Complexity revisited: learning from failures

Frans Kaashoek

Lec 26 --- Last one!

5/11/11

Credit: Jerry Saltzer

6.033 in one slide

Principles:End-to-end argument, Open Design, …

- Client/server
- RPC
- File abstraction
- Virtual memory
- Threads
- Coordination
- Protocol layering
- Routing protocols
- Reliable packet delivery
- Names
- Replication protocols
- Transactions
- Secure channels
- Hash

Case studies of successful systems: DNS, X Windows, Unix, MapReduce, BGP, TCP, Bittorrent, RAID, Databases, SSL, ….

Today:
Why do systems fail anyway?

- Complexity in computer systems has no hard edge
- Learning from failures: common problems
- Fighting back: avoiding the problems
- 6.033 theme song

Too many objectives

- Ease of use
- Availability
- Scalability
- Flexibility
- Mobility
- Security
- Networked
- Maintainability
- Performance
- Durable
- ….

Lack systematic methods

Many objectives

+

Few Methods

+

High d(technology)/dt

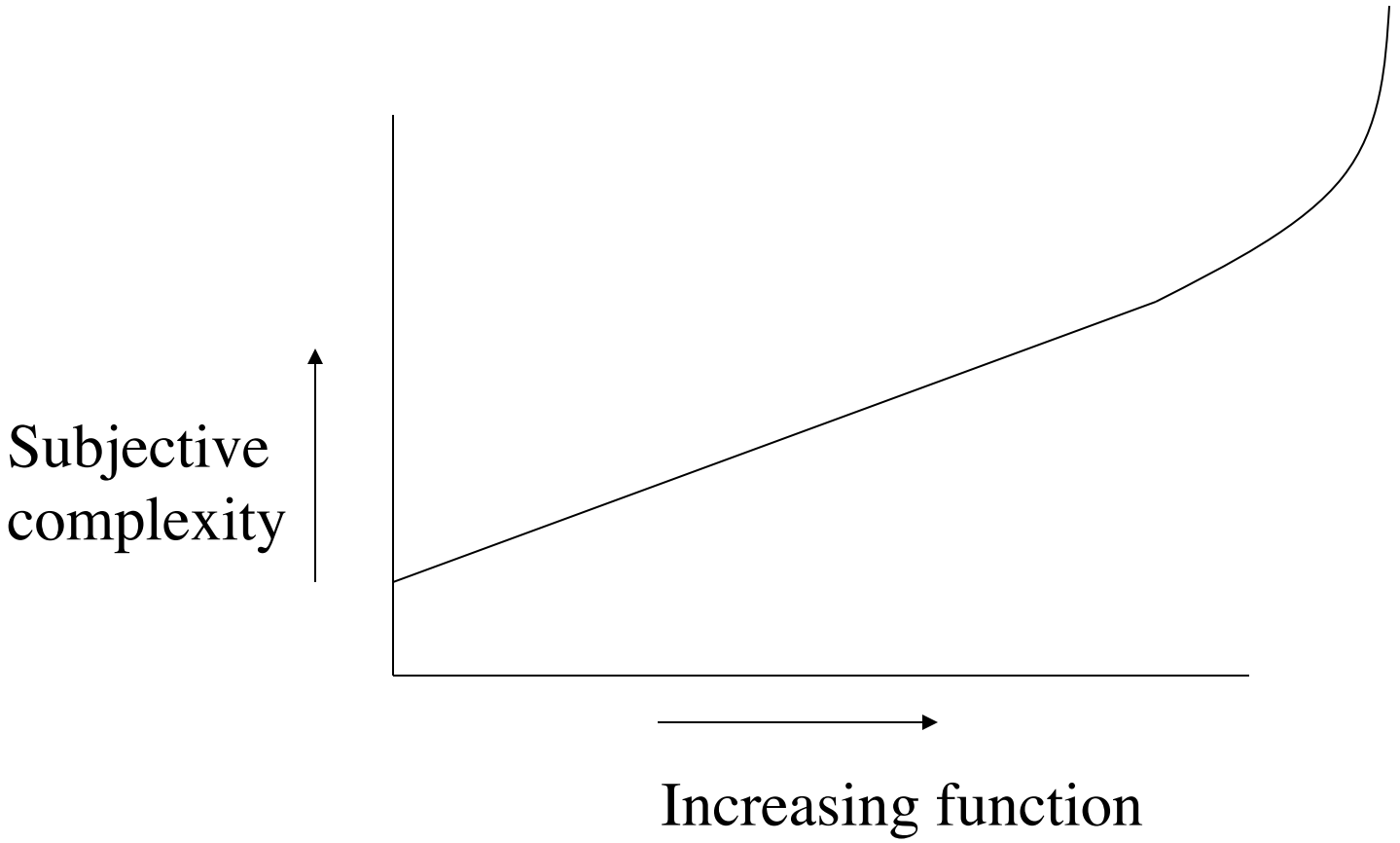
=

Very high risk of failure

[Brooks, Mythical Man Month]

The tarpit

Complexity: no hard edge



- It just gets worse, worse, and worse …

Learn from failure

"The concept of failure is central to design process, and it is by thinking in terms of obviating failure that successful designs are achieved…"
[Petroski]

Keep digging principle

- Complex systems systems fail for complex reasons
  - Find the cause …
  - Find a second cause …
  - Keep looking …
  - Find the mind-set.

[Petroski, Design Paradigms]

Pharaoh Sneferu's Pyramid project

Try 1: Meidum (52° angle)

Try 2: Dashur/Bent
(52° to 43.5° angle)

Try 3: Red pyramid (right angle: 43°)

United Airlines/Univac

- Automated reservations, ticketing, flight scheduling, fuel delivery, kitchens, and general administration
- Started 1966, target 1968, scrapped 1970, spend $50M
- Second-system effect (First: SABRE)

(Burroughs/TWA repeat)

CONFIRM

- Hilton, Marriott, Budget, American Airlines
- Hotel reservations linked with airline and car rental
- Started 1988, scrapped 1992, $125M
- Second system
- Dull tools (machine language)
- Bad-news diode

[Communications of the ACM 1994]

IBM Workplace OS for PPC

- Mach 3.0 + binary compatability with AIX + DOS, MacOS, OS/400 + new clock mgmt + new RPC + new I/O + new CPU
- Started in 1991, scrapped 1996 ($2B)
- 400 staff on kernel, 1500 elsewhere
- "Sheer complexity of class structure proved to be overwhelming"
- Inflexibility of frozen class structure
- Big-endian/Little-endian not solved

[Fleish HotOS 1997]

Advanced Automation System

- US Federal Aviation Administration
- Replaces 1972 Air Route Traffic Control System
- Started 1982, scrapped 1994 ($6B)
- All-or-nothing
- Changing specifications
- Grandiose expectations
- Contract monitors viewed contractors as adversaries
- Congressional meddling

London Ambulance Service

- Ambulance dispatching
- Started 1991, scrapped in1992 (20 lives lost in 2 days, 2.5M)
- Unrealistic schedule (5 months)
- Overambitious objectives
- Unidentifiable project manager
- Low bidder had no experience
- No testing/overlap with old system
- Users not consulted during design

[Report of the Inquiry Into The London Ambulance Service 1993]

More, too many to list …

- Portland, Oregan, Water Bureau, 30M, 2002
- Washington D.C., Payroll system, 34M 2002
- Southwick air traffic control system $1.6B 2002
- Sobey's grocery inventory, 50M, 2002
- King's County financial mgmt system, 38M, 2000)
- Australian submarine control system, 100M, 1999
- California lottery system, 52M
- Hamburg police computer system, 70M, 1998
- Kuala Lumpur total airport management system, $200M, 1998
- UK Dept. of Employment tracking, $72M, 1994
- Bank of America Masternet accounting system, $83M, 1988,
- FBI virtual case, 2004.
- FBI Sentinel case management software, 2006.
- UK National offender management IS, $155M, 2007 (restart)

Recurring problems

- Excessive generality and ambition
- Bad ideas get included
- Second-system effect
- Mythical Man Month
- Wrong modularity
- Bad-news diode
- Incommensurate scaling

Fighting back: control novelty

- Source of excessive novelty:
  - Second-system effect
  - Technology is better
  - Idea worked in isolation
  - Marketing pressure
- *Some* novelty is necessary; the difficult part is saying *No*.
- Don't be afraid to re-use existing components
  - Don't reinvent the wheel
  - Even if it takes some massaging

Fighting back: adopt sweeping simplifications

- Processor, Memory, Communication
- Dedicated servers
- N-level memories
- Best-effort network
- Delegate administration
- Fail-fast, pair-and-compare
- Don't overwrite
- Transactions
- Sign and encrypt

Fighting back: design for iteration, iterate the design

- Something simple working soon
  - Find out what the real problems are
- One new problem at a time
- Use iteration-friendly design
  - E.g., Failure/attack models

"Every successful complex system is found to have evolved from a successful simple system"

Example: Linux

- 1995: Linux hobbyist project
- Now: Google, Amazon servers, Android run Linux
- Fast iterative software development

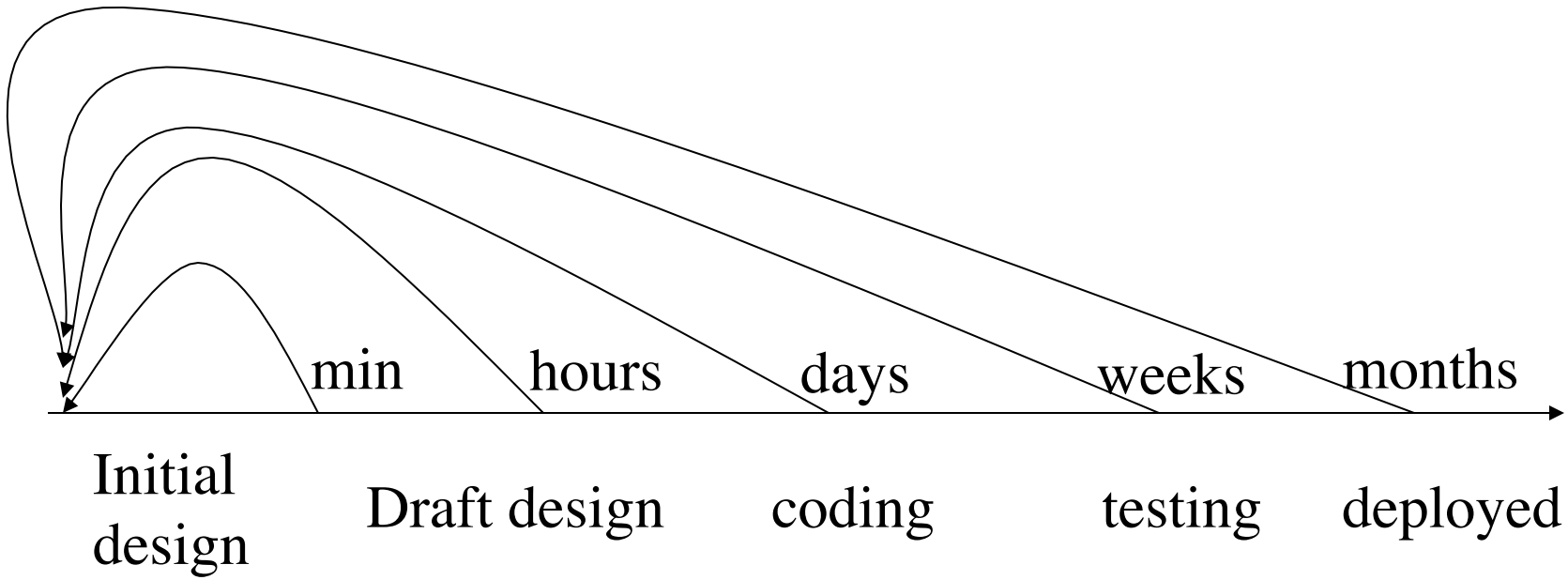
Version	Insertions		Deletions		#
	LOC	% of total	LOC	% of total	
2.6.13	224596	6.6%	79304	2.3%	
2.6.14	125947	3.5%	73173	2.1%	
2.6.15	295552	8.2%	184804	5.1%	
2.6.16	239086	6.4%	115931	3.1%	
2.6.17	158873	4.1%	99534	2.6%	
2.6.18	195217	5.0%	87806	2.2%	
2.6.19	264203	6.6%	173307	4.3%	
2.6.20	130823	3.2%	70608	1.7%	
2.6.21	179715	4.3%	99243	2.4%	
2.6.22	282043	6.6%	135117	3.2%	
2.6.23	194784	4.4%	204809	4.7%	
2.6.24	400742	9.1%	319644	7.3%	
2.6.25	458622	10.3%	247255	5.5%	
2.6.26	307233	6.6%	206375	4.4%	
2.6.27	649094	13.6%	609369	12.8%	
2.6.28	499163	10.4%	343133	7.1%	
2.6.29	443027	8.9%	262983	5.3%	
2.6.30	376948	7.3%	189751	3.7%	
2.6.31	306308	5.7%	100660	1.9%	
2.6.32	303351	5.5%	110817	2.0%	
2.6.33	266081	4.6%	134299	2.3%	
2.6.34	308185	5.3%	133734	2.3%	
2.6.35	227835	3.8%	285428	4.7%	
2.6.36	209973	3.5%	431638	7.2%	
2.6.37	245373	4.3%	134239	2.3%	

Fighting back: find bad ideas fast

- Question requirements
  - "And ferry itself across the Atlantic" [LHX light attack helicoper]
- Try ideas out, but don't hesitate to scrap
- Understand the design loop

Requires strong, knowledgeable management

The design loop



- Find flaws fast!

Fighting back: find flaws fast

- Plan, plan, plan (CHIPS, Intel processors)
- Simulate, simulate, simulate
  - Boeing 777 and F-16
- Design reviews, coding reviews, regression tests, daily/hourly builds, performance measurements
- Design the feedback system:
  - Alpha and beta tests
  - Incentives, not penalties, for reporting errors

Fighting back: conceptual integrity

- One mind controls the design
  - Macintosh
  - Visicalc spreadsheet
  - UNIX
  - Linux
- Good esthetics yields more successful systems
  - Parsimonious, Orthogonal, Elegant, Readable, …
- Few top designers can be more productive than a larger group of average designers.

Fighting back: learn from failures

- Take failures seriously and learn from it
- Example: Amazon outage
  - Elastic block store aggressively remirrors
  - Network configuration problem in NE availability zone effected primary and backup network
  - "Re-mirror storm", effected other regions
  - Took days to get under control
  - Amazon took failure analysis serious
- Counter examples: RSA, Sony PlayStation network

Fighting back: summary

- Principles that help avoiding failure
  - Limit novelty
  - Adopt sweeping simplifications
  - Get something simple working soon
  - Iteratively add capability
  - Give incentives for reporting errors
  - Descope early
  - Give control to (and keep it in) a small design team
- Strong outside pressures to violate these principles
  - Need strong knowledgeable managers

Admonition

Make sure that none of the systems you design can be used as disaster examples in future versions of this lecture

6.033 theme song

'Tis the gift to be simple, 'tis the gift to be free,
'Tis the gift to come down where we ought to be;
And when we find ourselves in the place just right,
'Twill be in the valley of love and delight.

When true simplicity is gained
To bow and to bend we shan't be ashamed;
To turn, turn will be our delight,
Till by turning, turning we come out right.

[Simple Fifts, traditional Shaker hymn]

Learn more about systems

- 6.823: computer architecture
- 6.824: distributed systems engineering
- 6.828: operating system engineering
- 6.829: computer networking
- 6.830: databases
- 6.858: computer system security