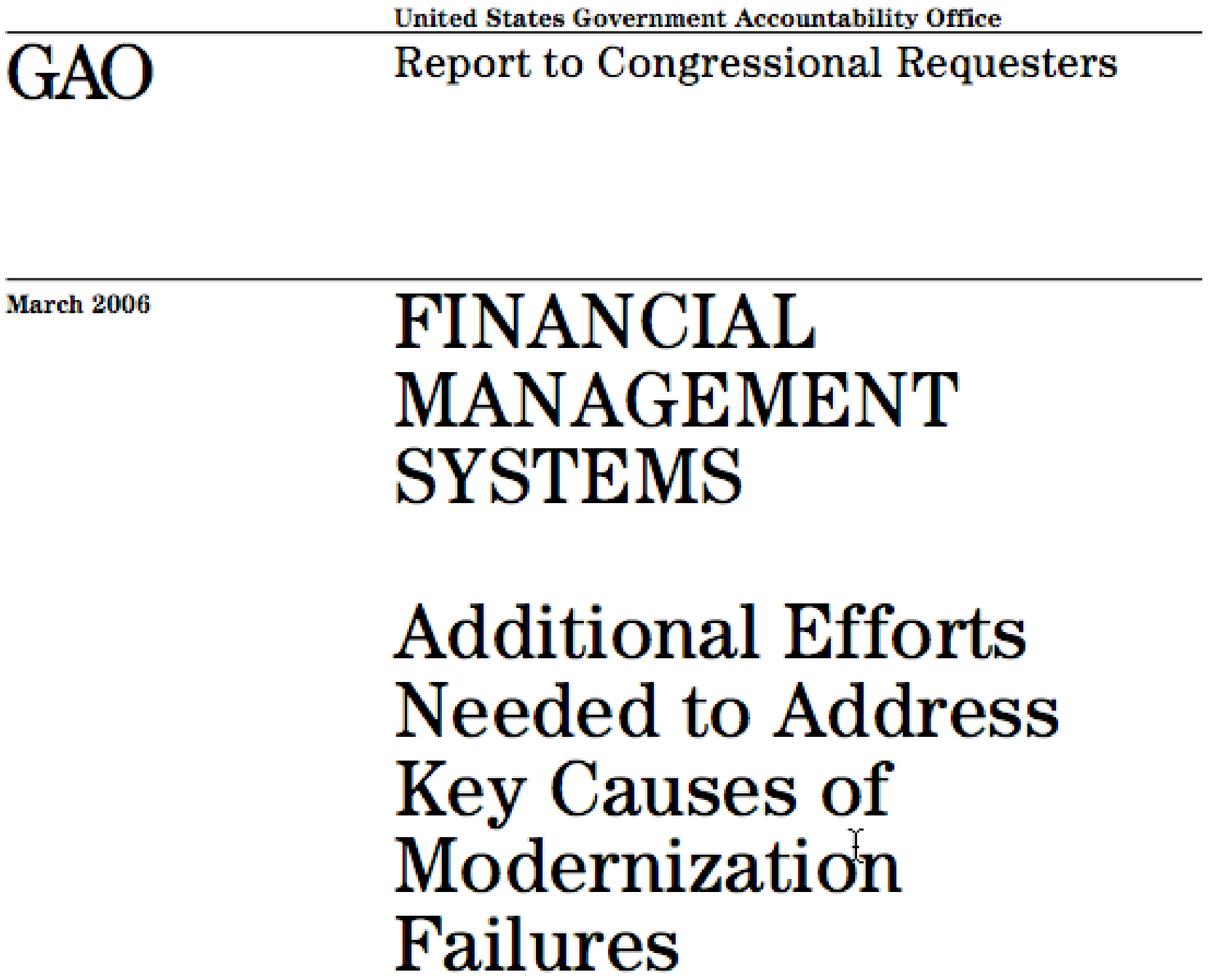



INCLUDES DESKTOP MANAGEMENT, SYSTEM SETUP, AND SERVER CONFIGURATION
For All Major Distributions
LINUX ANNOYANCES FOR GEEKS
O'REILLY
MICHAEL JANG

maybe government’s
doing better?

US government report, 2006





United States Government Accountability Office

GAO

Report to Congressional Requesters

March 2006

FINANCIAL MANAGEMENT SYSTEMS

Additional Efforts Needed to Address Key Causes of Modernization Failures

sample failures

Navy enterprise resource planning

- $1B wasted on systems that don't interoperate

NASA financial systems

- after 12 years and $120M spent, on third attempt expected to cost $1B
- still cannot produce auditable financial statements

Department of Veterans' Affairs

- supplies not available for patients due to bad inventory control
- implementation halted after spending $250M

FBI modernization attempts

reacting to 9/11

- had to send photos of suspected hijackers by fax
- no PCs for most employees, no secure email for images

Trilogy

- new network, thousands of PCs, software system ("VCF")
- contract awarded to SAIC

National Research Council report, 2004

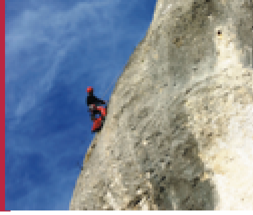
- agents can't take copies of cases into the field
- no bookmarking or history to help navigation, no sorting

outcome

- $600M later, no system; Sentinel ($425M) planned for 2009

maybe critical systems
are better?

runaway cannons

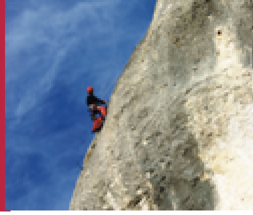


South Africa, October 2007

- antiaircraft cannon kills 9 soldiers and injures 14 others
- cause not known, but software suspected

http://blog.wired.com/defense/2007/10/robot-cannon-ki.html

air-traffic control



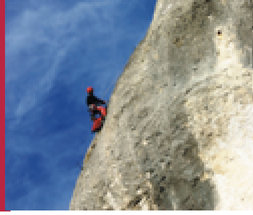
A radar system that was supposed to warn low-flying planes of nearby obstacles was plagued with problems and fixed nationwide only after a 1997 fatal airplane crash on Guam, according to a published report. In some cases, programming errors caused the Minimum Safe-Altitude Warning system not to operate over wide areas, including near busy airports such as those in Chicago and Dallas-Ft. Worth. In other cases, **false alarms were so numerous that air traffic controllers placed cardboard over warning speakers to silence the noise**. The Federal Aviation Administration was warned about the trouble after a business jet crashed outside Washington in 1994, but it did not take decisive action to resolve it until after a Korean Air jumbo jet slammed into a hill on approach to Guam in August 1997, killing 228.

AP, Oct 1999; http://ns.gov.gu/guam/indexmain.html

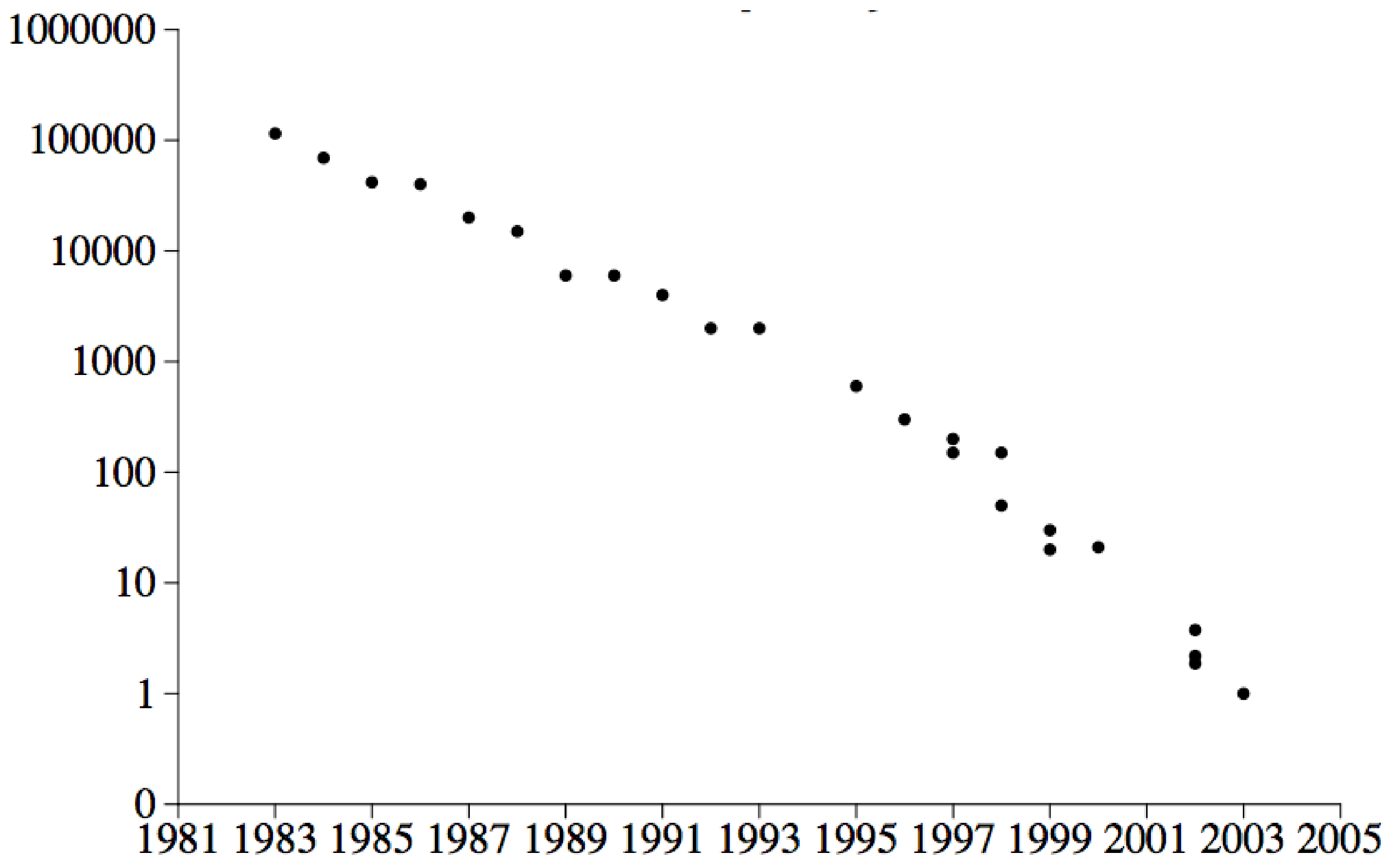
most aviation deaths from
"controlled flight into terrain"

how did we get here?
dtech/dt and criticality creep

storage costs

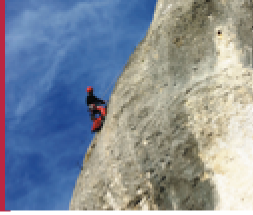


magnetic disks, US$/gigabyte

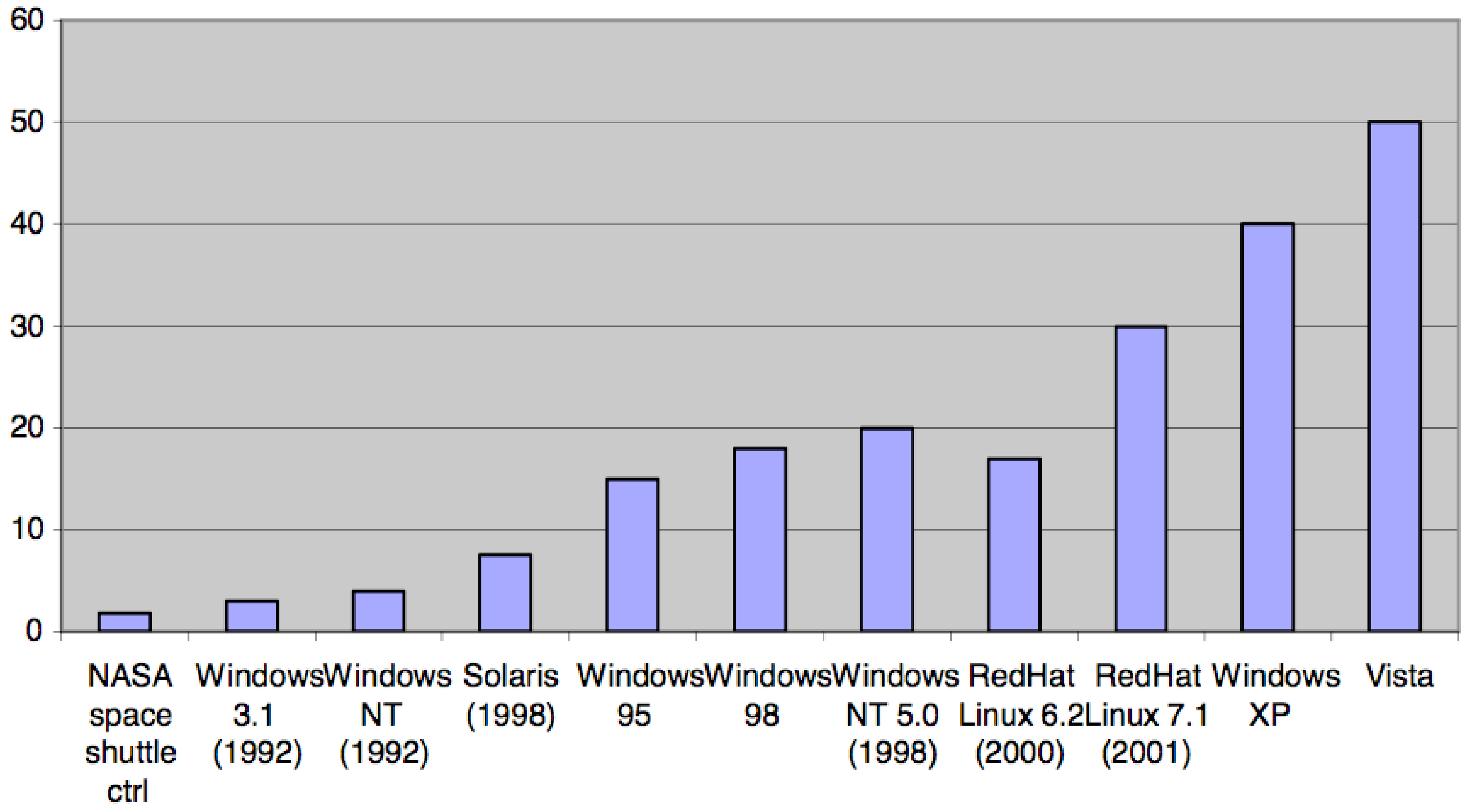


from Frans Kaashoek and Jerome Saltzer, *Topics in the Engineering of Computer Systems*, to appear.

operating system growth



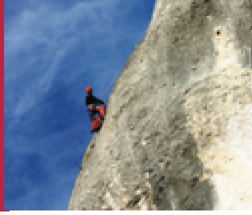
size in millions of lines of code



from Frans Kaashoek and Jerome Saltzer, *Topics in the Engineering of Computer Systems*, to appear.

texas A&M bonfire

bonfire history



traditional began in 1928

- small bonfire at annual football game

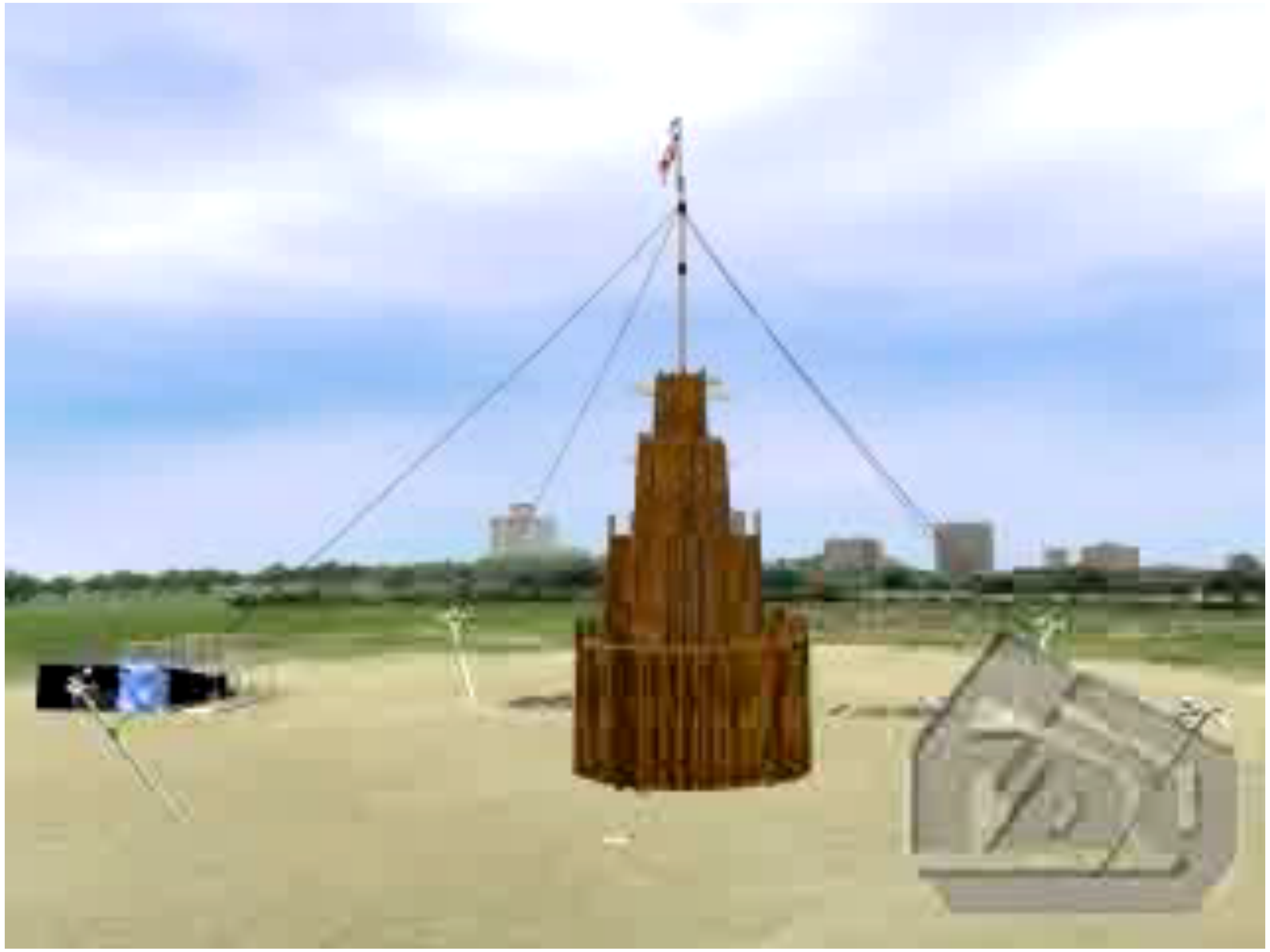
grew in size and complexity each year

- in 1990's required crane to erect

November 18, 1999

- collapsed killing 12 people

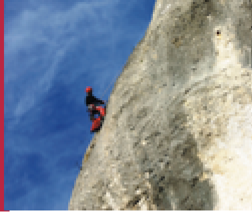
the collapse



http://www.fayengineering.com/structural.html fundamental challenges:

context, state space, coupling

software as system component



a software system is a component

- interacts with physical environment
- and organizational context of operators & users

sources of defects

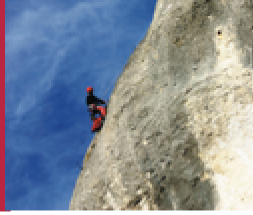
- < 3% of software failures due to bugs in code
- >90% from poor understanding of requirements

consequences

- requirements analysis is critical
- not just function, also assumptions

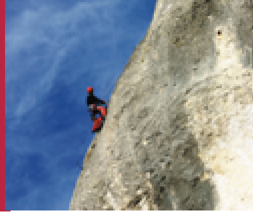
environmental assumptions

environmental assumptions



what happened

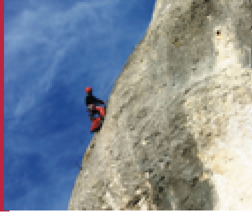
environmental assumptions



what happened

- Airbus A320, Warsaw 1993

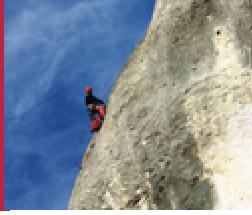
environmental assumptions



what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway

environmental assumptions



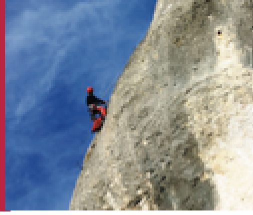
what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work

environmental assumptions

what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled



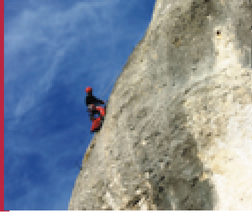
environmental assumptions

what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

why

environmental assumptions



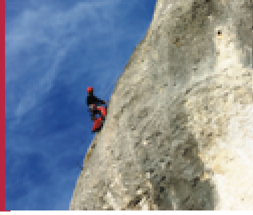
what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

why

airborne ⇔ disabled

environmental assumptions



what happened

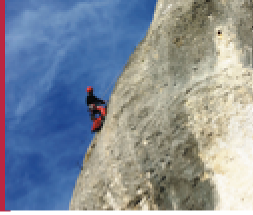
- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

why

airborne ⇔ disabled

airborne ⇔ not WheelPulse ⇔ disabled

environmental assumptions



what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

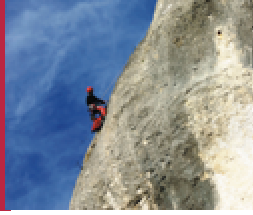
why

airborne ⇔ disabled

airborne ⇔ not WheelPulse ⇔ disabled

ENV

environmental assumptions



what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

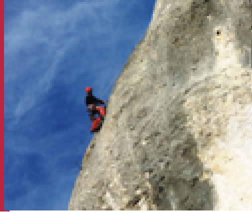
why

airborne ⇔ disabled

airborne ⇔ not WheelPulse ⇔ disabled

ENV MACHINE

environmental assumptions



what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

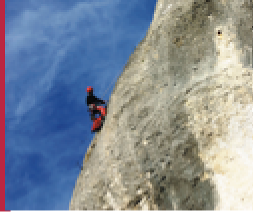
why

airborne ⇔ disabled

airborne ⇔ not WheelPulse ⇔ disabled

ENV MACHINE ✓

environmental assumptions



what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

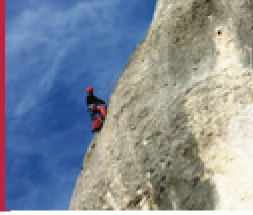
why

airborne ⇔ disabled

airborne ⇔ not WheelPulse ⇔ disabled

ENV ✗ MACHINE ✓

environmental assumptions



what happened

- Airbus A320, Warsaw 1993
- aircraft landed on wet runway
- aquaplaned, so brakes didn't work
- pilot applied reverse thrust, but disabled

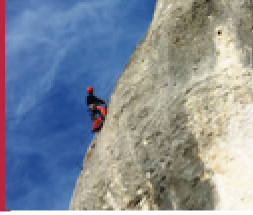
why

airborne ⇔ disabled

airborne ⇔ not WheelPulse ⇔ disabled

ENV ✗ MACHINE ✓

simplified; for full analysis, see Report on the Accident to Airbus A320-211 Aircraft in Warsaw on 14 September 1993, Main Commission Aircraft Accident Investigation, http://www.rvs.uni-bielefeld.de/publications/Incidents/DOCS/ComAndRep/Warsaw/warsaw-report.html



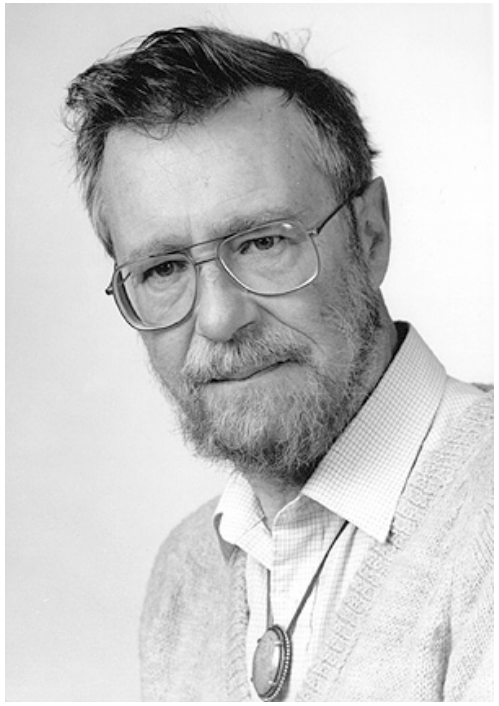
state space complexity

software systems have huge state space

- in lifetime, small proportion covered
- in testing, hardly any covered

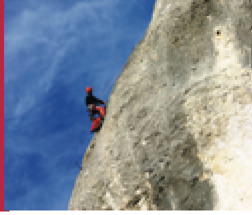
implications

- "Program testing can be used to show the **presence** of bugs, but never to show their **absence**!"
- often running in uncharted territory



*E.W. Dijkstra, Structured programming (EWD268)
http://www.cs.utexas.edu/users/EWD/

mechanical watch



state space is actually large

- many cogs, many positions

but rotational symmetry

- if works in one position, likely to work in others

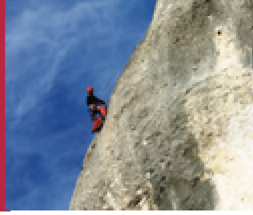
likely failure mode

- cogs wear down or break

unlikely failure mode

- design error causes error at 3:05pm

software watch



extract from Harel's watch model

- states & transitions

many states

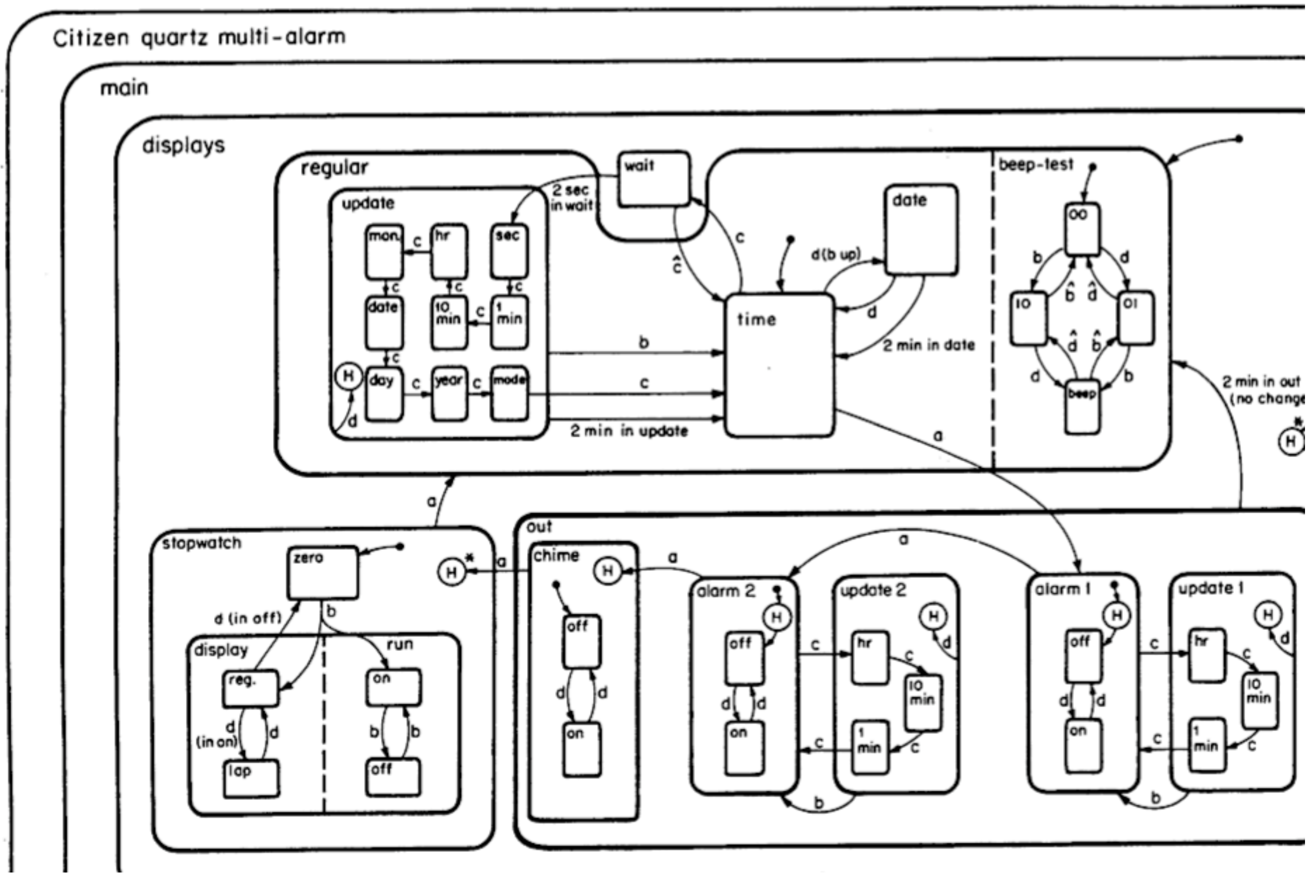
- some symmetry
- but many cases remain

likely failure mode

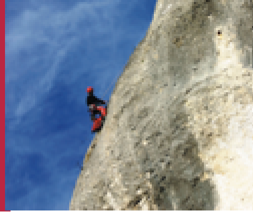
- design flaw in code

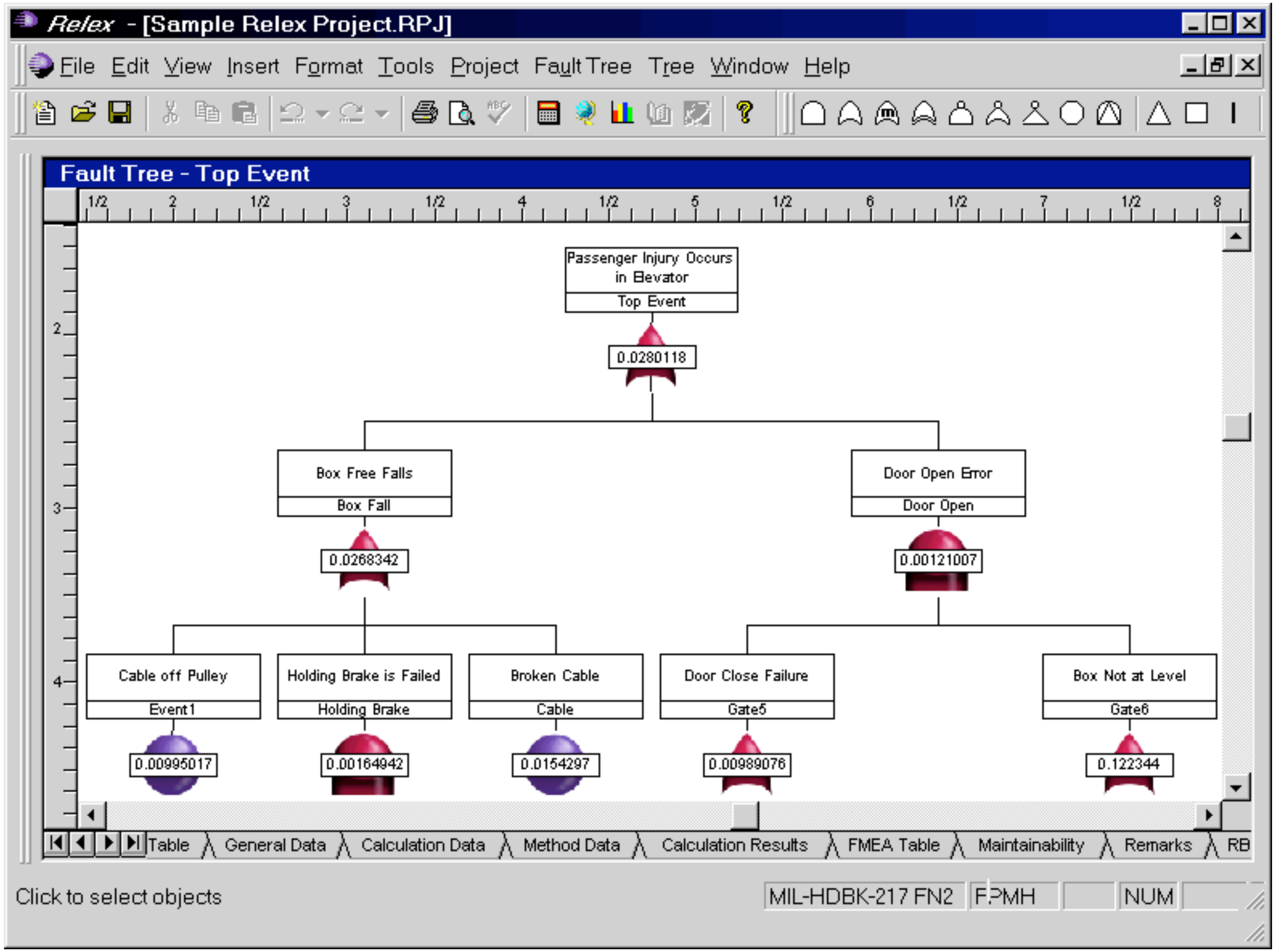
unlikely failure mode

- code wears out



a fault tree tool




Relex - [Sample Relex Project.RPJ]
File Edit View Insert Format Tools Project Fault Tree Tree Window Help
Fault Tree - Top Event
Table General Data Calculation Data Method Data Calculation Results FMEA Table Maintainability Remarks RB
Click to select objects
MIL-HDBK-217 FN2 FPMH NUM

Passenger Injury Occurs in Elevator — Top Event — 0.0280118

Box Free Falls — Box Fall — 0.0268342

Door Open Error — Door Open — 0.00121007

Cable off Pulley — Event1 — 0.00995017

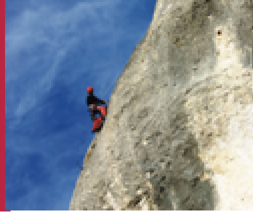
Holding Brake is Failed — Holding Brake — 0.00164942

Broken Cable — Cable — 0.0154297

Door Close Failure — Gate5 — 0.00989076

Box Not at Level — Gate6 — 0.122344

counting structures



suppose you're building a fault tree analyzer

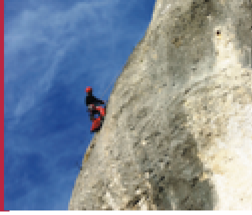
how many fault trees?

- with n nodes, can make n^{n-2} trees
- so for 10 nodes, 10^8 = 100 million trees
- actually much worse – sharing, AND/OR, etc

how about relations?

- table with C columns over N elements 2^{N^C} values
- so database with 3 tables, 3 columns, 3 elements has 2^{81} values!
- checking 1 billion/sec, would take about 100 million years

alternative to covering states?



"reliability growth modelling"

- determine operational profile
- pick random inputs weighted by profile

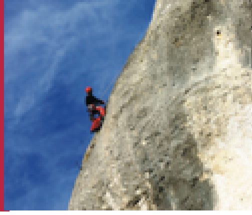
how long to test for?

- for probability of failure on demand (pfd) of 0.001
- with 99% confidence
- need about 6,600 demands without failure
- rises dramatically if failures have occurred

implication

- need huge number of tests for high confidence

coupling



what is coupling?

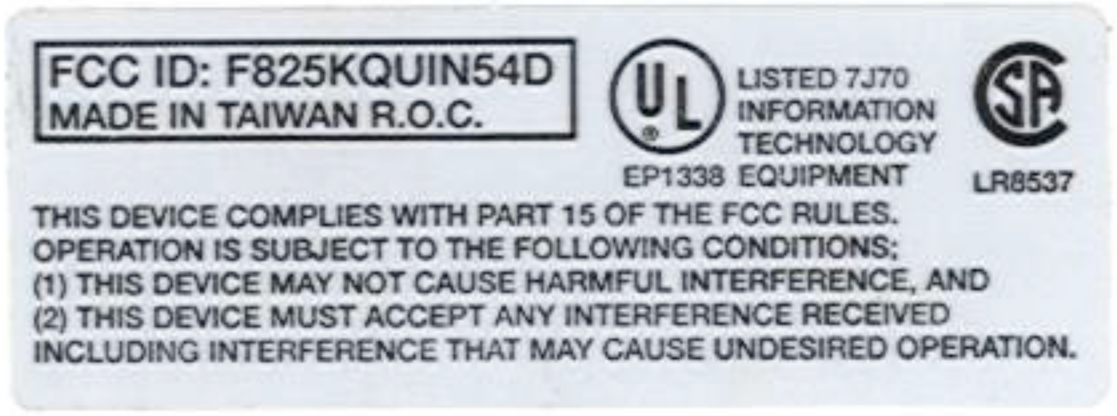
- when components of a system affect each other
- damages reliability, makes changes hard

physical components

- coupled in simple and predictable ways

software components

- coupled in complex and unpredictable ways



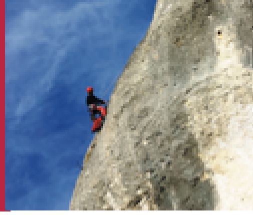
FCC ID: F825KQUIN54D
MADE IN TAIWAN R.O.C.

LISTED 7J70
INFORMATION
TECHNOLOGY
EP1338 EQUIPMENT

LR8537

THIS DEVICE COMPLIES WITH PART 15 OF THE FCC RULES.
OPERATION IS SUBJECT TO THE FOLLOWING CONDITIONS:
(1) THIS DEVICE MAY NOT CAUSE HARMFUL INTERFERENCE, AND
(2) THIS DEVICE MUST ACCEPT ANY INTERFERENCE RECEIVED
INCLUDING INTERFERENCE THAT MAY CAUSE UNDESIRED OPERATION.

USS Yorktown, 1997

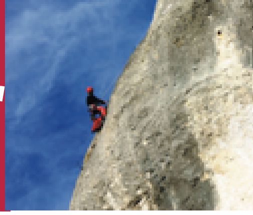


what happened

- bad data entered into spreadsheet
- divide-by-zero crashes application
- entire network went down
- ship dead in water for 3 hours

Government Computer News / July 13, 1998
Software glitches leave Navy Smart Ship dead in the water
Gregory Slabodkin, http://www.gcn.com/print/17_17/33727-1.html

dependences in internet explorer

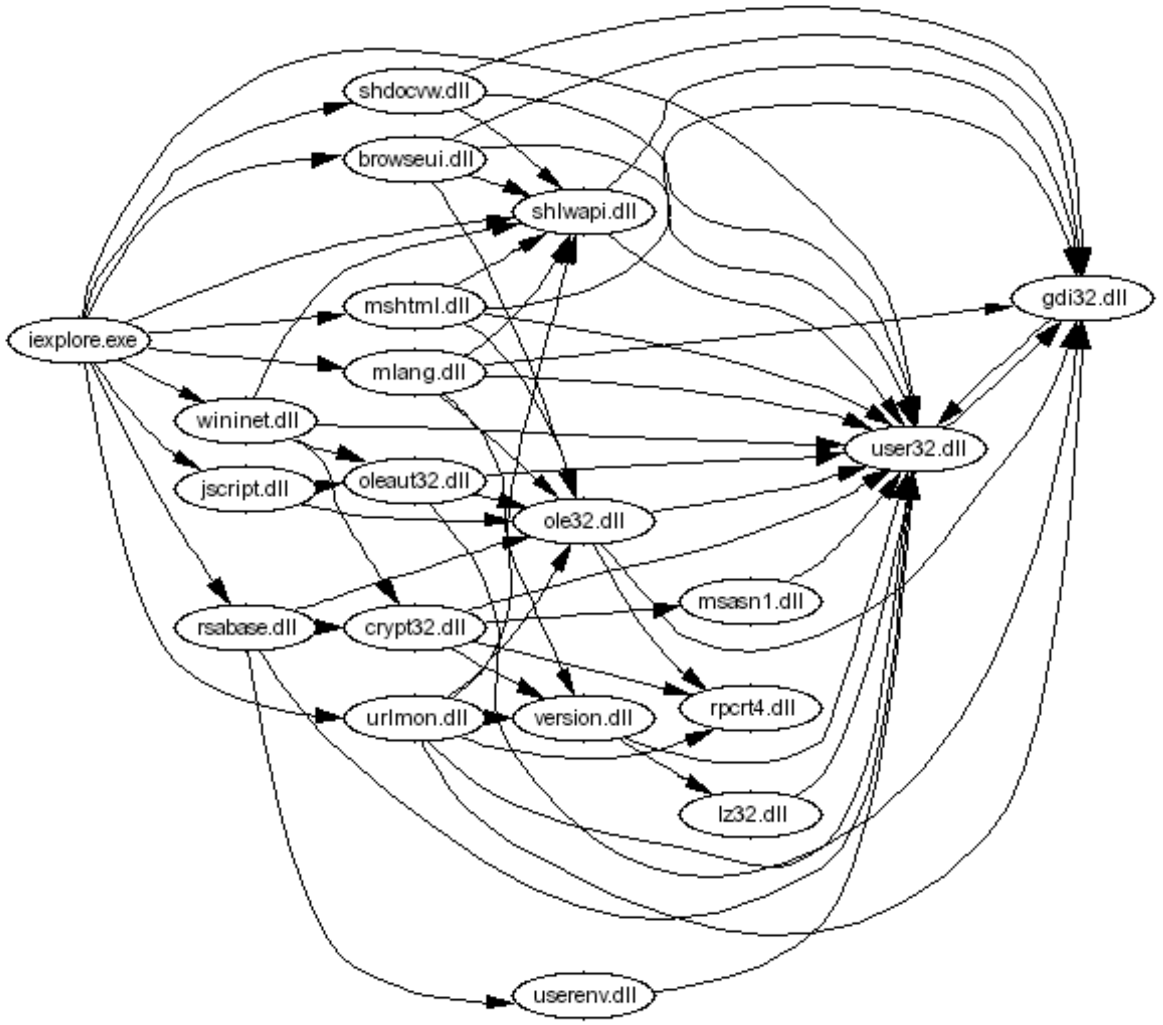


dependences between DLLs

- disciplined layering

why IE killed Netscape?

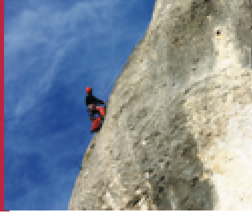
- spaghetti code in both
- but IE3 rebuilt from scratch



graph from http://www.spinellis.gr/blog/20031003
for Netscape story see:
Competing on Internet Time: Lessons From Netscape & Its Battle with Microsoft
by Michael A. Cusumano and David B. Yoffie

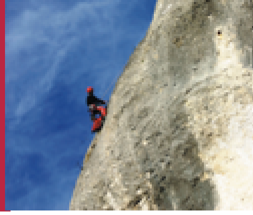
what can we do?
now and future

taming software



	today	future
requirements	pay attention to context explicit modelling designations & definitions	end-to-end arguments
state space	simplicity automated testing	model-based exploration
decoupling	safe languages data abstraction dependence diagrams	dependency management

pay attention to context

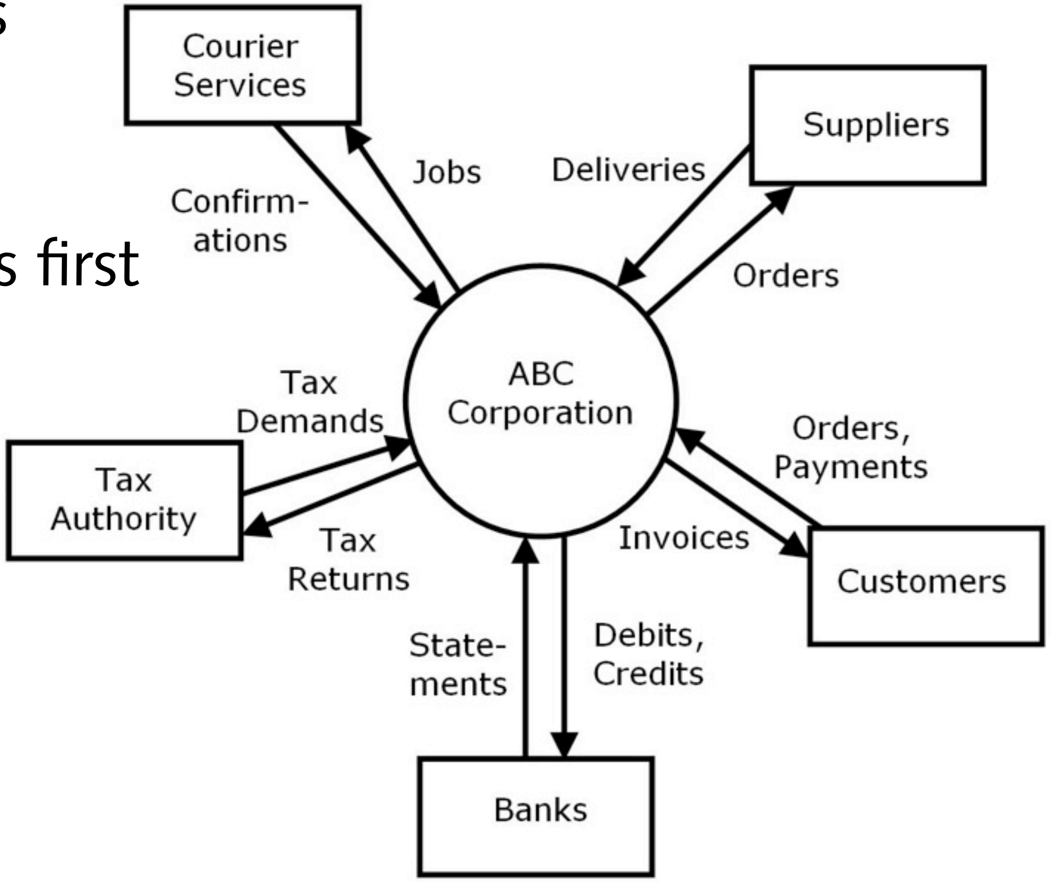


construct a context diagram

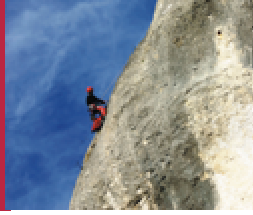
- all flows in and out of the system
- all users, operators, stakeholders

description before invention

- analyze existing business process first

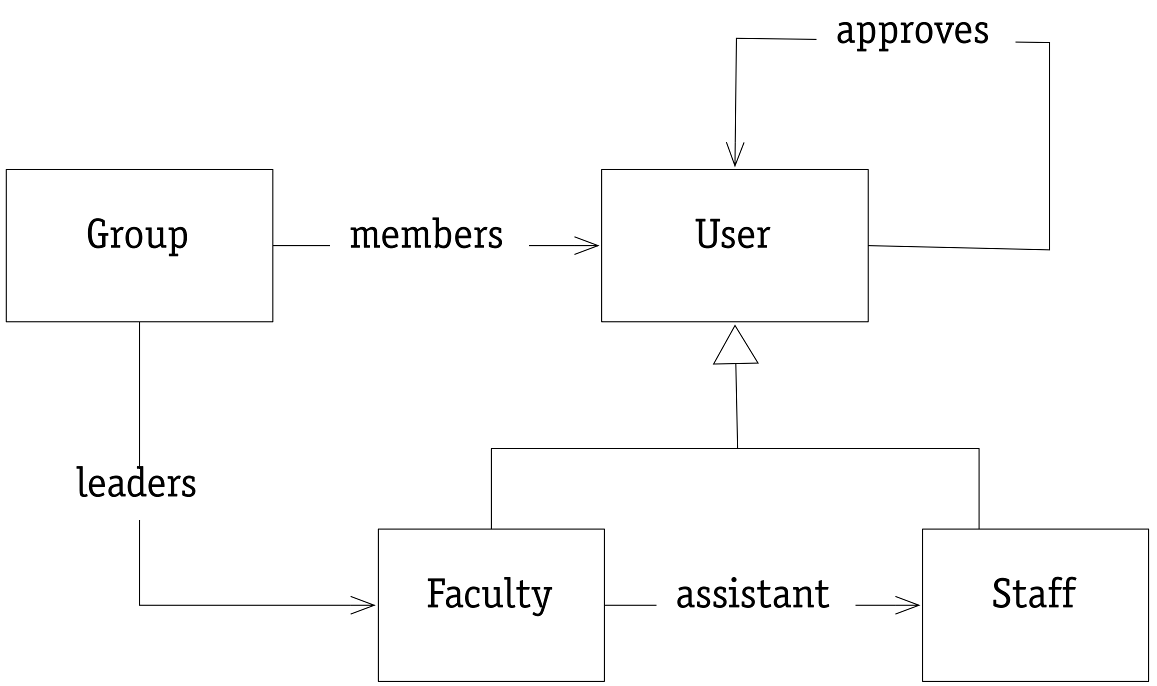


explicit modelling

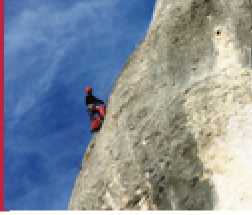


construct lightweight, precise models

- object models are most useful



designations & definitions



be clear about the meaning of terms

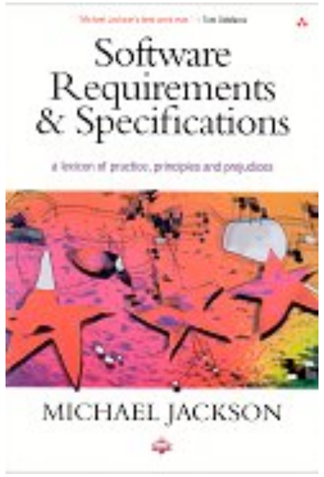
- designations: connect requirements to the world
- definitions: new terms from old ones

example

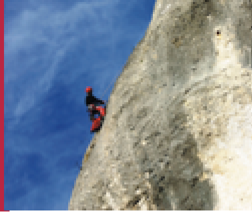
- a designation: shelved(b): book b is on a shelf in the library
- a definition: shelved(b) = owned(b) **and not** onLoan(b)

recommended reading

Software Requirements and Specifications:
A Lexicon of Principles, Practices and Prejudices.
Michael Jackson. Addison Wesley, 1995.

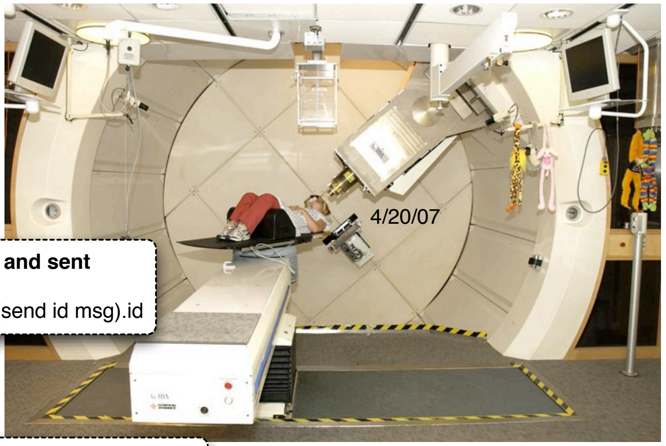

Software Requirements & Specifications
a lexicon of practice, principles and prejudices
MICHAEL JACKSON

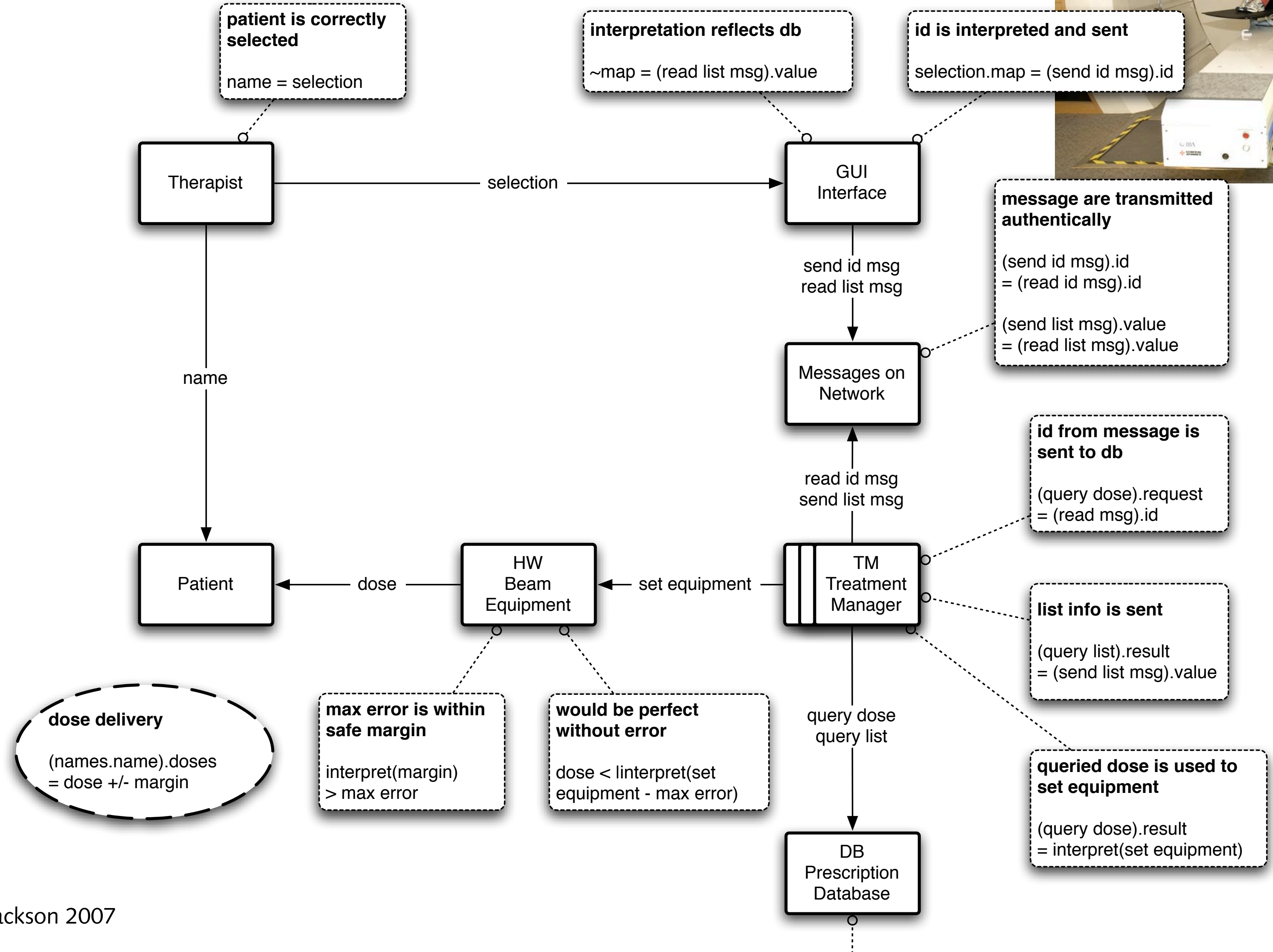
end-to-end arguments



does machine deliver right dose?

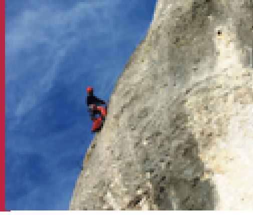
- code + physical plant + human operators



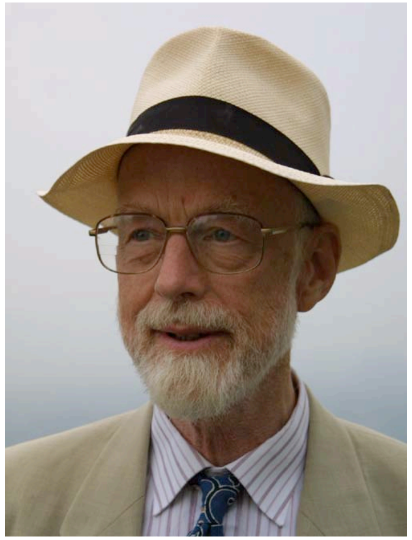


patient is correctly selected
name = selection
interpretation reflects db
~map = (read list msg).value
id is interpreted and sent
selection.map = (send id msg).id
Therapist
selection
GUI Interface
send id msg
read list msg
message are transmitted authentically
(send id msg).id = (read id msg).id
(send list msg).value = (read list msg).value
name
Messages on Network
id from message is sent to db
(query dose).request = (read msg).id
read id msg
send list msg
Patient
dose
HW Beam Equipment
set equipment
TM Treatment Manager
list info is sent
(query list).result = (send list msg).value
dose delivery
(names.name).doses = dose +/- margin
max error is within safe margin
interpret(margin) > max error
would be perfect without error
dose < Iinterpret(set equipment - max error)
query dose
query list
queried dose is used to set equipment
(query dose).result = interpret(set equipment)
DB Prescription Database

simplicity



"I gave desperate warnings against the obscurity, the complexity, and over-ambition of the new design, but my warnings went unheeded. I conclude that there are **two ways** of constructing a software design: **One way is to make it so simple there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies**"

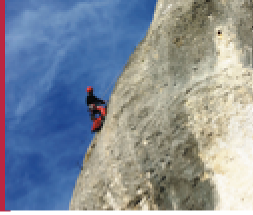


Tony Hoare, Turing Award Lecture, 1980

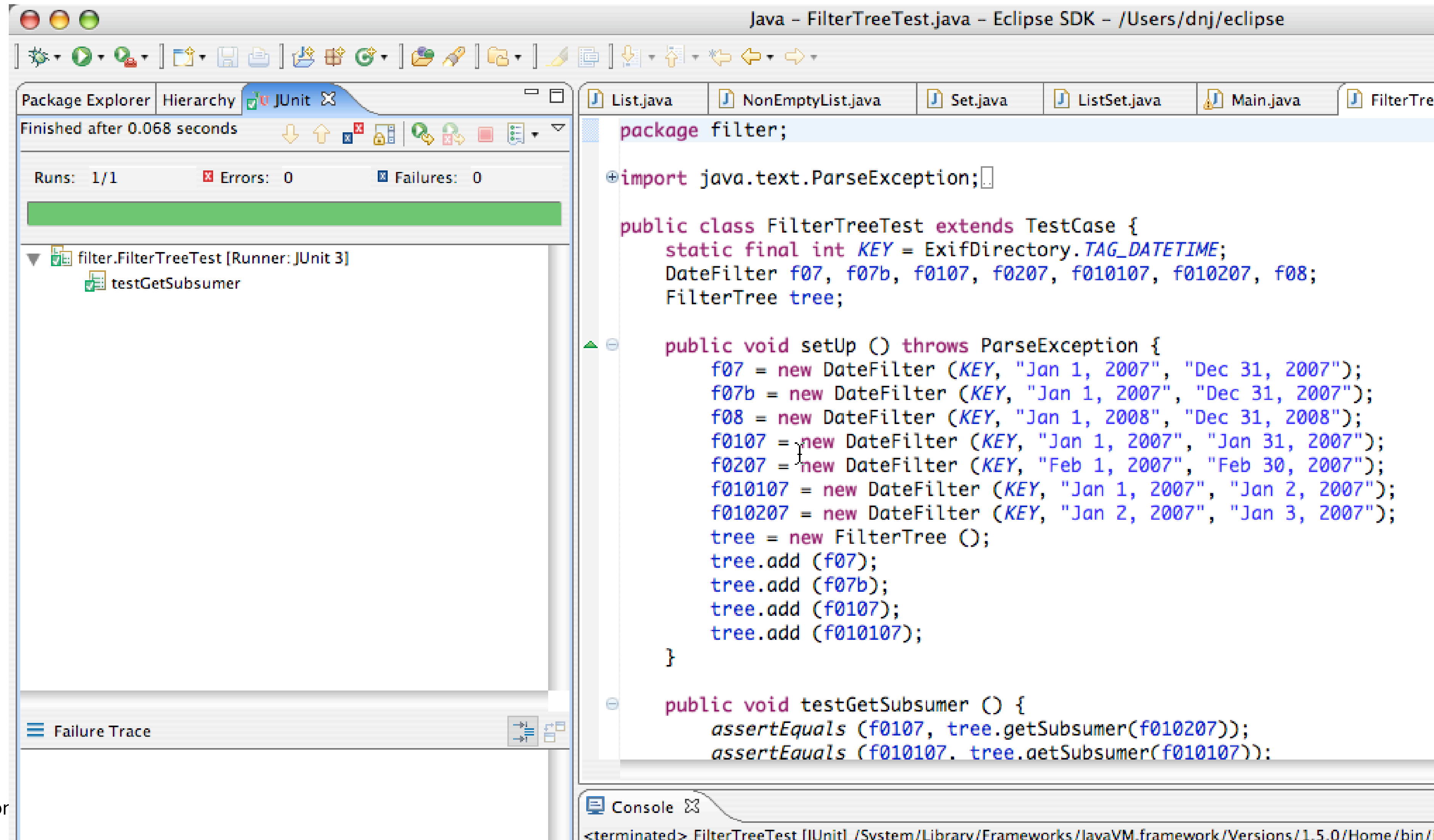
"Simplicity does not precede complexity, but follows it"

Alan Perlis

automated testing



write your tests so they can be **automated**
exploit code to **generate** as many tests as you can

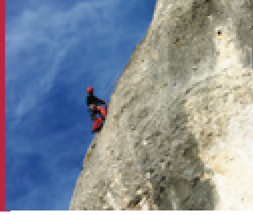


```
Finished after 0.068 seconds

Runs:  1/1          Errors:  0          Failures:  0

filter.FilterTreeTest [Runner: JUnit 3]
    testGetSubsumer
```

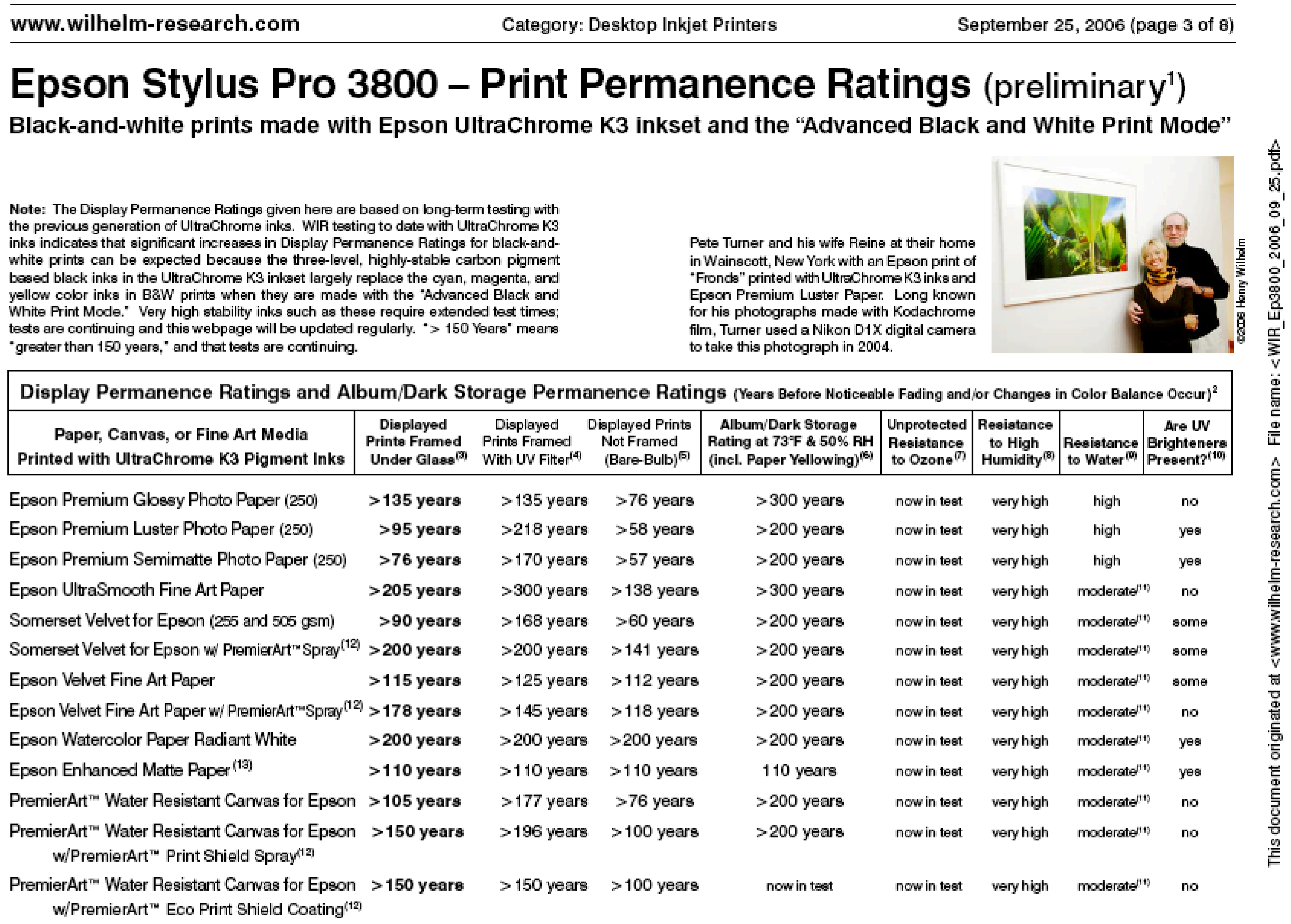
```
package filter;

import java.text.ParseException;

public class FilterTreeTest extends TestCase {
    static final int KEY = ExifDirectory.TAG_DATETIME;
    DateFilter f07, f07b, f0107, f0207, f010107, f010207, f08;
    FilterTree tree;

    public void setUp () throws ParseException {
        f07 = new DateFilter (KEY, "Jan 1, 2007", "Dec 31, 2007");
        f07b = new DateFilter (KEY, "Jan 1, 2007", "Dec 31, 2007");
        f08 = new DateFilter (KEY, "Jan 1, 2008", "Dec 31, 2008");
        f0107 = new DateFilter (KEY, "Jan 1, 2007", "Jan 31, 2007");
        f0207 = new DateFilter (KEY, "Feb 1, 2007", "Feb 30, 2007");
        f010107 = new DateFilter (KEY, "Jan 1, 2007", "Jan 2, 2007");
        f010207 = new DateFilter (KEY, "Jan 2, 2007", "Jan 3, 2007");
        tree = new FilterTree ();
        tree.add (f07);
        tree.add (f07b);
        tree.add (f0107);
        tree.add (f010107);
    }

    public void testGetSubsumer () {
        assertEquals (f0107, tree.getSubsumer(f010207));
        assertEquals (f010107, tree.getSubsumer(f010107));
```

```
<terminated> FilterTreeTest [JUnit] /System/Library/Frameworks/JavaVM.framework/Versions/1.5.0/Home/bin/
```

simulating 200 years of sunlight



can we do this for software?



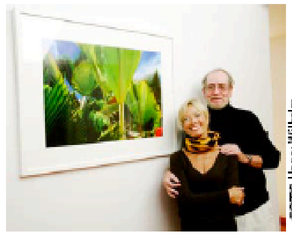
www.wilhelm-research.com Category: Desktop Inkjet Printers September 25, 2006 (page 3 of 8)

Epson Stylus Pro 3800 – Print Permanence Ratings (preliminary[1])

Black-and-white prints made with Epson UltraChrome K3 inkset and the "Advanced Black and White Print Mode"

Note: The Display Permanence Ratings given here are based on long-term testing with the previous generation of UltraChrome inks. WIR testing to date with UltraChrome K3 inks indicates that significant increases in Display Permanence Ratings for black-and-white prints can be expected because the three-level, highly-stable carbon pigment based black inks in the UltraChrome K3 inkset largely replace the cyan, magenta, and yellow color inks in B&W prints when they are made with the "Advanced Black and White Print Mode." Very high stability inks such as these require extended test times; tests are continuing and this webpage will be updated regularly. "> 150 Years" means "greater than 150 years," and that tests are continuing.

Pete Turner and his wife Reine at their home in Wainscott, New York with an Epson print of "Fronds" printed with UltraChrome K3 inks and Epson Premium Luster Paper. Long known for his photographs made with Kodachrome film, Turner used a Nikon D1X digital camera to take this photograph in 2004.



©2006 Henry Wilhelm

Display Permanence Ratings and Album/Dark Storage Permanence Ratings (Years Before Noticeable Fading and/or Changes in Color Balance Occur)[2]

Paper, Canvas, or Fine Art Media Printed with UltraChrome K3 Pigment Inks	Displayed Prints Framed Under Glass[3]	Displayed Prints Framed With UV Filter[4]	Displayed Prints Not Framed (Bare-Bulb)[5]	Album/Dark Storage Rating at 73°F & 50% RH (incl. Paper Yellowing)[6]	Unprotected Resistance to Ozone[7]	Resistance to High Humidity[8]	Resistance to Water[9]	Are UV Brighteners Present?[10]
Epson Premium Glossy Photo Paper (250)	**>135 years**	>135 years	>76 years	>300 years	now in test	very high	high	no
Epson Premium Luster Photo Paper (250)	**>95 years**	>218 years	>58 years	>200 years	now in test	very high	high	yes
Epson Premium Semimatte Photo Paper (250)	**>76 years**	>170 years	>57 years	>200 years	now in test	very high	high	yes
Epson UltraSmooth Fine Art Paper	**>205 years**	>300 years	>138 years	>300 years	now in test	very high	moderate[11]	no
Somerset Velvet for Epson (255 and 505 gsm)	**>90 years**	>168 years	>60 years	>200 years	now in test	very high	moderate[11]	some
Somerset Velvet for Epson w/ PremierArt™ Spray[12]	**>200 years**	>200 years	>141 years	>200 years	now in test	very high	moderate[11]	some
Epson Velvet Fine Art Paper	**>115 years**	>125 years	>112 years	>200 years	now in test	very high	moderate[11]	some
Epson Velvet Fine Art Paper w/ PremierArt™ Spray[12]	**>178 years**	>145 years	>118 years	>200 years	now in test	very high	moderate[11]	no
Epson Watercolor Paper Radiant White	**>200 years**	>200 years	>200 years	>200 years	now in test	very high	moderate[11]	yes
Epson Enhanced Matte Paper[13]	**>110 years**	>110 years	>110 years	110 years	now in test	very high	moderate[11]	yes
PremierArt™ Water Resistant Canvas for Epson	**>105 years**	>177 years	>76 years	>200 years	now in test	very high	moderate[11]	no
PremierArt™ Water Resistant Canvas for Epson w/PremierArt™ Print Shield Spray[12]	**>150 years**	>196 years	>100 years	>200 years	now in test	very high	moderate[11]	no
PremierArt™ Water Resistant Canvas for Epson w/PremierArt™ Eco Print Shield Coating[12]	**>150 years**	>150 years	>100 years	now in test	now in test	very high	moderate[11]	no

This document originated at <www.wilhelm-research.com>   File name: <WIR_Ep3800_2006_09_25.pdf>

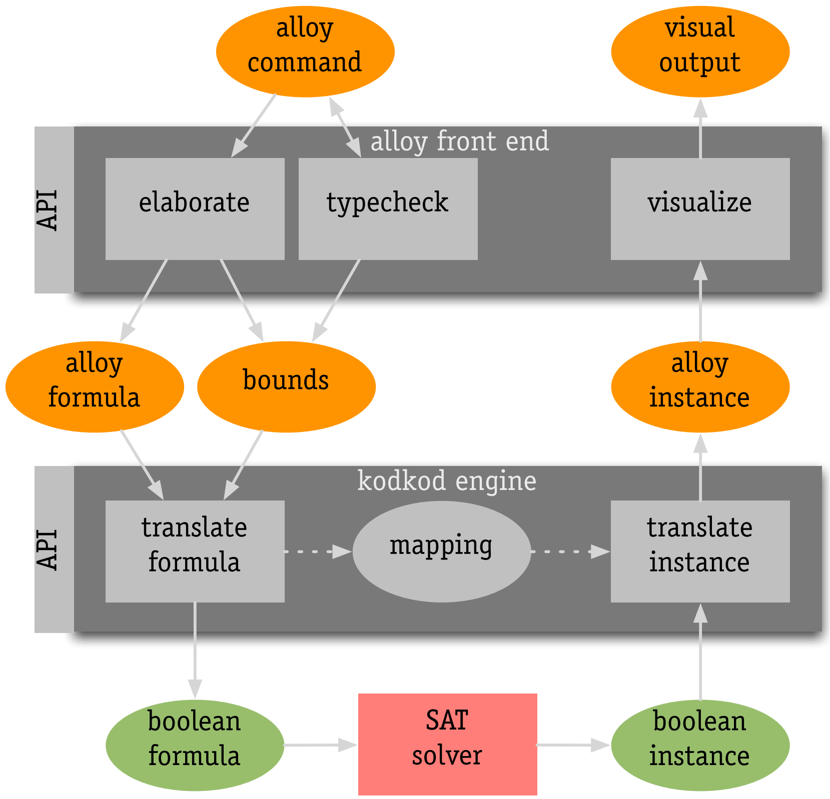
model-based exploration with alloy

testing the Galileo fault tree analyzer

- used by NASA on space station
- generated 250,000 trees (all 4-event)
- found 8 faults (tool), 3 (spec), 3 (oracle)

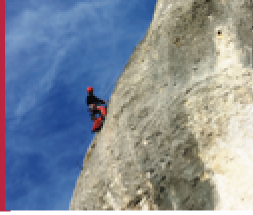
Mondex smartcard

- developed by NatWest Bank
- formal specification by Logica UK Ltd
- analysis with Alloy by Tahina Ramananandro
- all scenarios in scope of 5 (cards, users, etc)



K. Sullivan, J. Yang, D. Coppit, S. Khurshid, D. Jackson
Improving Software Assurance by Bounded Exhaustive Testing
International Symposium on Software Testing and Analysis, 2004

achieving decoupling



for system architects

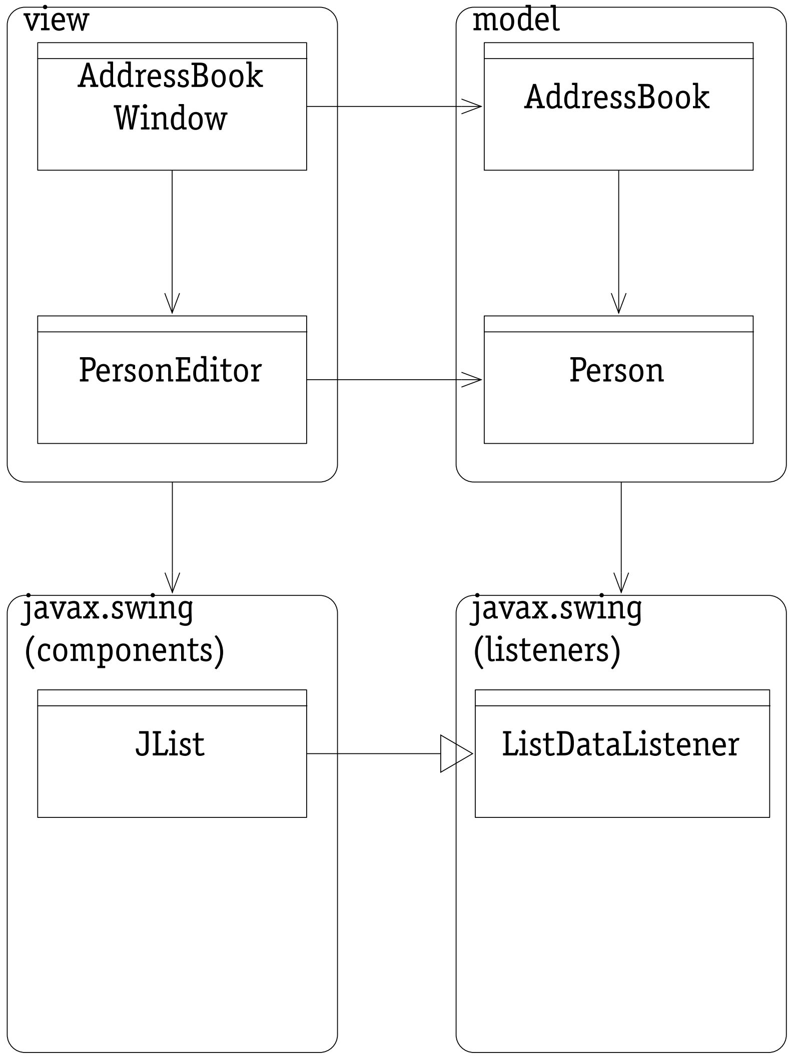
- choose a safe language: Java, not C++

for programmers

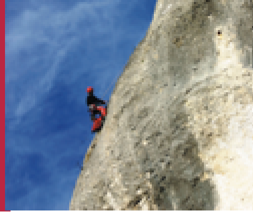
- use real data abstractions, not just objects
- all fields should be private

for designers, programmers, testers

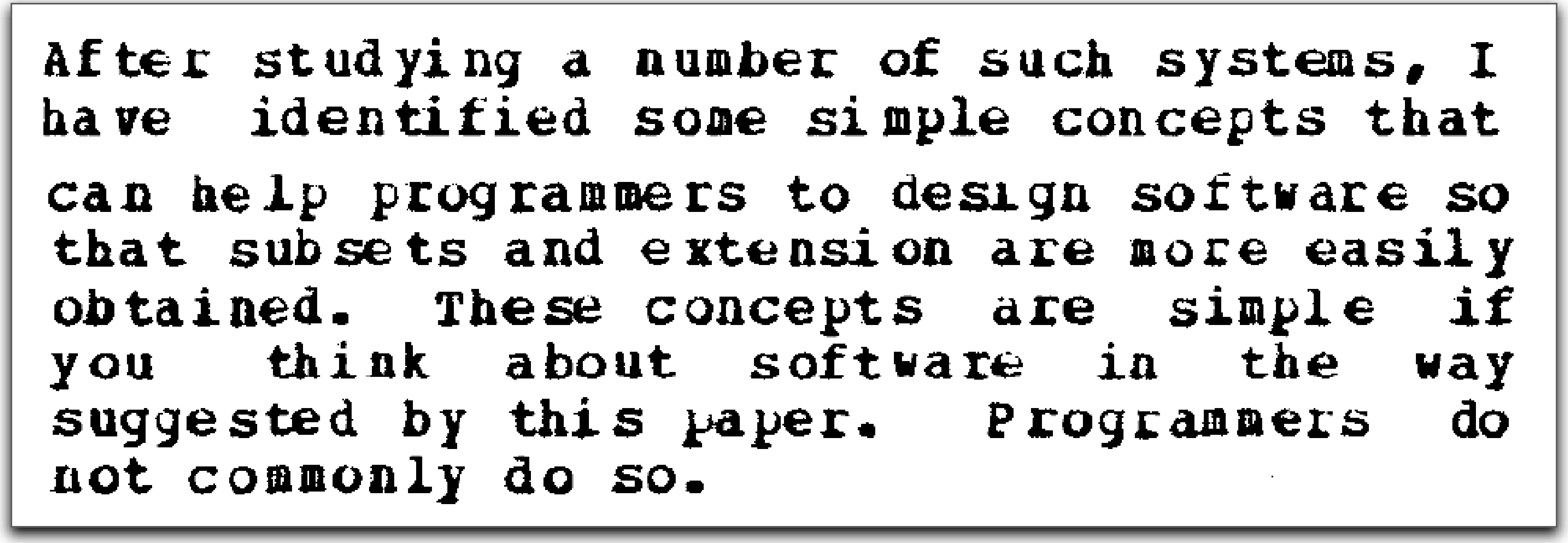
- construct a dependence diagram
- identify dependency liabilities
- focus testing on module interactions



learning about dependences



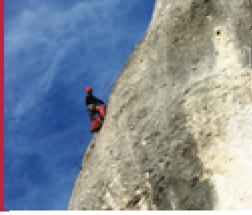
from Parnas's classic paper, 1979



After studying a number of such systems, I have identified some simple concepts that can help programmers to design software so that subsets and extension are more easily obtained. These concepts are simple if you think about software in the way suggested by this paper. Programmers do not commonly do so.

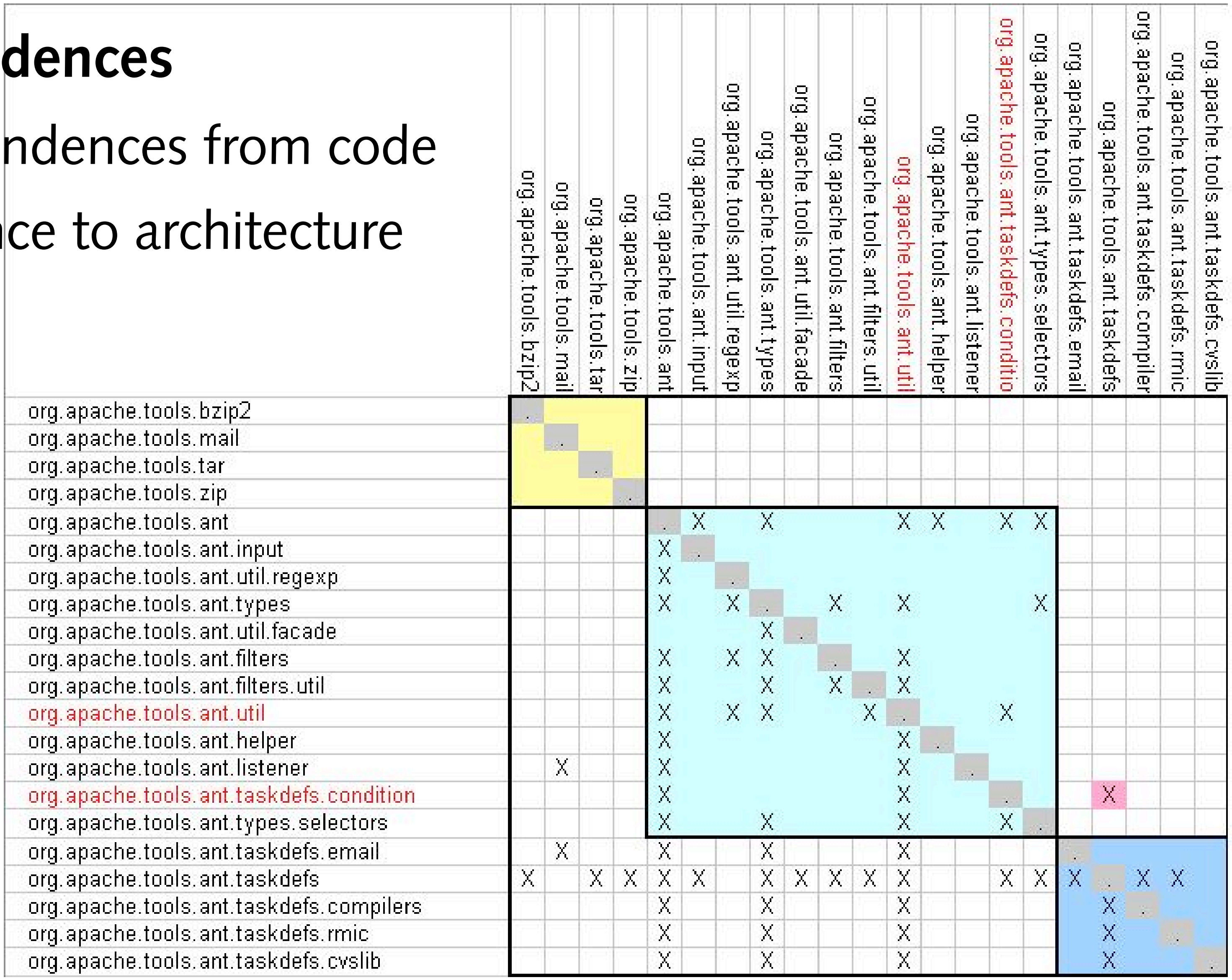
David L. Parnas. Designing software for ease of extension and contraction. IEEE Transactions on Software Engineering, SE-5, 2 (1979)

dependency management



controlling dependences

- tool extracts dependences from code
- checks conformance to architecture

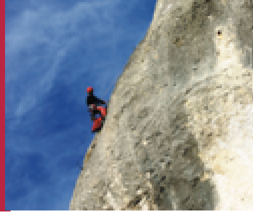


	org.apache.tools.bzip2	org.apache.tools.mail	org.apache.tools.tar	org.apache.tools.zip	org.apache.tools.ant	org.apache.tools.ant.input	org.apache.tools.ant.util.regexp	org.apache.tools.ant.types	org.apache.tools.ant.util.facade	org.apache.tools.ant.filters	org.apache.tools.ant.filters.util	org.apache.tools.ant.util	org.apache.tools.ant.helper	org.apache.tools.ant.listener	org.apache.tools.ant.taskdefs.conditio	org.apache.tools.ant.types.selectors	org.apache.tools.ant.taskdefs.email	org.apache.tools.ant.taskdefs	org.apache.tools.ant.taskdefs.compiler	org.apache.tools.ant.taskdefs.rmic	org.apache.tools.ant.taskdefs.cvslib
org.apache.tools.bzip2	.																				
org.apache.tools.mail		.																			
org.apache.tools.tar			.																		
org.apache.tools.zip				.																	
org.apache.tools.ant					.	X		X				X	X		X	X					
org.apache.tools.ant.input					X	.															
org.apache.tools.ant.util.regexp					X		.														
org.apache.tools.ant.types					X		X	.		X		X				X					
org.apache.tools.ant.util.facade								X	.												
org.apache.tools.ant.filters					X		X	X		.		X									
org.apache.tools.ant.filters.util					X			X		X	.	X									
org.apache.tools.ant.util					X		X	X			X	.			X						
org.apache.tools.ant.helper					X							X	.								
org.apache.tools.ant.listener		X			X							X		.							
org.apache.tools.ant.taskdefs.condition					X							X			.			X			
org.apache.tools.ant.types.selectors					X			X				X			X	.					
org.apache.tools.ant.taskdefs.email		X			X			X				X					.				
org.apache.tools.ant.taskdefs	X		X	X	X	X		X	X	X	X	X			X	X	X	.	X	X	
org.apache.tools.ant.taskdefs.compilers					X			X				X						X	.		
org.apache.tools.ant.taskdefs.rmic					X			X				X						X		.	
org.apache.tools.ant.taskdefs.cvslib					X			X				X						X			.

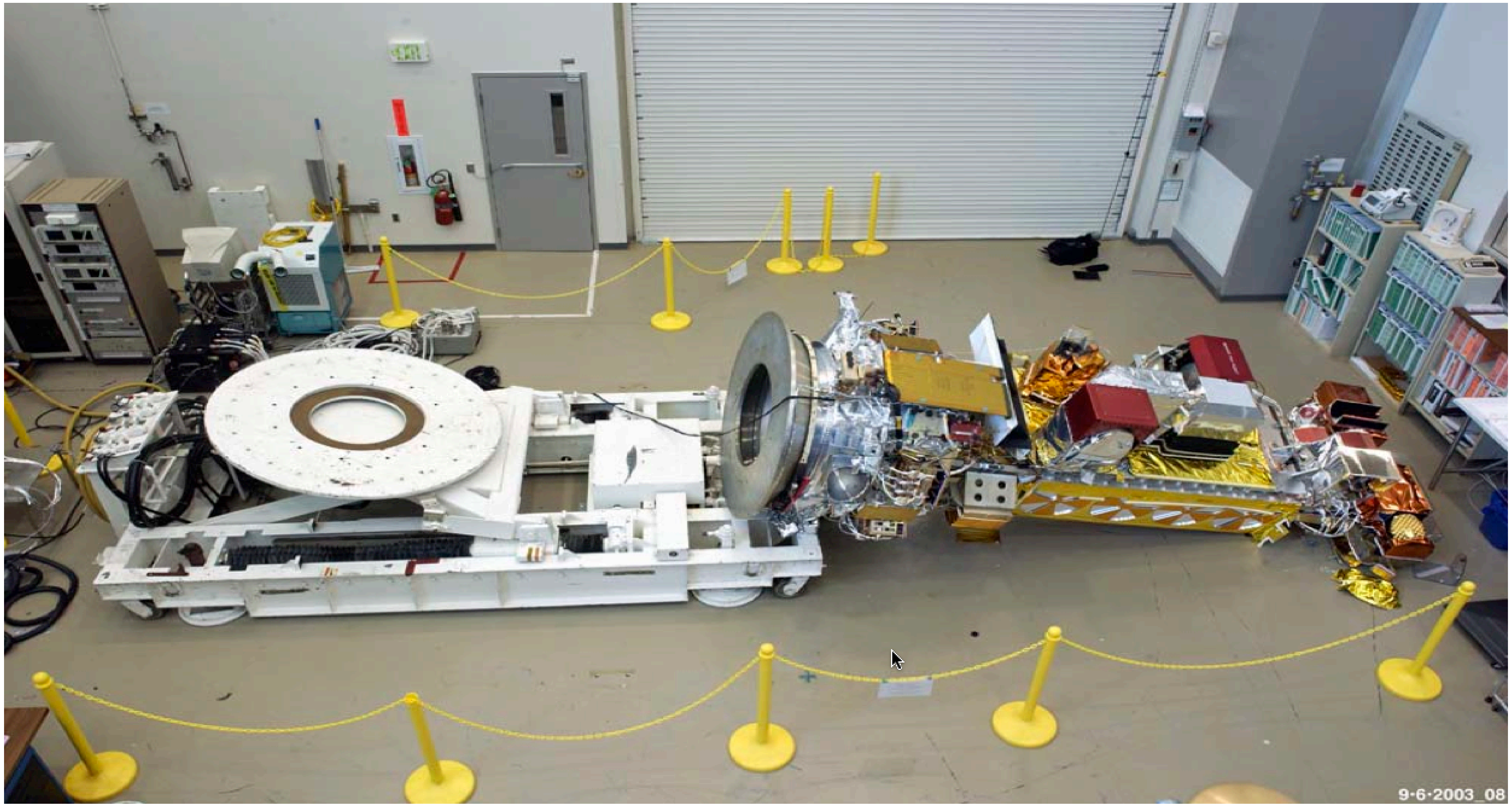
Lattix's LDM

and process matters too...

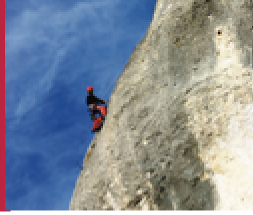
the importance of process

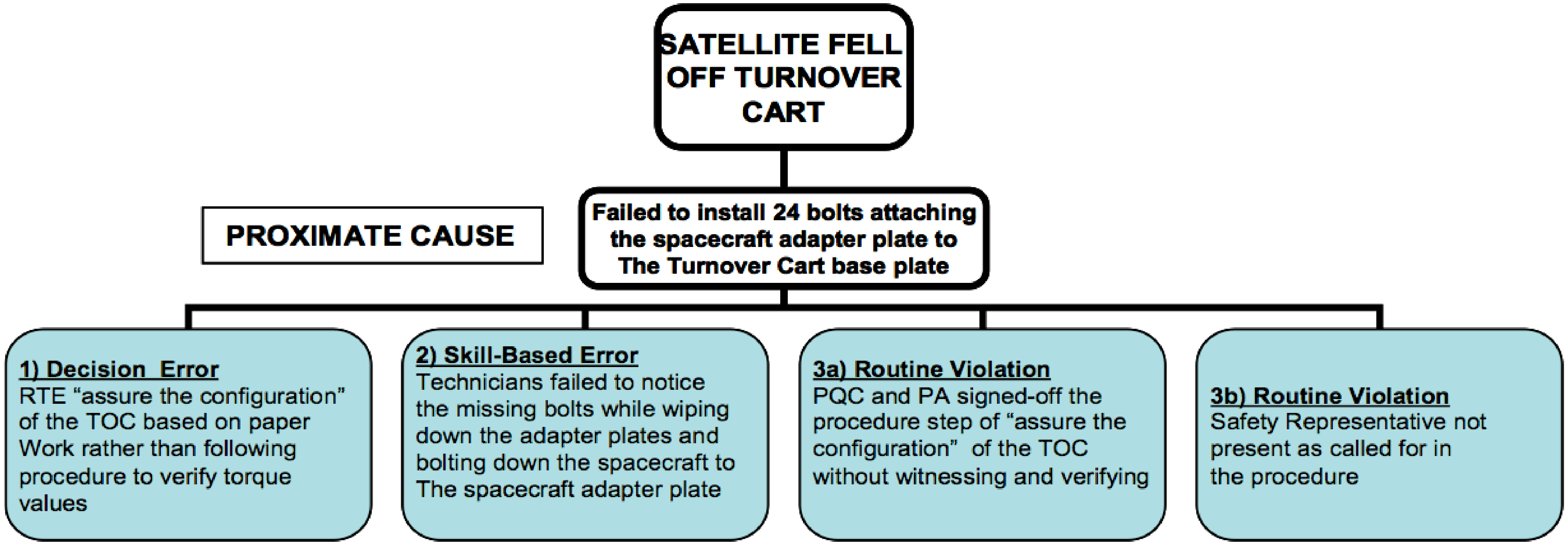


NOAA weather satellite at Lockheed Martin, September 2003



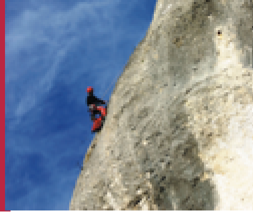
a fault tree




SATELLITE FELL OFF TURNOVER CART
PROXIMATE CAUSE
Failed to install 24 bolts attaching the spacecraft adapter plate to The Turnover Cart base plate
1) Decision Error
RTE "assure the configuration" of the TOC based on paper Work rather than following procedure to verify torque values
2) Skill-Based Error
Technicians failed to notice the missing bolts while wiping down the adapter plates and bolting down the spacecraft to The spacecraft adapter plate
3a) Routine Violation
PQC and PA signed-off the procedure step of "assure the configuration" of the TOC without witnessing and verifying
3b) Routine Violation
Safety Representative not present as called for in the procedure

NOAA N-Prime Mishap Investigation, Final Report
NASA, September 2004

conclusions



three challenges of software

- requirements, state space complexity, coupling

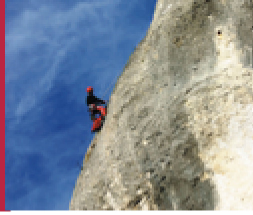
powerful tools we have today

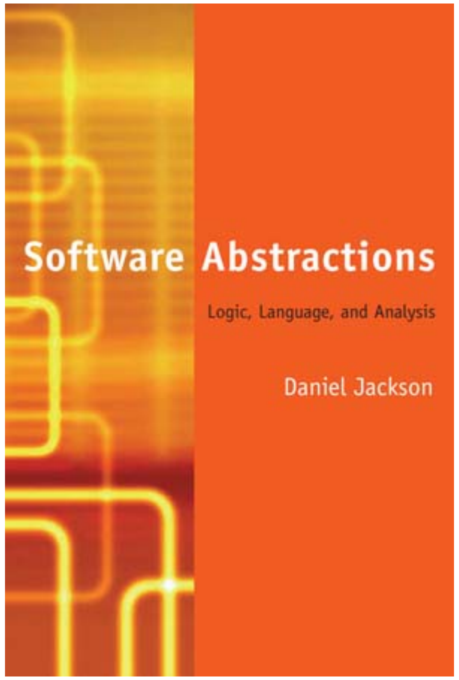
- models, test-case generation, dependency diagrams

in the future

- end-to-end cases
- model-based analysis and code generation
- automated dependency management

for more information





modelling and analysis with Alloy

- *Software Abstractions*, MIT Press, 2006

on requirements

- *Software Requirements and Specifications: A Lexicon of Principles, Practices and Prejudices.* Michael Jackson. Addison Wesley, 1995.

on decoupling

- Designing software for ease of extension and contraction. David Parnas. IEEE Transactions on Software Engineering, SE-5, 2 (1979).

on programming

- *Programming Pearls*. Jon Bentley. Addison Wesley, 1989.
- *Effective Java*. Joshua Bloch. Addison Wesley, 2001