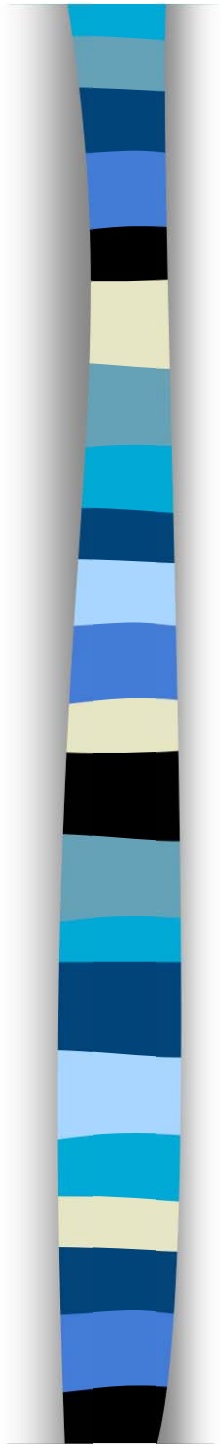


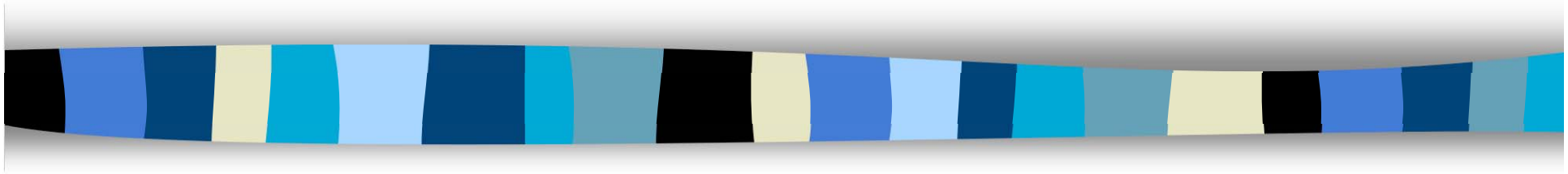
Dependability,

Diversity,

Disaster –

Whatever next?

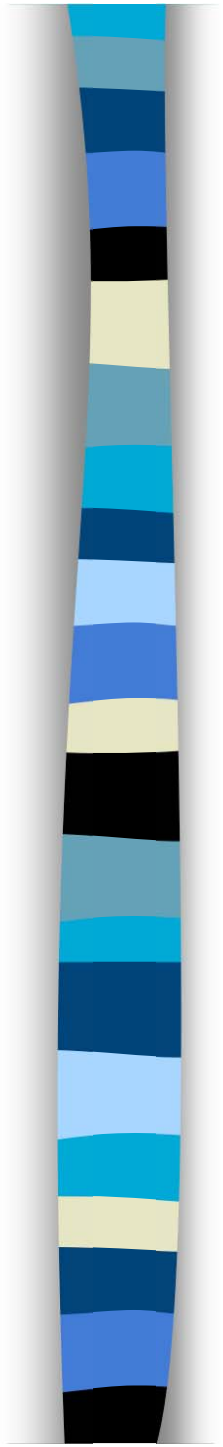
Tom Anderson



Centre for Software Reliability

School of Computing Science

University of Newcastle, UK



Dependability

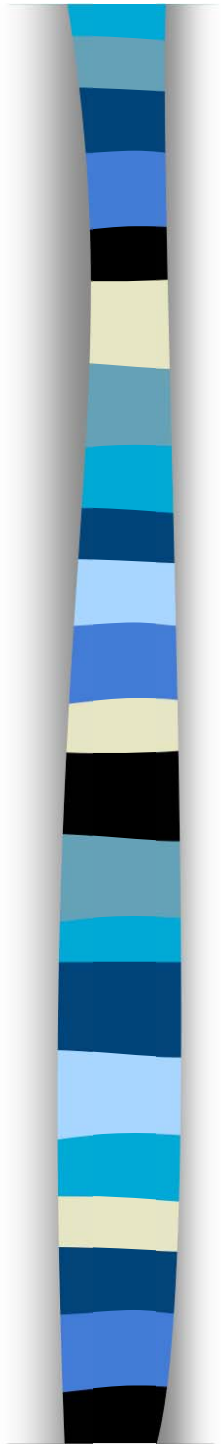
What it is and how to get it.

Diversity

Essential role and contribution.

Disasters

Past, present and looming.



Whatever next?

- Ambient/Pervasive IT services
  - comms, data, cycles ever cheaper
  - ubiquitous service provision
  - nano-bots
  - heterogeneous, dynamically self configured, mobile networks of autonomous activities
  - societal and technical system interaction
- Increasing
  - opportunities for disaster
  - need for dependability (and trust)
- Interfaces
  - component, system, SoS, S&S
- Protection
  - wrappers, interceptors

Dependability

- Of any delineated system
- System does what it's supposed to do (and doesn't do what it shouldn't)
- Whatever:
  – safety, security, reliability, …
- [Mostly]

Achieving dependable systems

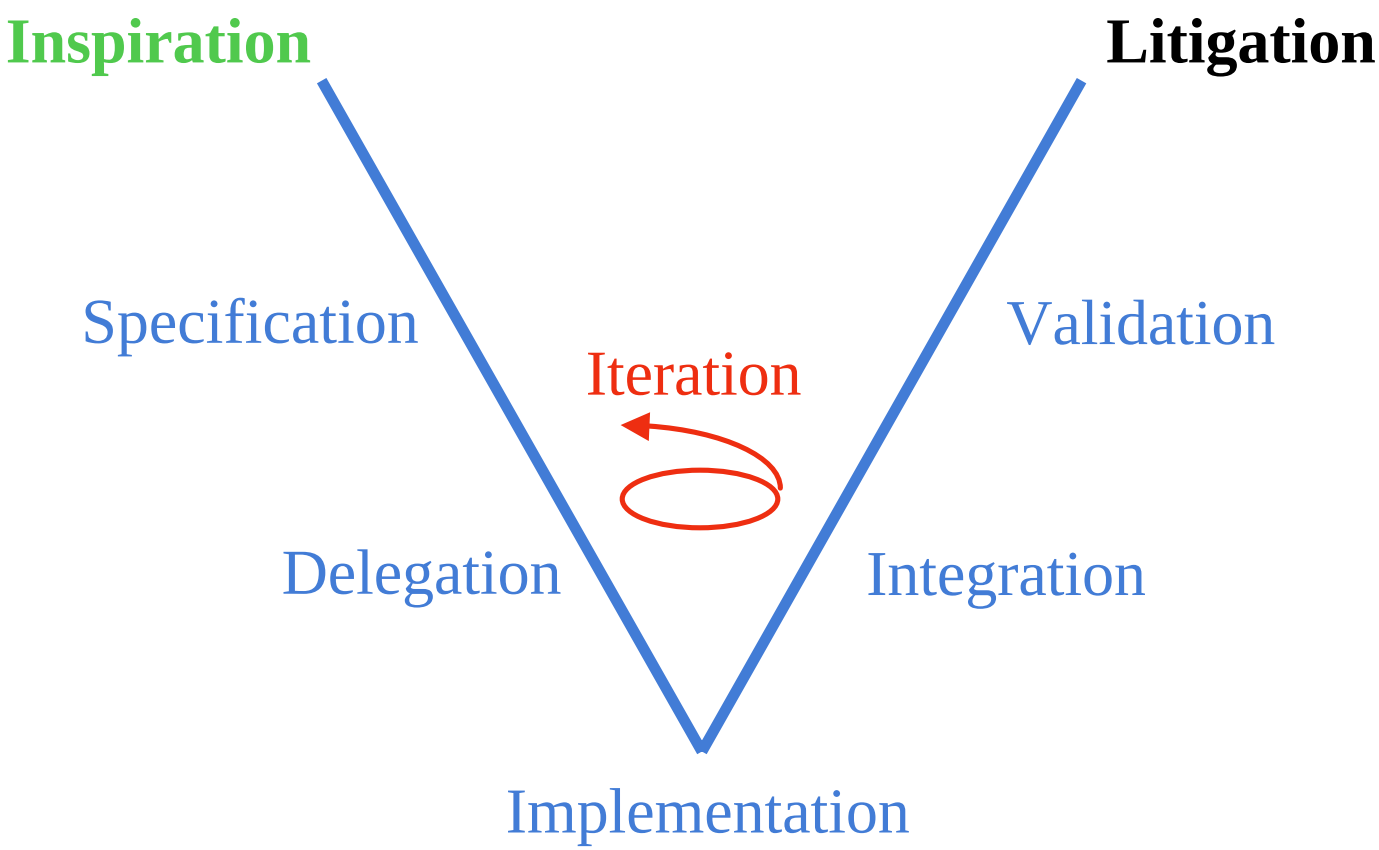
The subject of dependability needs to be treated holistically, to:

- cover all potentially relevant dependability attributes, since a balance is invariably needed
- allow for all types of faults – e.g. intermittent h/w faults, s/w specification faults, human-machine interaction faults (including incompetence and attacks)
- make appropriate use of the primary ways of achieving dependability (rigorous design, fault tolerance, verification and validation, system evaluation)
- cope with dependability threats (fault/error/failure “chains”) between systems, via system interaction, composition and creation
- overcome various linguistic and cultural divides (e.g. dependability/security/survivability/trustworthiness) - without necessarily imposing a common culture and terminology

In short

- Design and build with care
- Incorporate defences
- Find and fix mistakes
- Measure and evaluate

The *real* V diagram



Diversity

Diversity in development

– e.g. inspection and testing and static checking.

Diversity in operation

– temporal, physical and algorithmic redundancy.

The alternative is **total** dependency on a single mechanism.

Even Dijkstra noted that when you have to add up a table of numbers, performing a check by adding in the opposite direction was worthwhile (thereby following in the footsteps of Lardner, Turing and von Neumann).

In short

- Belt and braces
- Eggs in more than one basket
- Two heads better than one

Disasters

How long have you got?

- Past
- Present
- Looming

Past disasters

- Utility bills [0.0 or ∞] - lost goodwill
- Bank of New York (1985) - lost $5M
- Ariane V (1996) - lost mission ($150M)
- UK ATC NERC (2003) - lost years
- Therac 25 (1986) - lost lives (3)

Present disaster

Proprietary word processor

(maintaining a great tradition of lousy user interfaces e.g. IBM 360 JCL, Unix, C, the teco editor)

Looming potential disasters

- Health care and hospital IT systems
- Community services
- National infrastructure interdependencies
- Military C^{n}I
- Genetic programmers

Whatever next?

- Interacting networks of
- interacting IT service utilities
- implemented as interacting sub-systems
- which interact with individuals and organisations
- at all levels

Future systems

Risk of an explosion in complexity

Structural overload and melt-down

Near chaotic systems

- delivering ever more services
- which must be
  - accessible, dependable, trusted
- to individuals and to society

Current situation

How well are we doing?

User interfaces to

- Gadgets: VCR, DVD, digital camera
- Applications: e-mail, calendar, word-processing

Internal system interfaces

Specialist HCI applications

One approach

Don’t fight it

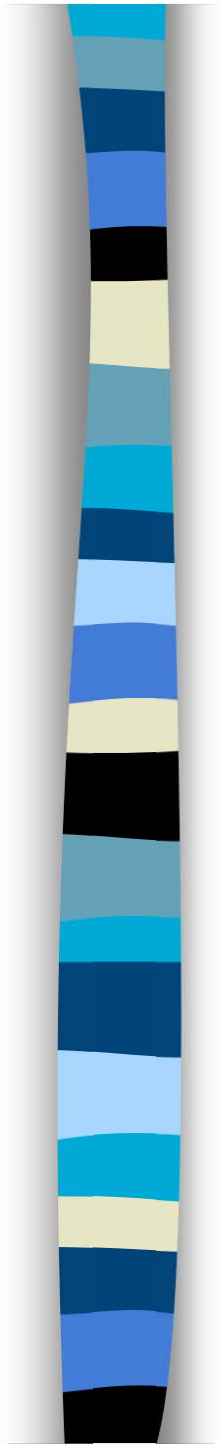
- go with the flow – exploit the new paradigm

Autonomy in cyber-space (the old wild west way?)

- B2B, P2P, publish/subscribe, dynamic contracts, …
- No architecture, no structure, only infrastructure

But all processes must be “wrapped”, enforced by

- Gatekeepers and patrol officers
  (authorised to quarantine or kill rogues)



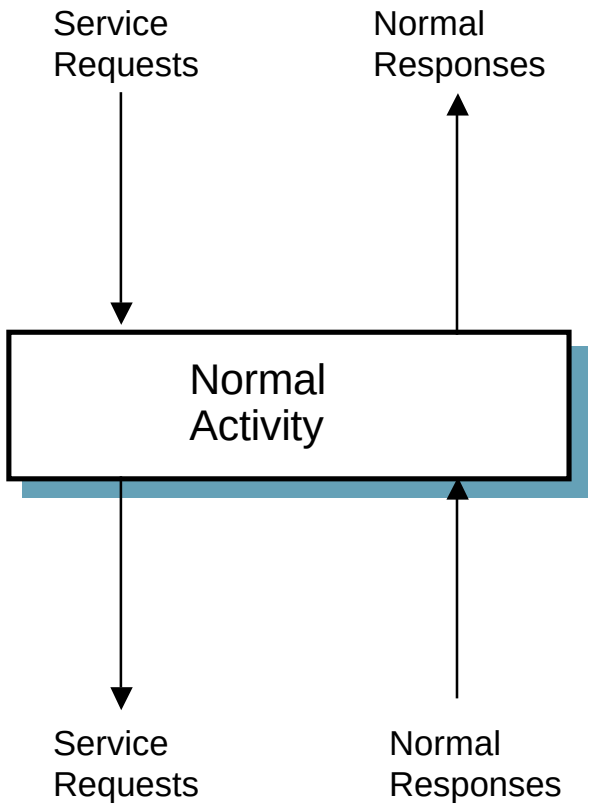
Wrapper technology

A wrapper component envelopes a process so that interactions are intercepted, and may be modified (internal process components can also be wrapped).

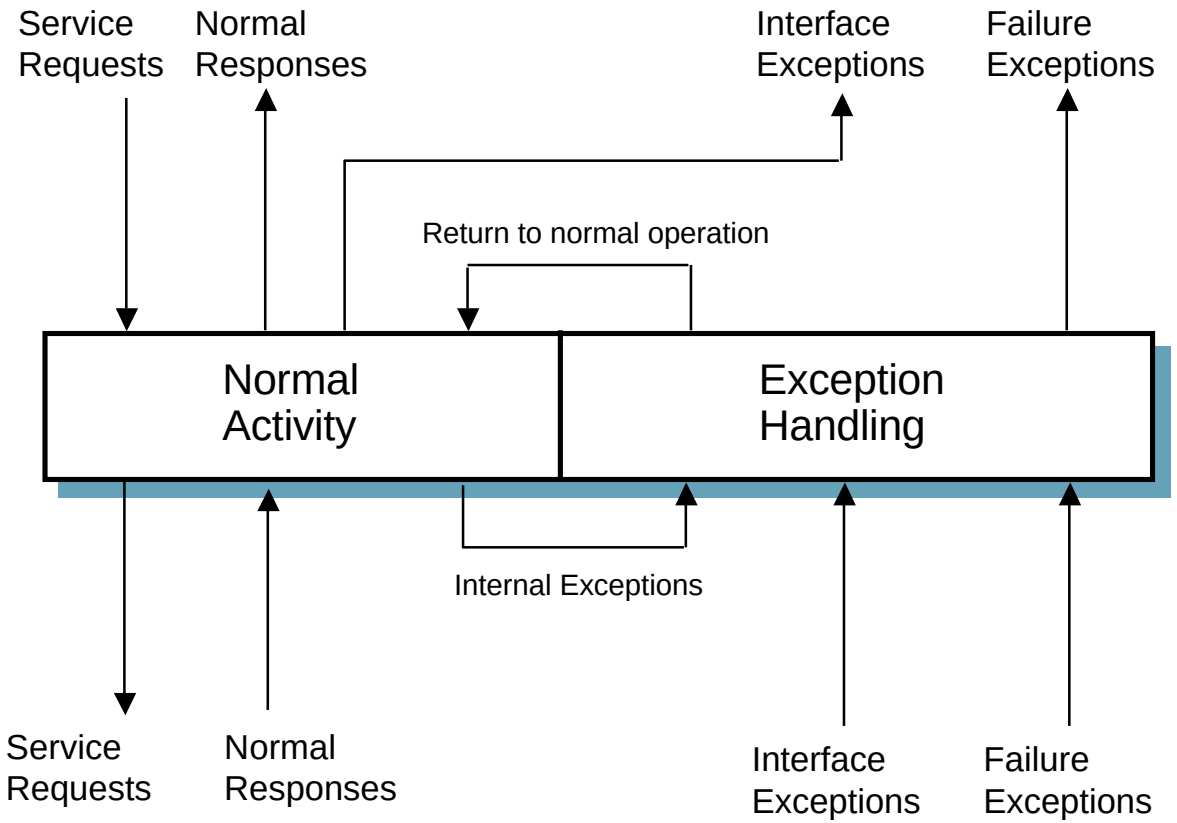
Human interactors must also operate via a wrapper

- protects you from the system
- protects the system from you
- can support and assist your interactions

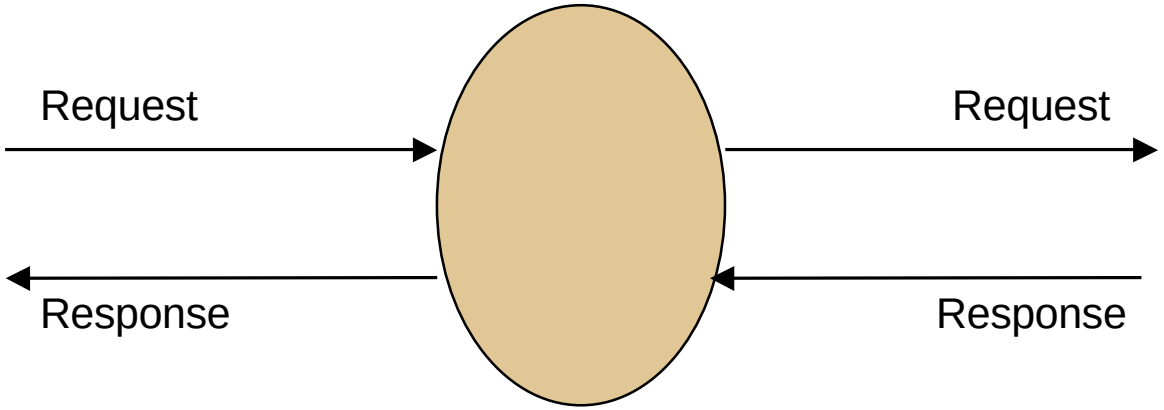
Basic Component



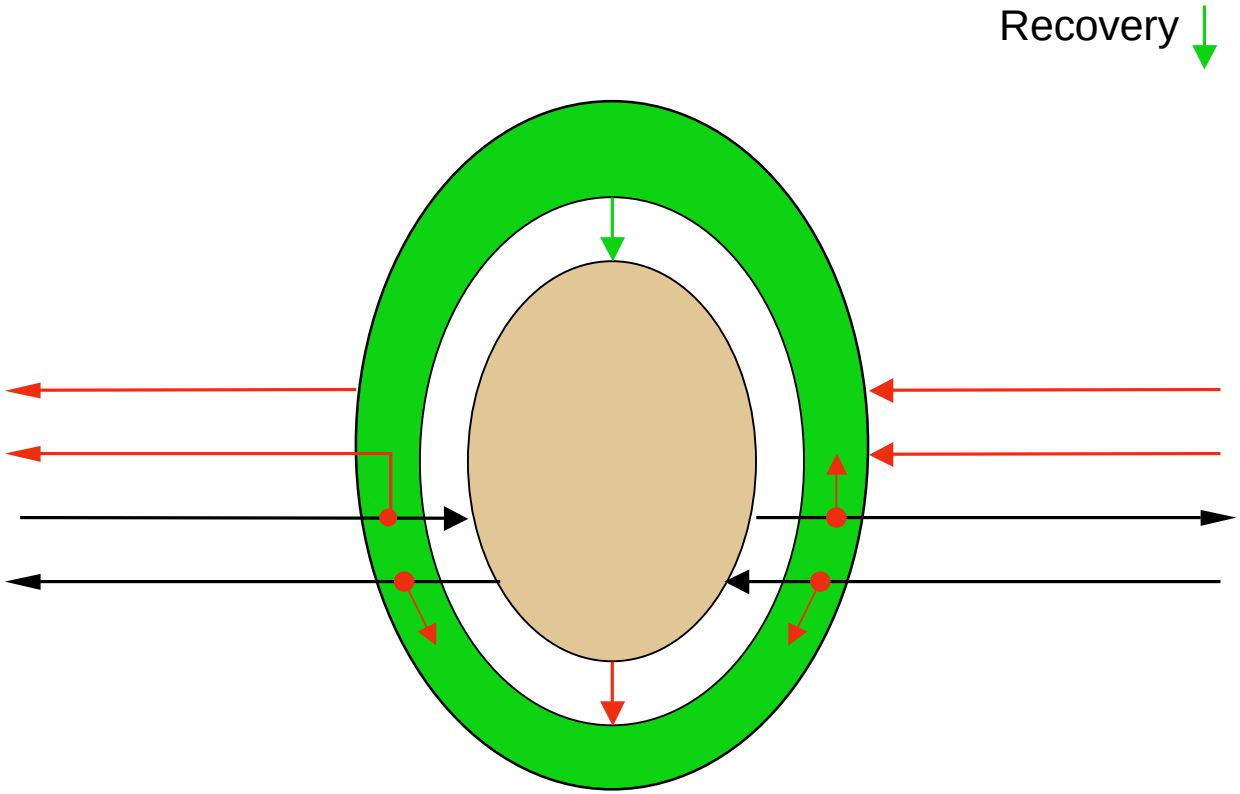
Idealised F.T. Component



Basic process



Wrapped process



Request/Response

Exception

Interface/Failure/Internal