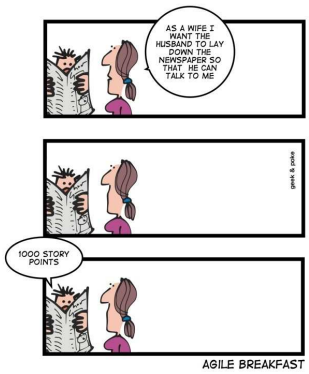


Software Requirements

www.cs.uoi.gr/~zarras/http://www.cs.uoi.gr/~zarras/se.htm

Slides material sources:

Software Engineering - Theory & Practice, S. L. Pfleeger

Introduction to Software Engineering, I. Sommerville

SWEBOK v3: IEEE Software Engineering Body of Knowledge

IEEE Recommended Practice for Software Requirements Specifications

Software requirements fundamentals

What is a software requirement?

What is a requirement?

At its most basic, a **requirement** is a property that must be exhibited by a software in order to solve some problem in the real world.

SWEBOK v3: IEEE Software Engineering Body of Knowledge

It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.

How can we classify requirements?

How can we classify requirements?

Product / Process requirements

Functional / Non-functional requirements

System / Software requirements

What makes them so important?

What makes them so important?

According to a large scale empirical study (> 8000 projects) done by Standish in 1995 they are amongst the **top factors** that **cause a project to fail**.

- Incomplete requirements (13.1%)
- Lack of user involvement (12.4%)
- Unrealistic expectations (9.9%)
- Lack of executive support (9.3%)
- Changing requirements and specifications (8.7 %)
- Lack of planning (8.1%)
- System no longer needed (7.5%)

What is a functional requirement?

What is a functional requirement?

A **functional requirement** describes a function/service that the software is to execute/provide.

Sometimes known as capability or feature.

What is a non functional requirement?

What is a non functional requirement?

A **non-functional requirement** is a constraint on the provided functions/services.

Sometimes known as constraints or quality requirements.

Can be further classified according to whether they are performance requirements, maintainability requirements, safety requirements, reliability requirements, security requirements, availability requirements, interoperability requirements…. or any other quality attribute….

What makes a good software requirement specification (SRS)?

It should be correct

IEEE Recommended Practice for Software Requirements Specifications

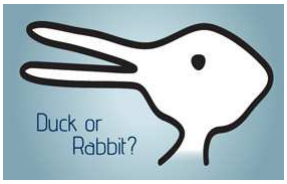
An SRS is correct if, and only if, every requirement stated therein is one that the software should/shall meet.

There is no tool or procedure that ensures correctness.

The SRS should be compared with any applicable superior specification, such as a system requirements specification, with other project documentation, and with other applicable standards, to ensure that it agrees.

Alternatively the customer or user can determine if the SRS correctly reflects the actual needs.

It should be unambiguous

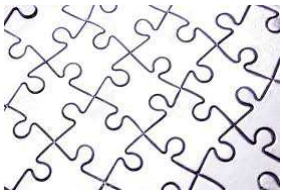


IEEE Recommended Practice for Software Requirements Specifications

An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation.

As a minimum, this requires that each concept/characteristic of the product be described using a single unique term.

It should be complete



IEEE Recommended Practice for Software Requirements Specifications

An SRS is complete if, and only if it includes:

All significant requirements.

Definition of the responses of the software to all realizable classes of input data in all realizable classes of situations. Note that it is important to specify the responses to both valid and invalid input values.

Full labels and references to all figures, tables, and diagrams in the SRS and definition of all terms and units of measure.

It should be consistent

IEEE Recommended Practice for Software Requirements Specifications

SRS is internally consistent if, and only if, no subset of individual requirements described in it conflict.

Consistency refers to internal consistency. If an SRS does not agree with some higher-level document, such as a system requirements specification, then it is not correct.

It should be consistent

IEEE Recommended Practice for Software Requirements Specifications

The specified characteristics of real-world concepts may conflict.

For example:

a) The format of an output report may be described in one requirement as tabular but in another as textual.
b) One requirement may state that all lights shall be green while another may state that all lights shall be blue.

It should be consistent

IEEE Recommended Practice for Software Requirements Specifications

There may be logical or temporal conflicts between two specified actions.

For example one requirement may state that A must always follow B, while another may require that A and B occur simultaneously.

It should be consistent

IEEE Recommended Practice for Software Requirements Specifications

Two or more requirements may describe the same real-world object but use different terms for that object.

For example, a program's request for a user input may be called a prompt in one requirement and a cue in another.

It should be ranked for importance

IEEE Recommended Practice for Software Requirements Specifications

An SRS is ranked for importance if and only if each requirement in it has an identifier to indicate the importance of that particular requirement.

Essential - Implies that the software will not be acceptable unless these requirements are provided in an agreed manner.

Conditional - Implies that these are requirements that would enhance the software product, but would not make it unacceptable if they are absent.

Optional - Implies a class of functions that may or may not be worthwhile. This gives the supplier the opportunity to propose something that exceeds the SRS.

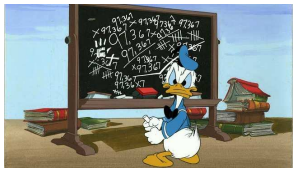
It should be ranked for stability

IEEE Recommended Practice for Software Requirements Specifications

An SRS is ranked for stability if and only if each requirement in it has an identifier to indicate the stability of that particular requirement.

Stability can be expressed in terms of the number of expected changes to any requirement based on experience or knowledge of forthcoming events that affect the organization, functions, and people supported by the software system.

It should be verifiable



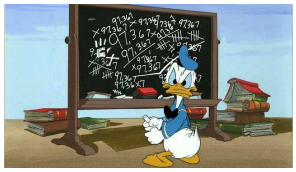
IEEE Recommended Practice for Software Requirements Specifications

An SRS is verifiable if, and only if, every requirement stated therein is verifiable.

A requirement is verifiable if, and only if, there exists some finite cost-effective process with which a person or machine can check that the software product meets the requirement.

In general any ambiguous requirement is not verifiable.

It should be verifiable



IEEE Recommended Practice for Software Requirements Specifications

Examples of non-verifiable requirements are statements such as:

"works well", "good human interface" and "shall usually happen"

An example of a verifiable statement is:

Output of the program shall be produced within 20 s of event X 60% of the time; and shall be produced within 30 s of event Y 100% of the time.

This statement can be verified because it uses concrete terms and measurable quantities.

It should be modifiable

IEEE Recommended Practice for Software Requirements Specifications

An SRS is modifiable if, and only if, its structure and style are such that any changes to the requirements can be made easily.

SRS must have a coherent and easy-to-use organization with a table of contents, an index, and explicit cross-referencing.

Not be redundant (i.e., the same requirement should not appear in more than one place in the SRS).

Express each requirement separately, rather than intermixed with other requirements.

It should be traceable

IEEE Recommended Practice for Software Requirements Specifications

An SRS is traceable if the origin of each of its requirements is clear and facilitates the referencing of each requirement in future development or documentation.

Backward traceability - each requirement must explicitly reference its source in earlier documents.

Forward traceability - each requirement in the SRS must have a unique name or reference number.

Software requirements process & techniques

Where do we start from?

Requirements elicitation

"Yes, I'm a real Genie... but you're asking me to understand your client's requirements and even I can't do that!"

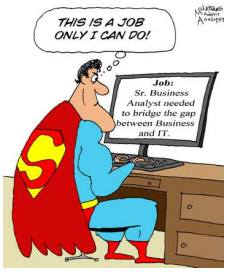
Requirements elicitation is the first stage in building an understanding of the problem the software is required to solve.

It is fundamentally a **human activity** and is where the stakeholders are identified and relationships established between the development team and the customer.

It is variously termed requirements capture, requirements discovery, and requirements acquisition.

How do we collect requirements?

Elicitation techniques



Interviews - Interviewing stakeholders is a "traditional" means of eliciting requirements.

Types of interviews

Closed interviews based on pre-determined list of questions

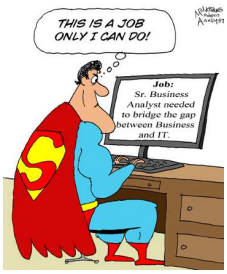
Open interviews where various issues are explored with stakeholders.

Effective interviewing

Be open-minded, avoid pre-conceived ideas and be willing to listen to stakeholders.

Prompt the interviewee using a springboard question, a requirements proposal, or by working together on a prototype system.

Elicitation techniques



Interviews in practice

Normally a mix of closed and open-ended interviewing.

Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.

Interviews are not good for understanding domain requirements

Requirements engineers cannot understand specific domain terminology;

Domain experts are so familiar with domain knowledge that find it hard to articulate what is useful or filter what isn't.

Elicitation techniques

Scenarios

Scenarios are real-life descriptions of **how a system can be used**.

They should include:

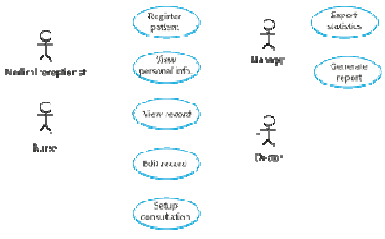
A description of the starting situation;

A description of the normal flow of events;

A description of what can go wrong;

A description of the state when the scenario finishes.

Elicitation techniques



Use-cases

A scenario based technique in UML which identify the actors in an interaction and which describe the interaction itself.

High-level graphical model supplemented by more detailed tabular description.

Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Elicitation techniques

Prototypes

This technique is a valuable tool for clarifying **ambiguous** requirements and assess **alternatives**.

They provide users with a context within which they can better understand what information they need to provide.

Elicitation techniques

Prototypes

- Paper mockups
- Screen designs
- Beta-test versions

Low fidelity prototypes are often preferred to avoid stakeholder "**anchoring**" on minor, incidental characteristics of a higher quality prototype that can limit design flexibility in unintended ways.

Elicitation techniques

Facilitated meetings

The purpose of these meetings is to try to achieve a summative effect, whereby a group of people can bring more insight into their software requirements than by working individually.

They can brainstorm and refine ideas that may be difficult to bring to the surface using interviews.

Another advantage is that **conflicting** requirements surface early on in a way that lets the stakeholders recognize where these occur.

Elicitation techniques

Facilitated meetings

May result in a richer and more consistent set of requirements than might otherwise be achievable.

However, meetings need to be **handled carefully** to prevent a situation in which the critical abilities of the team are eroded by **group loyalty**, or in which requirements reflecting the concerns of a few outspoken (and perhaps senior) people.

Elicitation techniques

Observation - ethnography

The importance of software context within the organizational/operational environment has led to observational techniques.

> Software engineers learn about user tasks by observing how users perform their tasks by interacting with each other and with software tools and other resources.
>
> These techniques are relatively **expensive** but also **instructive** because they illustrate that many user tasks and business processes are too subtle and complex for their actors to describe.

Ethnography is effective for understanding existing processes but **cannot identify new features**.

Elicitation techniques

"I'll go talk to the stakeholders and find out their requirements... in the meantime, you guys start coding."

User stories

This technique is commonly used in agile methods (esp. XP)

Refers to short, high-level descriptions of required functionality expressed in customer terms.

A typical user story has the form:

"As a <role>, I want <goal/desire> so that <benefit>."

Elicitation techniques

User stories

A user story contains just enough information so that the we can produce a reasonable estimate of the effort to implement it. The details are orally discussed

Before a user story is implemented, a detailed acceptance test must be provided to determine whether the goals of the user story have been fulfilled.

What do we do once we have the requirements?

Requirements analysis

"We have been having a hard time guessing the business requirements. I'm hoping our new analyst can help."

Requirements Classification

Requirements can be classified on a number of dimensions:

- Functional/Non Functional
- Product/Process
- Importance
- Stability
- Scope

Requirements analysis

"We have been having a hard time guessing the business requirements. I'm hoping our new analyst can help."

Conceptual modeling

The development of models of a real-world problem is key to software requirements analysis.

Their purpose is to aid in understanding the situation in which the problem occurs, as well as depicting a solution.

Hence, conceptual models comprise models of entities from the problem domain, configured to reflect their real-world relationships and dependencies.

Requirements analysis

"We have been having a hard time guessing the business requirements. I'm hoping our new analyst can help."

Conceptual modeling

Several kinds of models can be developed.

- Use case diagrams
- Data flow models
- State models
- Process/Activity models
- Data models
- and many others…..

Many of these modeling notations are part of *the* **Unified Modeling Language (UML)**.

Requirements analysis

"We have been having a hard time guessing the business requirements. I'm hoping our new analyst can help."

Architecture design and requirements allocation

At some point, the solution architecture must be derived. Architectural design is the point at which the requirements process overlaps with software or systems design.

The **requirements allocation** amounts to identifying the architecture/design components that will be responsible for satisfying the requirements.

Requirements analysis

"We have been having a hard time guessing the business requirements. I'm hoping our new analyst can help."

Requirements negotiation

Another term commonly used for this task is "conflict resolution."

This concerns resolving problems with requirements where conflicts occur between stakeholders requiring mutually incompatible features, between requirements and resources, or between functional and nonfunctional requirements, for example.

What do we do after the analysis?

Requirements specification

"Good news! He said he only needs a few more weeks to finish the first draft of the Requirements Document."

Requirements specification typically refers to the production of a document that can be systematically reviewed, evaluated, and approved.

For complex systems, particularly those involving substantial non-software components, as many as three different types of documents are produced: system definition, system requirements, and software requirements.

For simple software products, only the software requirements spec is required.

Requirements validation

The goal of this task is to check if the requirements specification is of **good quality** wrt required characteristics like correctness, completeness, unambiguity, traceability, verifiability, modifiability….


This is Joe... he's the requirements approver from the business side. Joe - do we have your approval?
THIS IS A JO
"We have been having a hard time guessing the business requirements. I'm hoping our new analyst can help."
"Good news! He said he only needs a few more weeks to finish the first draft of the Requirements Document."