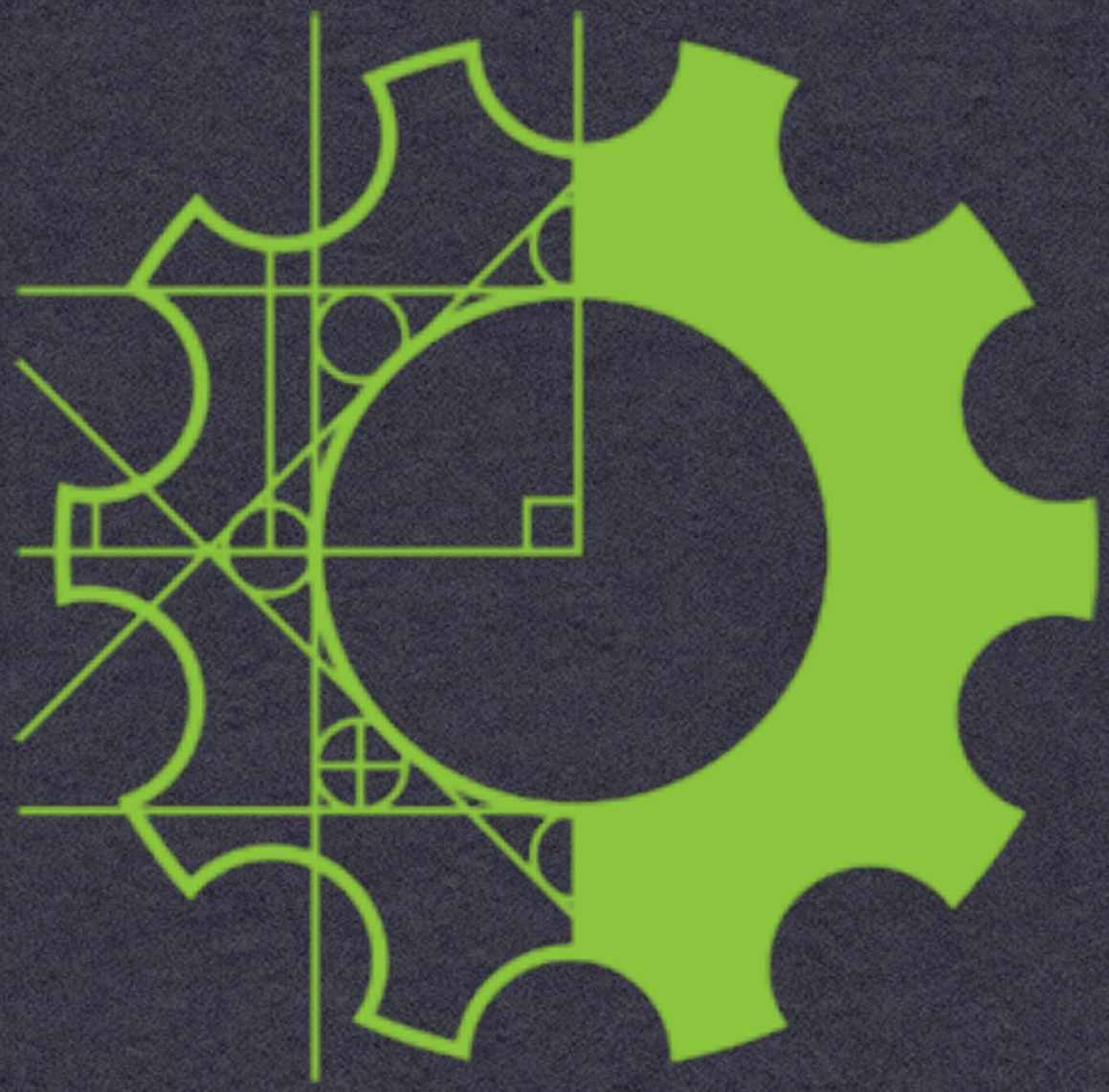


Specifying Software

Reid Holmes

A brief overview…

“Wicked” Problem

A ‘wicked problem’ is one that can only be clearly defined by solving it.

“Wicked” Problem

‘This paradox implies that one must solve a problem once to define it and then solve it again to create a solution that works.’ —Peters and Tripp

Wicked Characteristics

- No definitive formulation
- No stopping rule
- Solutions not true-or-false, just good-or-bad
- No ultimate test of a solution

“Wicked” Problem

‘The appropriate way to tackle wicked problems is to discuss them. **Consensus** emerges through the process of laying out **alternative understandings**, competing interests, priorities, and **constraints**’

—Mary Poppendieck

A specification is…

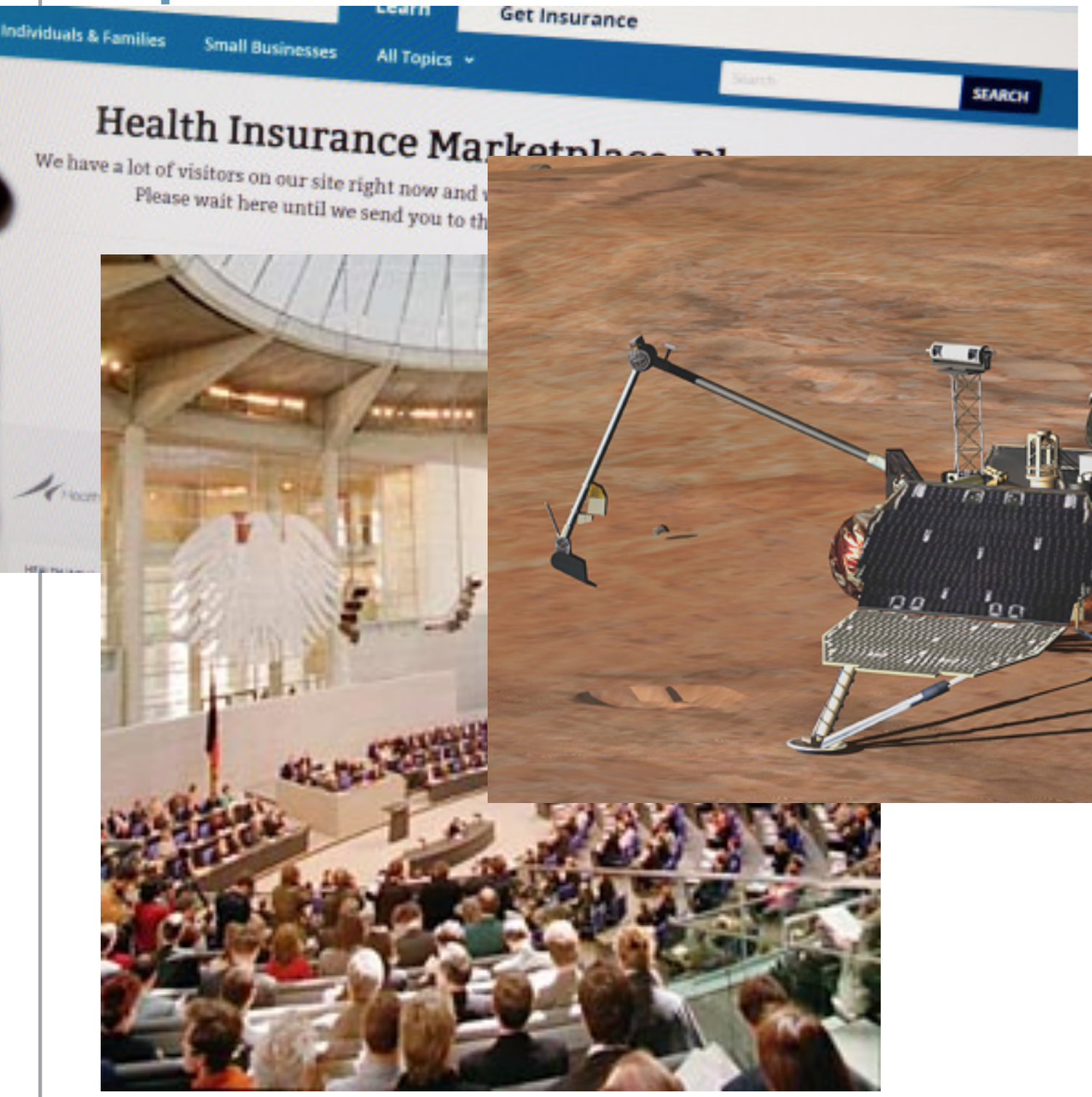
functional requirements

non-functional requirements

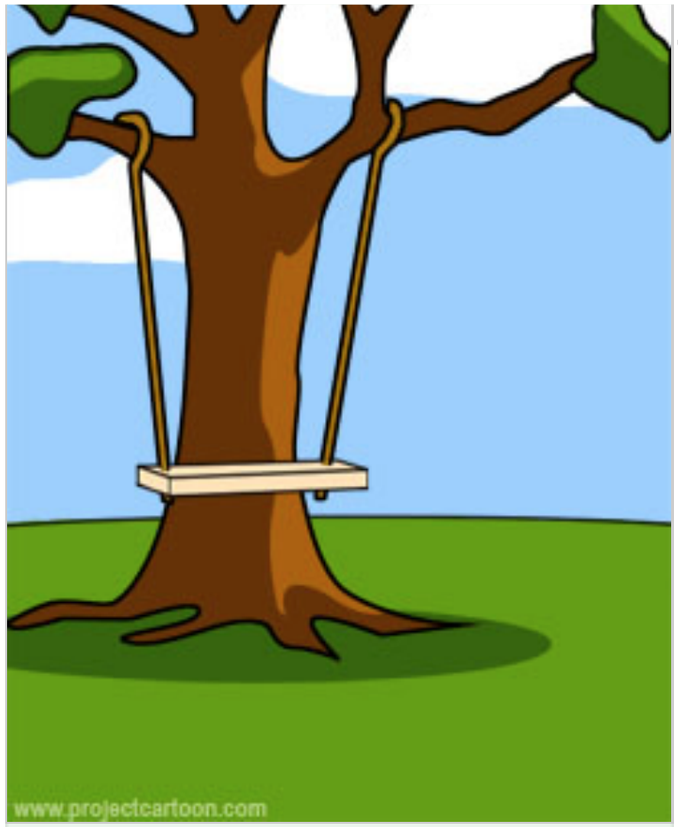
Specs describe what to do (but not how to do it)

- A perfect implementation is no good if it solves the wrong problem
- It is difficult to create specs that are:
  - complete
  - consistent
  - precise
  - concise

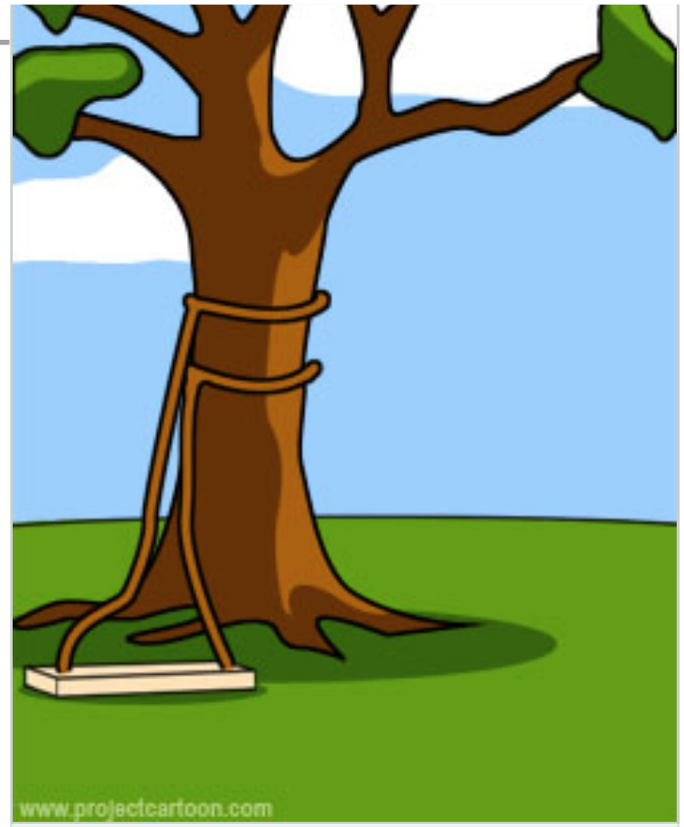
Specification failures are costly



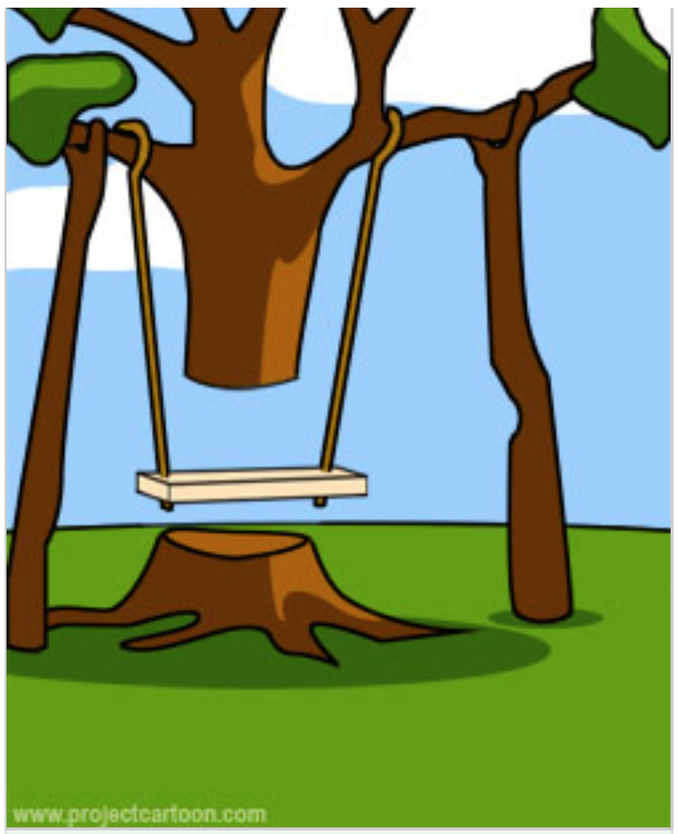
How the customer explained it



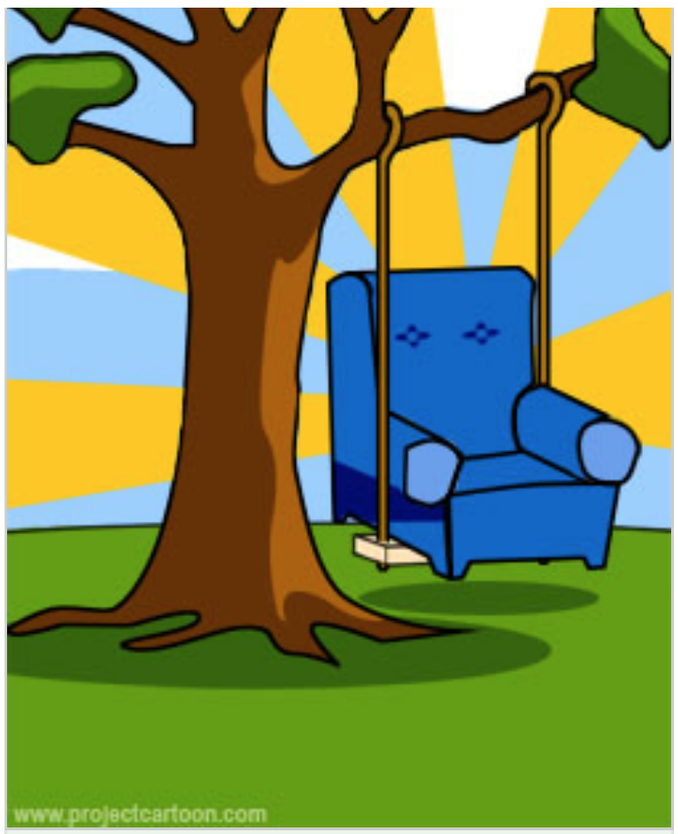
How the project leader understood it



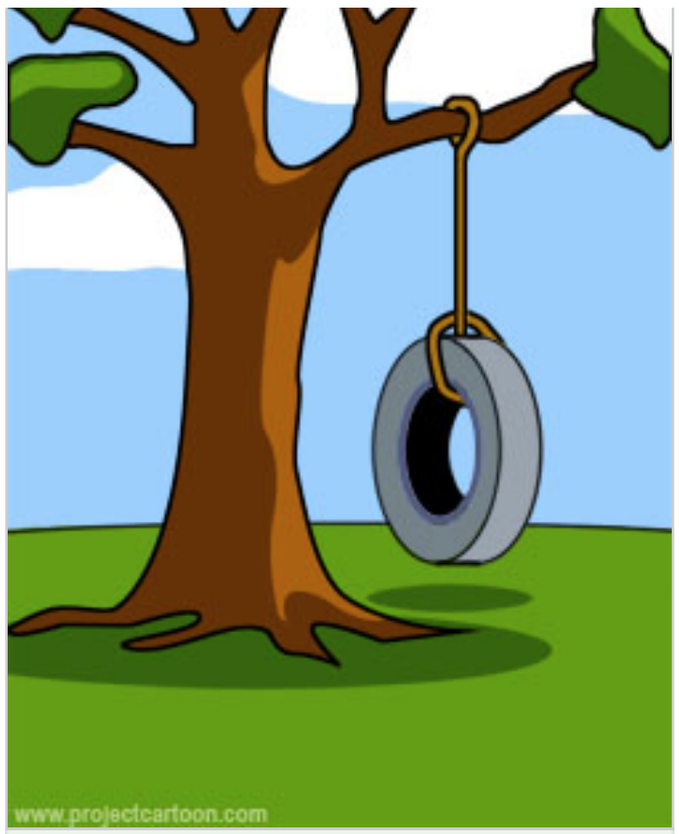
How the programmer wrote it



How the analyst designed it

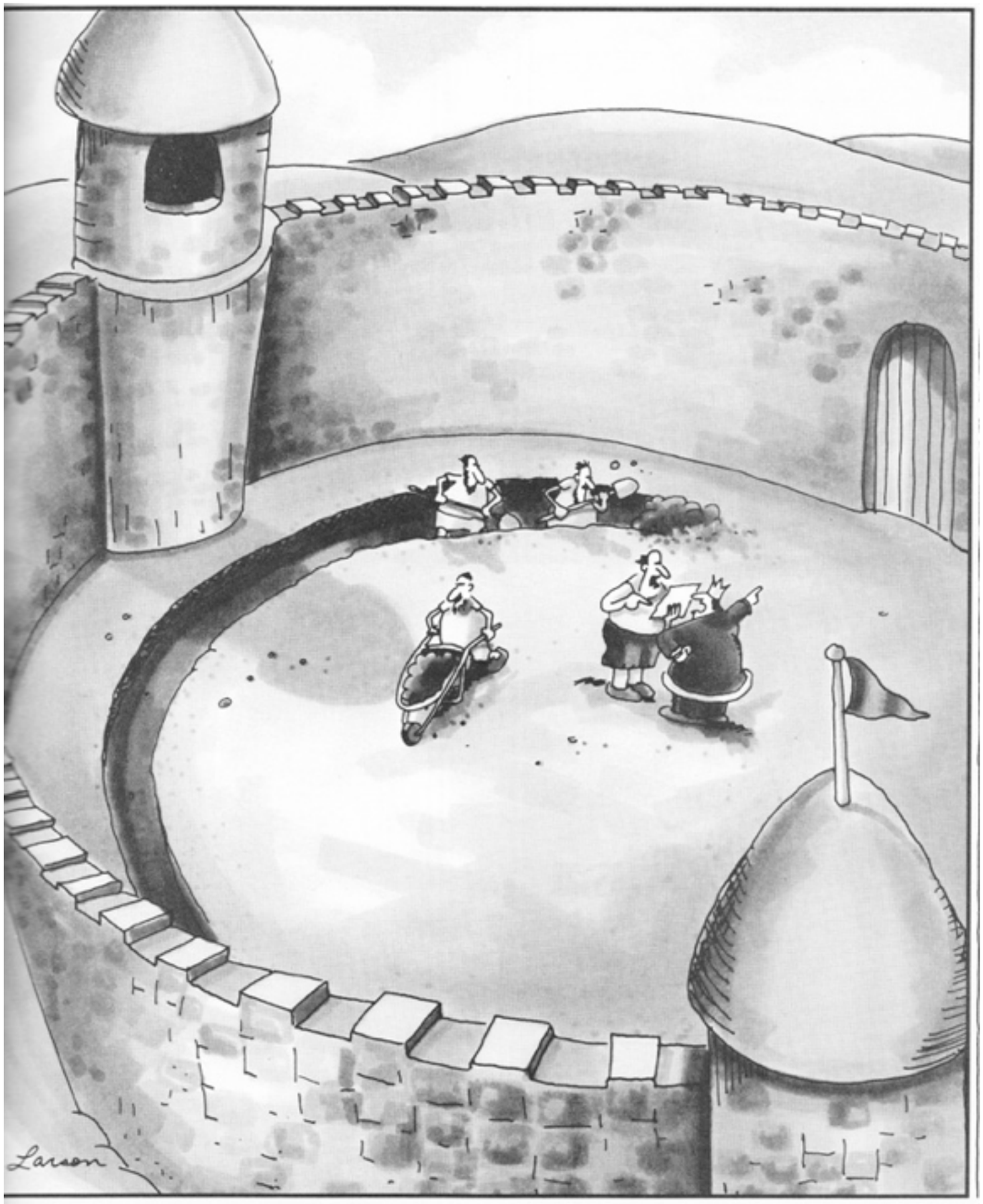


How the business consultant described it



What the customer really needed

Vague Reqs

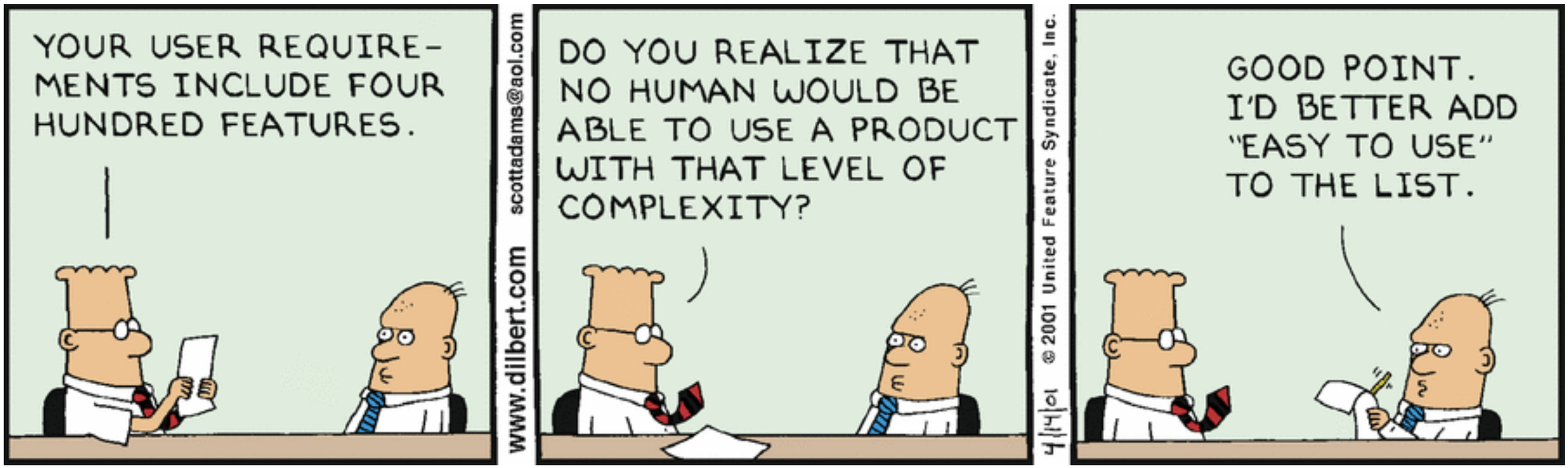


©Gary Larson

Ambiguous Requirements

©Bill Watterson
January 09, 1995

Changing Requirements

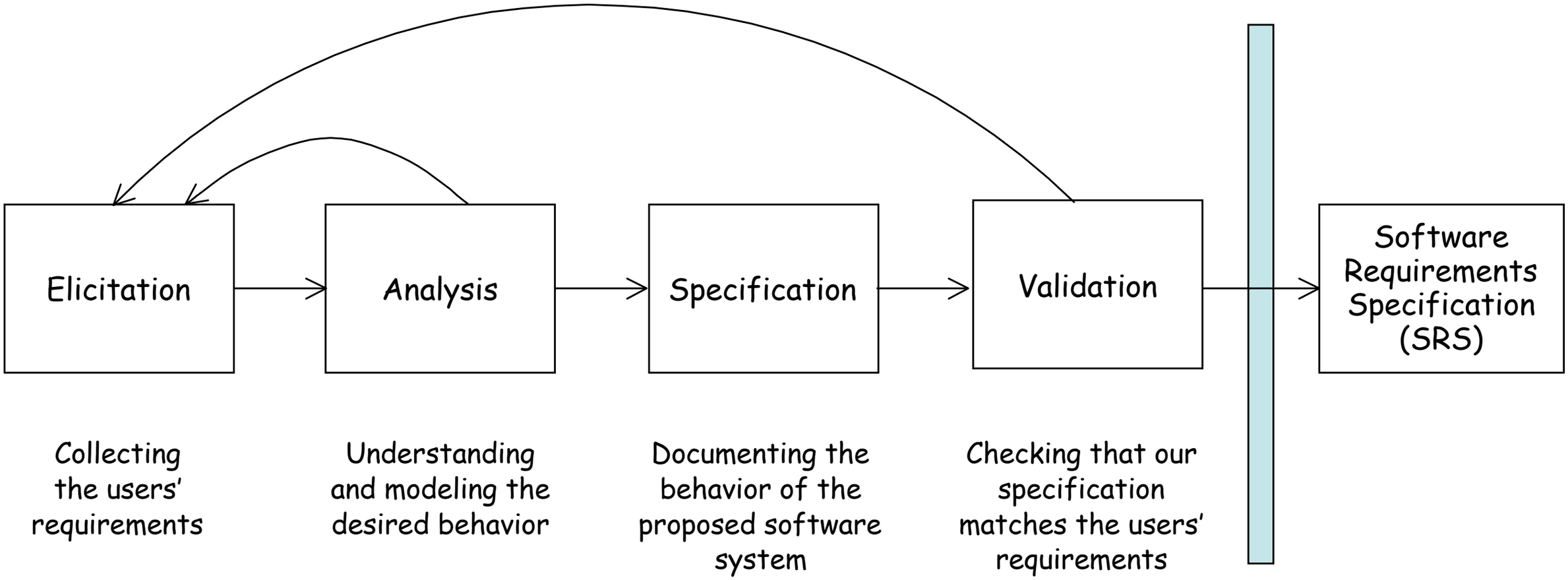


©Scott Adams
April 14, 2001

Specifications matter

- A specification:
  - connects customer and engineer
  - ensures parts of the implementation work together
  - defines the correctness of the implementation
- Therefore, everyone must understand the spec
  - Designers, developers, testers, managers, ops, customers…
- Good specifications are essential for a project to be successful

Requirements Process



Elicitation

- Required functionality: what the software should do
  - Record keeping, data computations / transformations, process control, query processing, commands to hardware devices, etc.
- Quality attributes: desired characteristics
  - Performance, efficiency, safety, security, usability, maintainability, reliability, robustness, availability
- Design constraints: customer-specified limits
  - Mandated hardware components, mandated adjacent systems, resource constraints, mandated development process, budget
- Environmental assumptions: assumed context
  - Working status of hardware / software components, assumptions about inputs (data format, rate of input, number of users), operating conditions
- Preferences
  - Priority rankings of requirements

Measuring Requirements

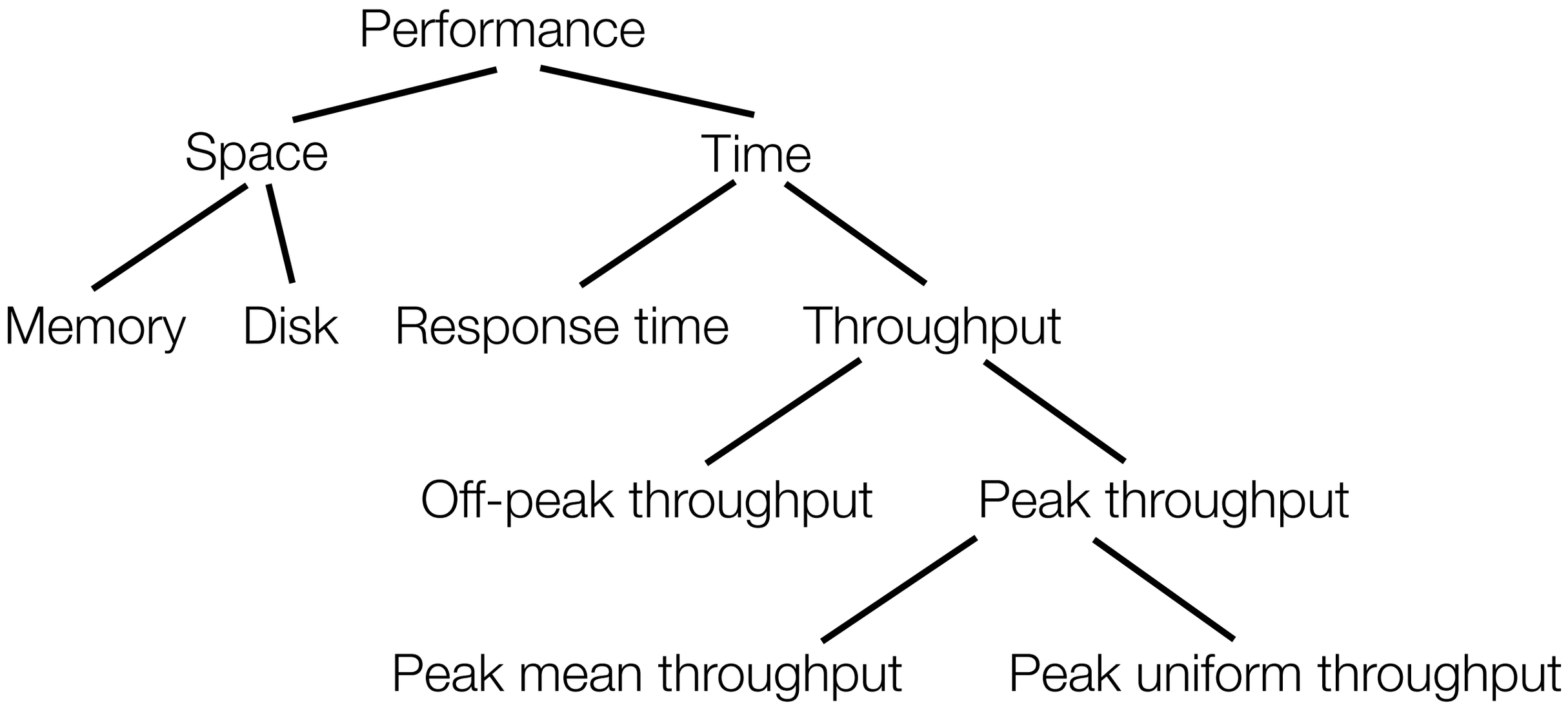
‘To measure is to know. If you can not measure it, you can not improve it.’

—Lord Kelvin

Measuring Requirements

- We must explicitly quantify requirements.
  - —> And verify that they are met.
- In practice, it turns out that measurable objectives are usually achieved.
  - Having explicit measures provides explicit goals.
  - Explicit goals arise from concrete discussions about values.

Performance



Usability

- Training effort
- Task time
  - Novice vs. experienced user
- Error rate (correctness of usage)
- Perceived satisfaction

Reliability

- Trying to maximize availability
  - Mean Time Between Failures (MTBF)
  - Minimum uptime
  - How do failures influence the rest of the system
- Operational availability
  - Availability - error downtime - admin downtime
  - Mean Time To Resume (MTTR) [from failure]

Complexity

- Lines of code
- Cyclomatic complexity
  - Independent closed loops on minimal graph
- Number of branches
- Order:
  - Number of direct/indirect in edges
  - Number of direct/indirect out edges
- Bug reintroduction rate