WHITE PAPER

The Necessity of Well-Written and Measurable Non-Functional Requirements (NFRs)

A practical guide for business and technology leaders

Prepared for use in IT resiliency and non-functional requirements consulting engagements.

Version: 1.0

Table of Contents

1. Introduction

Most IT delivery teams are good at building features. Where organizations struggle is defining how those features must behave in the real world — under load, during failures, under attack, and while being operated day-to-day.

That “how” is captured in Non-Functional Requirements (NFRs). When NFRs are vague, you get guesswork, inconsistent implementations, late surprises, and expensive rework. When NFRs are measurable, you get predictable outcomes, simpler design decisions, faster testing, and less operational pain.

This paper explains why measurable NFRs are critical to building resilient systems, and how to write them in a way that is clear, testable, and useful across engineering, security, and operations.

2. What is a Non-Functional Requirement (NFR)?

An NFR is a specification that defines operational and performance characteristics of an application, system, or platform.

Functional requirements describe what a system does (features and behaviors). NFRs describe how the system must perform and operate (quality attributes).

Strong NFRs reduce ambiguity by translating business concerns — like customer experience, compliance, or revenue impact — into technical expectations that can be designed, implemented, and verified.

3. What is a Resiliency Tier Concern?

Resiliency is not a single metric. It is a set of capabilities that allow a service to withstand disruption, recover quickly, and continue delivering value.

A Resiliency Tier Framework groups expectations into tiers (for example, Tier 0 through Tier 4) so organizations can apply stronger requirements where the business impact is higher.

Each tier is defined by measurable concerns such as IT Operations (manageability), Availability (uptime and continuity), Recovery (RTO/RPO), and Security (protection and response). The tier makes tradeoffs explicit: not every system needs the same level of investment, but every system needs clear expectations.

4. IT Operations

Well-defined NFRs for IT Operations improve how a system is monitored, managed, supported, and changed over time.

Operational NFRs commonly address maintainability, observability, automation, scalability, and supportability. They help teams move from reactive firefighting to proactive operations.

Example operational NFRs include requirements for standardized logging, alerting thresholds, runbooks, health checks, deployment automation, and routine patching windows.

5. Availability

Availability requirements define how continuously a system must operate under normal and abnormal conditions.

Availability NFRs commonly include uptime targets, fault tolerance, redundancy patterns, load balancing, and failover mechanisms. These requirements drive architectural decisions such as multi-zone deployments, redundancy, and automated scaling.

Example: If a service has a 99.99% uptime target, that implies a strict maximum annual downtime budget (about 52 minutes). That budget should influence design, testing, and operational readiness.

6. Recovery

Recovery (Disaster Recovery / DR) requirements define how quickly a system must be restored after a disruption and how much data loss is acceptable.

Recovery NFRs typically define Recovery Time Objective (RTO) and Recovery Point Objective (RPO), and the mechanisms required to meet them (backup, replication, failover).

Example: A critical service might require recovery within 30 minutes (RTO) with no more than 5 minutes of data loss (RPO). That has immediate design implications for replication, backup frequency, and testing discipline.

7. Security

Security requirements define how systems protect data, prevent misuse, and respond to threats.

Security NFRs commonly address confidentiality, integrity, availability, access control, auditability, encryption, vulnerability management, and incident response.

Security NFRs work alongside security policies. Policies explain what must be true. NFRs define what must be built, configured, and verified to make the policy real in the system.

8. What “Well-Written” Looks Like for NFRs

A well-written NFR is clear, measurable, and testable. It does not leave room for interpretation.

A practical structure is: Category + Condition/Scope + Measurable Target + Verification method.

Use the SMART principle: Specific, Measurable, Achievable, Relevant, and Time-bound.

Avoid vague language like “fast,” “secure,” or “highly available.” If it matters, quantify it and define how it will be tested.

- Here are some common pitfalls to avoid:
    - Combining multiple requirements into one statement (hard to test and enforce).
    - Using subjective terms without measurable thresholds.
    - Ignoring verification (no acceptance criteria, no test method).
    - Writing NFRs that are disconnected from business impact or tiering.
- Example NFR patterns:

1. **Performance:** Under a sustained load of 10,000 concurrent users, the service shall process 95% of requests within 2 seconds, measured at the API gateway over a 15-minute interval.
2. **Availability:** The service shall meet 99.95% monthly availability, excluding planned maintenance windows approved at least 7 days in advance.
3. **Recovery:** In the event of a regional outage, the service shall restore read/write capability within 30 minutes (RTO) with no more than 5 minutes of data loss (RPO).
4. **Security:** All sensitive data shall be encrypted in transit using TLS 1.2+ and at rest using AES-256 (or equivalent), verified via configuration review and automated security scanning.
5. **Operations:** The service shall emit structured logs (JSON) with correlation IDs for all requests and publish SLO-based alerts for error rate, latency, and saturation to the centralized monitoring platform.

9. Case Studies

Availability Study 1: Global streaming platform

A global streaming platform depends on near-continuous availability. Measurable NFRs around redundancy, automated failover, multi-region deployment, and real-time monitoring shaped the architecture. Chaos testing validated that the availability requirements held up under failure conditions.

Benefit: The platform maintained uninterrupted service during infrastructure failures and traffic spikes, reinforcing customer trust.

Availability Study 2: Electronic health record (EHR) access

A large healthcare provider required strict availability for patient record access. The system was designed with redundancy, data replication, and automated failover to meet a 24/7 access requirement and aggressive recovery targets.

Benefit: Clinicians could access records reliably during maintenance and unexpected failures, improving care delivery and safety.

Summary

In both examples, availability stopped being a vague goal and became an enforceable standard through measurable NFRs. That clarity drove better design, implementation, and validation.

10. Summary

NFRs are how organizations turn business expectations into engineered outcomes. They define the operational standards systems must meet to be reliable, resilient, and secure.

Well-written, measurable NFRs reduce misunderstandings, accelerate delivery, and prevent costly late-stage rework. They also create a common language across business stakeholders, engineers, security, and operations.

In short: if you want predictable resiliency, you need measurable operational requirements.

References (optional reading)

- Business Analysis Toolkit – Non-Functional Requirements overview.
- Forbes Technology Council – Discussion on why NFRs matter.
- General software engineering references on quality attributes, resilience, and SRE practices.