CSC 221 – Introduction to Software Engineering
debugging, bug finding and bug avoidance
Part 1

Alan Dix

www.hcilbook.com/alan/teaching/CSC221/

outline
• part 1 – general issues and heuristics
• part 2 – the system as it is
  • understand and document
• part 3 – locating and fixing bugs
• part 4 – bug engineering
  • design to expose, avoid and recover
  • including fail-fast programming

why is debugging difficult
• complexity of computation
• non locality of effect
• hidden state
• complex interactions
  
sounds familiar?

in systems
bugs are typically about interactions and unforeseen circumstances

debug cycle?

code

debug

does it work?

NO

YES

OK done?
bug seeking cycle

exposing bugs
general heuristics

- **discover bugs**
  - testing (random, targeted, boundary)
  - restrict environment (memory, load)
- **uncover bugs**
  - force bug to (re)appear (e.g. bad values, fail fast)
  - record data/environment to recreate bug
- **recover from bugs**
  - make system resilient, but log problems
... but don’t cover up bugs
  - quick fixes leave latent bugs

exposed bugs

general heuristics

- **isolate bugs**
  - simplify, factor
  - monitor, record
- **understand bugs**
  - why did it happen?
  - is the apparent bug the real one (non-locality)
... and only then
- **fix them ...**
  - but ...

exposed bugs

general heuristics

- **serious?**
  - frequent
  - costly
  - hard to recover

... and only then
- **fix them ...**
  - but ...

cost benefit

damage limitation
low cost ways to reduce impact

- **effect of bug**
  - serious
  - minor

- **serious?**
  - yes
  - no

- **cost to fix**
  - high
  - low

- **live with**
  - may get worse – fix!

- **fix!**