IT Architecture Module

Qualities & Constraints in IT Architecture

Part I –

- Overview
- Availability
- Performance

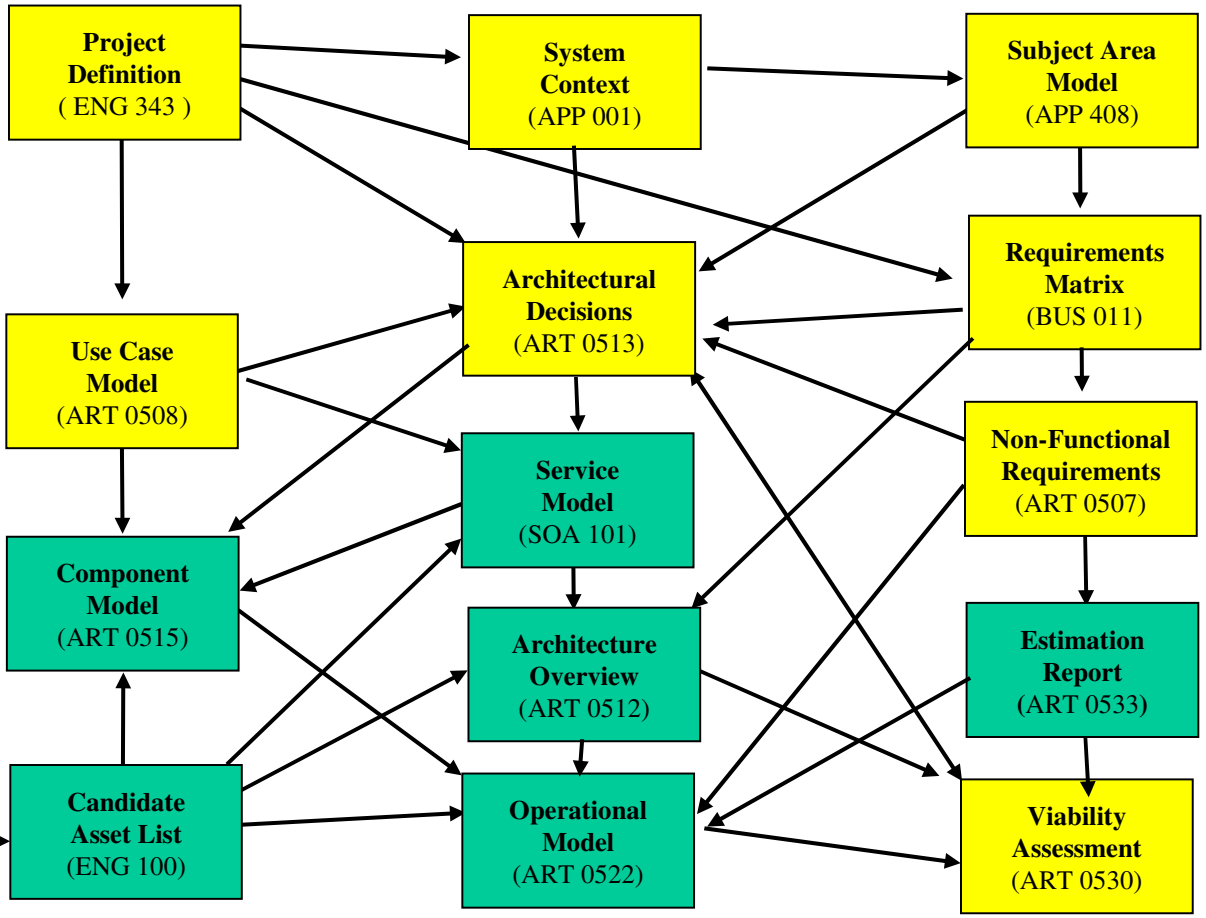
Agenda (Part I - today)

- *Qualities & Constraints in IT Architecture – overview*
  - What are “qualities and constraints” in IT Architecture?
  - Non-Functional Requirements and their quality
- Focus on *Availability*
  - Availability modelling
  - Availability design techniques
- Focus on *Performance*
  - The Performance Engineering Lifecycle
  - Volumetrics
  - Estimation and Modelling
  - Optional exercise

Agenda (Part II – to follow)

- Focus on *Security*
- Focus on *Usability & Accessibility*
- Focus on *Maintainability & Flexibility*

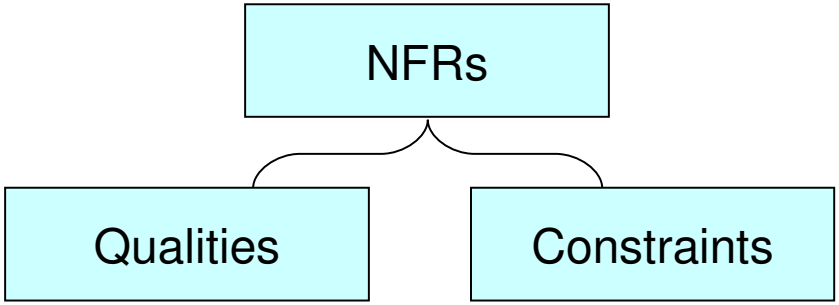
Work Product Dependency Diagram



Assets
Reference Architectures
ITT Solution Brief, Patterns
Redbooks, etc

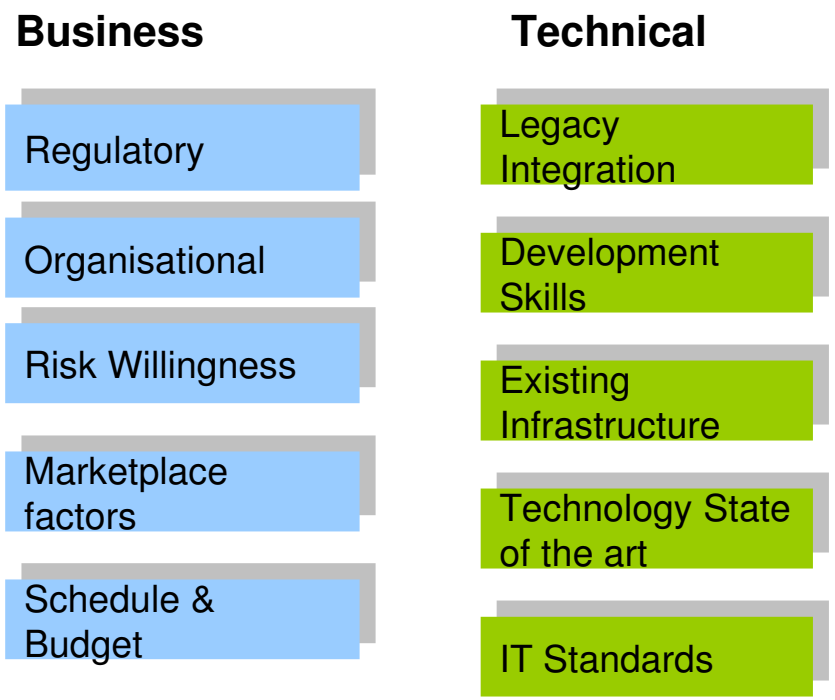
Qualities and Constraints are often referred to as '*Non-Functional Requirements*'

- Non-functional requirements (or NFRs) define the desirable qualities of a system ***and*** the constraints within which the system must be built
  - *Qualities* define the properties and characteristics which the delivered system should demonstrate
  - *Constraints* are the limitations, standards and environmental factors which must be taken into account in the solution



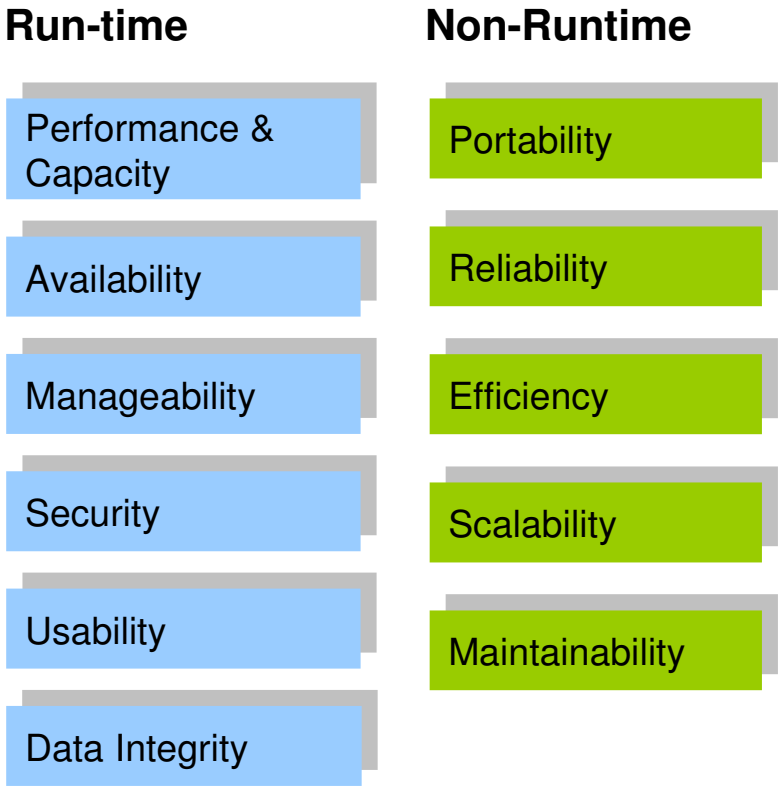
Constraints

- **The business aspects of the project, customer's business environment or IT organization that influence the architecture**
- **The technical environment and prevailing standards that the system, and the project, need to operate within**



Qualities

- **Runtime qualities are 'measurable' properties, often expressed as "Service Level Requirements".**
- **Qualities might also be related to the development, maintenance, or operational concerns that are not expressed at runtime.**



Quality-of-Service "metrics" have an impact on a company's bottom line – consider online services …

- Tangible metrics are ones which can be quantified as a measure of "Loss per transaction":
  - In the online world it's important to do a great job with buyers
  - People leave ".com" sites because of pages being unavailable or too slow
  - Even if a site is available and fast – it is *usable*?
  - Slow sites and/or poor navigation techniques cost companies sales
- Intangible metrics are less quantifiable and require estimation:
  - Consider a web site to be really just an extension of a company's BRAND
  - Visiting a web site is the same as visiting a store with the company's logo on it
  - Even if the experience produces no revenue, it can have an impact on return visits
  - Ideally, a customer should develop a mechanism for taking into account these "soft" costs in order to work out their quality of service requirements

The best technique for reducing the risk of poor quality of service is to consider the qualities from the start

- Build ‘quality’ into the solution starting with early design
  - Understand the risks to the project
  - Conduct quality of service engineering from the first elaboration of the architecture model
  - **Set guidelines for the developers (software & infrastructure)**
  - **Test the application/system at each major stage of development**
  - Make sure that the live support teams will be able to manage quality
- Fix it early, and save money and problems later …

(This is not a requirements engineering module, but …)
Common problems with Non-Functional "requirements"

- **Requirements are often vague or unactionable**
  - They need further elaboration, clarification, investigation (and possibly rejection)
  - It may be possible to derive clear, actionable intentions from them
- **Requirements can be statements of principle or good intention but come with little enforcement**
  - The organisation's governance models are central
- **Once captured, requirements are often treated as "musts" or "givens" whereas in fact they are "tradable" and may need to be challenged**
  - Classic example is "given" technology standards (e.g. "all applications in .NET") or infrastructure constraints ("64kbps links to offices")
- **Requirements are often of poor quality**
  - Watch out for these issues: Unrepresentative, unclear, inaccurate, inconsistent, incomplete or unnecessarily constraining
- **NFRs documents often become "dumping grounds" for things which don't have another home**
  - (regardless of quality or suitability)

In reality, “requirements” are actually “influences” whose characteristics we have to be clear about

- A “requirement” in the widest sense …

  stems from many sources … [**Context**]

  is either an aspiration or a constraint …[**Polarity**]

  may be negotiable (i.e. varying in importance) … [**Strength**]

  may be generic or specific … [**Level of generality**]

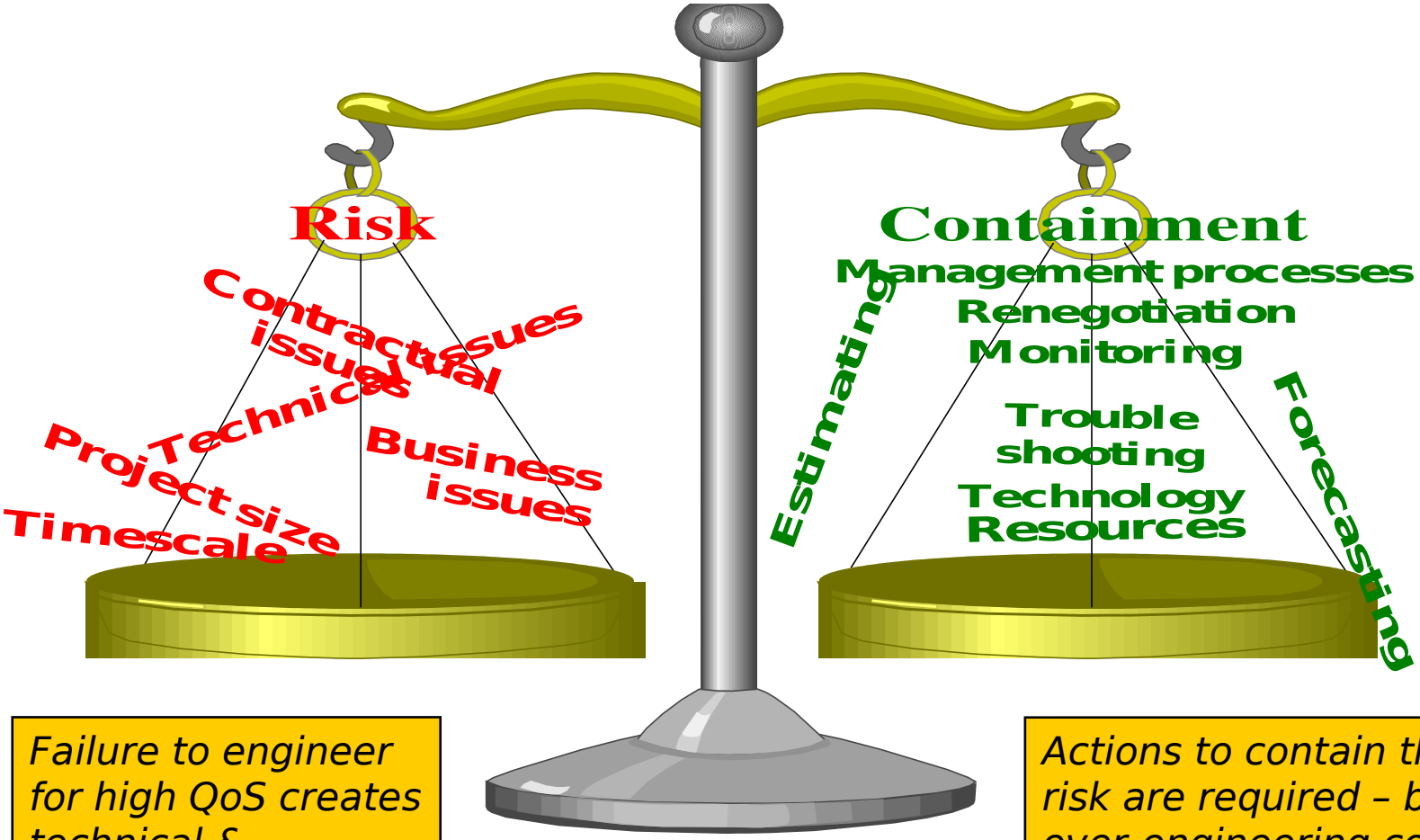
  may be directly actionable or difficult to interpret … [**Actionability**]

  may affect many components of the Solution … [**Affected objects**]

  may be helpful (good quality) or unhelpful (poor quality) [**Quality**]

- Unless we understand the real context and importance of each requirement, we risk producing the wrong solution

Beware: a BALANCE must be maintained between *risk* and *cost*



Failure to engineer for high QoS creates technical & business risks

Actions to contain the risk are required – but over-engineering could be unnecessarily costly

Availability

The reality of Availability is that customers directly relate it to the End User experience

The Availability of a system is a measure of its readiness for usage

There are certain key terms that are used to define Availability-related concepts

- **High Availability** is taken to mean a requirement for a system or service to be over 99% available – typically implies thorough design and may require redundant components
- **Disaster Recovery** means the recovery of essential services in the event of a major business disruption that has resulted from the occurrence of a disaster
- **Business Continuity** means the continued operation of business processes to a predetermined acceptable level in the event of a major business disruption
- **Unscheduled Outage** is a time period when the system is not ready for use and the users expect it to be. These are unplanned outages caused by 'Random Events'
- **Scheduled Outage** is a time period when the system is not ready for use and the users do not expect it to be. These are planned outages driven by predefined events
- **Continuous Operations** is the requirement for perpetual operations 365 days per year 24 hours per day with perhaps very rare scheduled outages
- **Fault Tolerance** is that property of a component, sub-system or system that means that normal service continues even though a fault has occurred within the system
- **Reliability** is the probability that an item will perform its intended function for a specified interval under stated conditions
- **Maintainability** (or **Recoverability**) is the probability that using prescribed procedures and resources, an item can be retained in, or restored to, a specific condition within a given period

Availability – a final word

- It is estimated that
  - ~20% of your total availability is a function of your use of technology
  - ~80% is a function of your people and processes
- E.g. someone says the:
  - Root cause was that firewall logs were full
  - The real reason was there was insufficient process in place to monitor the logs and clear them down
- Technology and design is important, however don’t assume that is your only challenge

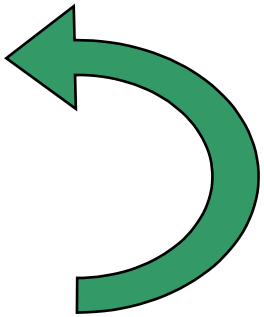
Performance

What is Performance?

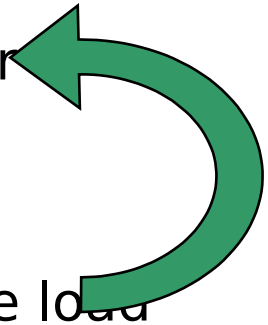
- Definition
  - “Performance. The degree to which a system or component accomplishes its designated functions within given constraints, such as speed, accuracy, or memory usage.” [IEEE-610.12]
- In general
  - *Timeliness* of response, and *predictability*, are the two main goals
  - “Faster” is not always enough, as in for example, a real time system requires extremely consistent performance
- An (old) quote (from ICCM):
  - “A manager's goal should always be to strike the right balance between system function, processing costs, people costs, and performance. This is why the technical aspects of performance can never be entirely divorced from organizational politics”

There are three main, heavily inter-related aspects of Performance to be considered

- Response Times
  - On-line response times
  - Batch run times
- Throughput
  - Transactions per second
  - Records processed per hour
- Capacity
  - Component sizing to handle load
  - Contingency and Scalability

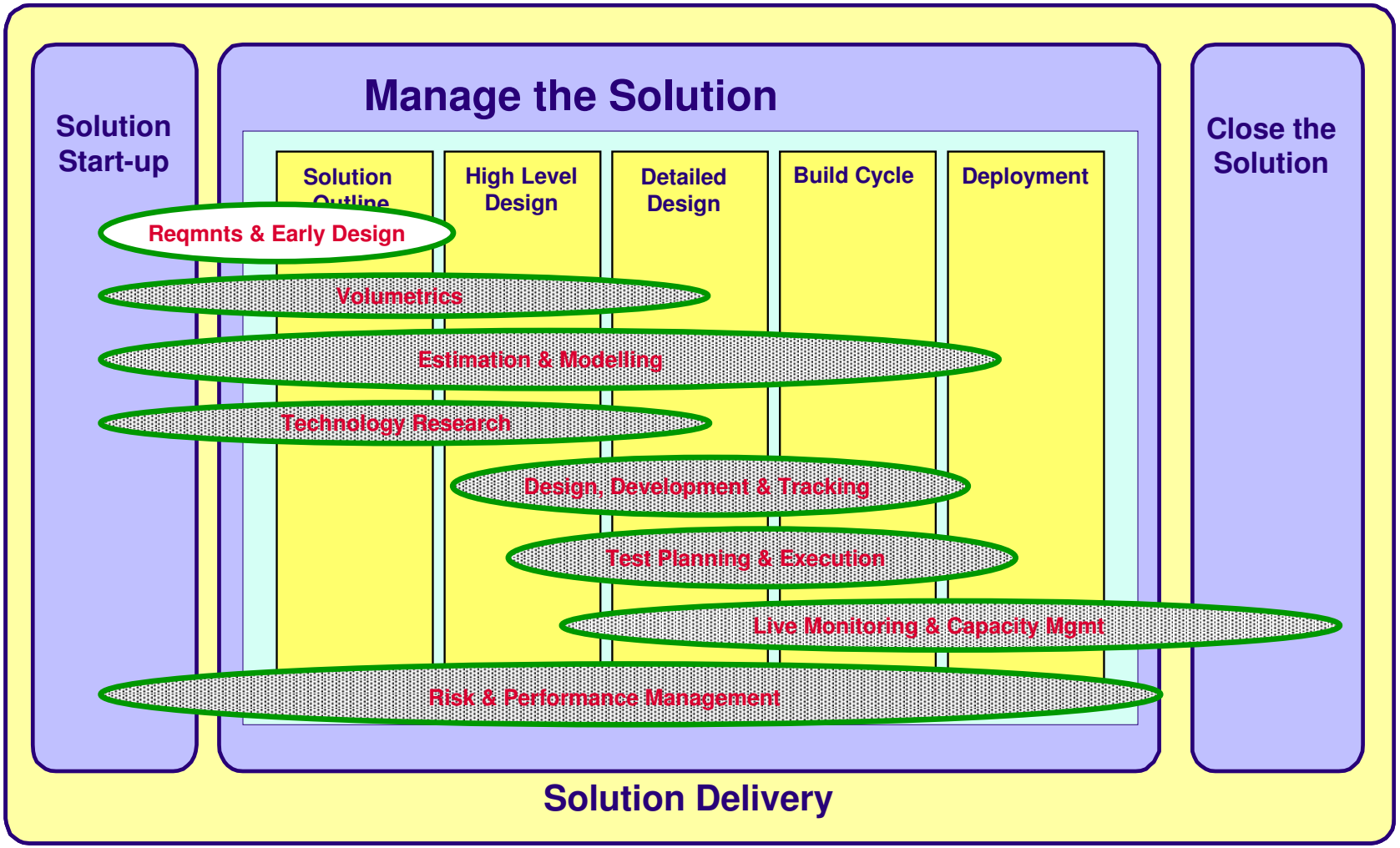


Must have adequate throughput to avoid poor response times



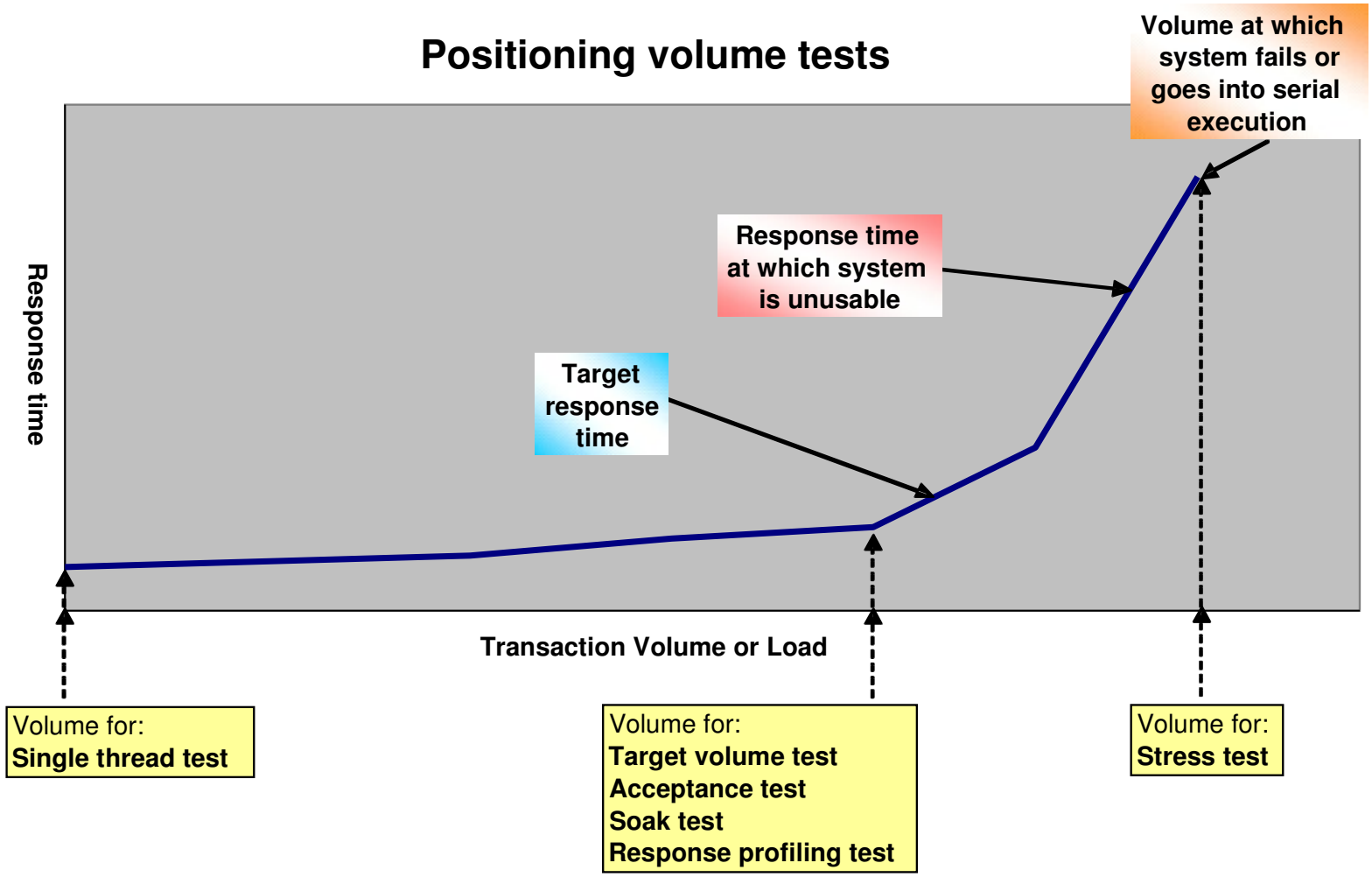
Sufficient capacity is required to meet throughput requirements

Major activities a Performance Engineer executes across the project lifecycle



A range of Performance Test types are used for different purposes

Test Planning & Execution



Performance ::
Volumetrics

The Importance of Numbers

Performance Architects rely on **VOLUMETRIC DATA** and **ASSUMPTIONS** in order to ….

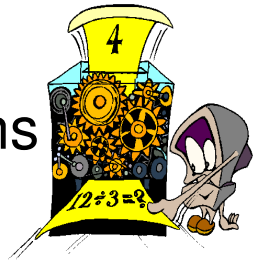
What do you do when these are vague or difficult to get?

Feed **performance and capacity models**, in order to ….

Or difficult to map down to the technical level?

Predict system performance
- *online and batch*

Size systems
Evaluate & improve designs
Plan capacity
Plan testing



Enterprises often cannot provide detailed volumetric information – often, it has to be derived (or guessed!)

Real questions IBM Performance Engineers have been asked by customers

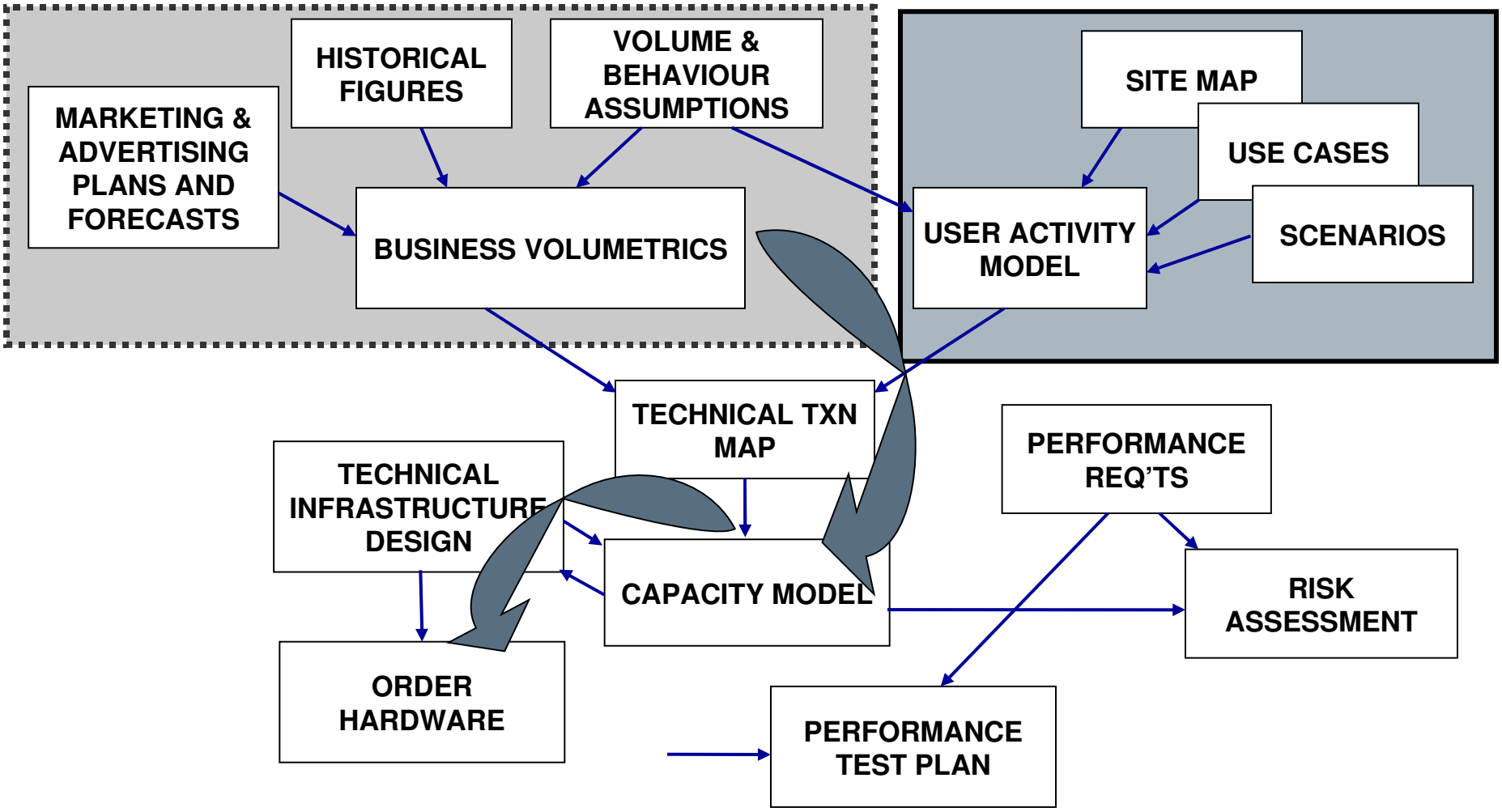
- *"We're just about to spend £20m on advertising our new brand. How many web servers do we need?"* - Insurance company

- *"Will this new digital audio broadcasting solution perform OK, given we don't know how we are going to use it yet?"* – Public service radio broadcaster

- *"How fast is the Internet?"* – Offshore bank

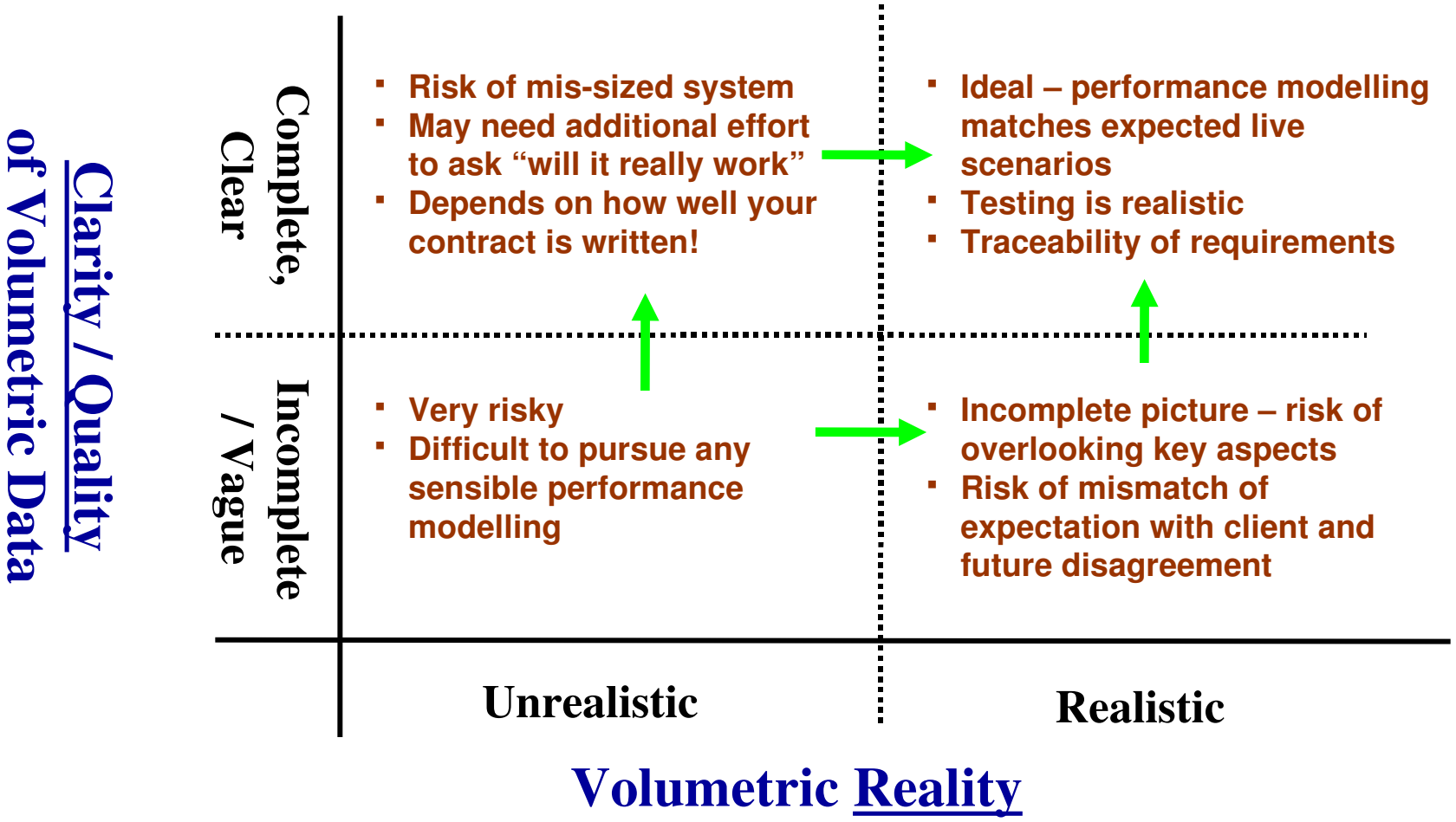
Volumetric data can be traced from various sources

An example "volumes map" used on an engagement



The "Volumetrics Magic Quadrant"

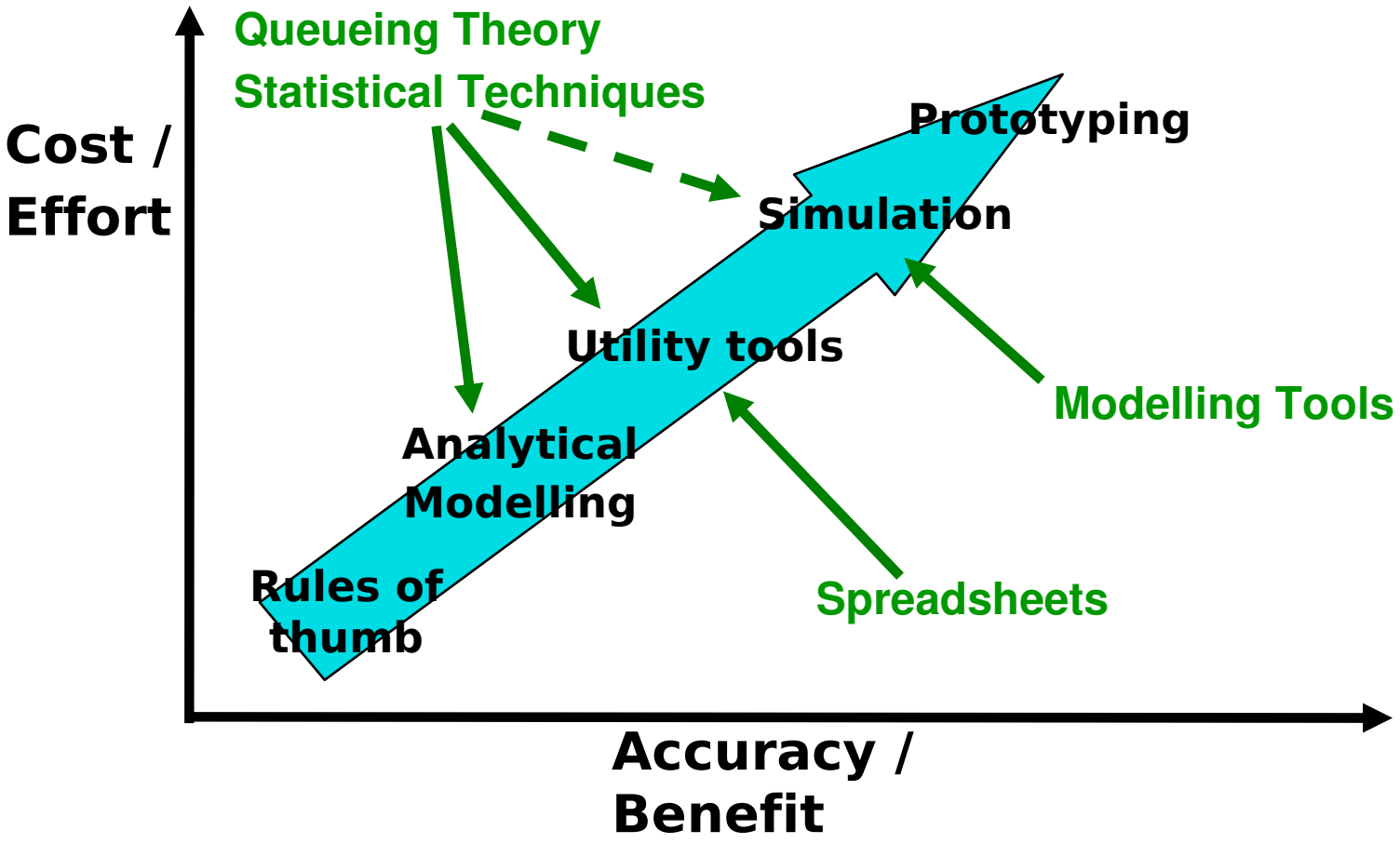
Relationship to Contracts and Targets



Performance :: Estimation and Modelling

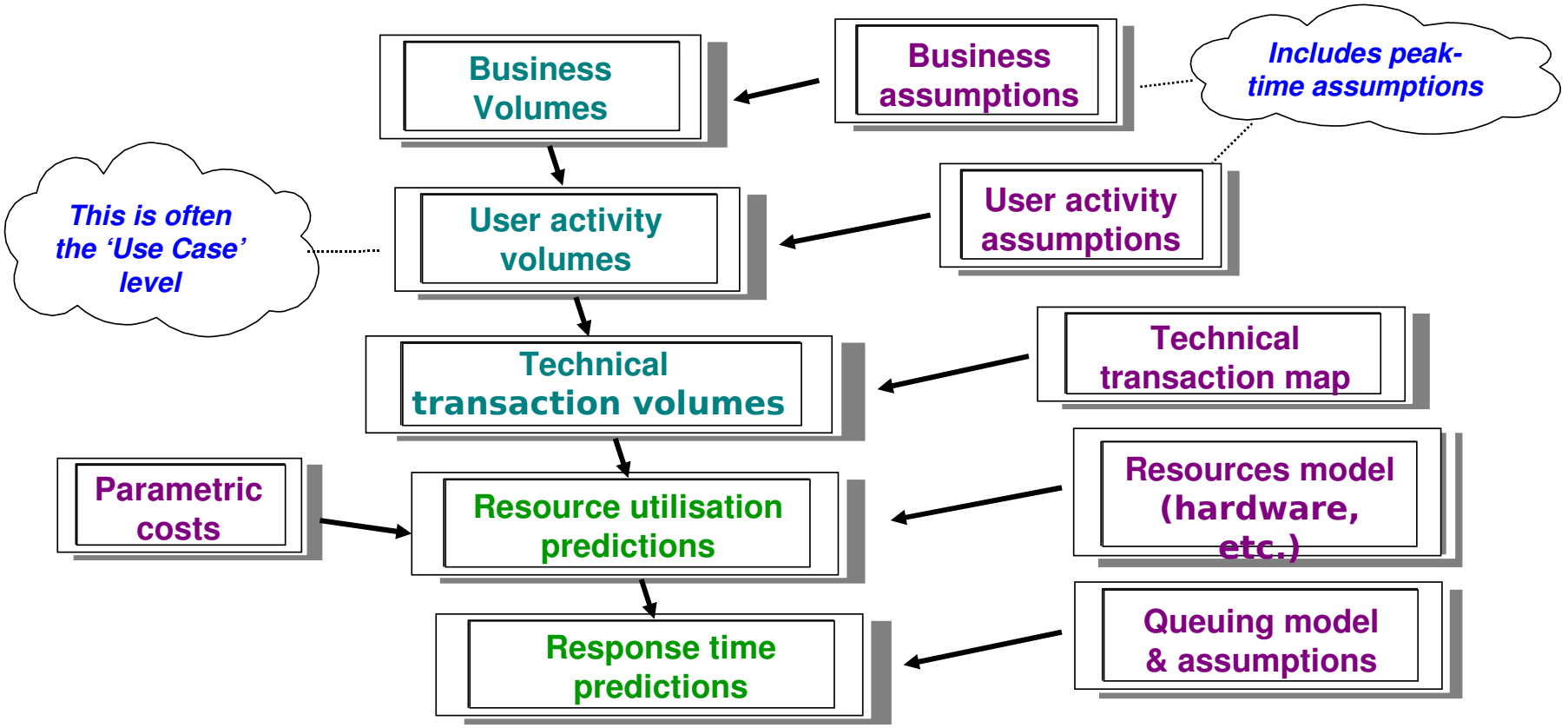
Performance characteristics of a system can be investigated in more detail by creating a model

- Different techniques are available different levels of effort to provide answers with different levels of reliability



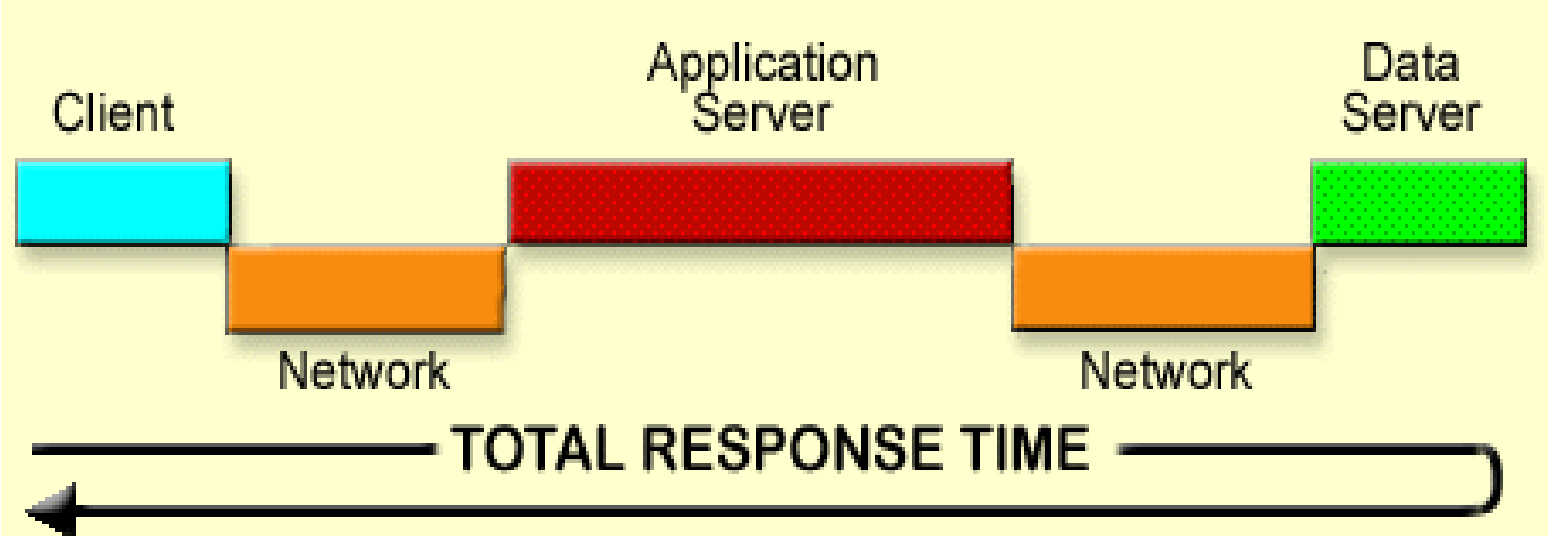
Analytical performance model typical structure

This is a outline (simplified) view of the relationships of the data sets and analysis steps required to build a analytic model



In actual models we need to sum response times for all components an end-to-end transaction relies on

- Utilisation of each resource based on the total workload and workmix
- End-to-end response times based on multiple steps in the end-to-end transaction path



Summary

Summary of Topic

- Despite continuing advances in technology, IT Architects spend significant amounts of time engineering systems to account for Quality of Service requirements
  - In the context of often significant constraints
  - Software and infrastructure designs need to be iterated together to achieve goals
- Non-functional requirements & service levels may be contractually binding
  - Failure to achieve targets may result in financial penalties for the IT provider, and/or lost business for the customer
  - If a design cannot be established which meets requirements, this is top severity project issue
- Modelling theory, techniques and tools are available to assist with evaluating design alternatives
  - Employing them successfully requires understanding of the systems elements, management of assumptions and appropriate modelling skills
- Regardless of the quality of design, the quality of implementation must be validated through testing
  - QoS design should inform test strategy and test planning
- *The effort expended should always be proportionate to the risk involved*