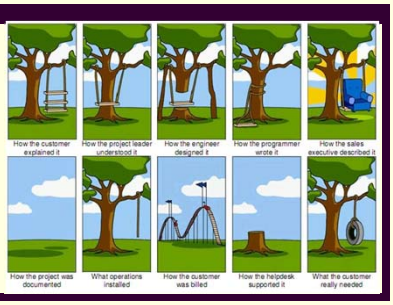
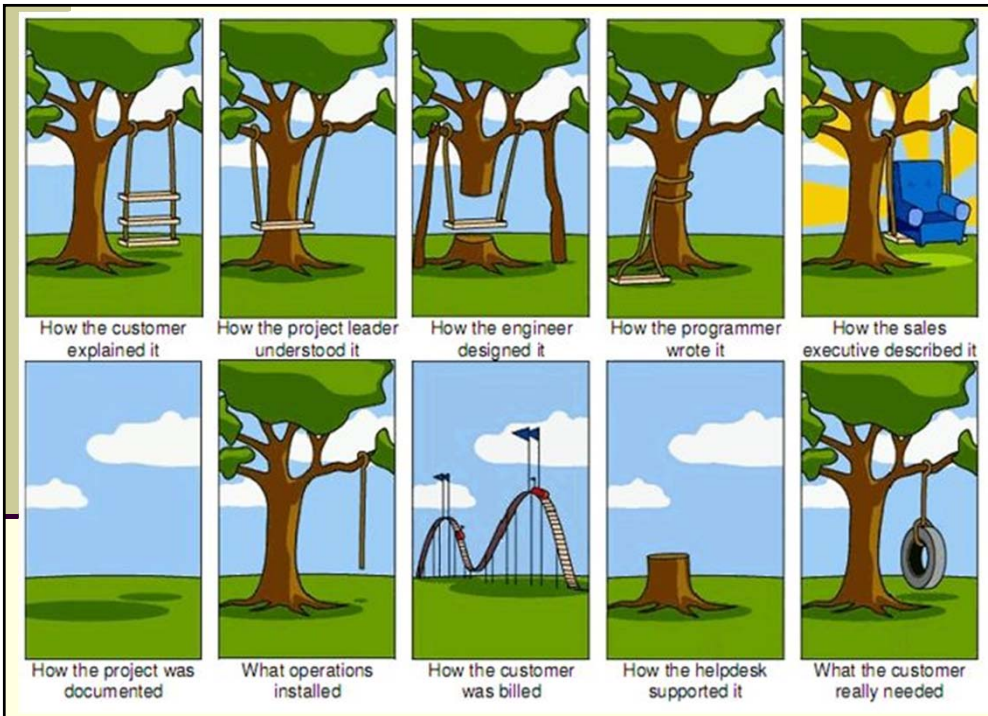
CS 351
Requirements Engineering

Instructor: **Joel Castellanos**
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/



4/13/2017



Designing Large Programs: Essence & Accidents

"The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements . . . No other part of the work so cripples the resulting system if done wrong. No other part is as difficult to rectify later" [Fred Brooks "No Silver Bullet: Essence and Accidents of Software Engineering." *Computer 20*, 4 (April 1987): 10-19].

"The inability to produce complete, correct, and unambiguous software requirements is still considered the major cause of software failure today" [Dorfman, M. & Thayer, R. H. *Software Requirements Engineering*. Los Alamitos, CA: IEEE Computer Society Press, 1997.].

3

Requirements Definition

The Institute of Electrical and Electronics Engineers (IEEE) defines a requirement as:

1) A condition or capability needed by a user to solve a problem or achieve an objective.
2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3) A documented representation of a condition or capability as in definition 1 or 2.

[Standard Glossary of Software Engineering Terminology (IEEE Std 610.12-1990)]

4

Requirements Engineering

Requirements engineering emphasizes the use of ***systematic*** and ***repeatable*** techniques that ensure the completeness, consistency, and relevance of the system requirements.

Requirements elicitation is the process of discovering, reviewing, documenting, and understanding the user's needs and constraints for the system.

Requirements analysis is the process of refining the user's needs and constraints.

Requirements specification is the process of documenting the user's needs and constraints clearly and precisely.

Requirements verification is the process of ensuring that the system requirements are complete, correct, consistent, and clear.

5

Roles in Requirements Development

Requirements engineering is complex because of the three roles involved in producing even a single requirement:

1) The requestor (a.k.a. "user" in the IEEE definition),
2) The developer (who will design & implement the system),
3) The author (who will document the requirements).

Typically, the requestor *imperfectly* understands the problem to be solved by the system but not how to develop a system. The developer *imperfectly* understands the tools and techniques required to construct and maintain a system but not the problem to be solved.

The author needs to create a statement that communicates unambiguously to the developer what the requestor desires. Hence, requirements address a fundamental communications problem.

6

Requestor Stakeholders

Typically, ***the requestor*** is a composite of multiple stakeholders:

Executives (who need to know the organization's business goals and constraints).

End Users (who need to know how the products will be used).

Marketers (who need to know the market demands).

Stakeholders can potentially include ***legal experts (especially***, ***government agencies***, and ***insurance experts***.

Often there are conflicts between the diverse needs of the requestor stakeholders. Requirements development must incorporate tradeoffs to resolve these conflicts.

7

Requirements are Pervasive

Requirements continuously affect all development and maintenance phases of a system's development by providing the primary information needed during them.

The requirements form a trigger mechanism for the development and maintenance efforts.

Testing, for instance, depends on a precise statement of quality and behavioral requirements to define the standard of correctness against which to test.

The longer the system's lifetime, the more it is exposed to changes in the requirements that result from changes in the needs and concerns of its stakeholders.

8

Agile Software Development Process

- **Understand What the Customer Wants**
  - Identify a purpose.
  - Identify givens: constraints, boundaries, standards.
  - Identify high-level goals.
  - Identify stakeholders.
  - Cost-Benefit Analysis.
  - Risk assessment
  - Requirements List
  - Proof of Concept
- **Build a Usable, Functional Application**
- **Deployment**

9

Proof of Concept (POC)

A **proof of concept** (also called **proof of principle)** is a realization of a certain method or idea to demonstrate its feasibility.

A proof of concept is usually small and may or may not be complete.

Proof of concept is documented evidence that a potential product or service can be successful.

10

Proof of Concept Goals

- Demonstrate to the customer what the developer thinks has been asked for.
- Give the customer a chance to try out a sample with enough functionality for the customer to be sure that what has been asked for is:
- What is wanted and
- Will be useful.
- Demonstrate that the developer has the skills to get the job done.

11