Nine Fundamental Principles of Requirements Engineering

IREB® International Requirements Engineering Board

CPRE Certified Professional for Requirements Engineering

Value Orientation:

1

Requirements are means to an end, not an end in itself

The act of writing requirements is not a goal in itself. To be useful, requirements need to add value. The value of a requirement is defined as its **benefit minus its cost**. The better the requirements help to fulfill the stakeholders' needs and desires and reduce rework costs, the higher the benefit.

To optimize the value of a requirement, Requirements Engineers elicit and document the needs of the stakeholders as precisely as possible. Depending on how extensive and precise the requirements are specified, the costs involved in eliciting and documenting them increase.

The economic value of RE is mostly indirect because early implementation reduces rework costs in the long term, but the costs for eliciting and documenting requirements are immediate.

Stakeholders:

RE is about satisfying the stakeholders' desires and needs

2

The core goal of RE is to *understand the stakeholders' desires and needs* and to *minimize the risk* of delivering a system that doesn't meet these. The stakeholders and their roles must be considered to fulfill this goal.

Every stakeholder has a *role* in the context of the system (e.g., user, client, customer, operator, or regulator) and thus different viewpoints on requirements. So, stakeholders have to be classified with respect to the degree of their influence (**critical, major, minor**) on the system's success. This helps with assessing the criticality of requirements and in negotiation conflicts.

It is vital for the success of RE to identify inconsistencies and conflicts between the requirements of different stakeholders and to resolve them, be it by finding a consensus, overruling, or specifying system variants for stakeholders with different needs.

Shared understanding:

Successful systems development is impossible without common basis

3

For successful RE implementation, a **shared understanding** of the problem between stakeholders, Requirements Engineers, and developers is needed.

We distinguish between *explicit shared understanding* (often used for plan-driven processes), which is achieved through carefully elicited, documented, and agreed requirements, and *implicit shared understanding* (often used in agile processes), which is based on shared knowledge about needs, visions, context, etc. Whether explicit or implicit, we can't rely blindly on shared understanding, as stakeholders may interpret issues differently. It is the task of RE to create *true* shared understanding.

Practices to achieve this include working with **glossaries** and **prototypes**, using **existing systems as references**, providing **examples of expected outcomes**, **estimating costs of implementing a requirement**, and **short feedback loops**.

Context:

4

Systems cannot be understood in isolation

Systems are always embedded in a larger **context**. Without understanding this **system context,** it's almost impossible to specify a **system** correctly. The system context in RE is defined as the part of a system's environment being relevant for understanding the system and its requirements. A user login for a banking system might come up with different requirements than one for the members area in a sports club.

Understanding the context of the system is important to identify the relevant interfaces to existing systems, regulations, and stakeholders of the intended system. Understanding the context of a system helps to fix the **context boundary** (defining what part of the system context has to be considered or not). It also fixes the **system boundary** (defining what is within the scope of the intended system and what is not).

It is important to understand that RE goes beyond considering requirements in the system boundary. RE also has to deal with aspects that are coming from the system context.

Problem, requirement, solution:

An inevitably intertwined triple

5

Problems, their **solutions**, and **requirements** are closely intertwined. Any situation in which people are unsatisfied with processes and results can be considered a **problem**. To eliminate the **problem**, a socio-technical system may come into action. This system needs **requirements** to make the system a **solution** to the **problem**.

It's important to understand that problems, requirements and solutions don't always arise in this chronological order. The design of an innovative system can be driven by a solution that ends up solving a known problem. Understanding the problem and the solution in turn leads to the elaboration of requirements that satisfy user needs and are implemented in a concrete solution.

Despite the intertwinement of problems, requirements, and solutions, Requirements Engineers strive to separate requirements concerns from solution concerns when thinking, communicating, and documenting in order to provide as little technical direction as possible.

Validation:

Non-validated requirements are useless

6

The ultimate goal of developing and deploying a system is to satisfy all stakeholders' desires and needs. However, waiting until the end is risky and bears the risk of unsatisfied stakeholders. Thus, the **validation of requirements** must start during RE.

Validation is a core activity of RE and describes the process of confirming that the documented requirements match the stakeholders' needs and whether the right requirements have been specified.

During the validation, it is crucial to check whether **all conflicts among stakeholders are resolved** and **priorities are set**, that the **stakeholders' desires and needs are covered by the requirements** and the **domain assumptions are reasonable**.

7

Evolution:

Changing requirements are no accident, but the normal case

Needs, businesses and capabilities are always exposed to evolution. As a consequence, the requirements for systems that satisfy needs, support businesses and use technical capabilities will also change. **If the requirements don't keep up with the evolution of the system, they will eventually lose value and become useless.**

Reasons for the changes are vast and can range from changed business processes to the detection of errors or faulty domain assumptions.

On the one hand, Requirements Engineers have to permit requirements to change, since ignoring their need to change would be futile. On the other hand, they have to keep requirements stable, because the cost for change can become high and development teams can't develop systematically with constantly changing requirements.

Innovation:

8

More of the same is not enough

As mentioned before, the main concern of RE is to satisfy stakeholder's desires and needs. However, it is crucial to **not fall into the role of the stakeholder's voice recorder**, specifying exactly what the stakeholders tell you since that would mean missing out on the opportunity of doing things better than before.

Requirements Engineers must walk the fine line between **staying innovative** without believing they know everything better than the stakeholders.

On a small scale, RE shapes innovative systems by striving for exciting new features and ease of use. Beyond that, Requirements Engineers also need to look for the big picture, exploring with the stakeholders whether there are any disruptive ways of doing things, leading to large-scale innovation.

9

Systematic and disciplined work:

We can't do without RE

RE is not an art but a discipline, which calls for RE to be performed in a systematic and disciplined way. Even when a system is developed in an ad hoc fashion, a systematic and disciplined approach to RE will improve the quality of the resulting system. **Agility and flexibility are not valid excuses for an unsystematic, ad hoc style of work in RE**.

However, **there is no “one size fits all” for RE**. Requirements Engineers need to conceptualize RE processes that are suited for the problem at hand. They shouldn’t always use the same process, practices and work products, but select those that help best with the given problem, context and working environment. Last but not least, processes and practices from past successful RE work shouldn’t be reused without reflecting upon them first.