(/)

> The devil is in the details.
>
> —Common proverb

Nonfunctional Requirements

Nonfunctional Requirements (NFRs) define system attributes such as security, reliability, performance, maintainability, scalability, and usability. They serve as constraints or restrictions on the design of the system across the different backlogs.

Also known as system qualities, nonfunctional requirements are just as critical as functional Epics (/epic/), Capabilities, Features (/features-and-capabilities/), and Stories (/story/). They ensure the usability and effectiveness of the entire system. Failing to meet any one of them can result in systems that fail to satisfy internal business, user, or market needs, or that do not fulfill mandatory requirements imposed by regulatory or standards agencies. In some cases, non-compliance can cause significant legal issues (privacy, security, safety, to name a few).

NFRs are persistent qualities and constraints that, unlike functional requirements, are typically revisited as part of the Definition of Done (DoD) for each Iteration (/iterations/), Program Increment (PI) (/program-increment/), or release (/release-on-demand/). NFRs influence all backlogs: Team (/team-backlog/), Program (/program-and-solution-backlogs/), Solution (/program-and-solution-backlogs/), and Portfolio (/portfolio-backlog/).

Proper definition and implementation of NFRs is critical. Over-specify them, and the solution may be too costly to be viable; under-specify or underachieve them, and the system will be inadequate for its intended use. An adaptive and incremental approach to exploring, defining, and implementing NFRs is a vital skill for Agile teams.

Details

Functional requirements expressed in user stories, features, and capabilities represent most of the work in building solutions that deliver value to the user. Although they may be a bit subtler, NFRs are just as important to system success as they describe critical operational qualities required for release (or manufacture, or sell).

NFRs are not themselves backlog items. They are constraints on development that limit some degree of design freedom for those building the system. These constraints are often defined in the acceptance criteria for multiple backlog items. For example, SAML-based Single Sign-on (SSO) is a requirement for all products in the suite. SSO is a functional requirement, while SAML is a constraint. And any backlog item building sign-on functionality would reference the SAML constraint in its acceptance criteria.

'FURPS' is a commonly referenced set of important quality attributes: Functionality, Usability, Reliability, Performance, and Supportability [1]. The NFR list is more exhaustive and includes compliance, security, resilience, privacy, accessibility, and others [2].

NFRs Occur with All Backlogs

NFRs are associated with backlogs throughout SAFe, as Figure 1 illustrates

NFRs are associated with backlogs throughout SAFe, as Figure 1 illustrates.

Figure 1. NFRs occur with all backlogs in SAFe

Because NFRs are significant attributes of the solution that the Agile Release Train (ART) (/agile-release-train/) and Value Streams (/value-streams/) create, they most often influence the backlogs of ARTs and Solution Trains (/solution-train/). System and Solution Architect and Engineering (/system-and-solution-architect-engineering/) are often responsible for defining and refining these NFRs.

As teams implement these features and capabilities, NFRs roll down to constraint team-level backlog items. Teams use Built-In Quality (/built-in-quality/) practices to accelerate NFR testing and make it continuous. They include the relevant NFRs into their DoD, use them as constraints on local design and implementation decisions, and take responsibility for some level of NFR testing on their own.

The portfolio backlog may also require NFRs. This is often the case for cross-system qualities, like single sign-on. Other examples include restrictions on open source usage, security requirements, and regulatory standards.

NFRs as Backlog Constraints

NFRs are modeled as backlog constraints in the framework, as is illustrated in Figure 2.

Figure 2. Backlogs are constrained by NFRs

NFRs may constrain any number of backlog items as described in the SAFe Requirements Model (/safe-requirements-model/). In order to know that the system complies with the constraint, most NFRs require one or more system qualities tests (ideally automated), as is illustrated in Figure 3.

Figure 3. The relationship between backlog items, NFRs, and system qualities tests

NFRs can begin as enablers that explore design alternatives and build infrastructure to support the constraint. Once implemented, the NFRs then constrain the system and new backlog items.

The Impact of NFRs on Solution Development

Nonfunctional requirements can have a substantial impact on solution development and testing. Architects and developers should use caution when specifying them. For example, a statement like “99.999 percent availability” may increase development effort exponentially more than “99.98 percent availability.” The impact of the NFR must be well understood by those defining requirements.

In many cases, applying Set-Based Design (/set-based-design/) can keep options open (/assume-variability-preserve-options/) by initially specifying NFRs as a range (e.g., 99.98 .. 99.999). Teams explore the solution space and gain additional knowledge that leads to a better economic decision. There may indeed be value in 99.999 reliability. Or lower reliability may be more cost-effective with adjustments made elsewhere in the system’s operational environment. Physical constraints such as weight, volume, or voltage work impact solution development in similar ways. Deferring decisions to better understand costs and value often lead to better economics.

The solution’s Economic Framework (/take-an-economic-view/#economicframework) should contain criteria to evaluate NFRs. NFRs should be viewed in the context of trade-offs with costs and other considerations. NFRs also affect Suppliers (/supplier/) and their knowledge and concerns should inform NFR specifications and the economic framework.

NFRs and Solution Intent

Solution Intent (/solution-intent/) is the single source of truth about the solution. As such, it includes NFRs as well as functional requirements. It also includes traceability links between NFRs, other requirements they impact, and tests used to verify them. NFRs play a key role in understanding the economics of fixed versus variable solution intent.

Figure 4. Solution Intent

Figure 4. Solution Intent

Similar to other requirements, some NFRs are fixed and well known in advance; others will evolve with the solution.

NFRs may impact a wide range of system functionality. Therefore, they're an important factor to consider when:

- Analyzing business epics, capabilities, and features
- Planning and building the Architectural Runway (/architectural-runway/)
- Refactoring (/refactoring/) to better reflect increasing solution domain knowledge
- Imposing constraints on manufacturing, deployment, support, installation, maintainability, and so on

The tools used to help develop the solution intent provide some mechanisms to establish an economic approach to define and implement NFRs:

- **Compliance** – Compliance drives many NFR decisions to ensure the solution meets regulatory, industry, and other relevant standards and guidelines. The solution intent often includes traceability from the system's NFRs to those regulatory standards.
- **Model-Based Systems Engineering (MBSE)** – MBSE can be used to simulate the effect of NFRs and support tracing to the designs the implement them and tests that validate them
- **Set-based Design (SBD)** – SBD provides different options for achieving NFRs as described earlier in this article

Specifying NFRs

Like all other requirements, NFRs must be quantified for clarity to ensure stakeholders' desires are clearly understood by everyone. Figure 5 from [3] quantifies NFR requirements. Step 1 defines the NFR's quality including its name, scale, and method to measure. Step 2 quantifies the NFR's measurable values including the current measured value (baseline), the value to achieve (target), and the value that becomes unacceptable (constraint). The example in Figure 5 shows the measurable efficiency of an autonomous vehicle's speed limit detection. On average, users currently set the speed manually .1 time per mile, overriding the automated solution. New system functionality will improve to .01 times per mile, but during the implementation should never fall below .15 times per mile.

Figure 5. Defining an NFR quality

The following criteria help define NFRs:

- **Bounded** – When they lack bounded context, NFRs may be irrelevant and lead to significant additional work. For example, an airplane's flight controls should be more rigid reliability than the infotainment system.
- **Independent** – NFRs should be independent of each other so that they can be evaluated and tested without consideration of or impact from other system qualities.
- **Negotiable** – Understanding NFR business drivers and bounded context mandates negotiability.
- **Testable** – NFRs must be stated with objective, measurable, and testable criteria.

Implementation Approaches

Many NFRs prescribe that some additional work must be done—either now or in the future—to satisfy them. Sometimes the NFR must be implemented all at once; other times the teams can take a more incremental approach. The trade-offs described in the economic framework should impact the implementation approach. Implementation should occur in a way that will allow several learning cycles to ascertain the right level of NFR.

- **All at once** – Some NFRs appear as new concerns and will require immediate implementation. For example, any regulatory charges require the organization to respond within the specified time constraints or risk being in violation.
- **Incremental story-by-story path** – At other times, the teams have options. For example, the need for improved performance can be addressed over time, one story at a time, as Figure 6 illustrates.

Figure 6. Incremental implementation of an NFR

NFR implementation is also impacted by the way ARTs have been organized. ARTs built around architectural layers will find it challenging to implement and test an NFR in its entirety. Trains organized around

capabilities, however, will find it easier to implement, test, and maintain systemic NFRs.

Applying Agile Architecture (/agile-architecture/) practices supports the development of NFRs and helps maintain flexibility as the requirements evolve.

Testing Nonfunctional Requirements

Like all system requirements, NFRs must be tested. Quadrant 4 of the Agile Testing (/agile-testing) Matrix, 'system qualities tests,' defines most NFR tests. Due to their scope, NFR tests often require collaboration between the System Team and the Agile Teams (/agile-teams/). To support building quality in, teams should automate wherever possible so that tests can be run continuously or on demand.

Learn More

[1] Leffingwell, Dean and Don Widrig. Managing Software Requirements: A Use Case Approach (second edition). Addison-Wesley, 2003.

[2] https://en.wikipedia.org/wiki/Non-functional_requirement (https://en.wikipedia.org/wiki/Non-functional_requirement)

[3] Leffingwell, Dean. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley, 2011.

[4] Leffingwell, Dean and Ryan Shriver, Nonfunctional Requirements (System Qualities) Agile Style, Agile 2010.

[5] Larman, Craig, and Bas Vodde. Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum. Addison-Wesley, 2010.

Last update: 30 June 2020

Partner Directory (https://www.scaledagile.com/find-a-partner/)

GET SOCIAL

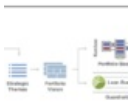
Twitter (https://twitter.com/ScaledAgile)

Linkedin (https://www.linkedin.com/company/scaled-agile-inc-/)

YouTube (https://www.youtube.com/user/scaledagile)

SlideShare (http://www.slideshare.net/ScaledAgile)

RECENT POSTS



Updates to Lean Portfolio Management (https://www.scaledagileframework.com/blog/updates-to-lean-portfolio-management/)

Aug, 20th 2020

COVID-19 is Redefining Face-to-Face Communication: Updated Framework Guidance (https://www.scaledagileframework.com/blog/covid-19-is-redefining-face-to-face-communication-updated-framework-guidance/)

Aug, 12th 2020

SAFe® for Marketing (https://www.scaledagileframework.com/blog/safe-for-marketing/)

Jul, 20th 2020

SCALED AGILE, INC

CONTACT US

5400 Airport Blvd., Suite 300
Boulder, CO 80301 USA

BUSINESS HOURS

Weekdays: 9am to 5pm
Weekends: CLOSED

Privacy Policy (https://www.scaledagile.com/privacy-policy/)
Cookie Policy (https://www.scaledagile.com/cookie-policy/)
Your California Consumer Rights (https://www.scaledagile.com/privacy-policy#noticetoCAconsumers)