
# Avoiding systemic catastrophes

## Predicting behaviour under high load through reasoning about timeliness and imperfection

Neil Davies

Nov 2022

# Success is more than “function”



My (initial) motivation:

- Watched too many good ideas “fail”
  - Unable to survive contact with the “real world”
  - Why?

Forget about “MBA-like” issues (marketing etc)

- They were *hazards* that manifest themselves at scale
  - Mitigations required architectural / different algorithmic approach.
  - Hazards were (in hindsight) ones that could have been resolved earlier.

**How to capture and qualify/quantify such hazards early**

# Fitness-for-purpose implies timeliness



There are *expectations* on all our interactions with any system:

- Phone call connects within *N* seconds
- Vending machine produces coffee with *M* seconds
- …

Yet being 'fit-for-purpose' does NOT imply perfection.

- Phone call can fail once every *N* attempts (e.g 1 in 50, 1 in 200)
- Vending machine can fail (randomly) once every *M* uses.
- …

**Fit-for-purpose: meeting (reasonable) expectations for *outcomes* of interest**

PREDICTABLE NETWORK SOLUTIONS

# ΔQ - Quality Attenuation



Mental Jujutsu move: ***accept and embrace the imperfection***.

Control and manage (design/reasoning) complexity: ***Abstraction / compositionality***

***Improper Random Variables*** (IRVs): (well founded) way of dealing with "rigidly defined areas of doubt and uncertainty"

Δ***Q***: captures timeliness and non-completion (failure) in a single variable / quantity.

Δ***Q:*** Quantifies outcomes.

Outcomes (and their associated timed observables) arose out stochastic process algebras.

# Surviving contact with “Real Word” ™

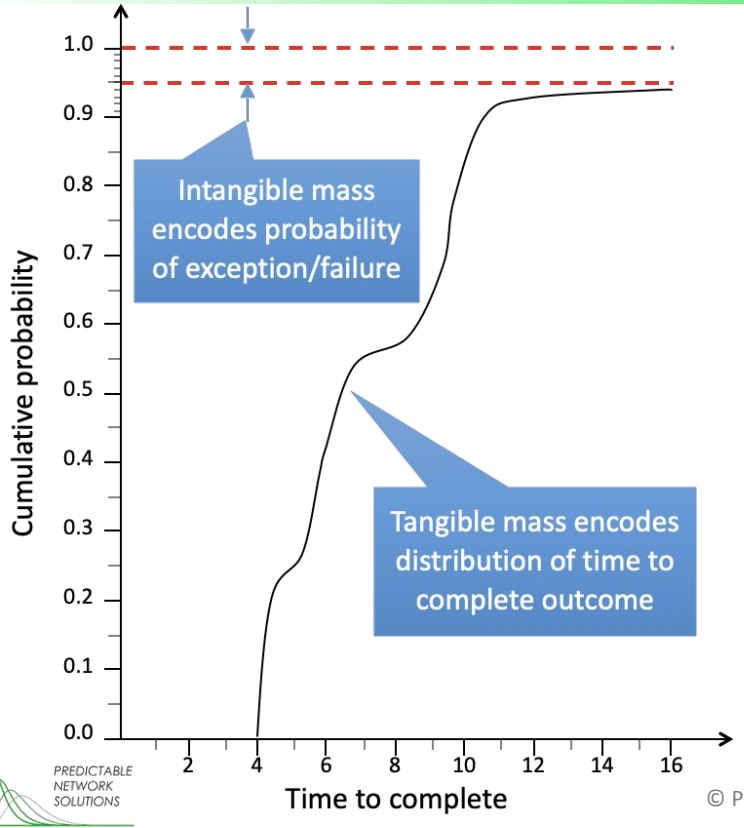


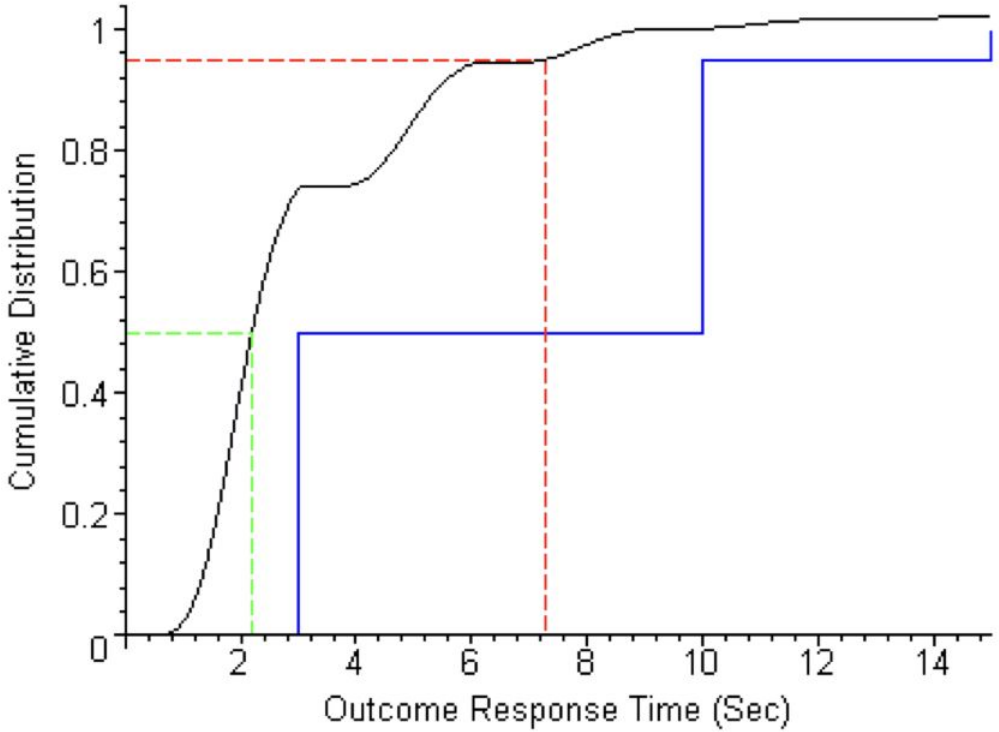
- Can represent more than “ICT” systems - been used to model “cyber-physical” systems.
- Embraces uncertainty, in:
  - Design - “quantified engineering judgement”
  - Implementation - value-related / resource contention effects
  - Operation - Observation v Expectation during system lifetime
- Infidelities and junk: numerical accuracy less important than:
  - Failure to identify *existence of* performance / scalability “cliffs”
    - Qualitative v quantitative infidelities
  - Issues that only exist in the model, not in ‘real’ system
    - Junk

**Knowing the boundaries is key for scaling / economic viability**

# ΔQ: combining the continuous and discrete QTA (Quantitative Timeliness Agreement)





# Why an (algebraic) Framework?



- All about the Humans / human organisation
  - Short term memory: 7±2 things at any one time
  - Teams & ‘silo-isation’ - information doesn’t flow
  - Critical interactions and (resource) competitions
    - only found late in system development
    - ‘Institutional knowledge’ lost over time
  - **Discovering issues late is costly** (to the point of fatal)
- Need tool support - not just for the design for lifecycle
  - Capture ‘causality’ - information dependencies
  - Capture ‘resource contention’ - scope/demand creep
  - Contain the risk of not meeting critical outcomes

As part of CI (continuous integration)

**Infrastructure needs to ‘run forever’**

**Use automation to reduce the demand on the most precious resource - humans**

Top Down
Bottom Up

## High level outcomes

- Easy to explain to stakeholders

## Data network and computation effects

- Incorporate Real World issues ($\Delta$Q of networks)
- Make resources finite ($\Delta$Q from contention)

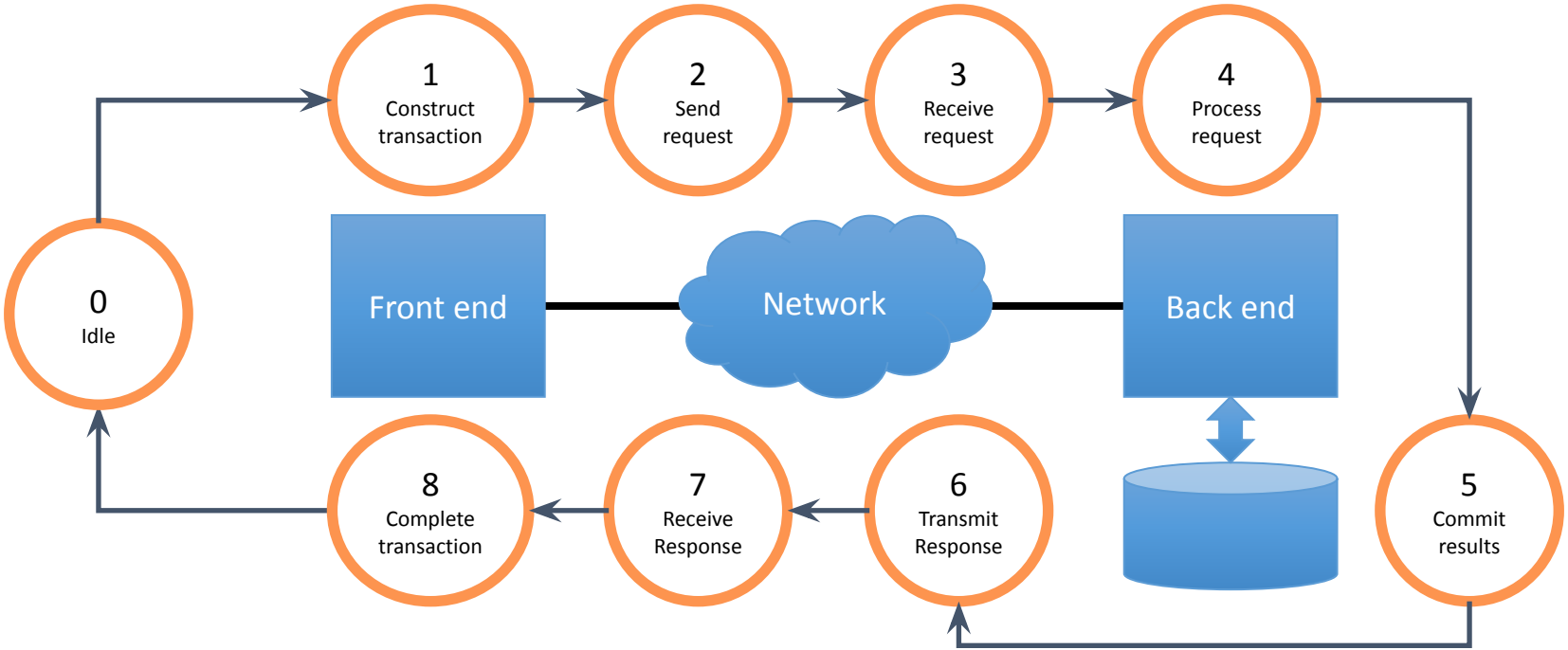
## Refine

- Either to trivial things - “know how to do that”
- Needs further analysis - part of the ongoing work

Desires, Aspirations and Requirements
Feasibility, Resources and time-to-complete

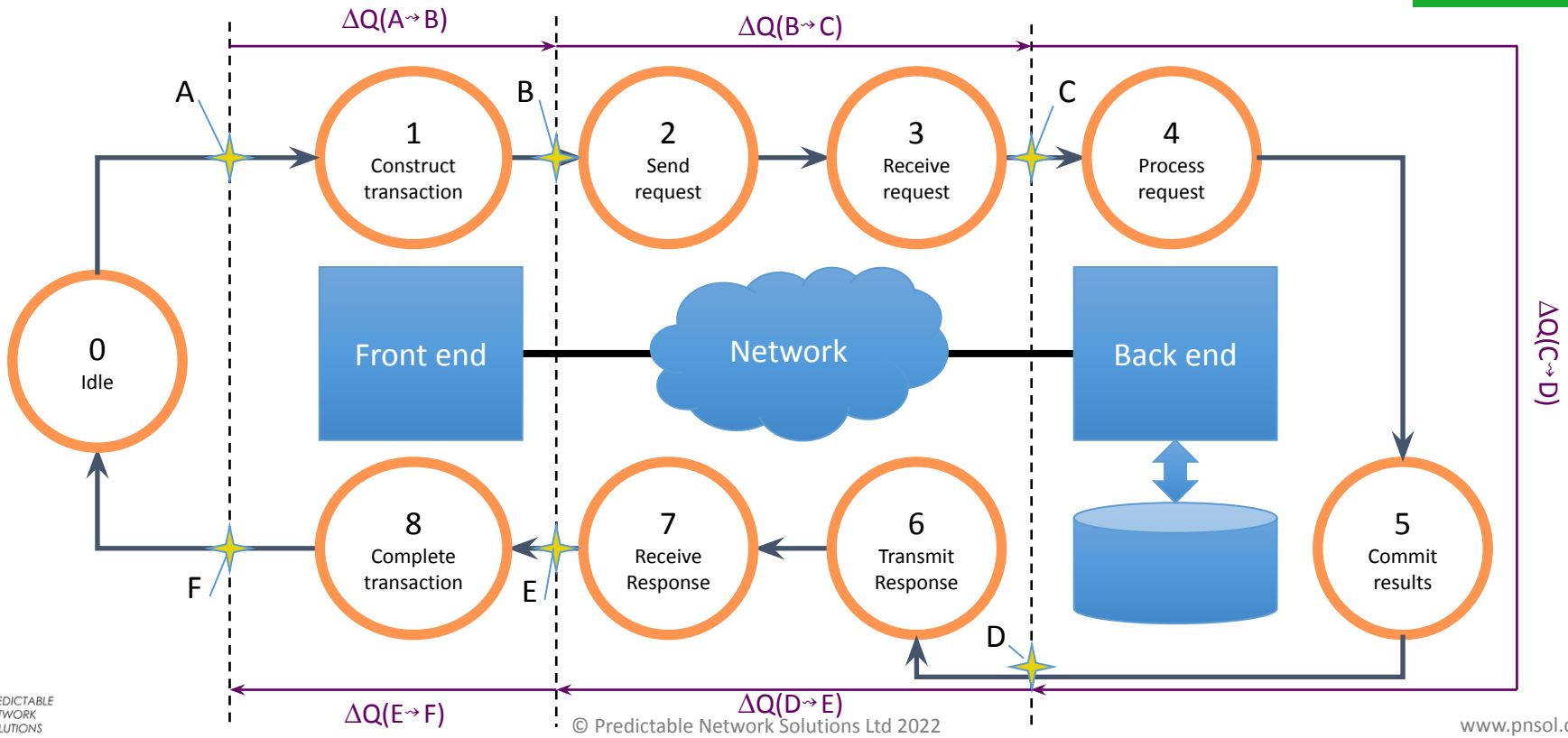
# Generic RPC





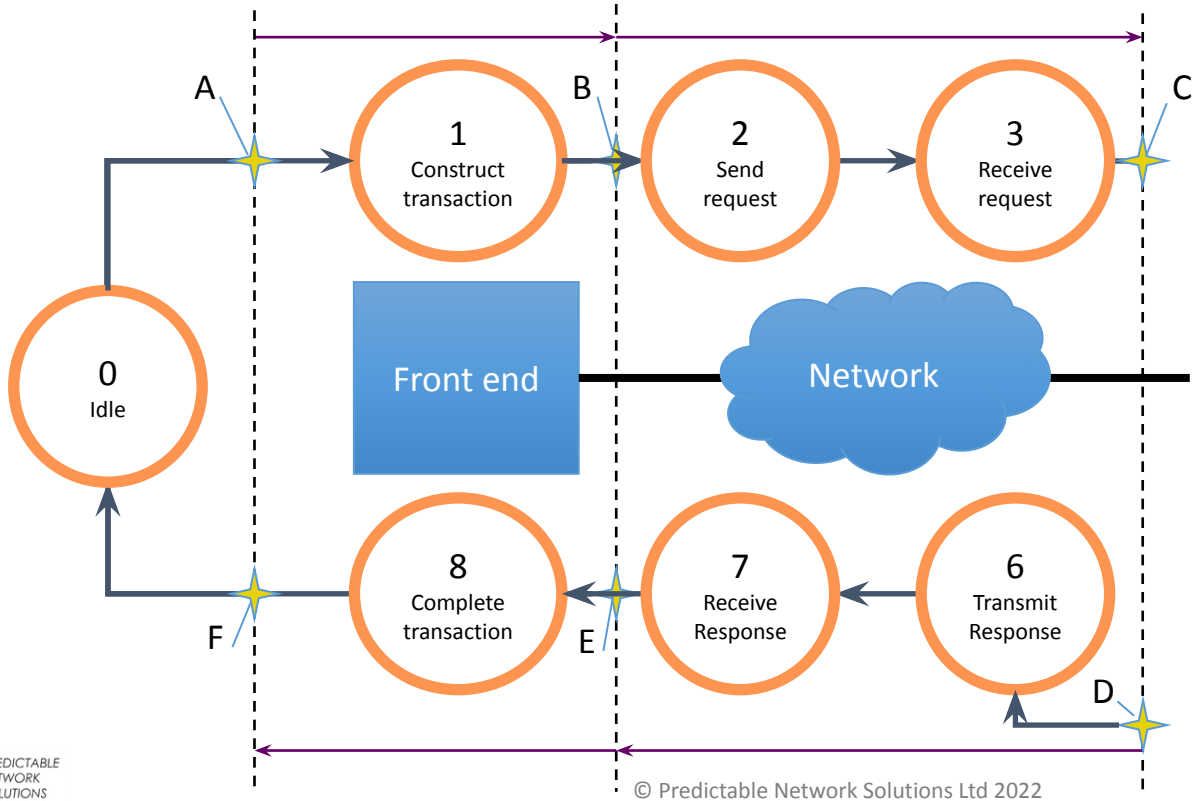
# Generic RPC - observables





# Generic RPC – abstract performance

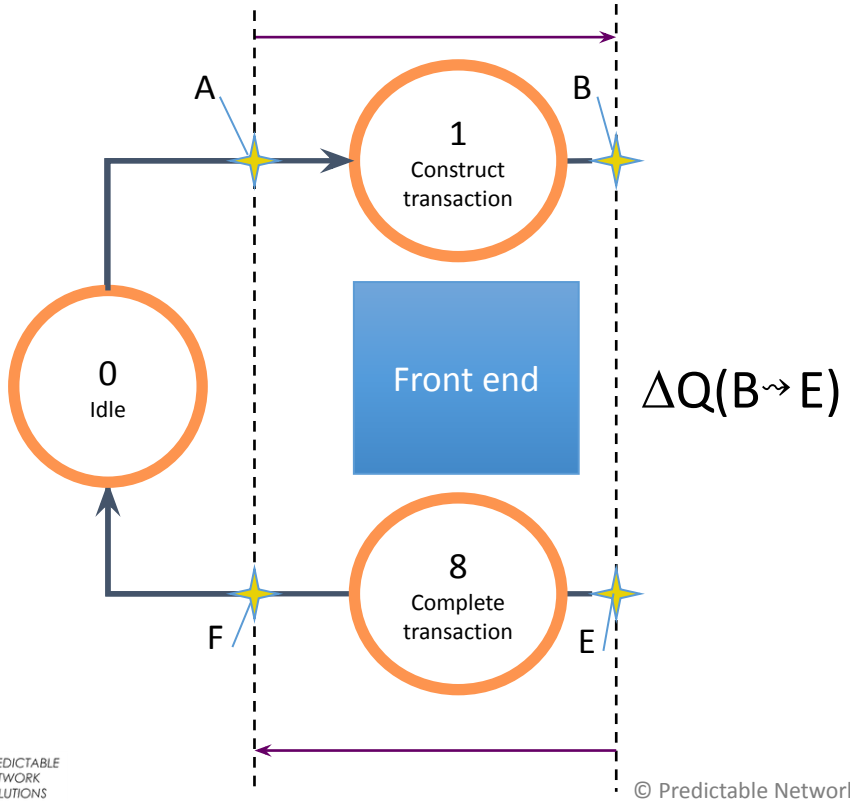





$\Delta Q(C \rightsquigarrow D)$

# Generic RPC – abstract performance

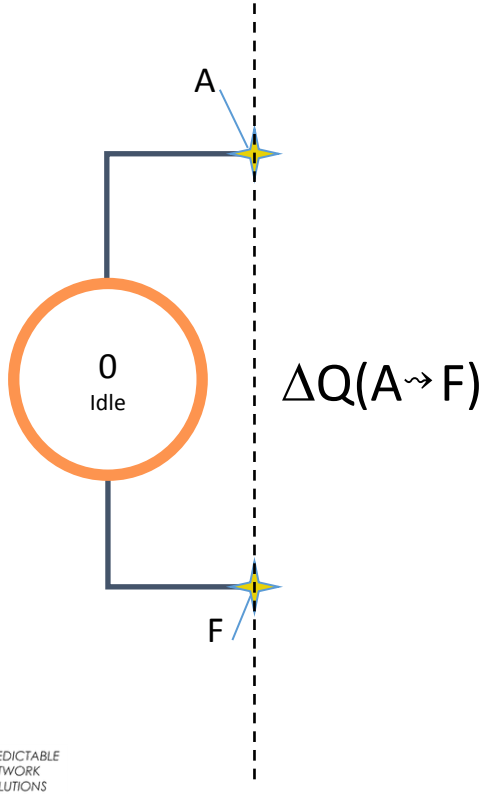





$\Delta Q(B \rightsquigarrow E)$

# Generic RPC – abstract performance

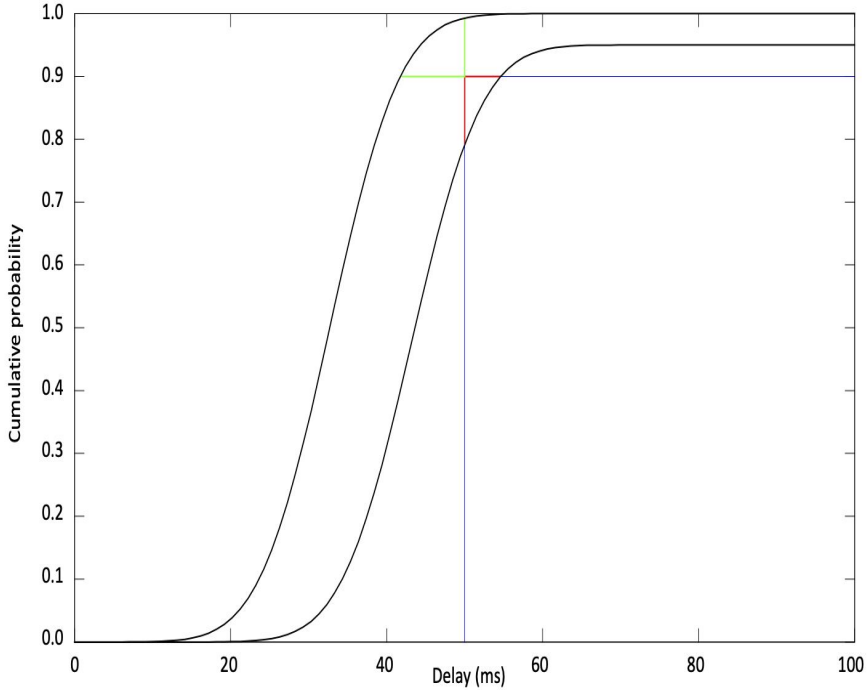




# Avoiding catastrophes - ex ante engineering



- Capturing causality
- Assigning initial ΔQ to key critical outcomes
  - Might be requirements (top down)
  - Might be constraints (bottom up)
  - Might be “engineering judgement” (best guess)
- Understanding the slack and the latent hazards
  - Quantified measure of timeliness / resource hazards



# Identifying root causes - post mortem analysis



Helping people get out of the situations they find themselves in

- (this is where we, PNSol, make most of our income)

Why?

- Typically (tens, hundreds) millions of £/$/€ at stake
- Companies (and investors) hate “stranded assets” of this magnitude

We’ve been standardising the measurement principles in the Broadband Forum

# Conclusions - future work



- ΔQ/Quality Attenuation based delivers insights
  - Making them useful takes subject-matter expertise
  - Needs to be captured as part of continuous integration
  - Biggest win in operational context - transferable insight to isolate issues
- Delivering fitness-for-purpose becomes a lot easier in the ΔQ framework
  - Notions like QTA, Slack and Hazard resonate with other “stakeholders”
- Tool support is in its early days
  - Algebraic based support (ΔQSD) for refinement/abstraction automatically tracking performance invariants
  - High data volume analysis for root-cause analysis