CS445 / SE463 / ECE 451 / CS645

Software requirements specification & analysis

Requirements elicitation

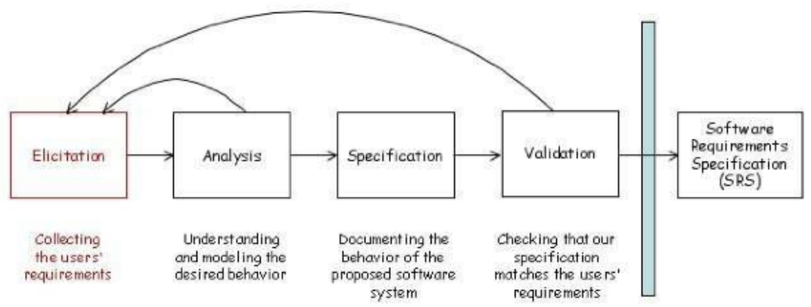
Fall 2016 — Mike Godfrey substituting for Dan Berry, but using "his" own slides

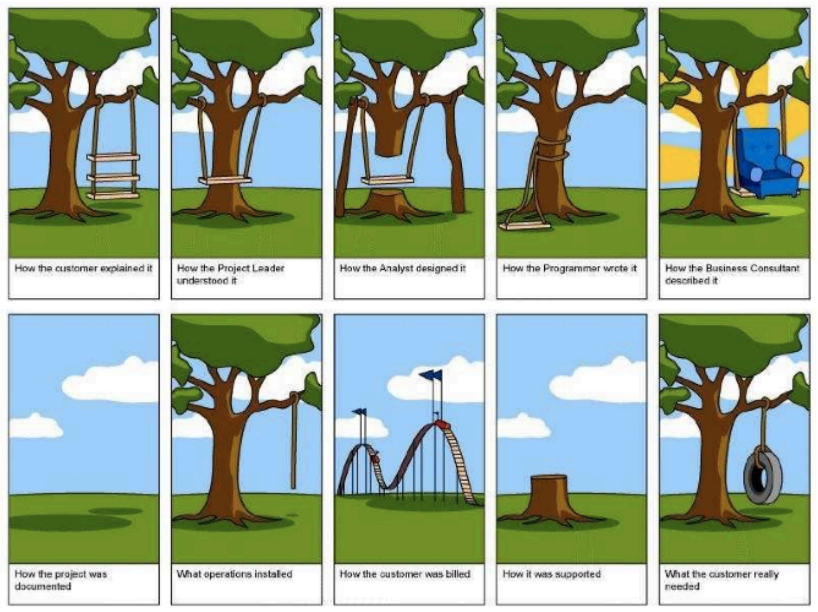
Overview

- Requirements elicitation
  - Types of requirements
  - Stakeholders
  - Requirements elicitation techniques
    - Artifact based
    - Stakeholder based
    - Model based
    - "Creative"

[Some material due to Prof. Steve Easterbrook, UofT]

The requirements engineering process


Elicitation
Analysis
Specification
Validation
Software Requirements Specification (SRS)
Collecting the users' requirements
Understanding and modeling the desired behavior
Documenting the behavior of the proposed software system
Checking that our specification matches the users' requirements


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Requirements

- The first step in engineering software is making sure that the software does what the customer wants it to do.
- A couple of views on the notion of requirements
  - A description of a problem within a context

    i.e., how we want the world to behave
  - A set of *capabilities* and/or *conditions* that must be achieved for the *user* or *customer* to consider the software an *acceptable solution* to their problem.

Types of requirements

- *Functions* describe what the software should do

  e.g., record or transform data, perform computations, answer a query, issue commands to hardware devices or adjacent software systems
- *Attributes* are product characteristics desired by the client (aka *quality requirements* aka *NFRs*)
  - Two products with nearly identical function sets are distinguished mainly by their attributes e.g., Corolla vs. Ferrari

Types of requirements

- *Constraints* are customer-specified limits on the solution space

  e.g., mandated hardware components, mandated adjacent systems
- *Assumptions* about the environment describe the assumed context of the product.
  - The product is expected to function correctly if placed in an environment that satisfies these assumptions.

Other info to be elicited

- A *preference* is a desirable, but optional, condition placed on any function or attribute.
  - Any final design that satisfies all functions and attributes is acceptable, but some acceptable designs are preferable to others.
  - Preferences allow designers to compare acceptable solutions and choose the better ones.
  - Preferences are not requirements *per se*, but they should be noted during elicitation

Other info to be elicited

- The client, just as anyone else, has *expectations*.
  - The difference between disappointment and delight over a product is how well expectations are matched upon delivery of the product.
  - Sometimes, a client's expectations are too high.
    - Perhaps, the client has developed unreasonable expectations for the product from having seen other products or movies.
    - It is our job to limit the client's expectations to something reasonable.
  - Expectations are not requirements per se, but they should also be noted during elicitation

Dear Mr. Architect:

Please design and build me a house. I am not quite sure of what I need, so you should use your discretion. My house should have somewhere between two and forty-five bedrooms. Just make sure the plans are such that the bedrooms can be easily added or deleted. When you bring the blueprints to me, I will make the final decision of what I want. Also, bring me the cost breakdown for each configuration so that I can arbitrarily pick one.

Keep in mind that the house I ultimately choose must cost less than the one I am currently living in. Make sure, however, that you correct all the deficiencies that exist in my current house (the floor of my kitchen vibrates when I walk across it, and the walls don't have nearly enough insulation in them).

As you design, also keep in mind that I want to keep yearly maintenance costs as low as possible. This should mean the incorporation of extra-cost features like aluminum, vinyl, or composite siding. (If you choose not to specify aluminum, be prepared to explain your decision in detail.)

Please take care that modern design practices and the latest materials are used in construction of the house, as I want it to be a showplace for the most up-to-date ideas and methods. Be alerted, however, that kitchen should be designed to accommodate, among other things, my 1952 Gibson refrigerator.

To ensure that you are building the correct house for our entire family, make certain that you contact each of our children, and also our in-laws. My mother-in-law will have very strong feelings about how the house should be designed, since she visits us at least once a year. Make sure that you weigh all of these options carefully and come to the right decision. I, however, retain the right to overrule any choices that you make.

Please don't bother me with small details right now. Your job is to develop the overall plans for the house: get the big picture. At this time, for example, it is not appropriate to be choosing the color of the carpet. However, keep in mind that my wife likes blue.

Also, do not worry at this time about acquiring the resources to build the house itself. Your first priority is to develop detailed plans and specifications. Once I approve these plans, however, I would expect the house to be under roof within 48 hours.

While you are designing this house specifically for me, keep in mind that sooner or later I will have to sell it to someone else. It therefore should have appeal to a wide variety of potential buyers. Please make sure before you finalize the plans that there is a consensus of the population in my area that they like the features this house has. I advise you to run up and look at my neighbor's house he constructed last year. We like it a great deal. It has many features that we would also like in our new home, particularly the 75-foot swimming pool. With careful engineering, I believe that you can design this into our new house without impacting the final cost.

Please prepare a complete set of blueprints. It is not necessary at this time to do the real design, since they will be used only for construction bids. Be advised, however, that you will be held accountable for any increase of construction costs as a result of later design changes.

You must be thrilled to be working on as an interesting project as this! To be able to use the latest techniques and materials and to be given such freedom in your designs is something that can't happen very often. Contact me as soon as possible with your complete ideas and plans.

Yours sincerely, the client.

PS: My wife has just told me that she disagrees with many of the instructions I've given you in this letter. As architect, it is your responsibility to resolve these differences. I have tried in the past and have been unable to accomplish this. If you can't handle this responsibility, I will have to find another architect.

PPS: Perhaps what I need is not a house at all, but a travel trailer. Please advise me as soon as possible if this is the case.

[An old USENET posting — MWG]

Elicitation

- To *elicit* means "to bring out, to evoke, to call forth"
- The purpose of elicitation is to get information about:
  - current work and current problems,
  - the requirements of the system, and
  - the domain and environment in which the system will operate.

Identify the stakeholders

- A *stakeholder* is anyone who has a stake in the ultimate success of the project.

Client	Inspectors
Customer	Market researchers
Users	Lawyers
Domain experts	Experts on "adjacent" systems
Software engineers / managers	Value-adders

- During elicitation, we *want* to talk to all of the stakeholders.
  - But sometimes we have to make do with *proxies*

Stakeholders: Client

- The *client* is the person paying for software *to be developed*
  - They are the ultimate stakeholder — almost always, the client has the last say in what the product does
  - For "bespoke" or customized systems, the client is the person with the chequebook ☺
  - For software intended for the mass market, the client may be the company developing the software
  - For in-house software, the client is probably the manager of the product's users
    - Since his/her employees will be the primary beneficiaries, it is reasonable for him/her to pay for the project

Stakeholders: Customer

- A *customer* is a person who buys software *after it is developed*
  - May be the same as the client
  - May be the same as the user; other times, the customer is an office manager who buys software for his / her staff
    - For what requirements will he / she pay? Which are trivial or are excessive?
  - Must be an active participant in the project (or have an active representative if there are many customers)
    - Sometimes a marketing person or experienced manager serves as a proxy

Stakeholders: Users

- *Users* (of both the current and future systems)
  - Experts on the existing system — tell us which features to keep and which need improvements, or
  - Experts on competitors' products — give suggestions about how to build a superior product.
  - May have special needs or requirements to be satisfied, e.g., regarding usability, desired features.
  - May want to consult special-interest groups: users with disabilities, users who have computer phobias, expert users, novice users, etc.

Stakeholders: Domain expert

- *Domain experts* know the problem domain well
  - Familiar with the problem that the software must solve

    e.g., financial experts for financial packages,
    aeronautical engineers for aircraft navigation systems,
    meteorologists for software that models the weather,
    travel agents for travel industry, etc.
  - … and are familiar with typical users and their expectations
  - … and are familiar with typical deployment environments

Stakeholder: Software engineer

- *Software engineer* == technology expert
  - May include managers too
  - Represents the rest of the development team (developers, testers)
  - Ensures that the project is technically and economically feasible
  - Accurately estimates the cost and development time of the product
  - Educates the customer about innovative hardware or software technologies, and recommends new functionality that takes advantage of these technologies

Other stakeholders

- *Inspectors* == experts on government and safety regulations
  - Familiar with government and safety regulations relevant to project

    e.g., safety inspectors, auditors, technical inspectors, government inspectors

- *Market researchers*
  - Experts who have conducted surveys to determine future trends and potential customers' needs.
  - May assume the role of client, if the software is being developed for the mass market and there is no identifiable customer.

Other stakeholders

- *Lawyers*
  - Familiar with legal requirements + relevant standards
  - Familiar with licensing, e.g., for use of open source components

- *Experts on adjacent systems*
  - Know about the interfaces to adjacent systems, and any special demands for interfacing with the product
  - May also have an interest in the product's functionality,

    e.g., if the product can help the adjacent system do its job better, by providing information it can use.

- *Value-adders*
  - People who build on our product

Why is elicitation hard?

Why is elicitation hard?

- Requirements are hard to articulate
- Human element
- Lack of imagination; may be fixed on one solution
- Lack of user motivation; resistance to change
- Demands may evolve
- Many varied sources of requirements
- Information, key stakeholders may not be available
- Hard to judge the quality of the result, or to judge when done
- Involves compromises and setting priorities (negotiation)
- Tension between developers (want everything set in stone ASAP) and stakeholders (want to hold on to flexibility as long as possible)

Why is elicitation hard? [according to Dan Berry]

- It's common to use a physically violent metaphor for elicitation, like extracting a painful tooth or trawling for tuna.
  - The implication is that drawing out the requirements is simply a matter of asking the right questions or taking appropriately strong measures from ignorant or recalcitrant stakeholders.
  - In practice, the real requirements (the ones you eventually decide that are the right ones) are usually ill-conceived and poorly understood by *everyone* at the start.
- It is through the acts of *discussing, analyzing, formalizing, reviewing, negotiating over, inventing, synthesizing,* and *explicitly documenting* that the various stakeholders come to a mutual understanding and agreement about what the real requirements are (and are not).

Elicitation techniques

Artifact-based
- Document studies
- Similar companies
- Norms
- Domain analysis
- Requirements taxonomies

Model-based
- Scenarios
- Modelling
- Analysis patterns
- Mockups & prototyping
- Pilot experiments

Stakeholder-based
- Stakeholder analysis
- Questionnaires
- Interviews
- Observation
- Task Demo
- Ask suppliers
- Domain workshop

Creativity-based
- Brainstorm
- Focus groups
- Design workshop

Artifact-based elicitation

- *Document studies*
- *Similar companies*
- *Norms*
- *Domain analysis*
- *Requirements taxonomies*

Artifact-based

- Scenarios
- Modelling
- Analysis patterns
- Mockups & prototyping
- Pilot experiments
- Stakeholder analysis
- Questionnaires
- Interviews
- Observation
- Task Demo
- Ask suppliers
- Domain workshop
- Brainstorm
- Focus groups
- Design workshop

Artifact-based elicitation

- *Idea:* Learn as much as we can by studying existing docs before asking for stakeholders' time.
  - System documentation
    e.g., existing requirements specifications, design documents, bug reports, change requests user manuals, work procedures, usage statistics, marketing data, performance figures
  - Environment documentation
    e.g., organization charts, business plans, policy manuals, financial reports, minutes of important meetings
  - Domain analysis
    e.g., Wikipedia (!!), textbooks, surveys, standards and regulations governing this domain

Norms

- General form of the use of a *norm* to state requirements:

Build a better X.

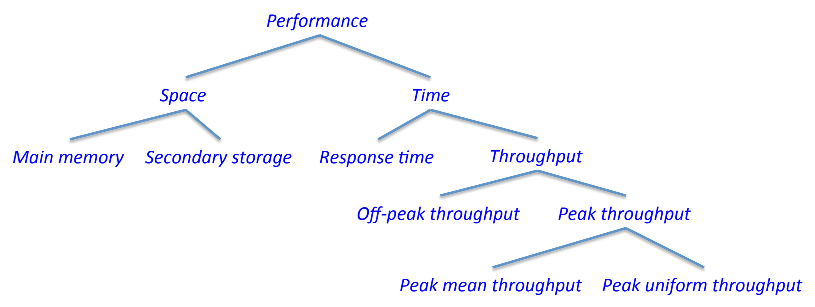
e.g., *"Build a better Facebook that allows users to encrypt their chat."*

- The norm can protect us from colossal blunders by starting with something that's clearly feasible … but there are dangers:
  - Norms can keep us from seeing a new way to solve the problem that X is solving by keeping us immersed in enhancing X.
  - Different people's perception of the norm may be different.

Requirements taxonomies

- *Requirements taxonomy* — classification of requirements; the classification can act as a checklist of details to be elicited.

e.g., Domain-*independent* taxonomy for performance-related NFRs


Performance
Space
Time
Main memory
Secondary storage
Response time
Throughput
Off-peak throughput
Peak throughput
Peak mean throughput
Peak uniform throughput