



An Overview of Human Error

Drawn from J. Reason, *Human Error*, Cambridge, 1990

Aaron Brown

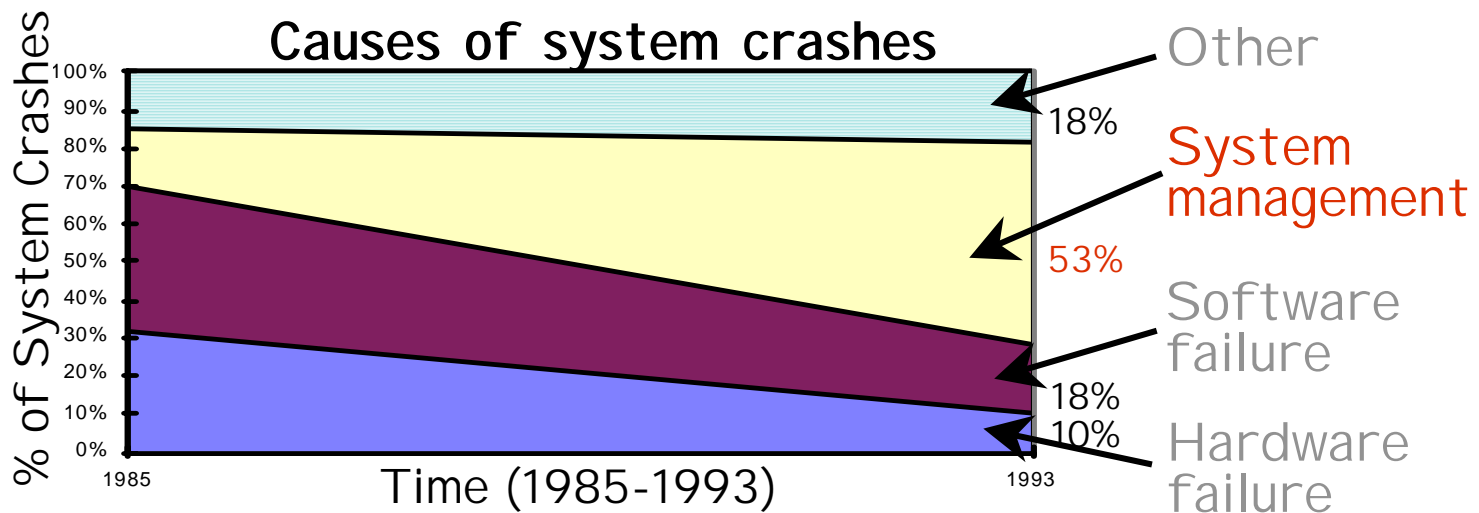
CS 294-4 ROC Seminar

Outline

- Human error and computer system failures
- A theory of human error
- Human error and accident theory
- Addressing human error

Dependability and human error

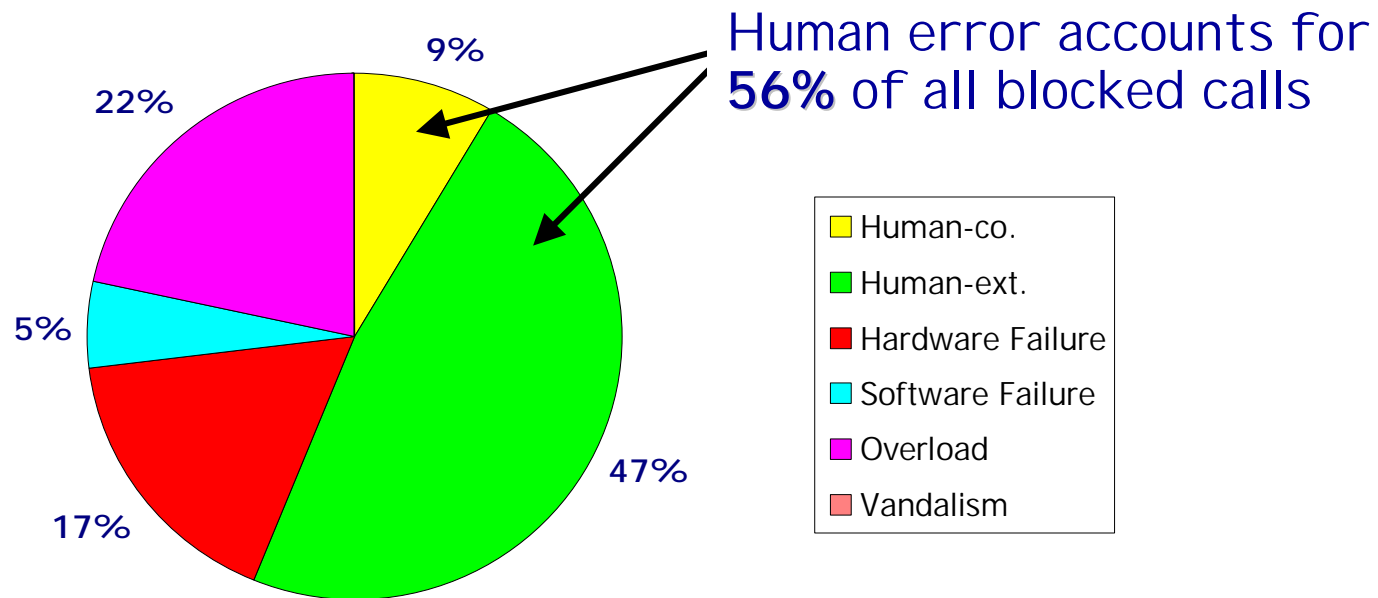
- Industry data shows that human error is the largest contributor to reduced dependability
 - HP HA labs: human error is #1 cause of failures (2001)
 - Oracle: half of DB failures due to human error (1999)
 - Gray/Tandem: 42% of failures from human administrator errors (1986)
 - Murphy/Gent study of VAX systems (1993):



Learning from other fields: PSTN

- **FCC-collected data on outages in the US public-switched telephone network**

- metric: breakdown of customer calls blocked by system outages (excluding natural disasters). Jan-June 2001









- comparison with 1992-4 data shows that human error is the only factor that is not improving over time



















Learning from other fields: PSTN

- PSTN trends: 1992-1994 vs. 2001

Cause	Trend	Minutes (millions of customer minutes/month)	
		1992-94	2001
Human error: company		98	176
Human error: external		100	75
Hardware		49	49
Software		15	12
Overload		314	60
Vandalism		5	3

Learning from experiments

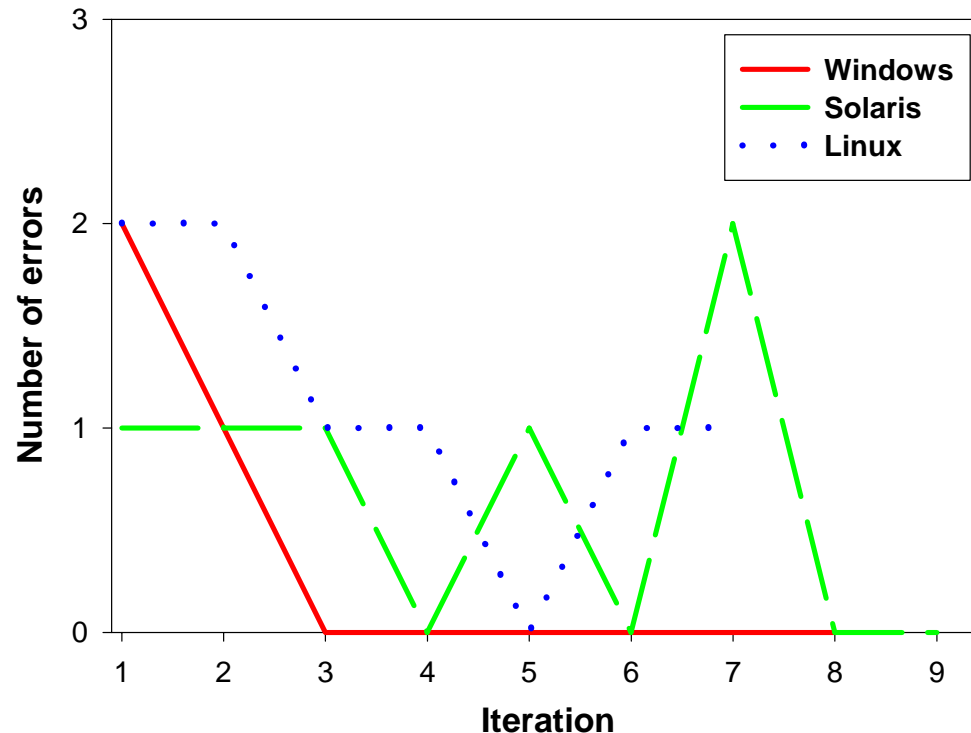
- Human error rates during maintenance of software RAID system
 - participants attempt to repair RAID disk failures
 - » by replacing broken disk and reconstructing data
 - each participant repeated task several times
 - data aggregated across 5 participants

Error type	Windows	Solaris	Linux
Fatal Data Loss			 
Unsuccessful Repair			
System ignored fatal input			
User Error - Intervention Required		 	
User Error - User Recovered		   	 
Total number of trials	35	33	31



Learning from experiments

- Errors occur despite experience:



- Training and familiarity don't eliminate errors
 - types of errors change: mistakes vs. slips/lapses



- System design affects error-susceptibility

Outline

- Human error and computer system failures
- **A theory of human error**
- Human error and accident theory
- Addressing human error



A theory of human error

(distilled from J. Reason, Human Error, 1990)

- **Preliminaries: the three stages of cognitive processing for tasks**

- 1) planning

- » a goal is identified and a sequence of actions is selected to reach the goal

- 2) storage

- » the selected plan is stored in memory until it is appropriate to carry it out

- 3) execution

- » the plan is implemented by the process of carrying out the actions specified by the plan



A theory of human error (2)

- Each cognitive stage has an associated form of error
 - **slips**: execution stage
 - » incorrect execution of a planned action
 - » example: miskeyed command
 - **lapses**: storage stage
 - » incorrect omission of a stored, planned action
 - » examples: skipping a step on a checklist, forgetting to restore normal valve settings after maintenance
 - **mistakes**: planning stage
 - » the plan is not suitable for achieving the desired goal
 - » example: TMI operators prematurely disabling HPI pumps



Origins of error: the GEMS model

- **GEMS: Generic Error-Modeling System**
 - an attempt to understand the origins of human error
- **GEMS identifies three *levels* of cognitive task processing**
 - **skill-based**: familiar, automatic procedural tasks
 - » usually low-level, like knowing to type “ls” to list files
 - **rule-based**: tasks approached by pattern-matching from a set of internal problem-solving rules
 - » “observed symptoms X mean system is in state Y”
 - » “if system state is Y, I should probably do Z to fix it”
 - **knowledge-based**: tasks approached by reasoning from first principles
 - » when rules and experience don’t apply



GEMS and errors

- **Errors can occur at each level**
 - **skill-based:** slips and lapses
 - » usually errors of inattention or misplaced attention
 - **rule-based:** mistakes
 - » usually a result of picking an inappropriate rule
 - » caused by misconstrued view of state, over-zealous pattern matching, frequency gambling, deficient rules
 - **knowledge-based:** mistakes
 - » due to incomplete/inaccurate understanding of system, confirmation bias, overconfidence, cognitive strain, ...
- **Errors can result from operating at wrong level**
 - humans are reluctant to move from RB to KB level even if rules aren't working



Error frequencies

- In raw frequencies, **SB** >> **RB** > **KB**
 - 61% of errors are at skill-based level
 - 27% of errors are at rule-based level
 - 11% of errors are at knowledge-based level
- But if we look at *opportunities* for error, the order reverses
 - humans perform vastly more SB tasks than RB, and vastly more RB than KB
 - » so a given KB task is more likely to result in error than a given RB or SB task

Error detection and correction

- **Basic detection mechanism is self-monitoring**
 - periodic attentional checks, measurement of progress toward goal, discovery of surprise inconsistencies, ...
- **Effectiveness of self-detection of errors**
 - SB errors: 75-95% detected, avg 86%
 - » but some lapse-type errors were resistant to detection
 - RB errors: 50-90% detected, avg 73%
 - KB errors: 50-80% detected, avg 70%
- **Including correction tells a different story:**
 - SB: ~70% of all errors detected and corrected
 - RB: ~50% detected and corrected
 - KB: ~25% detected and corrected



Outline

- Human error and computer system failures
- A theory of human error
- **Human error and accident theory**
- Addressing human error

Human error and accident theory

- Major systems accidents (“normal accidents”) start with an accumulation of latent errors
 - most of those latent errors are human errors
 - » latent slips/lapses, particularly in maintenance
 - example: misconfigured valves in TMI
 - » latent mistakes in system design, organization, and planning, particularly of emergency procedures
 - example: flowcharts that omit unforeseen paths
 - invisible latent errors change system reality without altering operator’s models
 - » seemingly-correct actions can then trigger accidents

Accident theory (2)

- **Accidents are exacerbated by human errors made during operator response**
 - RB errors made due to lack of experience with system in failure states
 - » training is rarely sufficient to develop a rule base that captures system response outside of normal bounds
 - KB reasoning is hindered by system complexity and cognitive strain
 - » system complexity prohibits mental modeling
 - » stress of an emergency encourages RB approaches and diminishes KB effectiveness
 - system visibility limited by automation and “defense in depth”
 - » results in improper rule choices and KB reasoning



Outline

- Human error and computer system failures
- A theory of human error
- Human error and accident theory
- **Addressing human error**
 - general guidelines
 - the ROC approach: system-level undo

Addressing human error

- **Challenges**

- humans are inherently fallible and errors are inevitable
- hard-to-detect latent errors can be more troublesome than front-line errors
- human psychology must not be ignored
 - » especially the SB/RB/KB distinction and human behavior at each level

- **General approach: error-tolerance rather than error-avoidance**

“It is now widely held among human reliability specialists that the most productive strategy for dealing with active errors is to focus upon controlling their consequences rather than upon striving for their elimination.”

(Reason, p. 246)



The Automation Irony

- Automation is not the cure for human error
 - automation addresses the easy SB/RB tasks, leaving the complex KB tasks for the human
 - » humans are ill-suited to KB tasks, especially under stress
 - automation hinders understanding and mental modeling
 - » decreases system visibility and increases complexity
 - » operators don't get hands-on control experience
 - » rule-set for RB tasks and models for KB tasks are weak
 - automation shifts the error source from operator errors to design errors
 - » harder to detect/tolerate/fix design errors



Building robustness to human error

- **Discover and correct latent errors**
 - must overcome human nature to wait until emergency to respond
- **Increase system visibility**
 - don't hide complexity behind automated mechanisms
- **Take errors into account in operator training**
 - include error scenarios
 - promote exploratory trial & error approaches
 - emphasize positive side of errors: learning from mistakes



Building robustness to human error

- **Reduce opportunities for error (Don Norman):**
 - get good conceptual model to user by consistent design
 - design tasks to match human limits: working memory, problem solving abilities
 - make visible what the options are, and what are the consequences of actions
 - exploit natural mappings: between intentions and possible actions, actual state and what is perceived, ...
 - use constraints to guide user to next action/decision
 - design for errors. Assume their occurrence. Plan for error recovery. Make it easy to reverse action and make hard to perform irreversible ones.
 - when all else fails, standardize: ease of use more important, only standardize as last resort



Building robustness to human error

- **Acknowledge human behavior in system design:**
 - interfaces should allow user to explore via experimentation
 - to help at KB level, provide tools do experiments/test hypotheses without having to do them on high-risk irreversible plant. Or make system state always reversible.
 - provide feedback to increase error observability (RB level)
 - at RB level, provide symbolic cues and confidence measures
 - for RB, try to give more elaborate, integrated cues to avoid “strong-but-wrong” RB error
 - provide overview displays at edge of periphery to avoid attentional capture at SB level
 - simultaneously present data in forms useful for SB/RB/KB
 - provide external memory aids to help at KB level, including externalized representation of different options/schemas



Human error: the ROC approach

- ROC is focusing on system-level techniques for human error tolerance
 - complimentary to UI innovations
- **Goal: provide forgiving operator environment**
 - expect human error and tolerate it
 - allow operator to experiment safely, test hypotheses
 - make it possible to detect and fix latent errors
- **Approach: undo for system administration**



Repairing the Past with Undo

- **The Three R's: undo meets time travel**
 - **Rewind:** roll system state backwards in time
 - **Repair:** fix latent or active error
 - » automatically or via human intervention
 - **Redo:** roll system state forward, replaying user interactions lost during rewind
- **This is not your ordinary word-processor undo!**
 - allows sysadmin to go back in time to fix latent errors after they're manifested



Undo details

- **Examples where Undo would help:**
 - reverse the effects of a mistyped command (`rm -rf *`)
 - roll back a software upgrade without losing user data
 - retroactively install virus filter on email server; effects of virus are squashed on redo
- **The 3 R's vs. checkpointing, reboot, logging**
 - checkpointing gives Rewind only
 - reboot may give Repair, but only for "Heisenbugs"
 - logging can give all 3 R's
 - » but need more than RDBMS logging, since system state changes are interdependent and non-transactional
 - » 3R-logging requires careful dependency tracking, and attention to state granularity and externalized events



Summary

- **Humans are critical to system dependability**
 - human error is the single largest cause of failures
- **Human error is inescapable: “to err is human”**
 - yet we blame the operator instead of fixing systems
- **Human error comes in many forms**
 - mistakes, slips, lapses at KB/RB/SB levels of operation
 - but is nearly always detectable
- **Best way to address human error is tolerance**
 - through mechanisms like undo
 - human-aware UI design can help too

