Rapid Specification of Quality Goals

David Gelperin
ClearSpecs Enterprises

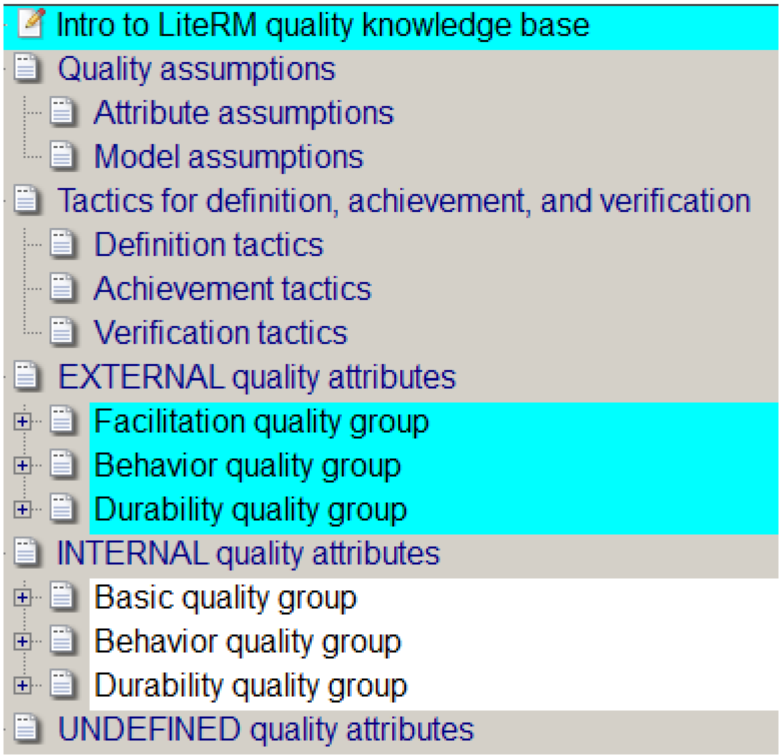
Quality attributes (the “ilities”) include security, safety, usability, etc. Quality goal specifications (specs) refer to requirements specs for quality attributes.

An efficient way to develop such specs is to tailor a comprehensive model of quality attributes.

For example, consider the LiteRM quality model described below. It contains over 60 attributes with each attribute having over 30 characteristics. This model is freely available at www.quality-aware.com.

The model partitions qualities (Figure 1) into external attributes such as safety, visible to users, and internal attributes such as code understandability, only visible to developers.

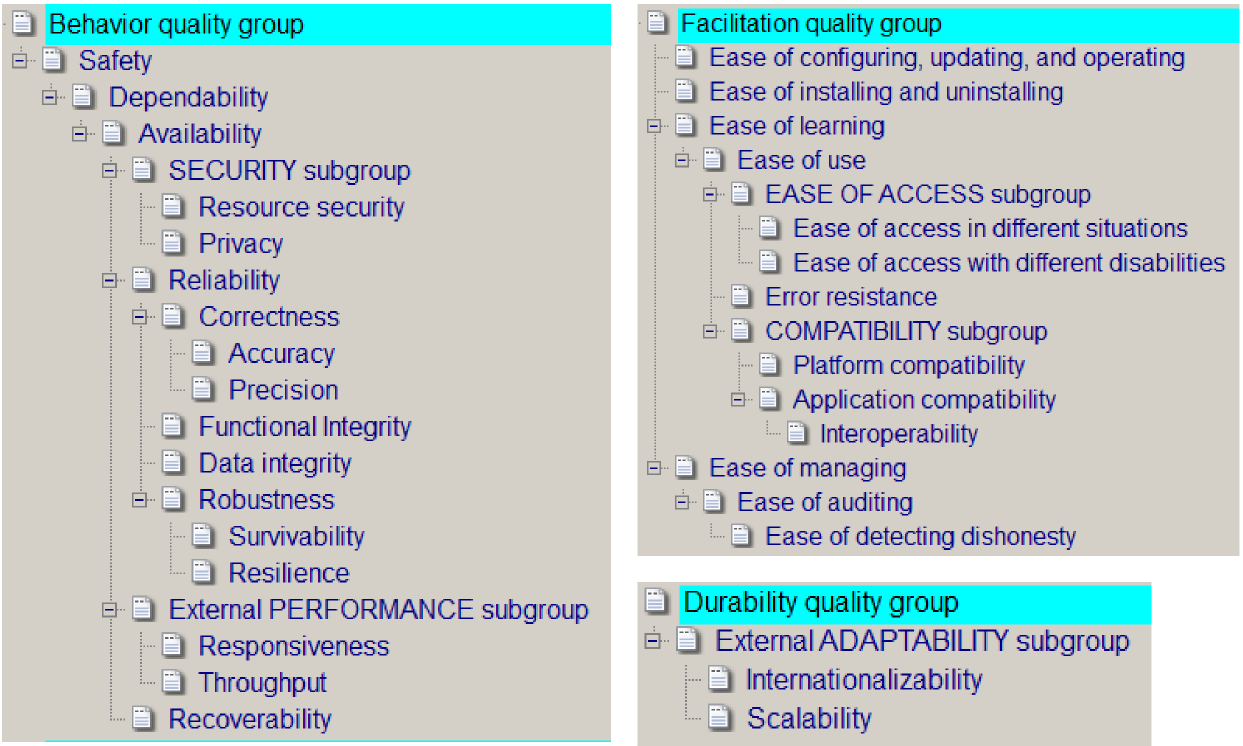
Figure 1. Top-level view of the LiteRM quality model



- Intro to LiteRM quality knowledge base
- Quality assumptions
  - Attribute assumptions
  - Model assumptions
- Tactics for definition, achievement, and verification
  - Definition tactics
  - Achievement tactics
  - Verification tactics
- EXTERNAL quality attributes
  - Facilitation quality group
  - Behavior quality group
  - Durability quality group
- INTERNAL quality attributes
  - Basic quality group
  - Behavior quality group
  - Durability quality group
- UNDEFINED quality attributes

The partitions (Figures 2 and 3) are then organized into groups and subgroups.

Figure 2. External quality groups and subgroups

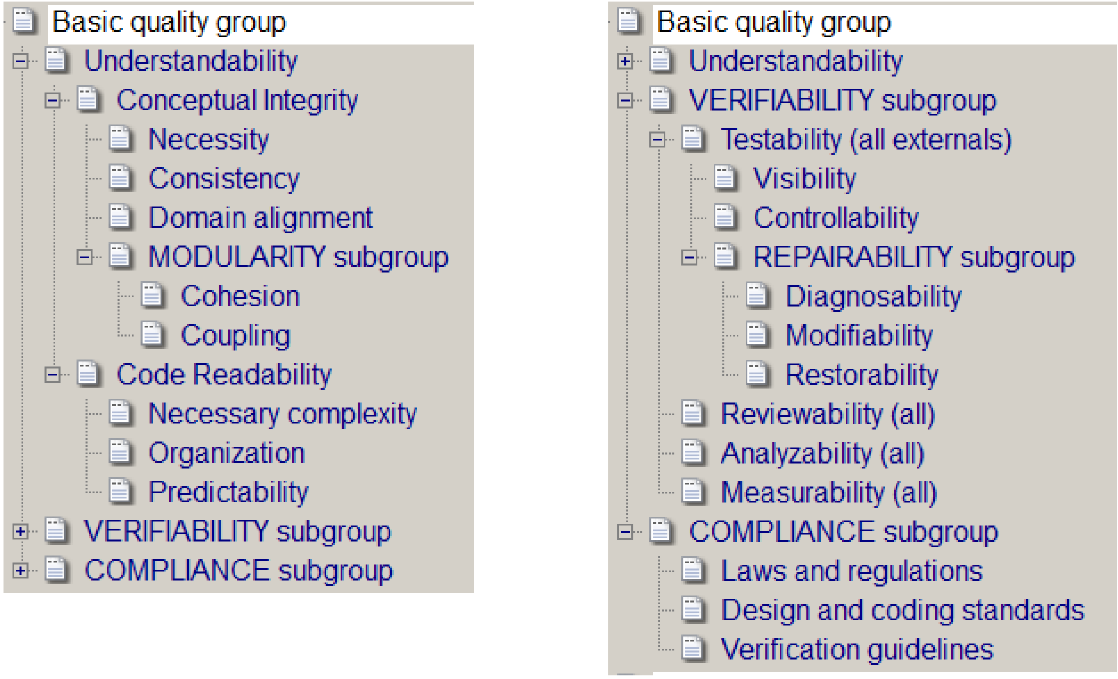


- Behavior quality group
  - Safety
    - Dependability
      - Availability
        - SECURITY subgroup
          - Resource security
          - Privacy
        - Reliability
          - Correctness
            - Accuracy
            - Precision
          - Functional Integrity
          - Data integrity
          - Robustness
            - Survivability
            - Resilience
        - External PERFORMANCE subgroup
          - Responsiveness
          - Throughput
        - Recoverability

- Facilitation quality group
  - Ease of configuring, updating, and operating
  - Ease of installing and uninstalling
  - Ease of learning
    - Ease of use
      - EASE OF ACCESS subgroup
        - Ease of access in different situations
        - Ease of access with different disabilities
      - Error resistance
      - COMPATIBILITY subgroup
        - Platform compatibility
        - Application compatibility
          - Interoperability
  - Ease of managing
    - Ease of auditing
      - Ease of detecting dishonesty

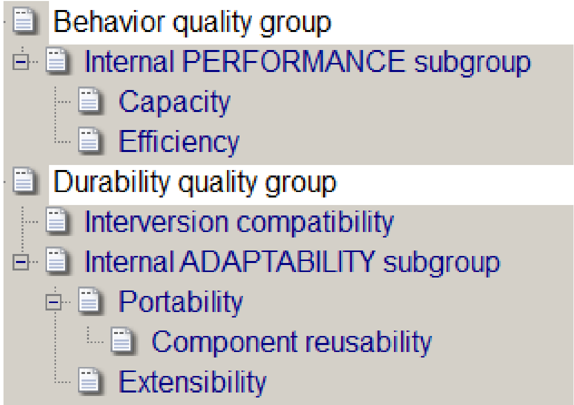
- Durability quality group
  - External ADAPTABILITY subgroup
    - Internationalizability
    - Scalability

Figure 3. Internal quality groups and subgroups



- Basic quality group
  - Understandability
    - Conceptual Integrity
      - Necessity
      - Consistency
      - Domain alignment
      - MODULARITY subgroup
        - Cohesion
        - Coupling
    - Code Readability
      - Necessary complexity
      - Organization
      - Predictability
  - VERIFIABILITY subgroup
  - COMPLIANCE subgroup

- Basic quality group
  - Understandability
  - VERIFIABILITY subgroup
    - Testability (all externals)
      - Visibility
      - Controllability
      - REPAIRABILITY subgroup
        - Diagnosability
        - Modifiability
        - Restorability
    - Reviewability (all)
    - Analyzability (all)
    - Measurability (all)
  - COMPLIANCE subgroup
    - Laws and regulations
    - Design and coding standards
    - Verification guidelines



- Behavior quality group
  - Internal PERFORMANCE subgroup
    - Capacity
    - Efficiency
- Durability quality group
  - Interversion compatibility
  - Internal ADAPTABILITY subgroup
    - Portability
      - Component reusability
    - Extensibility

Using safety, we show an example of the characteristics associated with an attribute.

Begin safety characteristics

Definition The ability of a system to do little or no harm to valuable assets

Software subfield safety engineering

Concerned stakeholders
customers, lawyers, tasked users, general public, designers

Assumptions/Rationale
Safety is a fragile quality because it depends on up to 39 other qualities
as well as on an accurate analysis of the hazards that must be managed

Quality attribute scenarios [described in **Software Architecture in Practice**]

Leading Indicators -- provide preoperational evidence of attribute goal attainment
Ratio of hazards added during HA technical review to hazard count after HA technical review
Ratio of defects found in safeguards during testing to number of safeguards

Operational Measures
Time since last "dangerous" failure or defect
Number of "dangerous" failures or defects detected per time interval
Greatest harm from a harmful event
Shortest harmful event free duration
Longest harmful event free duration
Expected length of harmful event free duration
Expected rate of harmful events
Ratio of actual loss to acceptable loss in a duration
Estimated residual risk

Supporting qualities – always or sometimes dependability, **ease of learning**

Note: While safety-critical functionality may be supported by these qualities,
at the same time they may conflict with non-safety-critical functionality.
For example, availability supports dependability, but it may cause non-safety-critical functionality

to be sacrificed so the system can continue to operate in a safe, but degraded mode.

Conflicting qualities adaptability qualities

Threats [identify using hazard analysis]

Mitigations [identify after identifying hazards]

Other achievement tactics

* identify valuable assets and hazards
* identify safety-critical and safety-related functions and constraints needed for safety e.g. "The Fire Detection System shall detect smoke above X ppm within 5 seconds**.**".
* isolate and protect safety-critical functions
* guard safety-critical functions with explicit conditions i.e. never with defaults such as "otherwise"
* identify safety-critical users
* eliminate or mitigate hazards i.e. identify appropriate control actions
* effectively execute control actions and receive accurate and sufficient feedback
* monitor system state to make sure safety-critical and safety-related functions are permitted
* alert users to dangerous actions with rotating warning messages
* precede each dangerous action with a delay so user can change their mind and cancel
* limit complexity
* design interfaces that prevent and detect user errors
* use warning labels and messages when appropriate

Verification tactics review hazards and mitigations for completeness and effectiveness during a safety audit, thoroughly test each safeguard, measure and track time since last "dangerous" failure or defect and number of "dangerous" failures or defects, verify all supporting qualities

Elicitation Questions

- What valuable assets are at risk
- Which functions are safety-critical or safety-related
- Who/What can perform these safety-critical or safety-related functions and under what conditions
- What harm can the system or its actors possibly do
- What can mitigate these hazards

Associated Tools

- Measurement
- Achievement
- Verification

Resources

"Quality Attributes" Technical Report CMU/SEI-95-TR-021 Chapter 6

Engineering a Safer World

Software Safety Primer

Risk Factors

a. Developer understanding =	[superficial, limited, deep]
b. Cost (implementation, verification, maintenance) =	[high, medium, low]
c. Feasibility (technical, cost, understanding) =	[low, medium, high]

Other Characteristics

a. Sources/Enterprise goals:	
b. Type = behavior quality	
c. **Associated scope** =	[system, <specific partitions>]
d. Design scope = crosscutting	[local, crosscutting]
e. **Consensus Priority** =	[critical, important, desirable]
f. Architecture-relevant = no	[yes, maybe, no]
g. Visibility group = externals	

States

a. Goal states are < @Unverified, Verified, Implemented, Inactive>

Notes

Past Goal Specs

Past Achievement and Verification Strategies

Current Goal Spec

Current Achievement and Verification Strategy

End safety characteristics

We propose that goals for relevant external qualities be identified and defined in a tailoring workshop that includes most stakeholders. Internal quality goals should be identified and defined in a meeting of technical managers and senior developers.

Stakeholders in the tailoring workshop should start by creating a project-relevant model of external quality attributes by:

1. Deleting external attributes and characteristics irrelevant to the project
2. Adding relevant external attributes and characteristics

3. Renaming and reorganizing the attributes and characteristics
4. Adding information to:
    a. Assumptions
    b. Indicators & Measures
    c. Threats & Mitigations
    d. Additional achievement tactics
    e. Verification tactics
    f. Other characteristics

The tailored model can be changed later based on new insights.

The tailored model can then be used to draft a set of external quality goals. Goals can be defined using characteristics such as assumptions, indicators, and measures. Other characteristic values such as priorities can also be defined.

When necessary information is not known, participants should receive research assignments. Responsibility should be assigned for drafting achievement and verification strategies after the workshop.

Research questions and draft strategies should be monitored by the requirements lead. Draft strategies should be reviewed for completeness and feasibility.

Early projects using this approach will do more original work. The organizations quality attributes model will grow richer as effective specs and other characteristic values are added based on project retrospectives. These effective specs and values can be reused on subsequent projects, making the tailoring progressively easier and more effective. The danger of inappropriate reuse should be considered.

A library of reusable support components (e.g. exception handlers), especially those that are crosscutting, should also be created.