Requirements

Introduction

★★☆☆ Can explain requirements

A *software requirement* specifies a need to be fulfilled by the software product.

A software project may be,

- **a *brown-field* project** i.e., develop a product to replace/update an existing software product
- **a *green-field* project** i.e., develop a totally new system with no precedent

In either case, requirements need to be gathered, analyzed, specified, and managed.

Requirements come from *stakeholders*.

> **Stakeholder**: A party that is potentially affected by the software project. e.g. users, sponsors, developers, interest groups, government agencies, etc.

Identifying requirements is often not easy. For example, stakeholders may not be aware of their precise needs, may not know how to communicate their requirements correctly, may not be willing to spend effort in identifying requirements, etc.

Non-Functional Requirements

★★☆☆ Can explain non-functional requirements

Requirements can be divided into two in the following way:

1. ***Functional requirements* specify what the system should do.**
2. ***Non-functional requirements* specify the constraints under which the system is developed and operated.**

Some examples of non-functional requirement categories:

- Data requirements e.g. size, volatility, persistency etc.,
- Environment requirements e.g. technical environment in which the system would operate in or needs to be compatible with.
- Accessibility, Capacity, Compliance with regulations, Documentation, Disaster recovery, Efficiency, Extensibility, Fault tolerance, Interoperability, Maintainability, Privacy, Portability, Quality, Reliability, Response time, Robustness, Scalability, Security, Stability, Testability, and more ...

Some concrete examples of NFRs

- Business/domain rules: e.g. the size of the minefield cannot be smaller than five.
- Constraints: e.g. the system should be backward compatible with data produced by earlier versions of the system; system testers are available only during the last month of the project; the total project cost should not exceed $1.5 million.
- Technical requirements: e.g. the system should work on both 32-bit and 64-bit environments.
- Performance requirements: e.g. the system should respond within two seconds.
- Quality requirements: e.g. the system should be usable by a novice who has never carried out an online purchase.
- Process requirements: e.g. the project is expected to adhere to a schedule that delivers a feature set every one month.
- Notes about project scope: e.g. the product is not required to handle the printing of reports.
- Any other noteworthy points: e.g. the game should not use images deemed offensive to those injured in real mine clearing activities.

You may have to spend an extra effort in digging NFRs out as early as possible because,

1. **NFRs are easier to miss** e.g., stakeholders tend to think of functional requirements first
2. sometimes **NFRs are critical to the success of the software.** E.g. A web application that is too slow or that has low security is unlikely to succeed even if it has all the right functionality.

Exercises

Quality of Requirements

★★★☆ Can explain quality of requirements

Here are some characteristics of well-defined requirements [📖 zielczynski]:

- Unambiguous
- Testable (verifiable)
- Clear (concise, terse, simple, precise)
- Correct
- Understandable
- Feasible (realistic, possible)
- Independent
- Atomic
- Necessary
- Implementation-free (i.e. abstract)

Besides these criteria for individual requirements, the set of requirements as a whole should be

- Consistent
- Non-redundant
- Complete

Prioritizing Requirements

★★★☆ Can explain prioritizing requirements

Requirements can be prioritized based on the importance and urgency, while keeping in mind the constraints of schedule, budget, staff resources, quality goals, and other constraints.

A common approach is to group requirements into priority categories. Note that all such scales are subjective, and stakeholders define the meaning of each level in the scale for the project at hand.

An example scheme for categorizing requirements:

- `Essential`: The product must have this requirement fulfilled or else it does not get user acceptance.
- `Typical`: Most similar systems have this feature although the product can survive without it.
- `Novel`: New features that could differentiate this product from the rest.

Other schemes:

- `High`, `Medium`, `Low`

- `Must-have`, `Nice-to-have`, `Unlikely-to-have`
- `Level 0`, `Level 1`, `Level 2`, ...

Some requirements can be discarded if they are considered 'out of *scope*'.

The requirement given below is for a Calendar application. Stakeholders of the software (e.g. product designers) might decide the following requirement is not in the scope of the software.

> The software records the actual time taken by each task and show the difference between the *actual* and *scheduled* time for the task.