Topic: *SMART Non-Functional Requirement (NFR)* Elicitation

Context and Motivation

Quality Attribute (QAs) describe *how* a system provides its functionality, not what it does (which is the purpose of functional requirements specifications such as use cases or user stories). Examples of key QAs are reliability, usability, efficiency (e.g., performance, scalability), maintainability, and portability.[1]

Expressive, unambiguous NFRs that find their way into project plans and the minds of the project participants are the main output of architectural analysis. Without such NFRs, architecture design work becomes a blind flight; the project team is at the mercy of the client or internal project sponsor who can come up with new and more advanced requirements as they wish.

Practical challenges of NFR/QA elicitation include:

- Many QAs are exist and usually the project budgets do not allow to specify all of them precisely. Furthermore, many conflicts between them exist (for instance, implementing a security feature has a performance cost); a prioritization is required, but hard to commit to and agree upon (for many stakeholders).
- QAs are often stated on inadequate level of abstraction, for instance per system and not per logical function or business process step. They keep on changing.
- Many attributes are hard to quantify. Hence, QAs often remain under-specified (so they are unknown and cannot serve as the base of design decisions and acceptance tests) or are over-specified (and therefore overly ambitious and hard to meet).

Therefore, it is desirable to establish criteria and templates that allow architects and other participants in architectural analysis to overcome these challenge so that NFR elicitation and architectural analysis become effective and efficient.

Concept(s) and Definition(s)

According to the Unified Process (UP), "a requirement describes a condition or capability to which a system must conform; either derived directly from user needs, or stated in a contract, standard, specification, or other formally imposed document." (source: https://www.ibm.com/developerworks/rational/library/4706.html).

An NFR then can be defined as an *attribute* (often also called a property) of a system or a *constraint* on such system. The term constraint is defined as something that restricts and limits the solution space beyond what is needed to meet the functional requirements and the specific qualities attributes that have been stated by the stakeholders.[3]

Some NFRs are system-wide and cut across viewpoints and elements; others are more local. External, observable qualities can be distinguished from system-internal quality properties not visible at the system boundary, which brings us back to the five ASR criteria from lesson 1: some of these criteria talked about "business value" and stakeholder concerns. Business value is a rather broad term and can have less obvious meanings depending on project context and purpose of the software under construction. For instance, what is the business value of software embedded in a vehicle? The "so that" part of user stories and the project vision are two examples of sources of information on business value.

arc42 recommends to have top three to five QAs in Section 1 of architecture descriptions[4], suggests a Section 2

[1] This list stems from ISO/IEC standard 9126[2]. ISO 9126, now superseded by a newer standard, also lists functionality as a quality category.
[3] Note that these definitions are adapted versions of the definitions given in Glinz (2007).
[4] http://docs.arc42.org/section-1/

dealing with constraints[5] and puts the detailed quality requirements section towards the end in Section 10[6].

First Examples

In the telecommunications order management system featured in the lecture in weeks 1 and 2 (Zimmermann et al. (2005)), requirements that deal with external and internal quality properties include:

- *Accuracy*: orders must not be lost, resource reservations must be undone
- *Efficiency* (here: performance): sub-second response times specified
- *Interoperability*: multiple platforms to be supported
- *Modifiability*: skills for selected technologies must be available locally

An example of a constraint is "only relational database management system X can be used because an enterprise-wide licensing agreement is in place and the required database administration skills have been built up".

Taxonomies

There are many different NFRs/QAs. Many of these pertain to the runtime, others deal with software support and maintenance. Therefore, many attempts have been made to organize the QA landscape (ordered from informal and ad hoc to formal and complete):

- The basic, easy-to-remember *FURPS+* classification introduced in the lecture and used in exercise 1, originally introduced in a software engineering book from the 1990s and explained in this article by P. Eeles[7].
- ISO 9126 and its successor ISO 25010[8], also used by in "Effektive Softarearchitekturen" Starke (2015) and recommended for use in arc42 descriptions.
- SEI quality utility trees, see this Technical Report, this article[9] by Arnon Rotem-Gal-Oz and arc42 tip 10-2[10].

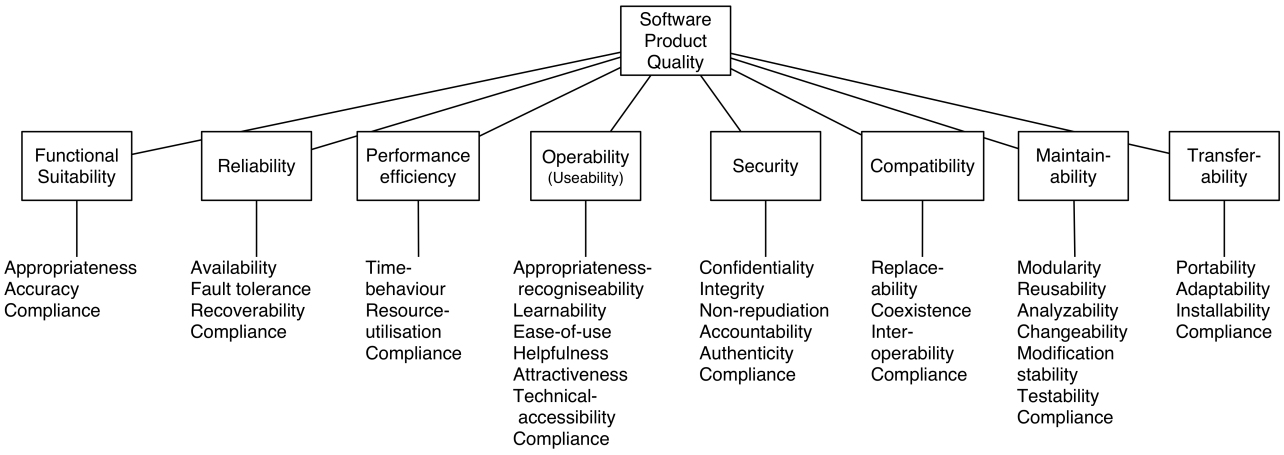

Figure 1: *NFR Classification from ISO 25010, from arc42 repo https://github.com/arc42/quality-requirements*

[5] http://docs.arc42.org/section-2/
[6] http://docs.arc42.org/section-10/
[7] https://www.ibm.com/developerworks/rational/library/4706.html
[8] http://iso25000.com/index.php/en/iso-25000-standards/iso-25010
[9] http://arnon.me/2010/05/utility-trees-hatching-quality-attributes/
[10] http://docs.arc42.org/tips/10-2/

SMART criteria (adopted for NFRs):

SMART criteria[11] are frequently used in project and people management. All five letters can have different meanings ; here, we use these meanings: *S* for specific, *M* for measurable, *A* for agreed upon, *R* for realistic, and *T* for time-bound.

The SMART criteria can be applied to NFR elicitation in a straightforward way and therefore serve as "meta-qualities" (quality attributes of/for quality attributes, that is):

- S: Which feature or part of the system should satisfy the requirement?
- M: How can testers and other stakeholders find out whether the requirement is met (or not)? Is the requirement quantified?
- A: Do all affected internal and external stakeholders agree on the "S" and the "M" wording? (issue for requirements engineering and project management, so out of scope here and now)
- R: Is it technically and economically feasible to achieve the "M" measure in the context of all features or system parts specified under "S"? (issue for requirements engineering and project management, so out of scope here and now)
- T: When should the NFR meet the "M" measure, is there a growth path from iteration to iteration? (also to be answered by project management, so out of scope here and now)

An example and a counter example are (which one is SMARTer than the other?):

- "The system should be highly usable, perform well and easy to maintain."
- "The 'place order' user story must complete in less than a second."

See lecture slides and exercise 2 for more examples.

SMARTness assessments can be recorded in the following way:

ASR	Specific (Y/N)?	Measurable (Y/N)?	Rationale for Answers	Improvement (if needed)
R1	[Y/N]	[Y/N]	(answer to question from above)	(required change or n/a)
Rn	[Y/N]	[Y/N]	...	...

Quantification Templates

To achieve the S and the M in SMART, several templates have been proposed:

- *Quality Attribute Scenarios (QAS)* were introduced in books Bass, Clements, and Kazman (2012) and technical reports from the Software Engineering Institute (SEI).
- User story annotations and quality stories are a more recent addition to the architect's toolbox, see this presentation[12] and this article[13].
- Other proposals include, for instance, the SOPHisten templates and PLANGUAGE (see below).

The degree of practical adoption of these templates varies.

[11] https://en.wikipedia.org/wiki/SMART_criteria
[12] https://sagra2016.files.wordpress.com/2016/10/zio-towardsopenleanarchitectureframework-sagranov2016v10p.pdf%3E
[13] https://www.ifs.hsr.ch/fileadmin/user_upload/customers/ifs.hsr.ch/Home/projekte/ARC-SummerSoCSubmission2015v11.pdf

Utility Trees

To elicit and refine NFRs/QAs incrementally and to maintain an overview ("big picture"), some architects recommend to display them in a hierarchical fashion called *quality utility trees*. When navigating in the tree from the root to the leaves, the NFRs/QAs get more and more precise; the leaves can be assessed according to their business value and technical risk (remember the first ASR criterion from lesson 1).

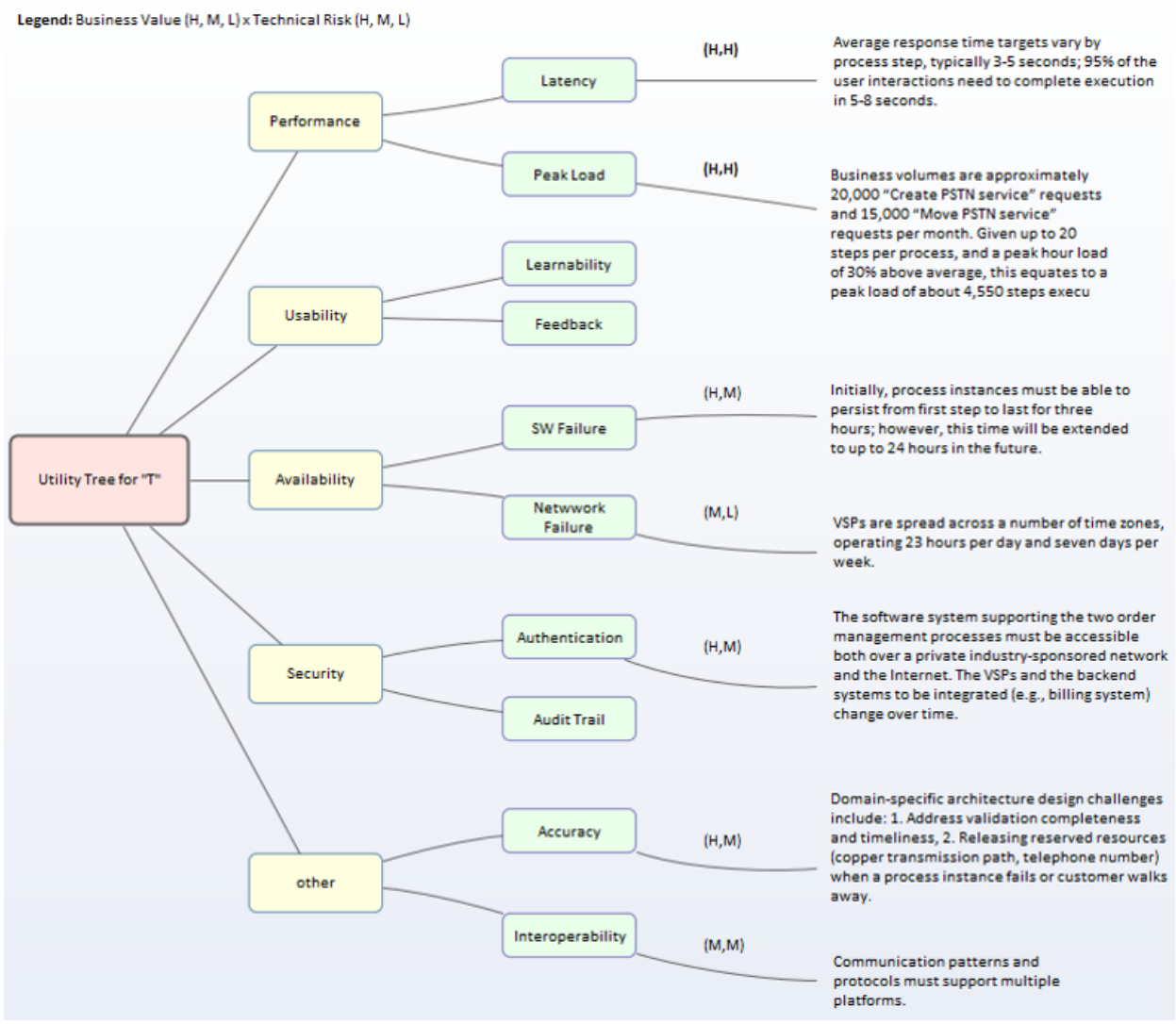

Figure 2: *Quality utility tree example: NFRs in T case prioritized*

See lecture slides and these resources:

- http://arnon.me/2010/05/utility-trees-hatching-quality-attributes/
- https://github.com/arc42/quality-requirements
- http://www.sei.cmu.edu/reports/95tr021.pdf

Application in Products and Projects

Usage of Concept/Topic

SEI Quality Attribute Scenarios (QAS) and Quality Attribute Workshop (QAW) Quality attributes and their quantification with QAS has been a core topic in software architecture since the inception of the field. An early tech-

nical report is http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6687 and the classical book from SEI authors is "Software Architecture in Practice", 3rd Edition[14] (Bass, Clements, and Kazman (2012)).

The example scenario "Scenario Refinement for Scenario Order Management at 'T' (Business-to-Business)" demonstrates the usage of the template. Note that we are still in analysis mode when writing QASs; the response should therefore not describe or even prescribe a design, but specify a requirement (postcondition). The response measure brings the "M" in SMART; the "S" is achieved by the stimulus and the stimulus source.

Scenario for Order Management SOA at "T" (Business-to-Business), NFR #5		
Scenario(s)		Response Time in Web Channel (serving VSPs)
Business Goals		Drive down cost of operations by interacting with VSPs efficiently
Relevant Quality Attributes		Performance (response time), scalability
Scenario Components	**Stimulus**	Order placed by VSP (system or end user)
	Stimulus Source	External to system (see functional requirements and architecture overview diagram: "Create PSTN", "Move PSTN")
	Environment (Worst Case)	Normal operation, at runtime
	Artifact (If Known)	Entire system, all logical layers and all physical tiers (including customer database in the backend)
	Response	Orders are accepted immediately, order number (identifier) is returned
	Response Measure	Order acknowledgment is sent in 3 seconds (or less). This performance is desired but not guaranteed to VSPs in any Service Level Agreement (SLA),
Questions		Can errors be identified and handled properly in this timeframe?
Issues		Backend systems might not have sufficient processing power

Figure 3: *Quality Attribute Scenario Example ("T" Case Study)*

See Utility Trees – Hatching quality attributes[15] by A. Rotem-Gal-Oz for another example and advice how to fill out the template (in the second half of the post; the first half deals with quality utility trees). A real-world system and exemplary QASs from it are presented in this video[16].

For a report on a trimmed down QA elicitation effort called *Mini QAW* see a SATURN presentation[17] and an IEEE Software article[18]. Another lightweight version (also not from the SEI) is reported here[19].

Quality Stories When functional requirements are elicited as agile user stories[20] following the M. Cohn template ("As as [role,stakeholder], I want to [capability/feature] so that [benefit, business value]."), one way of making NFRs specific is to simply annotate the user stories with them (see Figure "Quality Story Variant 1").

System-wide and/or cross cutting concerns can be specified in a quality story template that resembles the original one for user stories:

An example follows ("Quality Story Variant 2: Example]").

[14] https://www.pearson.com/us/higher-education/program/Bass-Software-Architecture-in-Practice-3rd-Edition/PGM317124.html
[15] http://arnon.me/2010/05/utility-trees-hatching-quality-attributes/
[16] https://www.youtube.com/watch?v=J3oig-AQGSg
[17] http://resources.sei.cmu.edu/library/asset-view.cfm?assetID=89553
[18] http://ieeexplore.ieee.org/document/7093052/?arnumber=7093052
[19] http://www.codingthearchitecture.com/2015/01/04/lightweight_quality_attribute_workshop.html
[20] https://www.agilealliance.org/glossary/user-stories/

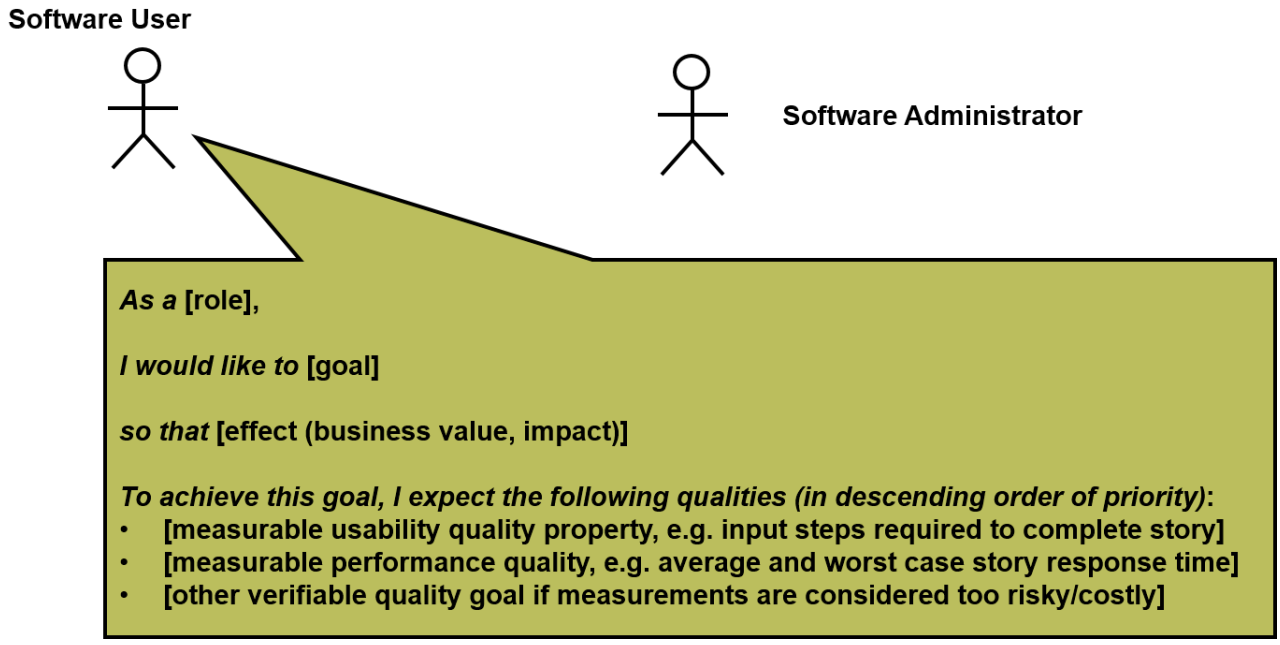

Figure 4: *Quality Story Variant 1: Annotated User Story*

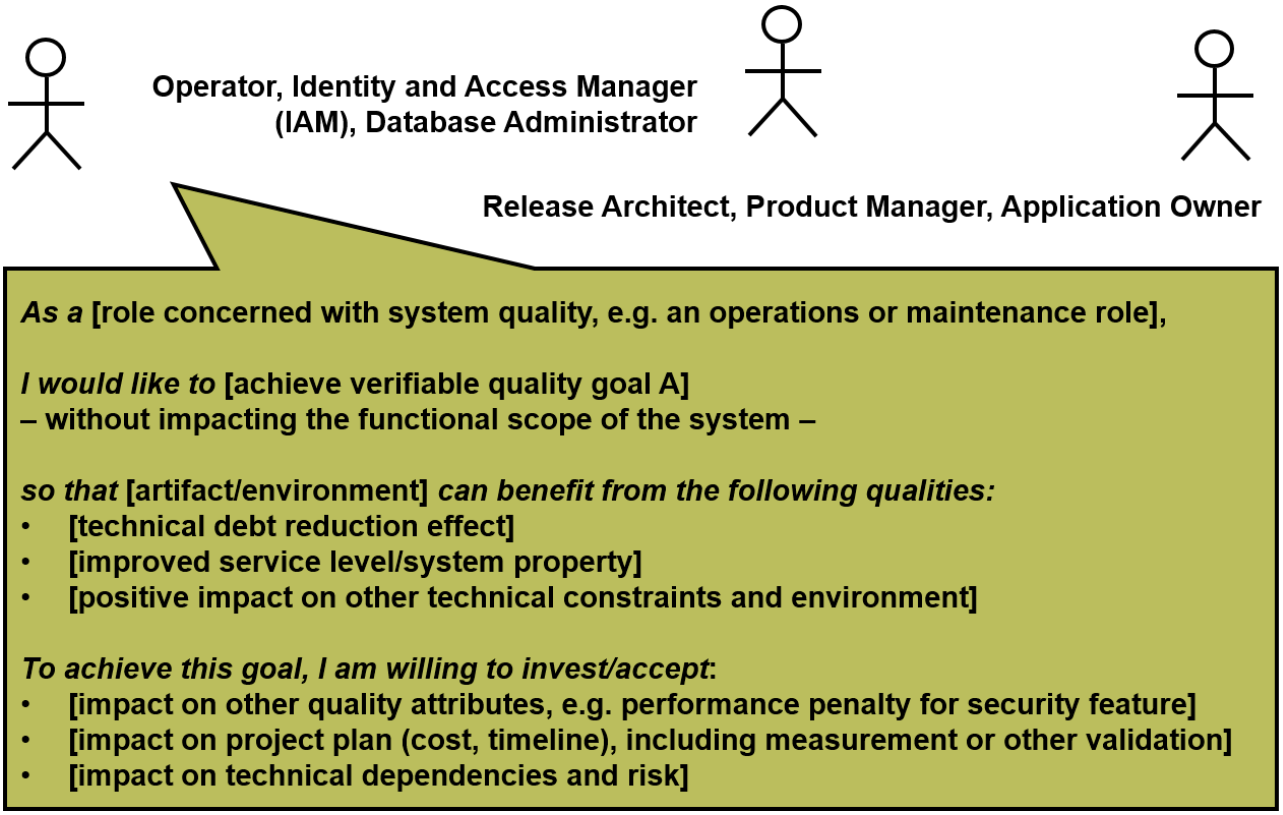

Figure 5: *Quality Story Variant 2 (Standalone): Template*

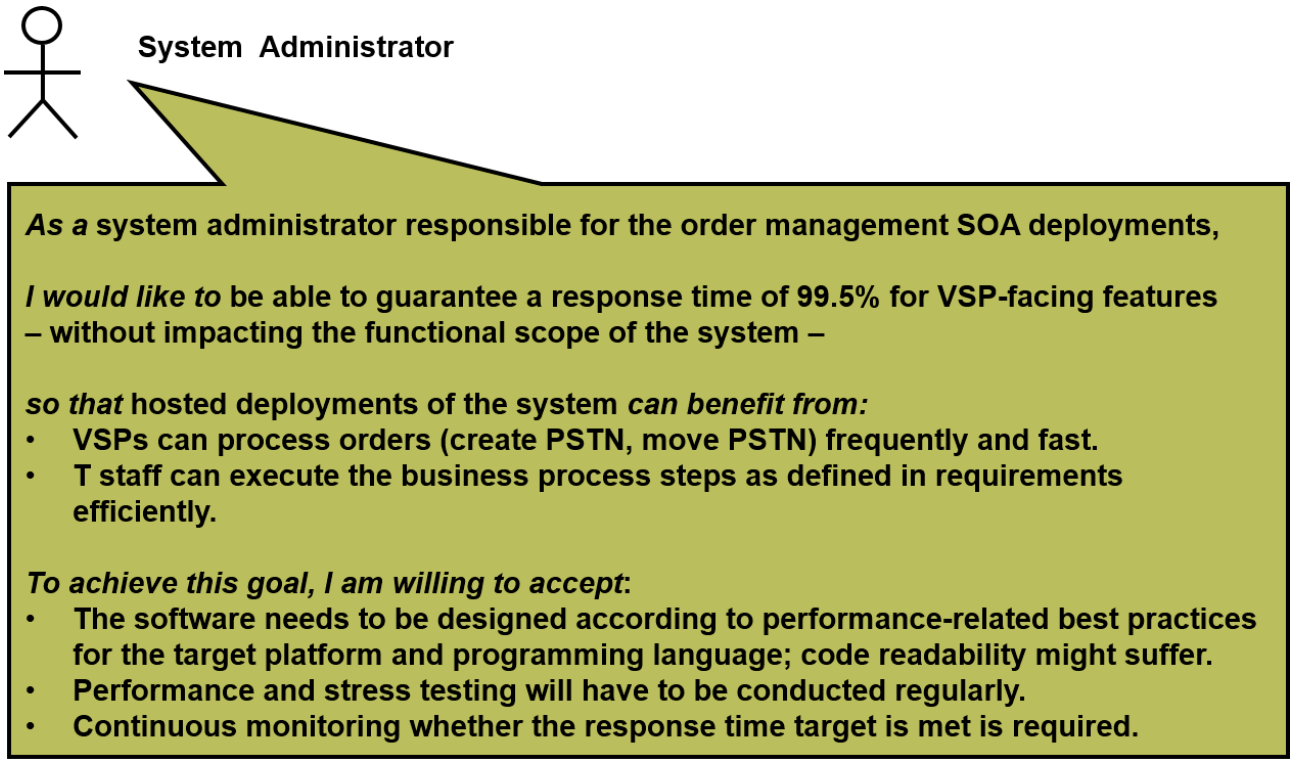

Figure 6: *Quality Story Variant 2 (Standalone): Example*

The quality story approach has been developed independently of Disciplined Agile (AD). One of the authors of DA discusses his take on agile NFRs here[21].

SOPHisten Templates (in German) The SOPHisten are a German speaking consultancy/community specializing on requirements engineering. They also propose an NFS elicitation template:

- Part 1: http://blog.sophist.de/2013/07/24/masterhaft-satzschablonen-fur-nicht-funktionale-anforderungen/
- Part 2 and 3: http://blog.sophist.de/2013/08/14/von-umgebung-und-mensch-satzschablonen-nummer-2-und-3-fur-nicht-funktionale-anforderungen-teil-2/

PLANGUAGE PLANGUAGE is a proposal by T. and K. Gilb from the 1980s/1990s that is still actively used, for instance at Intel. Their own website is a bit light on content, but links several articles. The approach has been picked up and applied by others:

- http://concepts.gilb.com/dl44
- http://www.seilevel.com/requirements/specifying-quality-requirements-with-planguage
- http://www.syque.com/quality_tools/tools/Tools104.htm

Risk vs. Value of Quantified NFRs (M. Glinz)

In an IEEE software article[22], M. Glinz (em. professor of software/requirements engineering at Zurich University) argues that *not* all NFRs should be quantified (the "M" in SMART), but that decision should be on criteria related to effort vs. business value and risk. He presents five situations and derives recommendations regarding adequate representations and verification techniques (Table 1 in referenced article).

[21] http://www.disciplinedagiledelivery.com/strategies-for-implementing-non-functional-requirements/
[22] https://files.ifi.uzh.ch/rerg/amadeus/publications/papers/Glinz_IEEE_Sw_2008.pdf

Tips and Tricks

When eliciting NFRs, the following rules of thumb apply:

- Do not use "information not available" or "too difficult to do" or "we are agile" as easy excuses for not eliciting NFRs/QAs; justify why certain ones are not specified with a risk and value argumentation.
- Make assumptions explicit; wrong numbers are better than none (also see FAQ section in exercise).
- Select one NFR/QA taxonomy when starting the NFR elicitation effort and stick to it to organize the NFR artifact; you can use it as a checklist too (was anything forgotten?).
- Follow the implementation advice given in the Disciplined Agile Delivery (DAD)[23] framework.
- Think about measurements that might not be obvious at first glance: number of screens and pop up windows in a user interface, time it takes to complete a certain task (like fixing a bug of a certain criticality), time it takes to download and install and run a particular software component or library (including time to understand and modify the samples that come with it).
- Take enterprise-wide requirements, constraints and principles into account, but do not accept and follow them blindly. If you think you have to deviate from corporate policies such as "two-factor authentication must be used on all client-facing channels" for good reasons, document the exempt request, negotiate it and seek for the required approvals.
- If some NFRs/QAs turn out to be hard to quantify exactly, even taking the advice in this fact sheet and in the referenced literature into account, consider to group the requirements into basic-intermediate-premium tiers; such approaches are commonly used to standardize and the availability needs, for instance in IT infrastructure delivery organizations that charge for meeting certain service level agreements.

See lecture slide "Practical Tips and Tricks (NFRs and QAs) and arc42[24] for more tips and implementation advice.

More Information

The literature on NFRs and QAs is rich but fragmented, see for instance several SEI TRs such as the one on Quality Attribute Workshops (QAW)[25] and books and an Application Architecture Guide[26] from Microsoft for more information on other QAs. The SOPHisten blog on requirements engineering has an NFR tag[27]. The IFS website Architectural Knowledge Hubs[28] has more pointers on agile architecting in general.

Performance and availability were defined and studied in VSS (e.g. related arch. tactics). Security in InfSys 1 to 3, usability in M. Stolze's lectures.

Related Topics and Concepts

- Agile Architecting
- Software engineering methods (OpenUP, DAD) and architecture design methods (ADD)
- Context and component modeling (next step)

[23] http://www.disciplinedagiledelivery.com/strategies-for-implementing-non-functional-requirements/
[24] http://docs.arc42.org/section-10/
[25] http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6687
[26] https://msdn.microsoft.com/en-us/library/ee658094.aspx
[27] http://blog.sophist.de/tag/nicht-funktionale-anforderungen/
[28] https://www.ifs.hsr.ch/index.php?id=13193&L=4

References

Bass, Len, Paul Clements, and Rick Kazman. 2012. *Software Architecture in Practice*. 3rd ed. Addison-Wesley.

Glinz, Martin. 2007. "On Non-Functional Requirements." In *15th IEEE International Requirements Engineering Conference, RE 2007, October 15-19th, 2007, New Delhi, India*, 21–26. IEEE Computer Society. doi:10.1109/RE.2007.45[29].

Starke, Gernot. 2015. *Effektive Software-Architekturen*. 7. Auflage. Hanser-Verlag.

Zimmermann, Olaf, Vadim Doubrovski, Jonas Grundler, and Kerard Hogg. 2005. "Service-Oriented Architecture and Business Process Choreography in an Order Management Scenario: Rationale, Concepts, Lessons Learned." ACM, 301–12.

Die SMART-Kriterien können unkompliziert auf NFR-Erhebungen angewendet werden und dienen daher als "Metaqualitäten" (Qualitätsmerkmale von / für Qualitätsattribute):

• S: Welche Funktion oder welcher Teil des Systems sollte die Anforderung erfüllen?

• M: Wie können Tester und andere Stakeholder herausfinden, ob die Anforderung erfüllt ist (oder nicht)? Ist die Anforderung quantifiziert?

• A: Stimmen alle betroffenen internen und externen Stakeholder dem "S" und dem "M" Word zu? (Ausgabe für Requirements Engineering und Projektmanagement, also hier und jetzt nicht mehr möglich)

• R: Ist es technisch und wirtschaftlich machbar, die "M" -Maßnahme im Zusammenhang mit allen unter "S" angegebenen Merkmalen oder Systemteilen zu erreichen? (Ausgabe für Requirements Engineering und Projektmanagement, also hier und jetzt nicht mehr möglich)

• T: Wann sollte der NFR das "M" -Maß erfüllen, gibt es einen Wachstumspfad von Iteration zu Iteration? (auch vom Projektmanagement zu beantworten, also hier und jetzt nicht mehr möglich)

[29] https://doi.org/10.1109/RE.2007.45