Requirement Engineering

Carsten Schmoll & Horst Rechner

Fraunhofer FOKUS - FOKUSforum - Best Practices

May 6., 2008

Requirement

A requirement is " a singular documented need of what a particular product or service should be or do. It is most commonly used in a formal sense in systems engineering or software engineering. It is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a user." -- [Wikipedia]

Sets of requirements are used as inputs into the design, implementation and test stages of product development.

What is Requirement Engineering?

Can be seen as transformation of the **requirements of the client** for his product (e.g. software) to **requirements for the system development**.

see [Man04] p.54

Requirements engineering includes the **analysis and management** of the requirements using **engineering techniques**.

Motivation

Why not just talk and start a cool product?

"When we had lost our goal completely from sight,
we doubled our efforts." -- Mark Twain.

- without an effective RE the developers will implement an imcomplete or wrong solution for the customer
- even with the most successful development techniques the customer will not want what you made
- extra work, hassles, complaints and other effects (loss of money, future projects) will probably be the result
- written facts are the key ("Worte sind Schall und Rauch")

Why Requirement Engineering?

Projects are defined by the following criteria:

- clearly defined goal / result
- limited in
  - money
  - human ressources
  - content
- individual
- high complexity
- system boundaries

Dedicated requirement engineering helps to clearly define those criteria in their variant for the project.

see also [Man04] p.22

Position of You in this Presentation

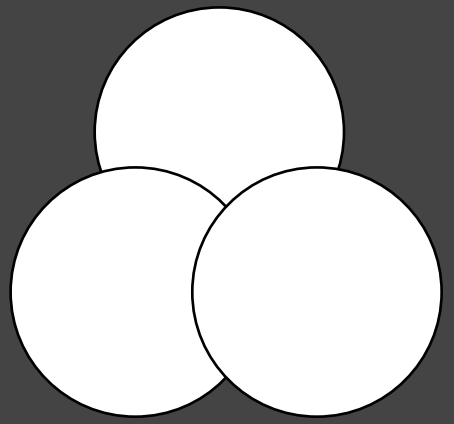
Let's assume the following position:

- you shall make a FOKUS customer happy
- you need to find out what shall be made
- you need to tell the developers what to do
- you need to make sure the customer says: "cool, that's what I wanted!" in the end

- we will focus on *Software* as the product here
- presented hints can also work vice versa
- basic rules also applicable to generic (non-IT) RE

The "Three" Sides of the Medal

- Engineering - what to do, how to do it
- Technical - what tools can support us
- Human - communication & politics
  - not all can be learned from books
  - only all three together make up very successful RE



Phases

Project Phases

(Requirement engineering helps us here:)

- **Analysis**
- **Requirement Specification**
- **Software Design**
- **Implementation**
- **Test**
- **Maintenance**

see [Man04] p.47

Which documents should be created in every IT development project?

(Requirement engineering helps us here:)

- **Calculation of costs**
- **Offer to the client**
- Assignment by the client
- **Requirement documentation**
- **Design documentation**
- **Testspecification**
- **Instruction manual**
- Bill of delivery
- **Acceptance protocol**

see [Man04] p.37

Phases of Requirements Engineering

These phases are part of the RE process:
(not necessarily one by one - they do overlap)

- Elicitation - gathering the requirements from stakeholders
- Analysis - checking for consistency and completeness
- Documentation / Specification
- Verification - making sure the specified requirements are correct
- Management - of the requirements.
  This includes: Technical documentation, Change Management
- Make it stick! - present it to customer
- Sync sync synchronize - as "close" as possible!

Kinds of Requirements

Functional

- defines a the targeted behavior of the system
- based on black-box view of the system
- may also have pre- and post-conditions

Non-functional

- = required constraints that "limit" the system design
- categories:
- technical - e.g. hwrdaware, OS, devices (screen)
- interface - e.g. color, text size
- quality - security or performance (e.g. reaction time)
- other delivery "parts" - e.g. training, support
- legal requirements - e.g. insurance, standards

Requirements Elicitation Techniques

- Creativity techniques
  - Brainstorming
  - Changing the perspective
- Observation techniques
  - On-Site-Contractor
  - Apprenticing
- Questioning techniques
  - **Questionnaire**
  - Interviews
  - On-Site-Customer
- Retrospective techniques
  - System archaeology
  - Reuse

Requirements Elicitation 1

- Make and circulate a *questions document*
  - only the right questions can yield useful answers
  - don't expect everyone to read it though before you read it out aloud (see below)

- *Get together* with the customer
  - if possible in any way *do it in real life*
  - get the right *stakeholders* "on board" (note them down)
  - be prepared about the project (learn about environment)
  - bring the right questions with you (above doc)
  - make *current situation analysis* there if not a new system
  - define clearly the roles of involved "partners" + show them
  - also check whom you *don't* need to ask/satisfy

Requirements Elicitation 2

- Organise a workshop
  - organiser brings the coffee ("feel good environment")
  - keep the number of stakeholders limited
  - be prepared for some level of conflicts among them too
  - limit the number of your contacts afterwards even more

Requirements Elicitation 3

- the "deepness" (= detail level) depends on many things
  - level of the project (prototype, demo, 24/7 application)
  - size of the project
  - "closeness" of the customer
  - style of development (waterfall, V-Model, UP, XP)
    - less agile - more complete spec needed

- *prioritize* during workshop into: **MUST / MAY / OPTIONAL**
- tell participants which documents will be the output
- beware of implicit requirements
  ("such buttons should of course have been big, round and red")

Sophist Template

For interfaces:

- *“Component A shall / should provide component B the ability to…”*

For functionality:

- *“Component A shall / should / will be able to…”*

shall - mandatory
should - desirable functionality
will - desirable functionality implemented in version Y

see [Soph08]

Analysis

- Part of Workshop wrapup
  - back at home
  - summarize and cleanup answers and minutes taken
  - note unclear points, missing items and risks in separate list or even separate document
  - put the many tiny details into annexes
  - circulate results openly, sync soon e.g. via telco
  - clearly state system boundaries and non-goals
    - boundaries will yield the system interfaces
  - if needed make a glossar (sparsingly, not: complete) to fix the common terminology

Reviews of Requirement Documents

- By the **customer** to avoid situations like
  - "You never asked us!"
  - "Why didn't you inform us in time?"
  - "We thought of this in a different way."
- To cover all feature aspects
- By the **technical staff** to avoid situations like
  - "That is how I thought it would be right."
  - "We never talked about this!"
  - "This is not possible with our system design."
- To ensure that requirements and design are aligned
- Goal: Distribution of the resposibility

see also [Man04] p.40

Documentation of Requirements

- Starting with an *informal* description
  doc/xls/odf templates available
  most companies have their own ones

- Later convert to *formal* description
  - e.g. UML use cases, UML activity diagrams
  - database systems help a lot to keep track
    - for each requirement state the same attributes:
      - identifier, short text, long text, author, initial date, latest change date, relation to other requirements, priority, riscs, release, status, questions, source
  - allow the traceability of project progress - project mgmt.

Maintenance of Requirements

- Important during the complete RE lifetime
  - reflect changes in your (office) documents
  - make stakeholders aware of the latest version
  - use simple to follow versioning scheme
  - make documents easily available (svn)
- keeping track even more important during design and implementation phase (trac)
- many more tiny requirements visible here
- practice shows that 3% of all requirements change *each month*
- keep list of open questions and to-be-clarified items

Tools

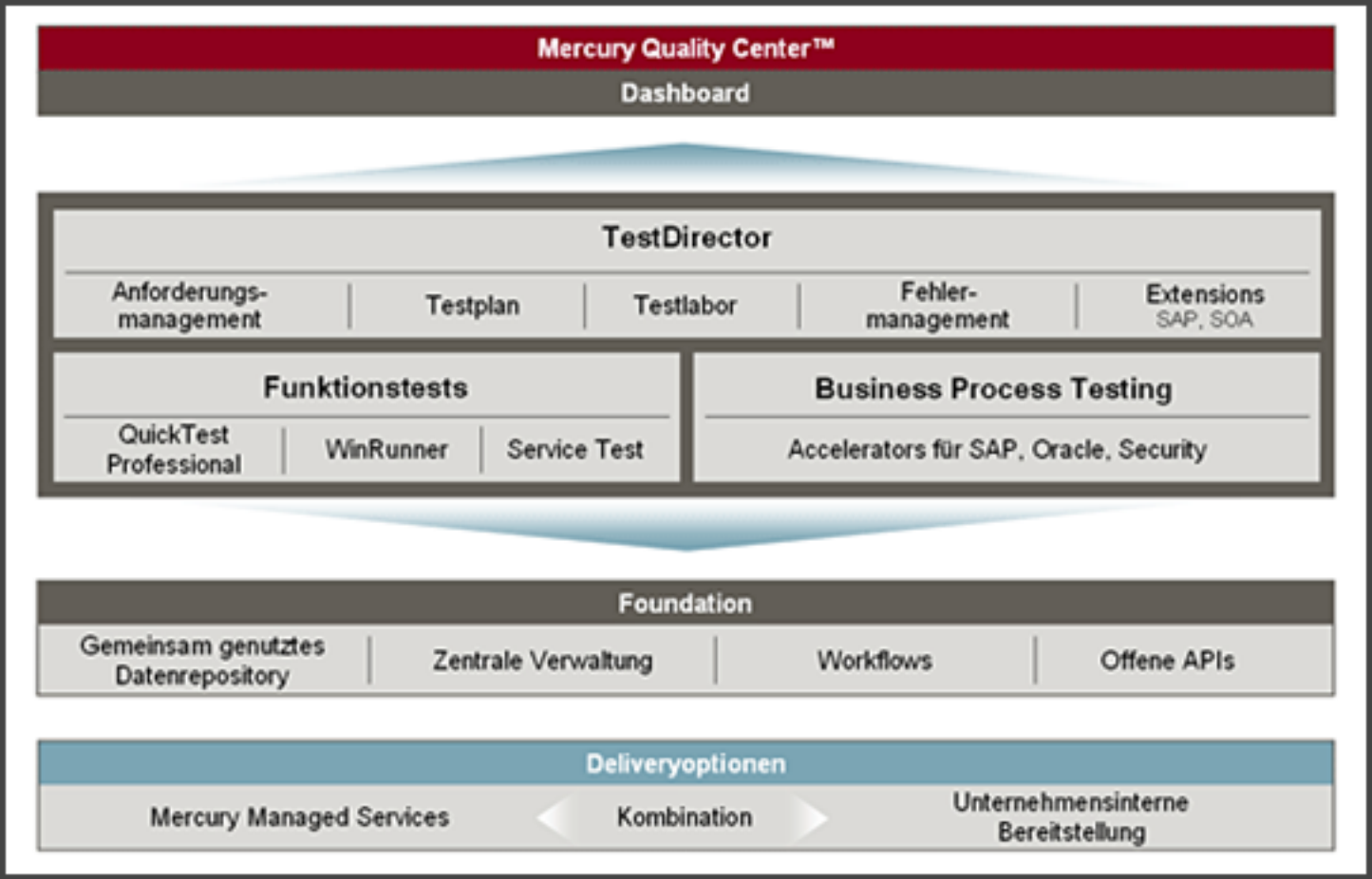
Tools in Use (mostly open source)

- depends on the type, style and size of the project
  - whatever makes both parties happy will do
  - usually will be some office solution for the informal part

- don't use systems like trac to communicate with your customer
- use trac to track project progress for the 500+ tiny requirements which your developers crunch down one by one (see lecture "Best Practices For Java Projects")

- you will want to have "at hand":
  - text processor, spread sheet, graphics app or flow charter, database, report generator, UML tools, groupware, a file repository, a ticket system (trac), Wiki

Test Director



Test Director - Requirement Documentation

Test Director - Test Planning (Verification)

Test Director - Testing (Verification)

Test Director - Defect Management (Verification)

Tips & Summary

Tips & Summary

- Document and Synchronize often among stakeholders!
- Don't try full RE the first time in a time critical project!
- Expect significant time overhead in your first try.
  - But it will pay out in the long run.
- Don't expect immediate embrace by all stakeholders!

- No single (open source) tool which integrates all aspects of requirement engineering (Elicitation, Analysis, Documentation / Specification, Verification and Management) know to us.
  - Commercial Toolchain (Integrator Fraunhofer IESE)
  - The best tools are those which all involved people can work fluently with! (even if it is only MS Excel)

Further reading (only German)

- [Man04]
  Pascal Mangold,
  IT-Projektmanagement kompakt,
  Spektrum Akademischer Verlag, 2. Auflage, 2004
- [Rupp08]
  Chris Rupp (SOPHIST GROUP),
  Systemanalyse kompakt,
  Spektrum Akademischer Verlag, 2. Auflage, 2008
- [Rupp07]
  Chris Rupp (SOPHIST GROUP),
  Requirements-Engineering und Management
  Hanser Verlag, 4. Auflage, 2007
- [Soph08]
  SOPHIST GROUP
  http://www.sophist.de
- [FUBSWT08]
  Vorlesung Softwaretechnik FU Berlin
  https://www.inf.fu-berlin.de/w/SE/VorlesungSoftwaretechnik

Thank you for your attention!

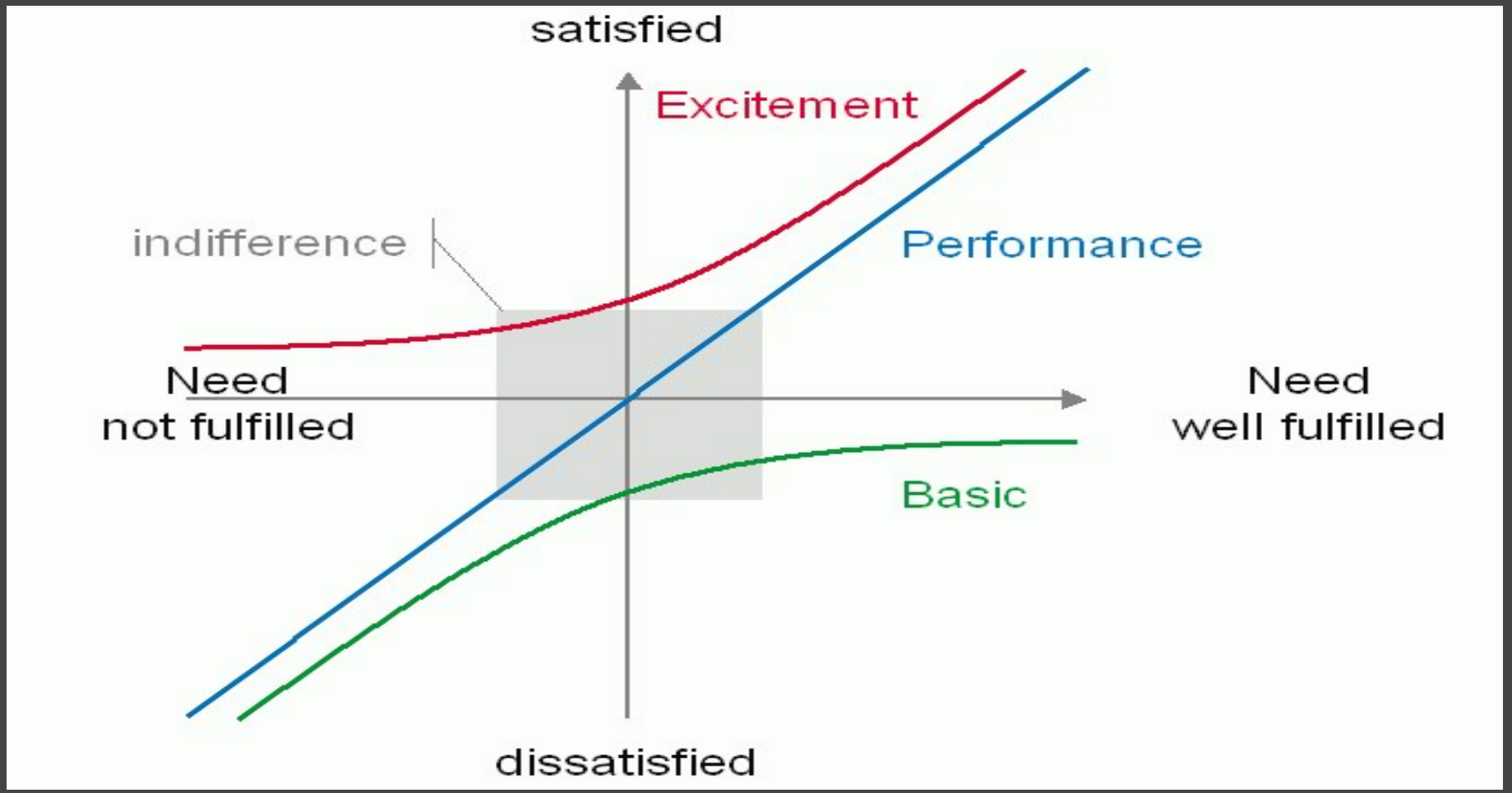
Additional Infos

Abstract

Requirement Engineering deals with the **gathering, analysis and documentation of requirements, their environmental conditions and the clients wishes** for their product. There are many areas where requirement engineering can be applied:
New projects, extension of existing projects / products, the setup of a new laboratory, the development / extension of software, even the preparation of a difficult dinner recipe for gourmets. A **good project roadmap and its execution with effective requirement acquisition** can avoid misunderstandings and excess work in later stages of the project and can lead to a **measurable, consistent result**. Associated topics are **communication and documentation of planned and finished work items -** which allows **continuous progress control**.
In this talk we will give you practical advise on effective requirement acquisition, take a close look at the **necessary tools** and show you **common pit holes**.

Kano-Model

- cool features are much more fun than "open+save" support!



Requirement Elicitation Exercise

Turm 1:

- Höhe an der Antennenspitze: 368m
- Kugel für Gäste mit Dreh-Café Aussichtsetage
- Durchmesser des Fußes: 32 m
- Durchmesser der Kugel: 40 m
- Material: Stahlbeton
- mehrere Aufzüge
- Geschwindigkeit der Aufzüge: 6 m/s
- Antennenanlage, Leistung bis 100kW
- Stromversorgung für Kugel
- Wasserversorgung für Kugel
- WCs in der Kugel
- Blitzschutzanlage
- Treppe im Inneren des runden Turmes

Turm 2:

- dreieckige Form (v.d.Seite)
- Grundriss quadratisch
- Stahlkonstruktion, genietet
- Aussichtsetage mit Café
- Höhe = 300m
- mehrere Aufzüge, mit Zwischenhalt am Café
- Geschwindigkeit der Aufzüge: 6 m/s
- Antennenanlage, Leistung bis 500kW
- Stromversorgung für Café
- Wasserversorgung für Café
- Blitzschutzanlage
- Treppe für Notfälle (Ausfall der Lifte)

- Play the client and have your colleague play the contractor who asks you questions.
- Afterwards let her/him paint what he/she thinks the tower looks like!
- Now compare with next slide - did your towers look similar?

How did your towers look like?