Meta-requirements for critical software requirements

A straw-draft specification – V4

Definitions:

1. A **mission statement** specifies the goals of a mission.
2. **High-risk missions** may result in significant financial loss and/or serious injury or death.
3. **Critical software** supports the goals of high-risk missions.
4. The **general public** are those not directly involved in a mission.
5. **HRGP missions** are those whose failures are likely to endanger the general public e.g., self-driving vehicles.
6. A **quality goal** is a quality attribute requirement e.g., a security requirement.
7. **Basic qualities** (as described in the LiteRM quality model – www.quality-aware.com/daves-q-a-stuff.php-) are internal qualities including understandability, verifiability, and compliance.
8. We consider the following **types of software requirements**:

Types and subtypes of software requirements	Primary Sources	Inherent Risks
Functions		
Domain functions		
happy paths	Customers	Medium
unhappy paths	Developers	Medium
Quality support functions		
external quality supports e.g., exception handlers	Developers	Medium
mixed quality supports e.g., safeguards	Developers	High
Quality goals		
External qualities	Customers & Developers	Medium
Internal qualities	Quality Mgmt	Low
Mixed qualities	Customers & Developers	High
Constraints		
Technical		
Design e.g., reused components	Developers	Medium
Implementation e.g., coding standards	Quality Mgmt	Low
Verification e.g., test coverage	Quality Mgmt	Low
Deployment e.g., secure packaging	Developers	Low
Project e.g., deadlines	Project Mgmt	Low
Supplier attributes	Quality Mgmt	Low

9. Each requirement has a set of **common properties** such as definition, achievement and verification tactics, and priority.
10. **Technical inspections are skeptical** if inspectors assume the work is defective and diligently search for the defects.
11. **Meta-requirements** are constraints on a requirement or set of requirements.
12.

Assumptions:

1. We don't fully understand the complex software we are building.
2. Effective development of complex software requires humility and an emphasis on understandability.
3. When software endangers the general public, our loved ones deserve our best efforts.
4. "As is well known to [some] software engineers (but not to the general public), by far the largest class of problems arises from errors made in the eliciting, recording, and analysis of requirements."[1]
5. Many non-functional requirements (**NFRs)** are poorly understood by developers.
6. Legacy norms (i.e., that's the way we have always done it) can be dangerous when dealing with significantly increased complexity.
7. The rationale for each meta-requirement is to maximize understanding and/or minimize defects in software requirements.
8. The rationale for the set of meta-requirements is to prevent unnecessary harm.
9.

Meta-requirements:

1. Software requirements must be organized by types and subtypes.
2. Each requirement must have:
    a. an identifier showing its type or subtype
    b. properties identifying priority, status, source, and associated requirements
    c.
3. Functional requirements must be specified by rules specifying trigger conditions and associated actions.
4. Trigger conditions must be skeptically inspected to assure that each condition is neither too broad nor too narrow and is effective in capturing intent. Each action must be skeptically inspected to assure that it is safe and effective when its trigger conditions exist.
5. Trigger conditions for a set of functional requirements must be analyzed and skeptically inspected for correctness, feasibility, completeness, consistency, and necessity.
6. All basic qualities must be required (and aggressively verified).
7. The production and logging of assurance evidence during operation (e.g., by self-checking) must be required.

[1] Daniel Jackson, Martyn Thomas, and Lynette I. Millett, Editors
Software for Dependable Systems: Sufficient Evidence?
National Research Council 2007 [page 40]

8. The properties of a quality goal must specify its
    a. definition
    b. required level of achievement
    c. measurement strategy
    d. hazards
    e. mitigations
    f. achievement strategy
    g. verification strategy
    h. relationships to quality functions and other quality goals
    i.
9.

10. The set of requirements must have:
    a. A requirements glossary to precisely define conditions, actions, states, and any other words, phrases, acronyms, and abbreviations used in requirements statements. Actions must be defined by constant, pre, and post conditions. The glossary entries must be complete and consistent. The glossary must be configuration managed and readily accessible to any stakeholder.
    b. A set of user type specifications, if the software is interactive
    c. A set of domain object specifications
    d. A set of domain facts and assumptions
    e. A set of facts about values, conditions, and relationships
    f. A set of associated regulations
    g.
11. Each requirement must be understandable, correct, complete, concise, necessary, feasible, verifiable, and prioritized.
12. The set of requirements must be complete, consistent, feasible, and verifiable.
13. The requirements and their associated information must be skeptically inspected.
14. The set of requirements and their associated information for critical software supporting HRGP missions must be **made freely available to the general public** no later than the start of software design based on these requirements.
15.