IT Spektrum
vormals OBJEKTspektrum
Digitaler Wandel & Software-Architektur für Profis

The Requirements Engineering Workshop

supported by:
IREB International Requirements Engineering Board
Universität Zürich[UZH]
Institut für Informatik

More specialist posters for IT professionals can be found here:
http://www.sigs-datacom.de/wissen/fachposter.html

Prof. Dr. Martin Glinz, University of Zurich, Chair of the IREB Counci
Dr. Kim Lauenroth, adesso AG, First Chair of the IREB e.V.

REQUIREMENTS ENGINEERING (RE)

What is Requirements Engineering (RE)? Requirements engineering is concerned with **systems requirements** and has three objectives:

1. Knowing the **relevant requirements**, **achieving a consensus** among the stakeholders about these requirements, documenting them according to given standards, and managing them **systematically**,
2. **Understanding** and **documenting** the stakeholders' **desires** and **needs**,
3. Specifying and managing requirements to minimize **the risk** of delivering a system that does not meet the stakeholders' desires and needs.

Why do we need RE?

Why don't we build systems in the same way?
Because the risk of direct development is generally too high. We need the requirements level because most systems are too big and comprehensive to develop and design intellectually at a purely technical level.

Can't we just develop in an agile way?
Even with agile development, we have to start by being clear about what our stakeholders want. For each formulated requirement, we work with them to design the future system.

Isn't that too expensive and time-consuming?
The costs of RE are balanced by higher returns: Based on good requirements, we can develop systems that are tailor-made to match the wishes and requirements of stakeholders. In addition, we reduce the costs of adjustments, troubleshooting and rewriting unusable code.

Which systems?

We usually deal with requirements for software systems or software-intensive systems. But RE can be applied and used for any type of system. In the context of RE, we also classify components and services as systems.

Documents

...are a blessing
- Documentation is a thinking and communication tool: accurate reporting assists communication, with identifying gaps and errors and uncovering design possibilities and opportunities for innovation.
- Documentation is a knowledge repository that is useful for things such as testing and further development.

...and also a curse
- Documentation is sometimes an obligation imposed on us by norms or standards.
- Documentation uses up resources that are not directly useful to the development objective.

Therefore
- Only as much as is needed.
- Match the documentation format to the process and requirements!
- It doesn't necessarily have to be a comprehensive requirements document. Sometimes a stack of cards on which requirements are noted is enough.

RE processes

Processes are important but not an end in themselves
Successful system development requires methodology and discipline. Which is why processes are important. However, they are not an end in themselves, but are only useful when they support us in developing a system that corresponds to the wishes and requirements of our stakeholders.

Criteria for designing the RE process

- limited stakeholder availability
- critical system
- well-defined procedures
- well-defined sectors
- commitment to binding tenders

Pre-RE
- V-model-type or waterfall-type processes
- Requirement specifications

Iterative RE
- Design Thinking
- User-Centred Design
- Agile Processes

- high level of stakeholder availability
- innovative system
- unfamiliar environment
- rapidly changing environment
- Agile software processes

Practical tip: Mixed approaches are a successful compromise; defined/critical system components can be specified in advance, innovative/unknown system components can be specified iteratively.

Requirements

Requirements are the basic units that we use to express stakeholders' needs and the required system capabilities.

What types of requirements are there?
- **Functional requirements** define a function that must be provided by a system.
  We make a distinction between structural and data perspectives, functional perspectives and behavioural perspectives.
- **Quality requirements** define the qualitative characteristics of a system.
- **Constraints** are additional conditions for implementing a system
  (e.g. existing interfaces, standards or legislation)

RE and design

Two types of design
- **Technical design** refers to the technical implementation of the software solution at both large-scale (software architecture) and small-scale (detailed design).
- **Product design** refers to the design of a product or system in terms of its features, performance, outer form and use.

Product design is an RE task
- **Classical RE** considers product design as a design task that comes after RE.
- In **modern RE** there has been a shift in thinking: product design configures the core properties of a system and is based on the wishes and needs of the system stakeholders.

Requirements Management (RM)

Elicited and documented requirements need to be managed
- Where and how to store and find them?
- How to change or adjust them?
- How to track them: where does a requirement come from? Where is it implemented/tested? Does it depend on other requirements?
- How to prioritise: What is important and how important is it? What will be implemented and when?

Good to know
- Many people regard RM as a synonym for RE. We regard RM as an (important) part of RE.
- RM takes place across several projects: as long as a product exists, its requirements need to be managed.

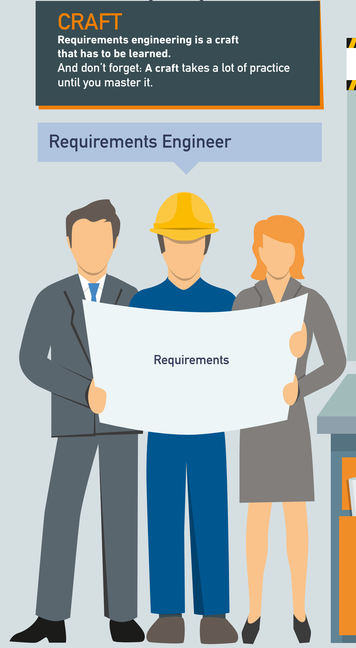
Tools

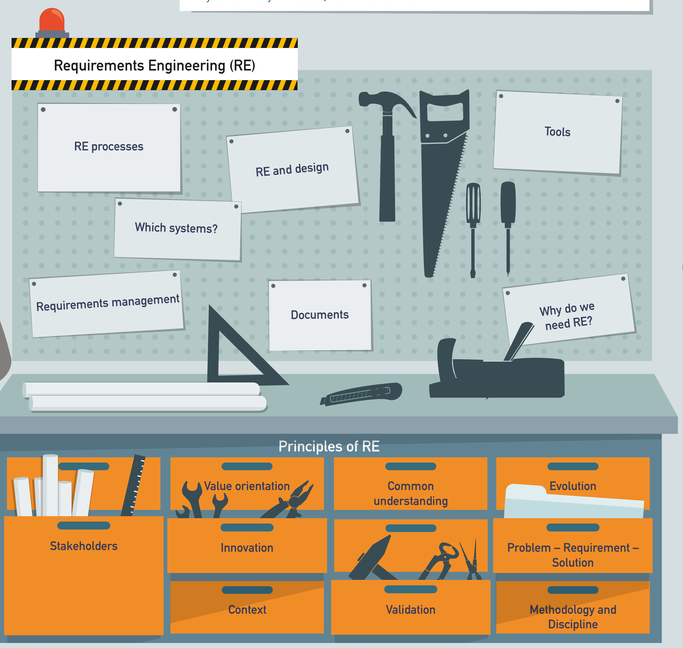
... are helpful
- RE tools relieve the burden of routine tasks, support the implementation of practices and facilitate cooperation

... are useless if
- you don't know how handle them,
- you don't master the appropriate/suitable RE practices.

... have to be systematically selected
- Determine which activities/practices are to be supported,
- Search for the most appropriate RE tool,
- Systematically introduce, train and use the tool.

CRAFT
Requirements engineering is a craft that has to be learned.
And don't forget: **A craft** takes a lot of practice until you master it.





Requirements Engineer

Is this actually a profession?
In practice, very few people have the job title of Requirements Engineer.

A Requirements Engineer is someone who
- designs systems based on requirements,
- has in-depth knowledge of RE,
- is able to define RE processes,
- selects and deploys appropriate RE practices

Requirements engineers work on RE tasks
for example, as an Application Specialist, Business Analyst, Product Owner, Project Leader, Systems Engineer, etc.

Learning RE

RE should and can be learned
- CPRE certificate (www.ireb.org) ☺
- Read RE books
- Compare notes with more experienced colleagues

Acquire skills in different disciplines
For example: software architecture, testing, project management, business analysis, usability engineering.

"Soft skills" are useful
Listening, moderation and negotiation techniques, motivational and organisational psychology

RE PRACTICES

RE provides practices for methodically and systematically eliciting, documenting, validating, negotiating, and managing requirements.
- Select appropriate practices for each specific RE task from the set of possible practices.
- Do not deploy practices side by side or one after the other, but coordinated with each other.
- Combining different practices can be useful for different aspects of the problem.

Elicitation

From the right sources
- Stakeholders – the most important source for requirements
- Also: documents, legislation, existing or related systems, processes, bug reports, feedback...

With selected suitable techniques
E.g. interviews, surveys, workshops, field observation, artifact analysis, storyboarding, prototypes...

Bear in mind
- Basic knowledge of the application domain of the system is required.
- *Active elicitation*, rather than passive observation and recording.
- Requirements can *emerge* during the elicitation process, they are not there a priori.

Documentation

The process determines the form and scope
- A plan driven, waterfall-like procedure: typically a comprehensive requirements specification in natural language with models.
- An agile procedure: typically a structured collection of individual requirements (for example, user stories), as well as a vision document

What to document?
- Functional aspect: required functions, required/supplied data, expected behaviour
- Quality aspect, such as reliability, user friendliness, security, response times, data rates...
- Constraints aspect, e.g. technical, organisational, legal, cultural...

Validation

Ensuring the quality of the requirements
- **Content accuracy:** Suitable? Consistent? No gaps? Reproducible? Testable? Actionable?
- **Appropriateness of the form:** complies with prescribed standards and rules? Understandable? Well-structured? As unambiguous as possible?
- **Consensus:** Requirements specified in cooperation with the stakeholders? Conflicts identified and resolved?

Avoid misunderstandings
- Implicit common understanding is frequently necessary.
- Random validation to check whether this understanding actually exists.

Negotiation

Agreement is not a law of nature
- Requirements have to be *consolidated* and agreed by all stakeholders.
- For this, conflicts must be detected and re-solved in a suitable manner.
- Coordination and testing of requirements are closely interconnected.

Conflict management is necessary
- **Identify:** where do the differences lie?
- **Analyse:** what type of conflict is it? Different opinions? Different functional requirements? Value system or power struggles?
- **Resolution:** e.g. consensus (win-win), compromise, negotiate, exercise authority, implement variants.
- **Documentation:** Document conflict resolution

Management

Accountancy in RE
- **Organise:** Uniquely identify each requirement; record meta-data (author, date, source, status...)
- **Store and find:** Store requirements in such a way that they can be systematically searched for and found
- **Traceability:** Helps with answering important questions: Where do requirements come from? How interdependent are requirements? Where will each requirement be implemented/tested?
- **Changes:** establish and implement a process for changing existing requirements
- **Prioritising:** which requirements are important, and how important are they?

PRINCIPLES

RE is based on a series of underlying principles, which are applied independently of the processes, practices, documents and tools used.

Stakeholders

The central source of requirements.
People or organisations that have a direct influence on the requirements of a system are called **stakeholders**.

They can have several **roles**:
user, customer, operator, marketer, legislator, regulator, developer...

- Referring only to customers or users **does not adequately reflect this**.
- In RE, involving the right people in the right roles is crucial.

Value orientation

RE is not an end in itself, but the means to an end.
The **value** of a requirement is measured in terms of its use, minus the costs of its identification, documentation, testing and administration.

The **usefulness** of a requirement is the degree to which it helps

- to build systems that meet the wishes and needs of its stakeholders,
- to minimise the risk of failure.

Note: The impact and value of RE are indirect, only the RE has a cost!

Common understanding

The fundamental prerequisite for any successful development.
- Created, strengthened and secured by *requirements.*

Explicit common understanding
- Through carefully determined, documented and negotiated requirements
- The goal for a plan-driven, waterfall-like approach

Implicit common understanding
- Common basic knowledge; knowledge of visions, requirements, ideas,...
- Necessary for agile procedures, when requirements are not specified in detail
- Greater risk of misunderstanding

Evolution

Changing requirements are not an accident but the norm.
Systems and their requirements are prone to **evolve**. Therefore we must
- keep requirements **stable** (otherwise orderly development is not possible),
- ... and at the same time allow requirements to be **changed**.

Possible solutions
- Standardise requirement documents, establish a change process for requirements
- Agile development: Changes will be incorporated in future sprints.

Innovation

More of the same is not enough.
- "Give the customer exactly what he wants" – with all the mistakes and shortcomings?
- "We know exactly what the customer needs" – which is why we know-it-alls are so popular.
- "Our new system does the same old crap as before, but much faster" – may progress be praised.

➜ We don't just want to satisfy people, but **make them happy** and **excite them** ➜ That's why we need **innovative requirements**.

Innovation can be specifically promoted.
➜ **RE creates innovative systems**
- small innovations (killer features)
- as well as big innovations (disruptive systems).

Problem – Requirement – Solution

Unavoidably interlinked.
Problem: If the actual situation is unsatisfactory – we have a problem.

Requirements: What do our stakeholders need to eliminate/simplify the problem?

Solution: A (technical) system, which meets the requirements.

Principle 1: Problems, requirements and solutions are closely interconnected.

Principle 2: Separate problems, requirements and solutions as far as possible in thinking, communicating and documentation.

Context

Systems cannot be considered in isolation.
Systems are embedded in a **context**.
➜ A system can only be specified if its context is understood.

The **boundary** between the system and its **context** is often blurred and the task of RE is to draw a precise context boundary.

Fulfilling all the requirements for a system is not enough.
- the assumptions about the system context must actually be true.
- context phenomena must be adequately mapped based on requirements in terms of data and states in the system to be specified.

Validation

Each requirement must be validated.
1. Have the desires and needs of the stakeholders been mapped to requirements?
2. Do the requirements adequately represent the desires and needs of the stakeholders? Have conflicting requirements been resolved or prioritised?
3. Are the requirements internally consistent?
4. The ultimate validation question: Are the stakeholders sufficiently satisfied with the operation of the installed system?

The validation questions **1** – **3** are used to reduce the risk of **4** not applying.

Methodology and Discipline

Otherwise it won't work..
Requirements must be
- systematically identified, documented, reviewed and managed
- as part of an appropriate process and using appropriate practices.

Even for agile development
Agile development also requires methodology and discipline, but in a different form.

Methodology does not mean commonplace
- There's no "One size fits all" – an appropriate procedure must be defined for each RE.
- No unreflective 1:1 takeover of RE practices from previous projects.