On 6 April 2011 the first deep dive related to non-functional requirements was organized by the design and architecture expert groups of the Microsoft Based Solutions cluster U31. Because non-functional requirements are an aspect of software which concerns both analysts and architects there was also a mixed bunch of analysts and architects present.

The session began with a presentation by Andre Evertse on what non-functional requirements are. The statement below is what RUP has to say on the differences.

Functional requirements specify actions that a system must be able to perform, without taking physical constraints into consideration. These are often best described in a Use-Case Model and in use cases. Functional requirements thus specify the input and output behavior of a system. Requirements that are not functional, such as the ones listed below, are sometimes called **non-functional requirements**. Many requirements are non-functional, and describe only attributes of the system or attributes of the system environment.

Tabel 1:
http://deliver2.capgemini.com/components/Classic_RUP/index.htm#core.base_rup/guidances/concepts/requirements_62E28784.html

The system analyst is responsible for both, functional and non-functional requirements. However there are more roles deeply involved in the process of documenting and gathering these requirements, such as the requirements specifier and the software architect.

Classification

The term non functional requirements is a broad term, therefore there are classifications developed to order the requirements. In the table below the most known classifications are shown

FURPS+ (RUP)	ISO/IEC 9126	Volere
Functionality Usability Reliability Performance Supportability Design Requirement Implementation Requirement Interface Requirement Physical Requirement	Functionality Usability Reliability Efficiency Maintainability Portability	Functional and Data Requirements Look and Feel Requirements Usability and Humanity Requirements Performance Requirements Maintainability and Support Requirements Operational and Environmental Requirements Security Requirements Cultural and Political Requirements Legal Requirements

Within our internal standard IRMA we use the following classification which is a combination of these classifications.

- Usage
  - User groups
  - Usability
  - Reliability
  - Usage Efficiency
  - Documentation usage
- Maintenance
  - Scalability
  - Software Maintenance
  - System Maintenance
  - Install ability
  - Testability
  - Adaptability
  - Maintenance Documentation
  - Maintenance Compliancy
  - Common Maintenance Requirements
- Design Constraints
  - Mechanisms
  - Structure
  - Common design constraints requirements

A guideline for filling in these requirements can be found in the supplementary specification guideline, that document also contains a table which shows where which part of the IRMA classification comes from.

Now you know that non functional requirements are attributes of a system that don't affect the functional way it works. Next to that you know what classifications there are and which are used within Capgemini project standards. Now it is time to get the requirements clear.

Documenting non functional requirements

When documenting non functional requirements it is essential to be smart (**S**pecific, **M**easurable, **A**ttainable, **R**ealisable, **T**ime), the requirement "the system must react fast" is not smart, it does not say what is fast, what is a reaction, when should this requirement apply.

Therefore Robert van Lieshout developed a way of documenting non-functional requirements in a way they can be tested. His way states that the following properties of a non functional requirement should be clear.

Requirement: What is the requirement? i.e. the systems is recovered form a crash within 10 minutes
Scale: time in minutes measured in as an average over 3 crashes within 1 month
Measurement conditions: in a production environment with a full database
Norm: target 10 minutes, challenge: 6 minutes

As you can image the requirement above isn't yet completely smart (what is full? What kind of crash?) but is gives some more details then a normal requirement. When you think of this you probably see that getting requirements smart is very difficult, but it is essential to get them as smart as possible. When you don't get the requirement smart you end up in discussions with the customer on that requirement when it is too late (or expensive) to adjust the system.

And lastly IRMA has some tips & tricks

- Know and use IRMA tips
- Define them in cooperation with software architect and stakeholders
- Document them when you see them
- Try to be complete
- Don't try to specify them all by yourself. When necessary, contact a specialist. For example for the Usability aspects.
- Make them SMART
- Requirements must be realizable against reasonable costs

After the theory we started with a case of the NVIS project to experience how difficult it is to get non functional requirements. And I can say from my experience that is was very difficult indeed.

Conclusion

In conclusion I can say that it was a very interesting and informative evening that is highly recommended for analysts and architects. So if you're interested you can apply by putting your name (by editing the page)on this WIKI page.

Author: Remco Boksebeld