

# Functional requirements, non-functional requirements, and architecture should not be separated –A position paper

Barbara Paech,\* Allen H. Dutoit,\*\* Daniel Kerkow,\* Antje von Knethen\*

\*Fraunhofer IESE  
{paech,kerkow,vknethen}@iese.fhg.de

\*\*Technische Universität München, Institut für Informatik  
dutoit@in.tum.de

**Abstract.** Requirements engineering approaches have for a long time mainly focused on functional requirements. During the last 5 years, several approaches dealing specifically with non-functional requirements have emerged. They support the elicitation, documentation, verification and validation of non-functional requirements: sometimes only concentrating on the non-functional requirements, sometimes in conjunction with functional requirements, and sometimes in conjunction with architecture. The position we put forward in this paper is that functional requirements, non-functional requirements, and architecture must be treated together.

It is well known that functional requirements (FRs) and non-functional requirements (NFRs) constrain each other and therefore should be treated together. Similarly, it is well known that both FRs and NFRs must be realized through the architecture. However, typically, the development of an architecture is not considered part of requirements engineering. In this paper we argue that FRs, NFRs and architectural decisions (ADs) must be developed in a tightly integrated approach. In the rest of the paper, we first sketch the solutions published so far that deal with NFRs and FRs or ADs. Then, we motivate our case with an example. We argue that none of the existing approaches has truly addressed all three issues in a coherent and integrated manner. Finally, we discuss the most critical research questions that result from considering such an integrated approach.

## Existing Solutions

When surveying existing approaches to NFRs we distinguish them according to their support for the different tasks: elicitation, documentation, architecture alignment, quality assurance, change management and project management. We have not found any approaches for project or change management specific to NFRs. Similarly, there are approaches for quality assurance of specific NFRs (e.g. usability testing), but no

general technique for continuous quality assurance for NFRs. The other activities are sketched in the following.

**Elicitation** covers the following questions: how to identify NFRs, how to ensure consensus of all stakeholders (about the NFRs and their respective priorities), how to relate the NFRs, the FRs and the architecture. Major techniques for dealing with these questions are on the one hand *decomposition and operationalization* and on the other hand *negotiation techniques*. Examples for the latter are WinWin [BI96, IBR01] and *structured client reviews (SCRAM)* [SR98]. Decomposition encompasses the refinement of NFRs to more detailed NFRs, while operationalization results in strategies for achieving the NFRs, namely process strategies, such as prototyping for usability NFR, and product strategies resulting in additional FRs and architecture requirements. For the purpose of this paper we do not distinguish between decomposition and operationalization. Both involve general techniques like *goal graphs* (e.g. [Yu97]) which are elaborated in the NFR-framework [CNYM00] or *classification* e.g. as provided by the standard ISO/IEC 9126 [ISO9126]. The latter are often enhanced with *domain-specific knowledge* as in the language extended lexicon (LEL) advocated by [CLN01] or a general knowledge base on NFRs (e.g. [BI96]). *GQM* [BCR94] is a general technique for decomposing high-level NFRs into verifiable metrics.

**Documentation** involves the following issues: how to describe NFRs, which additional information is necessary to deal with them? [BKLW95] distinguishes different facets on how to describe NFRs, namely *concerns, system and environmental properties relevant for the NFR, and methods* how to deal with a NFR. Several approaches advocate *capture the stakeholders* of the NFR [CLN01, IBR01, SM98]. A popular technique for describing NFRs is the usage of *scenarios* (e.g. [KBKCW99, SM98, SR98, PBG01]). For each of the non-functional requirements there are specific notations, often mathematical, corresponding to *metrics* (e.g. using lines of code to measure the complexity of a system or using mean time to failure to measure reliability).

**Architecture alignment** answers the question of how to take NFRs into account when selecting an architecture. ATAM [KBKCW99] provide a systematic method of *evaluating scenarios* against an architecture. [GEM01] proposes to look at the implications of NFRs on the architecture in terms of *components, bus, system features (CBSE)*. *Design patterns* are another popular solution for relating NFRs and architecture [GY00].

We have not found an approach addressing NFRs, FRs and ADs in an integrated fashion. Next, we give an example of why the three entities are highly intertwined, illustrating why it is not sufficient to first concentrate on two of them and then concentrate on the third.

## Motivating Example

The following example is taken from the Ariane 501 case, the first prototype of the Ariane 5 series, which exploded 40 seconds after lift-off because of an arithmetic overflow [L96]. Based on the available literature, we put forward a likely sequence of development steps that lead to the Ariane 501 failure. Keep in mind, however, that this example has been simplified for brevity, focusing on the dependencies among FRs, NFRs, and ADs. The sequence of events leading to the system failure and the actual causes for this incident are much more complex than our narrative suggests.

The focus of our example is the navigation system of the rocket. The primary function of the rocket is to put in a payload (e.g., a satellite) on a specified orbit and latitude. To achieve this, the rocket must follow a precise trajectory during flight. Hence, the first requirement identified for the navigation system is the following:

- **FR1 (requirement):** The navigation system shall calculate course corrections based on differences between the actual trajectory of the rocket and the planned trajectory for the specific payload.

Before the launch, the navigation system must also compute the starting position of the rocket to take into account the rotation of the earth and wind. Hence, when considering the horizontal velocity of the rocket, the navigation system must consider two cases, the velocity before launch, which is extremely small, and the velocity after launch, which is several orders of magnitude larger. In both cases, it is possible to compute the maximum value for the velocity based on physics and the properties of the rocket:

- **NFR1 (domain constraint):** The maximum horizontal velocity before launch is at most  $v_{\text{before}_{\text{max}}}$
- **NFR2 (domain constraint):** The maximum horizontal velocity after launch is at most  $v_{\text{after}_{\text{max}}}$

This distinction leads to the ADs to distribute this functionality to two subsystems: The alignment subsystem computes the starting position of the rocket. The inertial reference subsystem computes course corrections. Right before launch, the alignment subsystem hands over the starting position to the inertial reference subsystem.

- **AD1 (subsystem decision):** The alignment subsystem computes the initial position of the rocket.
- **AD2 (subsystem decision):** The inertial reference subsystem computes course corrections after launch.

Only NFR1 is relevant to the alignment subsystem, hence, the following AD:

- **AD3 (type decision):** Horizontal velocity in the alignment subsystem is represented as a 16-bit integer.

However, the alignment subsystem calculations are complex and take about 45 minutes to initialize. This triggers the FR that despite of this 45-minute penalty it is important to resume the count down only a few minutes after it is stopped, as the orbits required by specific payloads can have narrow launch windows.

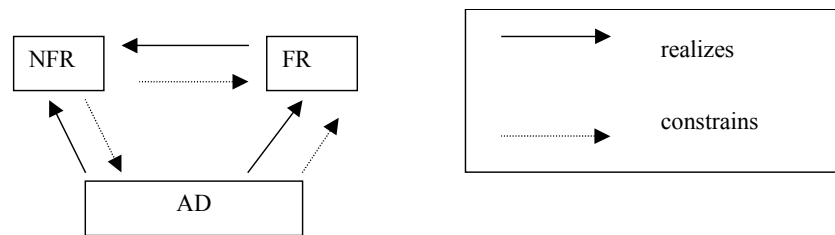
- **FR2 (requirement):** The system shall be able to resume count down within 2 minutes of stoppage.

To realize this FR, the developer decide to extend the functionality of the alignment subsystem to continue these calculations until about 50 seconds after the coordinate handover. When the count down is stopped after the handover but before lift off, controllers have then enough time to reset the alignment subsystem without paying the 45-minute penalty. This results in a new FR and AD:

- **FR3 (requirement):** The system shall proceed with the alignment calculations for 50 seconds after handover.
- **AD4 (subsystem decision):** The alignment subsystem takes over the additional functionality.

However, the addition of FR3 and AD4 puts the decision AD3 at risk, because now the alignment subsystem also has to deal with horizontal velocity values exceeding  $v_{before\_ma}$  and up to  $v_{after\_max}$ . Thus, NFR2 suddenly constrains the decision AD3. In case of Ariane 5, developers did not note this before flight 501.

Altogether, this example shows that ADs realize FRs and NFRs (as for AD1, AD3 and FR1 and NFR1). But in addition, ADs constrain FRs (as for AD2 and FR2) resulting in new FRs and ADs (in this case FR3 and AD4). Moreover, NFRs constrain ADs (as for NFR2 and AD3). Figure 1 illustrates these relationships in general. An example for the realization of an NFR through an FR is an authentication FR realizing a security NFR.



**Figure 1: Relationships between FRs, NFRs and ADs**

With this example, we illustrated how FRs, NFRs and ADs are treated in an iterative, nonlinear, and non-incremental fashion. Hence, we argue for integrated methods that support developers in dealing with this reality.

## Research questions for an integrated approach

A goal of the recently initiated research project EMPRESS (<http://www.empress-itea.org>) is to develop an integrated approach for dealing with NFRs, FRs, and ADs. The following research questions drive this project:

- *What are descriptions for FRs, NFRs and ADs that allow identify, verify and validate as well as maintain dependencies easily?* In particular, measurable definitions of NFRs and suitable architecture views are still open problems.
- *How to make explicit different views of different stakeholders?* One particular problem for the integration is achieving the common understanding of requirements and architecture. Usually, the analysts eliciting requirements and the architects designing the architecture are different persons, as they require broadly different skills. The close collaboration of these different types of specialists require mechanisms for sharing knowledge, for example, in the form of different views on the NFRs, FRs, and ADs.
- *How many different abstraction levels are required when refining and aligning FRs, NFRs and ADs?* How early can dependencies be identified? Goal graphs support decomposition, but do not give guidance on how many decomposition steps should be made. In particular, it is not clear how to relate the FRs, NFRs and high-level ADs to more detailed ADs.
- *How to describe the solution space?* Often ADs are made early and unnecessarily restrict further FRs. Therefore, the different options for realizing high-level FRs and NFRs should be described not in too much detail. Goal graphs allow show dependencies between single FRs, NFRs and ADs, but do not allow compare comprehensive options packaging ADs, FRs and NFRs.

In addition to these questions on the integration, questions arise from the embedding of an integrated method in the overall software-development cycle, for example:

- *What additional information has to be captured to support change of NFRs?* For example, traceability (e.g. [GF94]) is usually thought as a simple bi-directional relationship between elements (e.g., indicating which requirements impact which design element). As illustrated in the example, with NFRs, FRs, and ADs, the relationships can be much more complex.
- *How to ensure completeness and consistency of the considered option?* This can be supported e.g. by a knowledge base that should also include rationale to support trade-off decisions (e.g. [DP02]).
- *How to inspect or test NFR?* Different communities have investigated specific quality assurance techniques for specific NFRs (e.g. security, safety). Integrated quality assurance requires integration of these techniques to mirror the (de)composition of NFRs into more refined NFRs and additional FRs as well as ADs. As illustrated in the example, it is not sufficient to only consider NFRs when

designing test plans, as many implicit dependencies can have been introduced among NFRs by specific ADs.

### Acknowledgements

We thank our colleagues at Fh IESE in the EMPRESS project for fruitful discussions regarding the literature survey and the reviewers for helpful comments.

### References

- [BCR94] V.R. Basili, G. Caldiera, & H.D. Rombach, "Goal Question Metric Paradigm", In J.J. Marciniak (ed.), *Encyclopedia of Software Engineering*, vol.1, pp.528–532, John Wiley & Sons, 1994.
- [BI96] B.Boehm, H. In: *Identifying Quality Requirement Conflicts*, IEEE Software, March, pp.25-35, 1996.
- [BKLW95] M. R. Barbacci, M. H. Klein, T. Longstaff and C. Weinstock, "Quality Attributes", Technical Report CMU/SEI-95-TR-021, Software Engineering Institute, Carnegie Mellon University, December 1995.
- [CLN01] L.M. Cysneiros, J.C. Leite, J.S. Neto: A Framework for Integrating non-functional requirements into conceptual models, *Requirements Engineering Journal*, no. 6, pp. 97-115, 2001.
- [CNYM00] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, "Non-Functional Requirements in Software Engineering", Kluwer Academic Publishers, 2000.
- [DP02] A.H. Dutoit, B. Paech. "Rationale-based Use Case Specification", *Requirements Engineering Journal*, Special Issue on REFSQ'2001, 2002.
- [GEM01] P. Grünbacher, A. Egyed, N. Medvidovic: Reconciling Software requirements and architectures: The CBSP approach, RE'01, pp. 202-211, 2001.
- [GF94] O. Gotel and A. Finkelstein, "An Analysis of the Requirements Traceability Problem," ICRE'94., pp. 94–101, 1994.
- [GY00] D. Gross, E. Yu, "From Non-functional requirements to design through patterns", REFSQ'00, pp.86-97, 2000
- [IBR01] H. In, B. W. Boehm, T. Rodgers, M. Deutsch, "Applying WinWin to Quality Requirements: A Case Study", ICSE 2001, pp. 555-564, 2001
- [ISO9126] ISO 9126, "Information technology - Software product evaluation - Quality characteristics and guidelines for their use", 1991
- [KBKCW99] R. Kazman, M. Barbacci, M. Klein, S.J. Carriere, S.G. Woods, "Experience with Performing Architecture Tradeoff Analysis"; ICSE 99, pp.54-63, 1999
- [L96] J.-L. Lions, ARIANE 5 Flight 501 Failure: Report by the Inquiry Board, <http://ravel.esrin.esa.it/docs/esa-x-1819eng.pdf>, 1996.
- [PBG01] K. Pohl, M. Brandenburg, A. Gülich, „Integrating requirements and architecture information: a scenario and meta-model based approach. REFSQ'01, pp. 68-84, 2001
- [SM98] A. Sutcliffe and S. Minocha, "Scenario-based Analysis of Non-Functional Requirements", REFSQ'98, 1998.
- [SR98] A. Sutcliffe & M. Ryan, "Experience with SCRAM, a Scenario Requirements Analysis Method," In ICRE'98, April 1998.
- [Yu97] E. Yu, "Towards Modeling and Reasoning Support for Early-Phase requirements Engineering", RE'97, pp.226-235, 1997