

the intended environment

response to human error

#### How the lessons learnt can be related to software in a more general context Safety is not the same thing as Reliability "Operates desirably" is not the same as "Operates as specified" Causes of *unsafe* behaviour include not only: Causes of *undesirable* behaviour include not only: - Malfunction (ie failure to - Malfunction (ie the traditional operate as specified) "bug") as also . ... but also . Human error Bad response to user error Operation outside intended conditions Bad response to operation outside intended conditions Unintended environmental Bad environmental interinteractions dependencies - Inherently bad specifications Inherently unsafe specifications

#### IV&V Australia Pty Ltd The independent software testing specialists

# How the lessons learnt can be related to software in a more general context

- Defining "safe" can be difficult
- "Safety" is a property of systems not software
- Public expectations for safety are far greater than realistically achievable reliability
- Fully defining the desired behaviour can be difficult
- "Desirable" behaviour is a property of systems as a whole, not just particular software
- User expectations extend well beyond just performing the specified functionality



#### TVGV Australia Fty Ltu The independent sortware testing specialists

# Fully defining the desired behaviour can be difficult ...

- Specifications are never complete there are usually implied requirements:
  - The system doesn't mislead its users
  - The system doesn't impose gratuitous restrictions on its users
  - Screens and printouts aren't visually corrupted
  - Screens and printouts don't have spelling mistakes
- · Specifications often have implications which are not initially obvious
  - Bad interaction between functions
  - Missing functionality
  - "Just not right"

Bugs can be related to bad, missing, or implied requirements



#### IVAV Australia Ptv Ltd. The independent software testing specialists

# "Desirable" behaviour is a property of systems as a whole, not just particular software ...

- "Good" software should not make unnecessary assumptions about its environment
  - Performance and availability of hardware
  - How other parts of the system are configured
- · "Good" software should not unnecessarily affect its environment
  - Change configuration settings of other software
  - Affect operation of other software
  - Have additional unintended output

Bugs can be related to poor interaction with the rest of the system



# User expectations extend well beyond just

# performing the specified functionality ...

- The software should respond gracefully to error
  - User error
  - Hardware error
- The software should respond gracefully when operated outside its intended environment or conditions
  - Degrade rather than fail
  - Warn the user of reduced capability
- The software should fail gracefully
  - Visibly and transparently
  - With no unnecessary consequential damage

Bugs can be related to poor response to error, misuse, or failure



#### The independent solve

#### The role of testers

- Functional testing is supposed to find "bugs" – that is, failures to operate as specified
- However, it often uncovers undesirable characteristics that are not specifically related to the spec
- When these are reported, they are often dismissed as "not a bug", and outside the role of the tester
- Testers need a better way to categorise these problems so that they are not dismissed

"It is like that by design"

"The users will never do that"

"It worked in my environment"

"You haven't configured it correctly"

"Read the user manual"

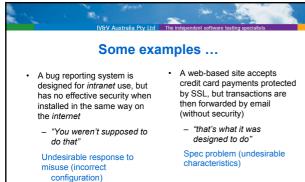


### A classification of "bugs"

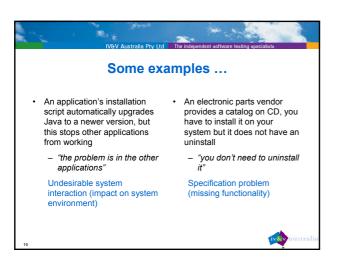
- Non-conformance to spec
  - Written specification (the traditional "bug")
  - Implied specification
- · Specification problem
  - Undesirable characteristics
  - Missing functionality
  - ,
- Undesirable system interaction
  - Sensitivity to system environment
  - Impact on system environment

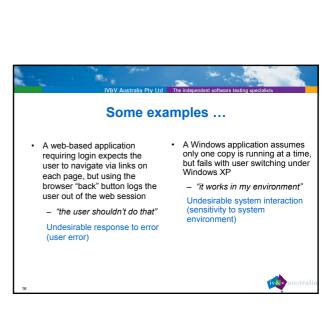
- Undesirable response to error
  - User error
  - Hardware error
- Undesirable response to misuse
  - Unintended use scenarios
  - Incorrect configuration
- Undesirable response to system failure





Some examples ... A test system for some prototype A multi-user operating system hardware tests a data bus by reading and writing back static leaves all open files truncated to zero length after a crash data, but the test passes even (even unchanged files) when the data bus is not - "it won't happen in practice" physically connected "it's not possible" Undesirable response to system failure Undesirable response to error (hardware error)





# So ... Is it a Bug? • Many "bugs" found by functional testing are NOT spec violations, and are often dismissed as not functional test issues • However they can be due to: - bad, missing, or implied requirements - bad interaction with the rest of the system - poor response to error, misuse, or failure \*\*Report them as such!\*\*