Understanding Quality Goals

David Gelperin, CTO
ClearSpecs Enterprises

1. Quality crisis and candidate causes (7)
2. Requirements risk management (4)
3. Tactics for managing quality awareness (10)
4. Quality Assumption Reviews (15)
5. Resolving the quality crisis (3)

Developers don’t understand quality goals

Many developers have a shallow understanding of how to achieve quality goals such as safety, security, and availability.

Developer ignorance endangers project results.

Customers want functions

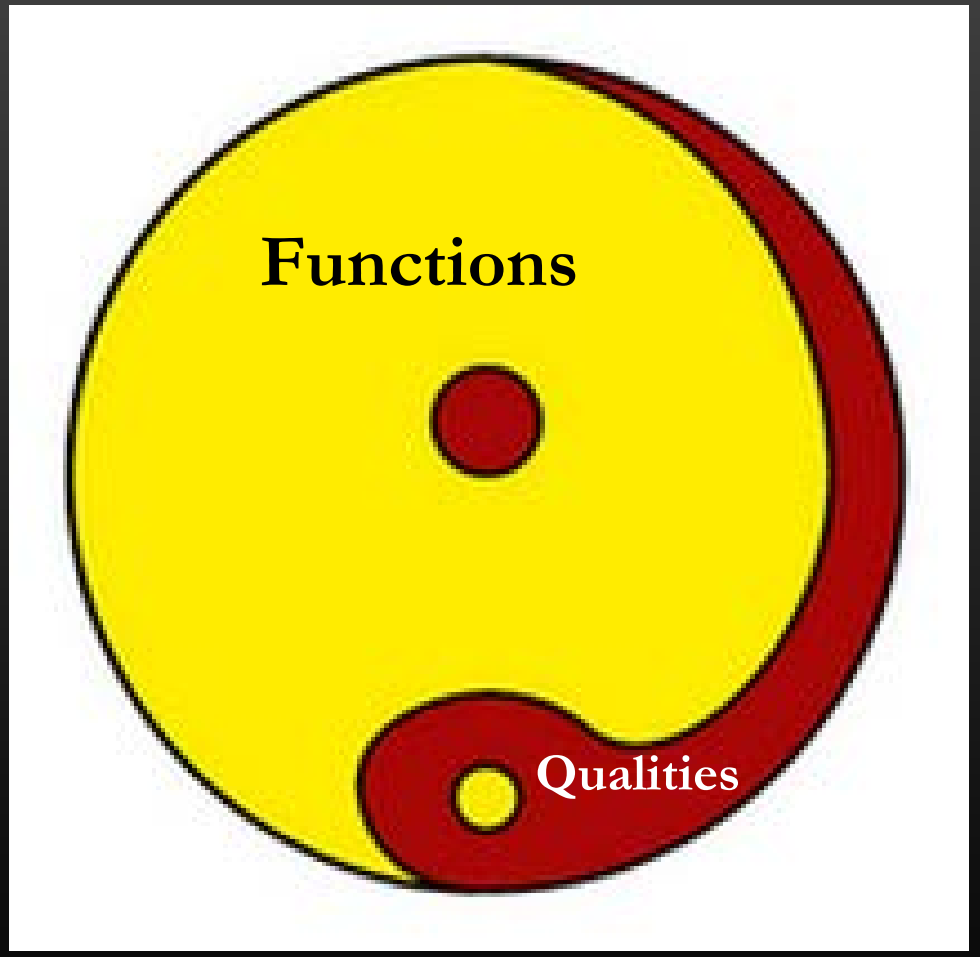
The Quality Crisis

There is a self-inflicted quality crisis in progress.
Websites are hacked or fail to handle large loads *and we shrug*.
Failure teaches us little and changes nothing.

This may be a “henny penny” (or not)

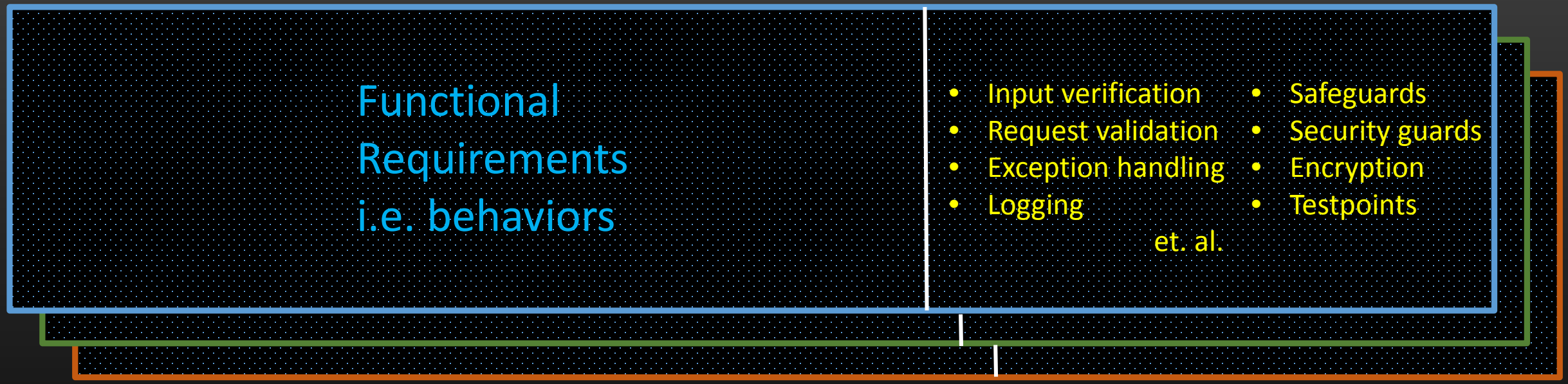
The resolution: Increased quality awareness and learning

Over-emphasis on functionality



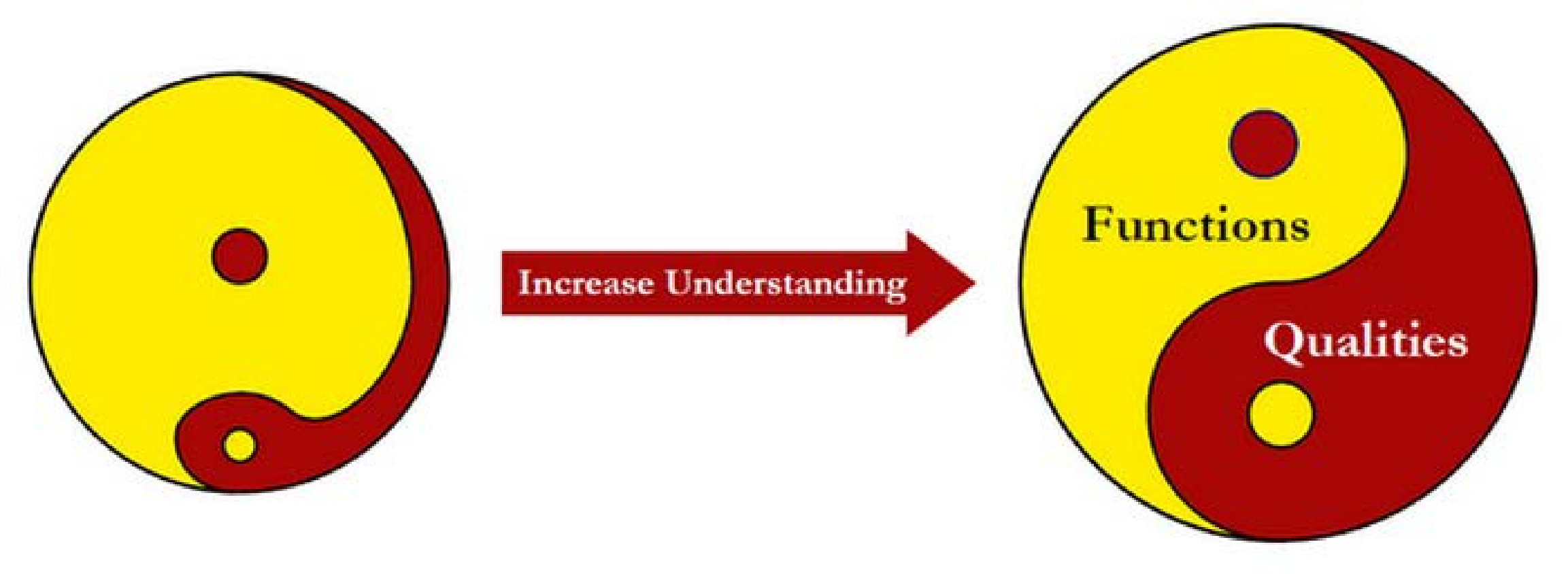
What programmers need to know

Delete reservation & refund payment (for example)



Functional components are crosscut by quality supports e.g. exception handlers

Becoming Quality-Aware

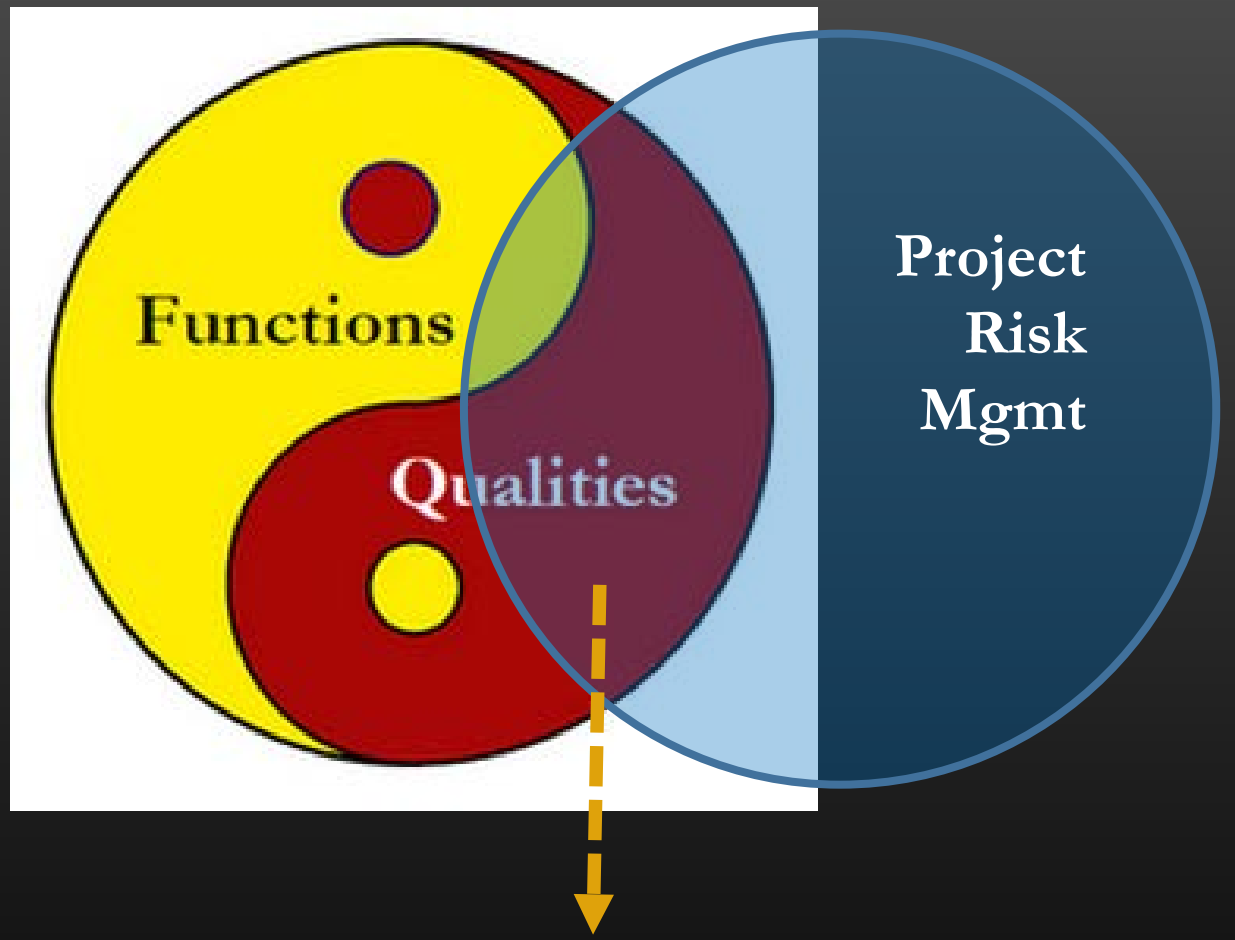


Understanding that functionality is just the beginning

Candidate causes of shallow understanding

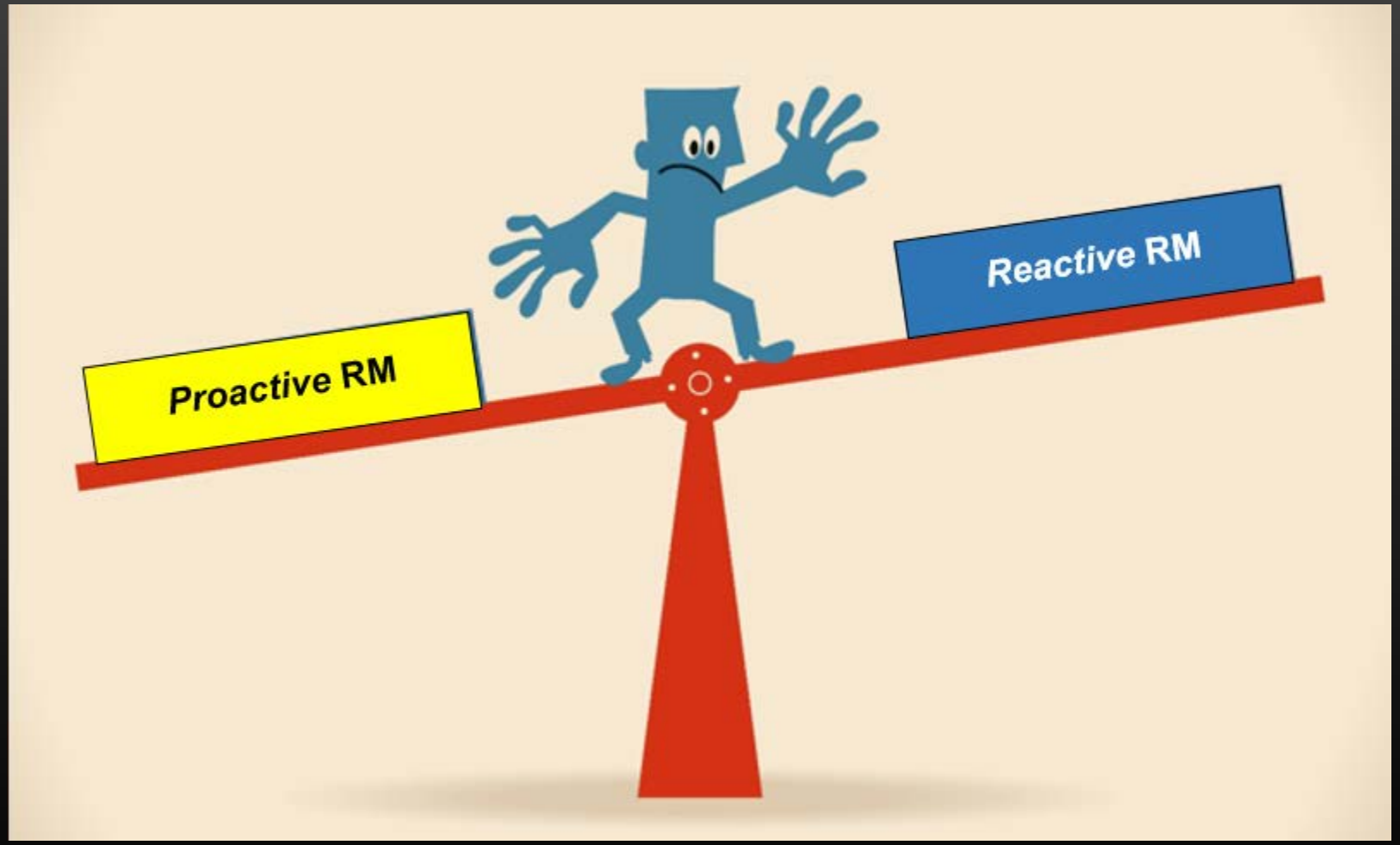
- Goals are numerous (50 +) and crosscutting (pervasive)
- Some are software engineering subfields
- Attributes (e.g. quality level) are numerous (20 +) and complex
- Goals may conflict (e.g. create infeasible pairs) or support
- Goals have ad hoc mitigation and support tactics
- Tactics are situation-specific
- Goals are hard to verify
- The nature, scope, and verification of quality goals are not taught
- Achieving quality goals is drudgery i.e. no fun
- Quality goals are "ignored" by Agile

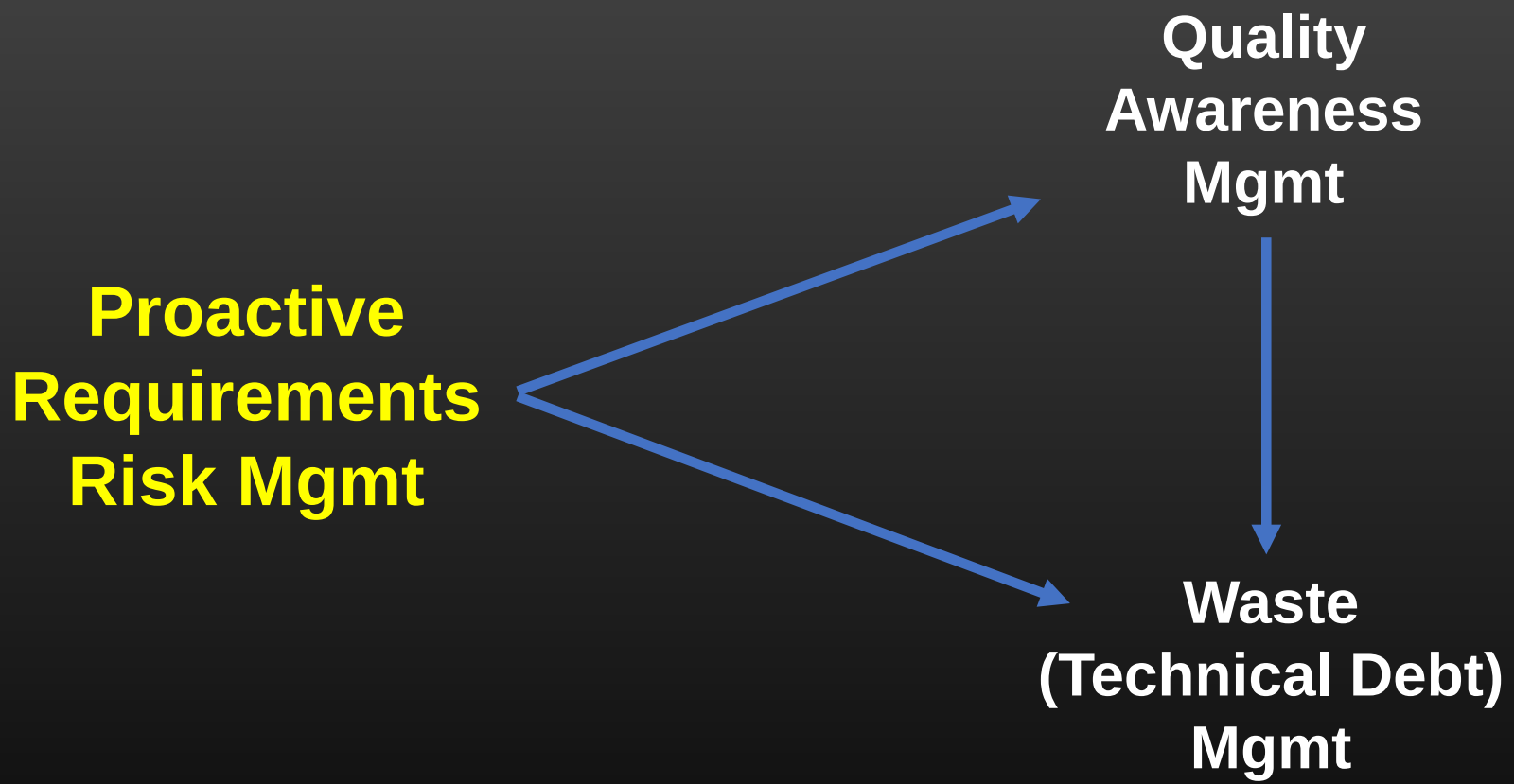
Qualities are dangerous



Requirements
Risk Mgmt (RM)

Searching for the Golden Mean





Tactics for managing quality awareness

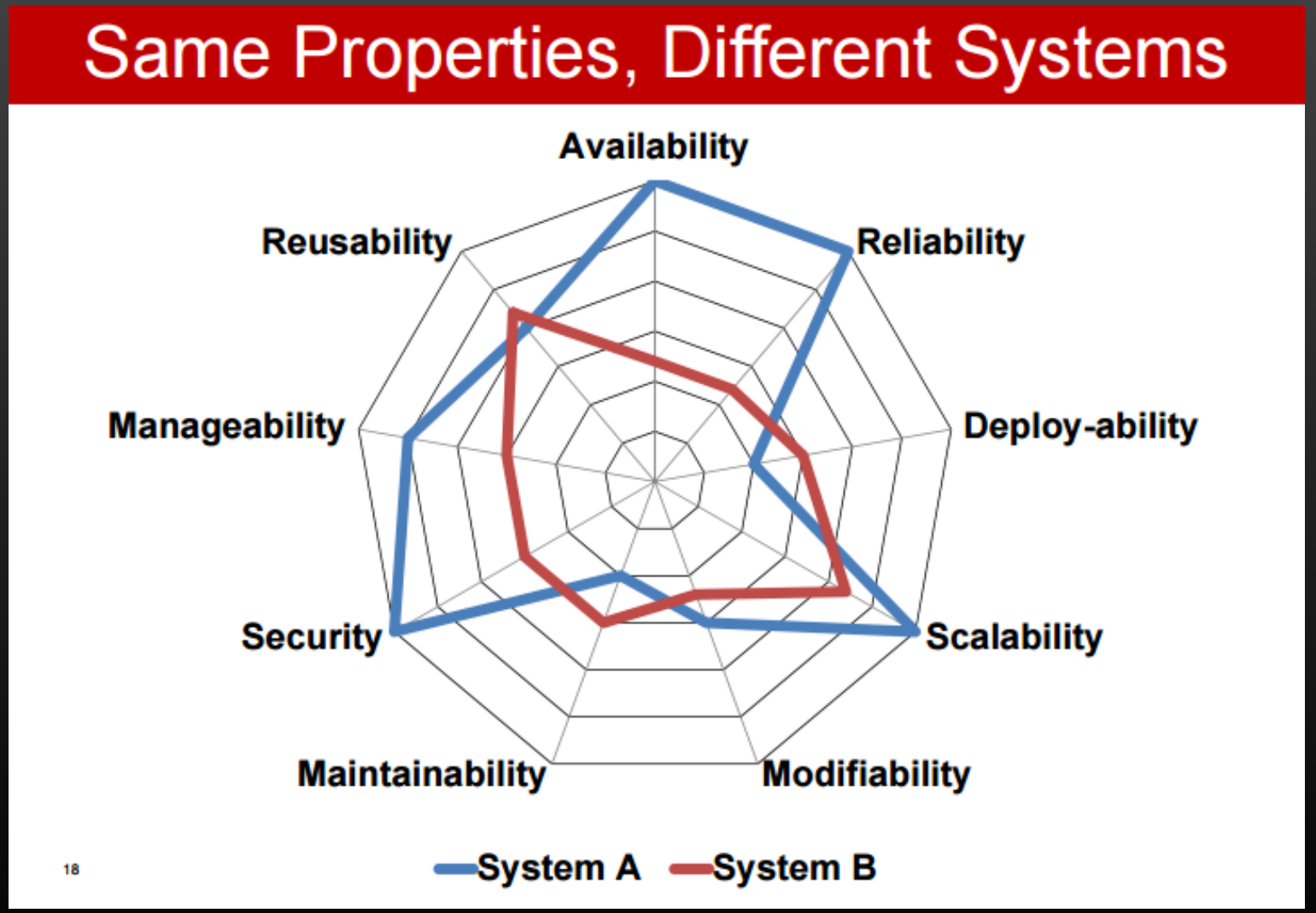
a. *Holding **quality assumptions reviews*** (temporary tactic)
b. *Creating **rich project quality models*** (RPQMs)
c. Holding (mini) "quality attribute workshops“ using RPQMs
d. Creating "quality scenarios"
e. *Sketching **verification strategies** for project quality goals*
f. *Recording **quality learnings***
g. Holding brown-bag lunch sessions to share quality experiences
h. Running "quality discussion groups" that read book chapters and papers

Find resources at quality-aware.com

b. Creating Rich Project Quality Models (1)

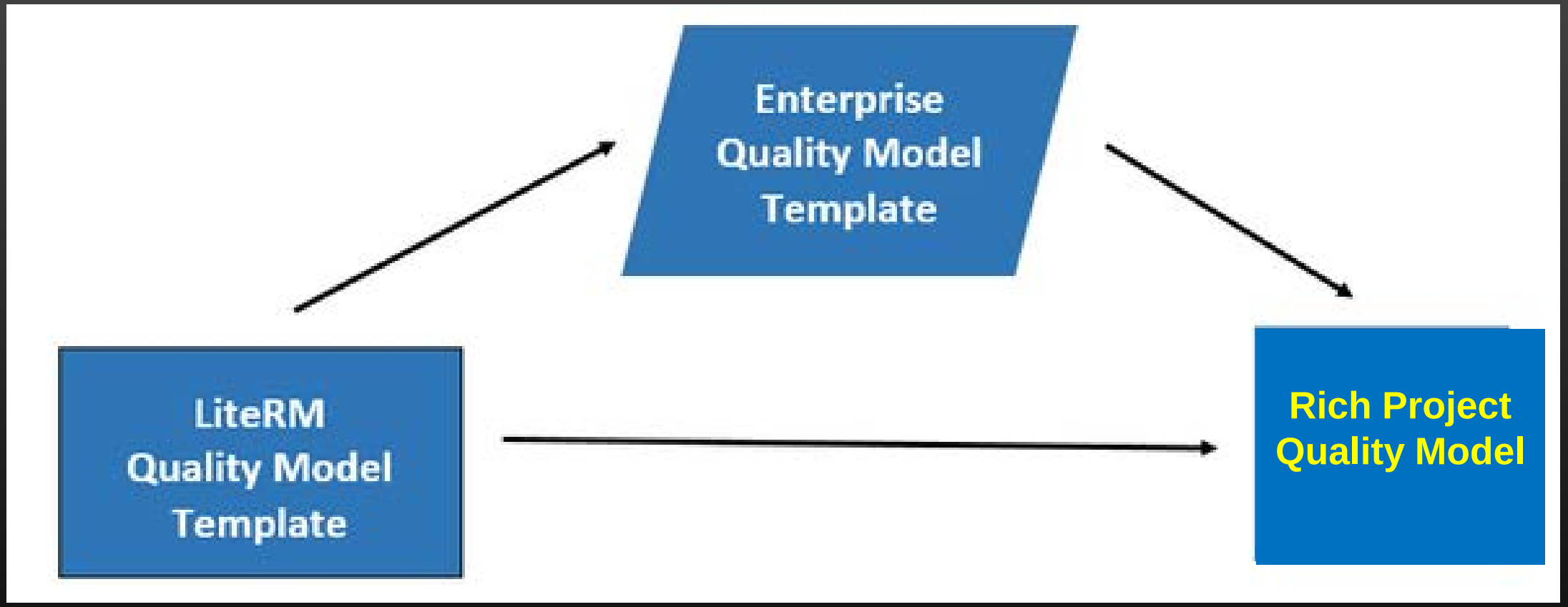
ISO 25010 and its peers are ***meager*** quality model templates

b. Creating Rich Project Quality Models (2)



This is a ***meager*** project quality model

b. Creating Rich Project Quality Models (3)



LiteRM is a ***rich*** quality model template

b. Creating Rich Project Quality Models (4)

Rich quality model templates
are analogous to

f. Recording Quality Learnings

Rich quality model templates
(and development standards)
are natural places to record
quality lessons.

Without them, lessons are lost.

e. Verification Q & A

Isn't verification just a fancy word for testing?

No

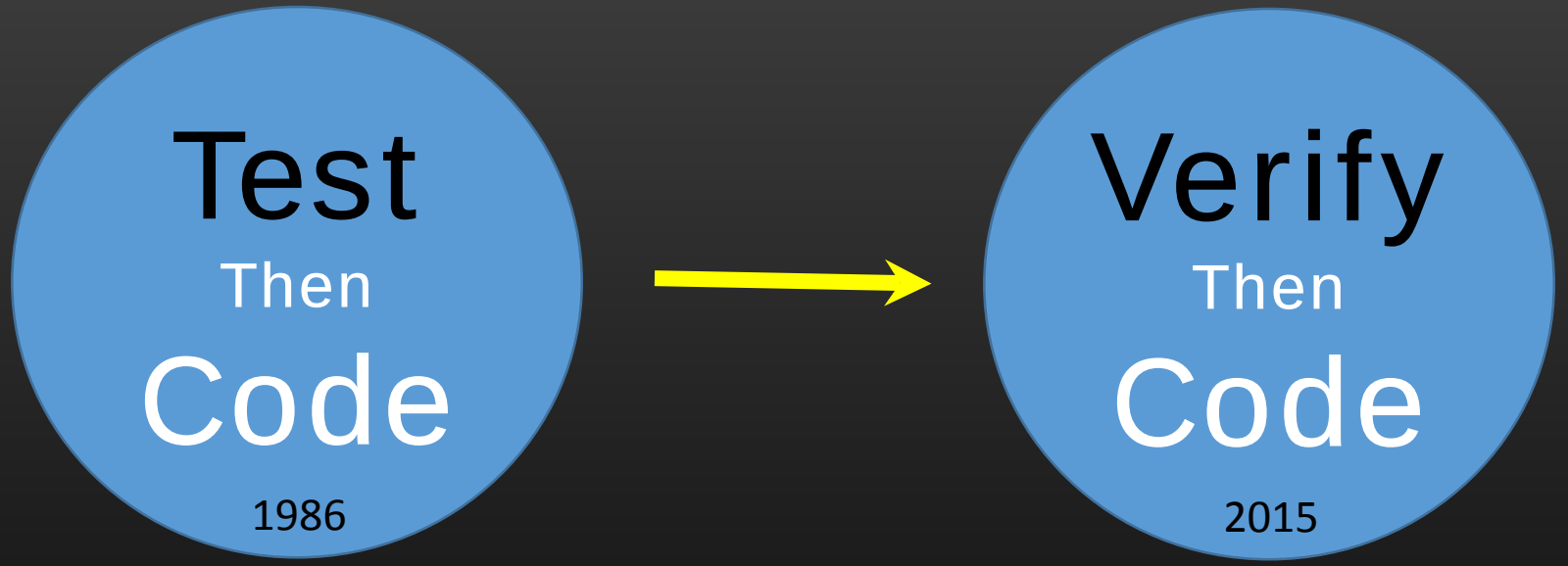
Then, what is it?

A composite checking activity that includes analysis e.g. hazard analysis, technical reviews e.g. code inspections, and measurement e.g. mean time between failures, as well as testing.

Why should I care?

Quality goals must be verified. Testing alone is inadequate
e.g. security testing is not enough, also need threat analysis and security guard code inspections.

e. Sketch & Execute Verification Strategies Early



Preventative Testing (TDD)

Preventative Verification (VDD)

e. Why verify early?

"IT projects that applied NFR (Non-Functional Requirement) verification techniques relatively early in development were more successful on average than IT projects that did not apply verification techniques (or applied them relatively late in development)"

- Eltjo Poort, Nick Martens, Inge van de Weerd and Hans van Vliet
"How Architects See Non-Functional Requirements: Beware of Modifiability" *REFSQ 2012* [Best Paper]

e. Sketching a Verification Strategy

[See sample on last page of handout]

a. Quality Assumptions Review – Why?

- To synchronize understanding of quality goals, how they differ from functional requirements, and how they are achieved.
- To build a culture of quality-awareness
- To prepare for quality goal identification

a. Quality Assumptions Review – How?

1. Create a list of basic assumptions [see handout]
2. Distribute assumptions prior to review and encourage discussions
3. During the review, various reviewers:
   a. Read assumption
   b. Ask for questions
   c. Ask for problems with assumption or its statement
   d. Ask for disagreements and reasons
4. Record open issues and comments during the review
5. Distribute resolutions after the review

Assumption 3

Quality goals, their relationships, and attributes are invariant across application domains and should be described in a rich quality model template. Relevance, priorities, levels, and implementations must be determined while developing the project quality model.

Assumption 4

Most quality goals (e.g. safety, throughput) crosscut functions and thus can't be implemented in a single increment and have an increasing cost to implement or repair across iterations.

Assumption 5

Some quality goals entail levels of achievement e.g. security, privacy, and performance. For example, if security is relevant, should it be minimal using a user name and password, moderate using user name, password, and security questions, or strong using a retinal scanner?

Assumption 6

Understanding quality goals entails understanding their feasibility (i.e. achievability). For example, is 99.999% reliability possible is your situation?

Assumption 7

Some pairs of qualities conflict e.g. understandability and performance, while others support e.g. security supports privacy.

Assumption 8

Some qualities are essential for all applications for example (1) compliance with required rules, (2) domain sufficiency, (3) understandability, and (4) verifiability.

Assumption 9

Some qualities are universal in the sense that it is hard to imagine a system where they don't matter. These include the essential qualities (assumption 8) as well as: ease of use and learning, modularity, reliability, and responsiveness.

Assumption 10

Quality goals are achieved using tactics e.g. supports and self-checking results. There are at least four types of supports:

1. other qualities
2. system functions e.g. logging & exception handling
3. sets of rules e.g. coding standards & design patterns
4. warning labels e.g. “are you sure that …”

Assumption 11

Many quality goals are harder to understand, express, achieve, and verify than most functional goals.

Even when the quality goal is precise and you know the candidate supports, you must select the specific supports needed to achieve the goal based on the system-specific execution environment.

Assumption 12

Some quality goals (e.g. availability, performance, privacy, reliability, safety, security, and usability) are much harder to understand than other quality goals. This is because each is a software engineering subfield with an extensive body of knowledge.

Assumption 13

Verifying quality goals is much more difficult than verifying functional goals. Quality goals, levels, and tactics must be verified with composite strategies that include technical review, analysis, measurement, and test. Testing alone is inadequate.

Assumption 15

Quality goals should be monitored with built-in tests, exception logging, and service call tracking.

Assumption 17 (summary)

Quality goals and functional goals have little in common
[See comparison in handout].

Resolving the Quality Crisis

The challenge is to raise developer awareness of quality goals to the same level as their awareness of functional goals. This implies early understanding of:

- high-priority quality goals and their attributes
- conflicts between qualities and how they should be resolved
- critical supports for each quality level
- effects of the critical supports on each domain function
- how qualities will be verified

Quality-Aware Development

Refers to any development process starting and continuing with activities aimed at helping stakeholders be quality-aware.

For example, Quality-Aware Agile is a hybrid process that begins with the identification of relevant quality goals along with their levels, priorities, and supports.

It should take less than a week to draft a rich, but rough, project quality model.

Quality-Aware Agile

The Agile process that follows considers the selected quality goals and supports when designing and coding the architecture and individual modules.

Each iteration includes the identification and verified achievement of incremental quality goals that depend on the functions chosen for the increment and the project's quality model.

Incremental learning about final quality goals should be recorded in the quality model.

Wrap Up

Thanks for your participation.

Please help resolve the quality crisis
by raising quality awareness in your organization.

Resources available at quality-aware.com

Questions or Comments

david@clearspecs.com