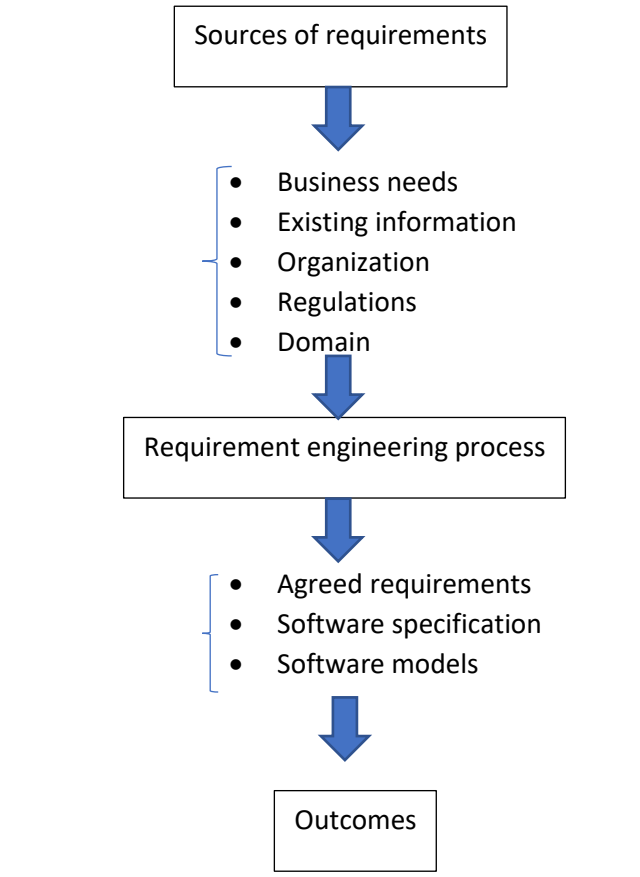
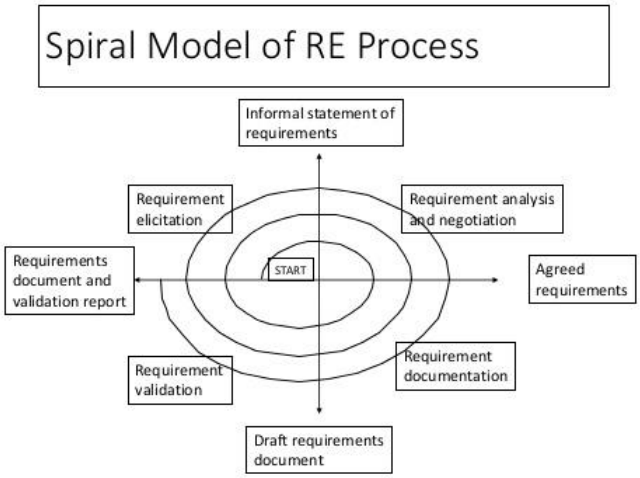
Contenido

Software Requirements

A requirement is something the product must do or a quality it must have, to solve a specific problem by a user. Any requirement must be correct and complete. It must be documented.





Stakeholders are the ones who participate in Requirements Engineering.

- **Client**: the one who pays for the development.
- **Customer**: the one who buys the software product.
- **User**: the one who uses the software product. He/she has direct contact with the product.

- Other stakeholders: consultants, management, core team, technical experts, marketing, lawyers, etc.

More stakeholders → harder to reach correct requirements.

Types of requirements

- Functional requirements → What the users need for the system to work. What the user pays for.
- Non functional requirements → How the system will do what it needs to do → look & feel, usability, security, legal… The user does not pay for it, but it is compulsory to do.

When to start requirements engineering?

Kick off meeting → purpose of project + Facts + Scope + Constraints + Stakeholders + Terminology + Risks + Costs

Scope of the work

Work: business area affected by the product (people + process + product)

Scope→ decide what will be studied and what not. We must identify areas of interest and adjacent systems.

Adjacent systems

Organization, individual, computer system, tech or a combination.

- Active adjacent systems → Humans → Initial Business Events
- Autonomous Adjacent Systems → one day data flow (no response / feedback)
- Cooperative Adjacent Systems → request-response communication

Business Events

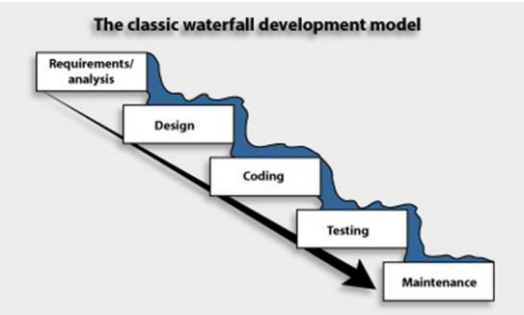
Business events take place outside the scope of work → arrival of incoming information.

Business events are known to the stakeholders. Every business event has a response → Business Use Case (BUC), which must be elicited with stakeholders.

Software Development Process

Ad-hoc → wild west approach. No organization, bad for big projects.

Waterfall → classic, rigid. Ignores changing needs (no going back), lack of user involvement once the specification is written. Doesn't accommodate prototyping, reuse, etc. This model is only appropriate when the requirements are well-understood.



Prototyping →Prototypes are pilot projects.

Throw away projects: usually non functional, no code, just graphical interface.

Evolutionary projects: there is an evolution behind the prototype until the final app is done.

Prototyping is used for:

- understanding the requirements for the user interface
- examining feasibility of a proposed design approach
- exploring system performance issues

Problems:

- users treat the prototype as the solution
- a prototype is only a partial specification

Incremental process → Requirements are static, each iteration adds new functionality. Gaming is a good example. We introduce new functionalities/requirements per release.

Evolutionary Process → Each iteration has its requirements. New requirements are created when bugs appear, new versions running all the time. (O&M: operation and maintenance)

Spiral Model→ CYCLE: Plan → Requirements → Risks → Develop + Test

in the cycle we always ask ourselves: shall we go ahead?

RUP Structure: iteractive process, waterfall, targets, no me he enterado de nada

Agile Software Development used by big companies

RUP → Waterfall + iterations + workflows; used by many companies (IBM)

Types of projects

- Rabbit: very agile, iterations, small upgrades, short life, not much documentation.
- Horse: Common longevity, order and formal.
- Elephant: Arrangements and regulations, very formal, lots of communication and documentation.

Requirements Elicitation

Roles of software engineers

- Explorer → Domain + requirements
- Mediator → Find compromise
- Documenter → About elicitation process
- Developer → Developing software
- Validator → Compare to stakeholders' wishes
- Facilitator → Find relevant questions

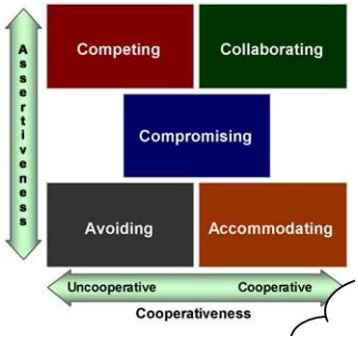
Guidelines to elicitation

- System Feasibility (have organizational/political considerations)
- Identifying and consult stakeholders. Record sources.
- Business requirements
- Collect multiple viewpoints.
- Prototype poorly understood requirements.

- Define optional process
- Reuse requirements.

Requirement Negotiation

There are five dominant stakeholder position: competing, collaborating, compromising (avg), avoiding, accommodating. We must aim for the middle position (compromising).



Assertiveness = focus on own concerns; Cooperativeness = focus on satisfying concerns of others.

Aspects to consider when prioritizing:

- Importance
- Penalty
- Cost
- Time
- Risk
- Volatility
- Financial, strategical aspects
- Combination of different aspects

Prioritization techniques:

- AHP: Systematic process (calculation) for finding which requirements have the highest priority by comparing in hierarchy. Too many requirements may be a problem for this technique.
- Hundred dollar test: distribute 100 imaginary units between all requirements. Not functional when there are too many requirements. Test cannot be done twice.
- Wiegers Method: establish benefit/penalty and cost/risk values. Open interpretation of each stakeholders.
- Ranking: Give each requirement a priority number. Difficult to manage with many stakeholders.
- Numeral assignment: Divide requirements in 3 groups (critical, standard, optional). Stakeholders may think everything is critical.
- Top Ten: For every stakeholder creates a list of 10 requirements to be fulfilled. Important to choose the requirements in a proportional basis (must not take all of 1 stakeholder).

Elicitation techniques

(process of getting information)

Interview: simple, traditional. Large amounts of data are collected quickly. It needs effective questioning. It is efficient to collect large amounts of data in short time period.

Questionnaire: open/closed questions, checklist style. We ensure to get info about essential elements. Must avoid redundancy.

Task analysis: top-down approach. Gives contextual description of activities.

Domain analysis: Explore domain. Create understanding between analyst and stakeholders. It requires to know well the domain.

Introspection: What the analyst believes the stakeholders and users need and want. Effective when the domain is well known by the analyst. No interaction with stakeholders (dangerous).

Group work: Collaborative meetings where people are put together. It involves stakeholders in collaborative meetings. Effective because they involve stakeholders but difficult to organize, requires experience.

Brainstorming: Informal discussion to generate a lot of ideas quickly. We must know what is useful or not.

JAD: It is like brainstorming but with the goal already stablished from the start. Fast response. More related to business than tech issues.

Workshops: group meetings to develop and discover requirements. Encourages innovative thinking and expression. Must find the right workshop to suit your needs.

Observation: Learn by seeing users. Users may act differently.

Protocol analysis: perform an activity describing the process behind them.

Apprenticing: Learn and perform task with an experienced user. Useful when analyst is inexperienced with the domain. Requires a lot of involvement.

Prototyping: creating prototypes for stakeholders; gather detailed information and relevant feedback. Encourage stakeholders to take an active role. Expensive (time and cost).

Scenarios: narrative descriptions of current and future processes. Useful for validating requirements. Requires iterative approach.

Risks

Risk can also mean opportunity, no risk no win. It is a potential problem, it may happen or no.

Importance of risk: less quality, more cost, delays, project failure.

Risks at software project level

- Project risks: related to project plan. Schedule will slip and costs will increase.
- Technical risks: Implementation problems. Quality and timelines.
- Business risks: Problems in project cost/benefit.

Risk categories

- Known risks: risks that can be uncovered after evaluation.
- Predictable risks: risks that are extrapolated from past project experiences.
- Unpredictable risks: sometimes we can't predict them (corona 😊). Extremely difficult to identify.

Risks in requirement engineering

Requirement elicitation

- Inadequate customer representation.
- Overlooking crucial requirements. Missing functional and non functional requirements.
- Modeling only functional requirements.
- Not enough time spent on requirements development.

Requirement analysis

- Requirement prioritization: prioritize every requirement.
- Technical difficulties, unfamiliar technologies.

Requirement specification

- Requirement understanding among stakeholders. Analysis and right questions.
- Don´t put solutions on SRS, put business problems.
- Don´t press forward without "To be determined" sections finish.

Requirement validation

Find problems at the earliest point

Software requirements specification (SRS)

It is a compilation of all requirements (clear, complete, testable). Is often done using a tool, as specifications are assembled and not written at once.

Requirements specification is done during elicitation process, converting potential requirements into formalized requirements.

We must use quality gateways in order to double check requirements.

What to put in SRS?

- Project drivers (stakeholders)
- Project constraints (names, facts, assumptions)
- Projects issues (risks and costs)
- Functional requirements (what to do)
- Non-functional requirements (how to do)

Software requirements validation

RFTS → Right From The Start. Use quality gateways when eliciting and specifying. Quality must be applied in all the process, not in the end.

Each requirement is tested. If accepted→specification; if not → return to source. Wrong specification means wrong product (irrelevant, inconsistent…).

Attributes to check

- Complete: (are they missing components?) name, description, originator,… we must use at least 2 prio. Techniques.
- Traceable: all requirements must be able to identify so we can check them.
- Relevant
- Consistent: definition of the essential terms and use of a consistent definition.
- Non ambiguous: does the requirement have fit criteria? Fit criterion → unambiguous goal that the product must meet.

- Viable: it is acceptable for all stakeholders?
- Don´t use solutions: we must take notes of what we need and not about how to do.
- Gold plating: “aparenta pero nou” Unnecessary features, nice things that look nice but not necessary.
- Creeping requirements. Dangerous, as they appear as new requirements suddenly.

Others

Product context diagram

Use Case (UC) diagram