CS 4530: Fundamentals of Software Engineering
Module 1.2: Capturing User Requirements

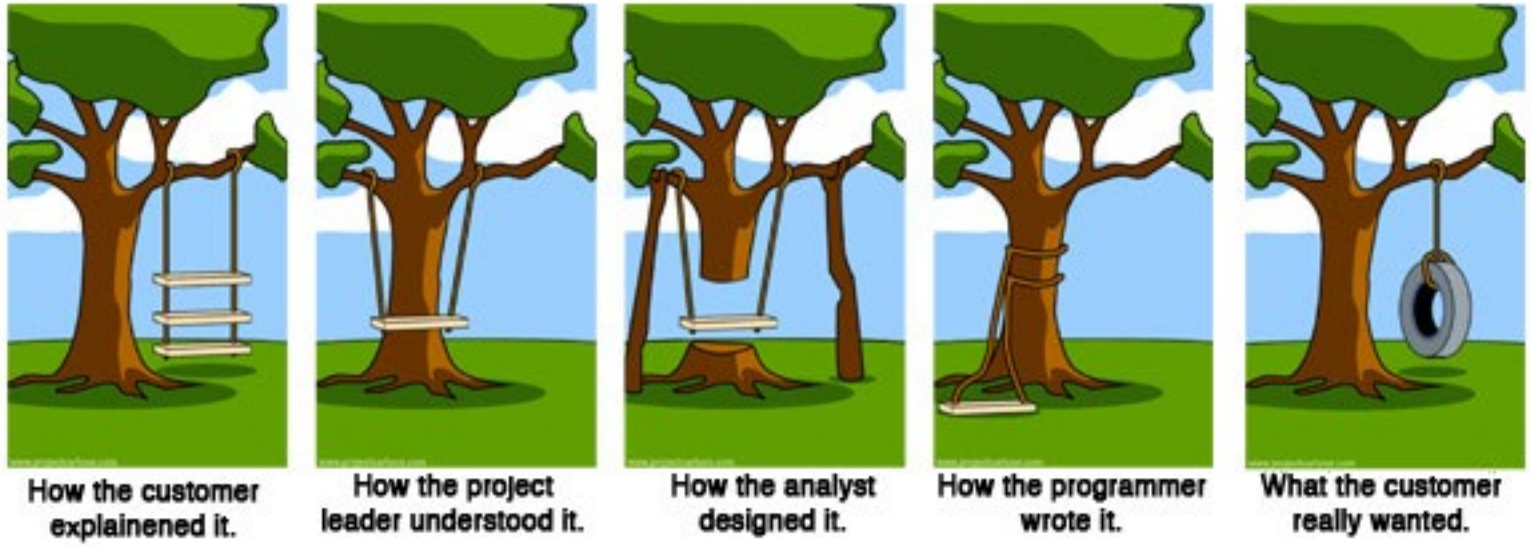
Jonathan Bell, Adeel Bhutta, Mitch Wand

Khoury College of Computer Sciences

Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Explain the overall purposes of requirements analysis
  - Enumerate and explain 3 major dimensions of risk in Requirements Analysis
  - Explain the difference between functional and non-functional requirements, and give examples of each
  - Explain the notion of a user story, with examples. (including conditions of satisfaction)

Overall question:
How to make sure we are building the right thing



Requirements Analysis

Planning & Design

Implementation

Why is requirements analysis hard?

Problems of understanding

Do users know what they want?

Do users know what we don't know?

Do we know who are users even are?

Problems of scope

What are we building?

What non-functional quality attributes are included?

Problems of volatility

Changing requirements over time

Soliciting Requirements

Option 1: Users tell developers what they want

- Client determines the problem and the solution
- Requirements might be formally provided in the form of a contract or statement of work
- Client might provide all requirements, or just some subset (e.g. “must be HIPPA compliant”)

Soliciting Requirements

Option 2: The developers try to figure out what the user really wants or needs.

- Interview users, ask questions about their problems, propose potential solutions, examine those solutions
- Embed your client in your design team, or better yet, become an anthropologist in your client's environment
- Build requirements documents that demonstrate your understanding of the requirements, iterate
- Empowers your team with credibility and authority

Always need to document the requirements

- Documentation helps our whole team make sure they are building the right thing
- Documentation can help specify implicit requirements
- Documentation can also serve as an artifact to iterate on with a client

Documentation should also capture non-functional requirements

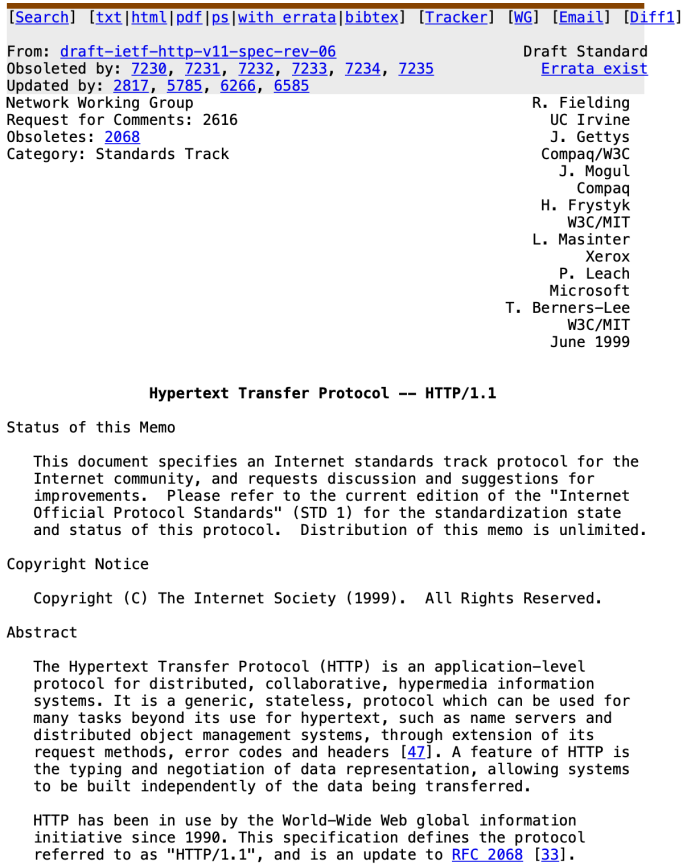
- Qualities that reflect the execution of the system
  - Accessibility
  - Availability
  - Capacity
  - Efficiency
  - Performance
  - Privacy
  - Response Time
  - Security
  - Supportability
  - Usability
- Example: “A 4-core server with 16 GB RAM should be able to service at least 200 simultaneous clients with less than 300ms latency”

Documentation should also capture non-functional requirements (2)

- Qualities that reflect the evolution of the system
  - Testability
  - Maintainability
  - Extensibility
  - Scalability
- Example: “A 3rd party component built conforming to the API defined in the Canvas LMS specification can create, modify, and delete assignments on behalf of an authenticated user”

Formal Specifications is one way to document the requirements

- Define all expected behaviors under all expected conditions
- Works best when domain is well-understood



[Search] [txt|html|pdf|ps|with errata|bibtex] [Tracker] [WG] [Email] [Diff1]

From: draft-ietf-http-v11-spec-rev-06 Draft Standard
Obsoleted by: 7230, 7231, 7232, 7233, 7234, 7235 Errata exist
Updated by: 2817, 5785, 6266, 6585
Network Working Group R. Fielding
Request for Comments: 2616 UC Irvine
Obsoletes: 2068 J. Gettys
Category: Standards Track Compaq/W3C
J. Mogul
Compaq
H. Frystyk
W3C/MIT
L. Masinter
Xerox
P. Leach
Microsoft
T. Berners-Lee
W3C/MIT
June 1999

Hypertext Transfer Protocol -- HTTP/1.1

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers [47]. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.

HTTP has been in use by the World-Wide Web global information initiative since 1990. This specification defines the protocol referred to as "HTTP/1.1", and is an update to RFC 2068 [33].

1.2 Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [34].

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocols it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for its protocols is said to be "unconditionally compliant"; one that satisfies all the MUST level requirements but not all the SHOULD level requirements for its protocols is said to be "conditionally compliant."

1.3 Terminology

This specification uses a number of terms to refer to the roles played by participants in, and objects of, the HTTP communication.

connection
A transport layer virtual circuit established between two programs for the purpose of communication.

message
The basic unit of HTTP communication, consisting of a structured sequence of octets matching the syntax defined in section 4 and transmitted via the connection.

request
An HTTP request message, as defined in section 5.

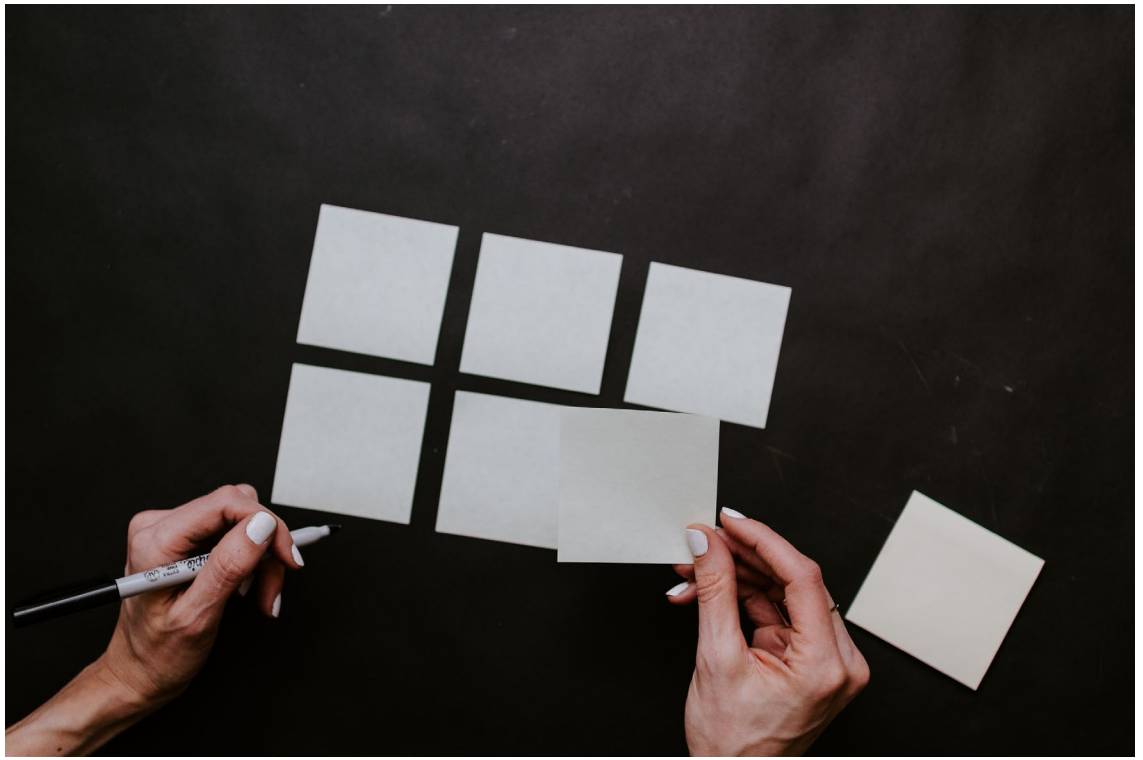
response
An HTTP response message, as defined in section 6.

User Stories document requirements from a *user's* point of view

Specifying what should happen, for whom, and why

As a <role> I can <capability>,
so that <receive benefit>

Conditions of Satisfaction:
Given <interaction with software, state of environment>, I expect <behavior and side effects>



Writing User Stories: INVEST

- Independent
- Negotiable
- Valuable (has value to client)
- Estimatable
- Small
- Testable

As a <role> I can <capability>,
so that <receive benefit>

Example: a Transcript database User Story

- User story: tells what the user wants to do, and why.
- Example:

As a College Administrator, I want a database to keep track of students, the courses they have taken, and the grades they received in those courses, so that I can advise them on their studies.

Satisfaction Conditions

- Satisfaction Conditions list the capabilities the user expects, in the user's terms.
- Example:

My database should allow me to do the following:

- **Add a new student to the database**
- **Add a new student with the same name as an existing student.**
- **Retrieve the transcript for a student**
- **Delete a student from the database**
- **Add a new grade for an existing student**
- **Find out the grade that a student got in a course that they took**

Non-Functional Requirements:

- What other properties might a customer want to know about the product?
  - How quickly can a transcript be retrieval? (Performance)
  - How many student transcripts can our system store? (Scalability)
  - How long did I spend on the phone with support to set up the software? (Usability)
  - After my system is setup, is the access controlled at all? (Security)
  - Are these any times when I can’t use this system? (Availability)

Requirements: Which to pick?

- There are four knobs you can adjust when negotiating requirements:
  - Project scope
  - Project duration
  - Project quality
  - Project cost
- Usually cost is most constrained: you have a budget to spend, and you have a headcount of developers to pay
- Determining feasible scope, timeline and maximizing quality is the subject of much software engineering research, see next lesson

Learning Goals for this Lesson

- At the end of this lesson, you should be able to
  - Explain the overall purposes of requirements analysis
  - Enumerate and explain 3 major dimensions of risk in Requirements Analysis
  - Explain the difference between functional and non-functional requirements, and give examples of each
  - Explain the notion of a user story, with examples. (including conditions of satisfaction)