CS 4530: Fundamentals of Software Engineering Module 1.2: Requirements and User Stories

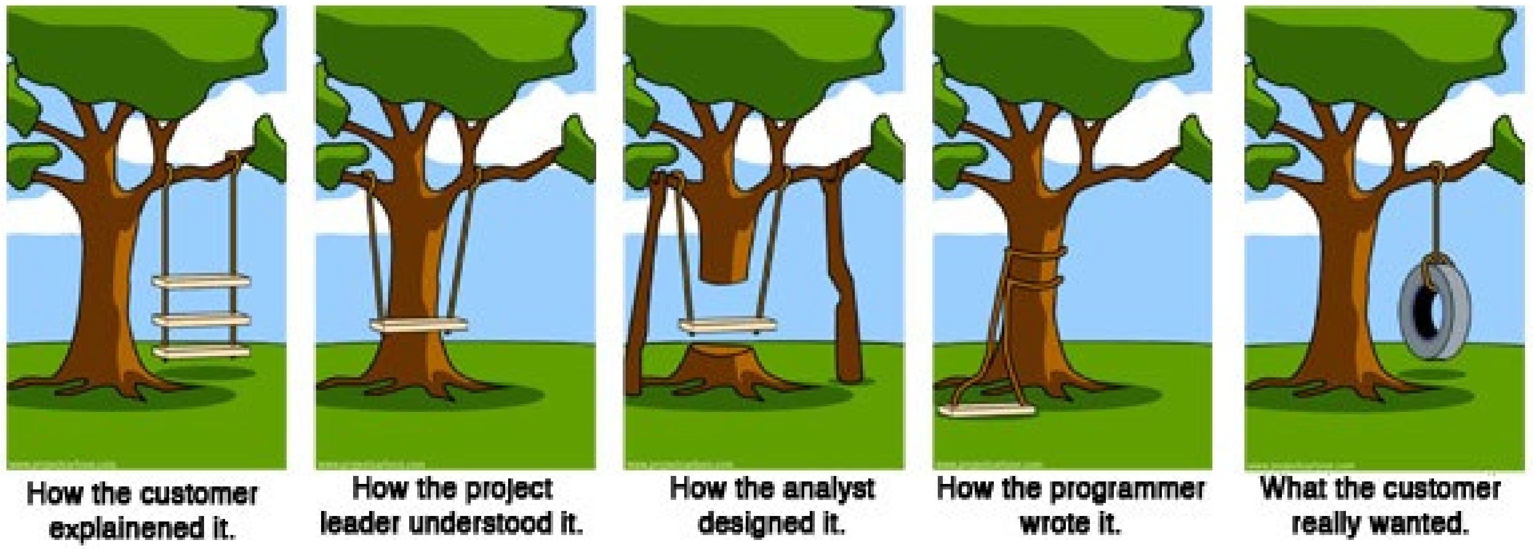
Adeel Bhutta, Joydeep Mitra and Mitch Wand

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
  - Use Value Sensitive Design to uncover requirements based on the different kinds of investigations.
  - Explain the difference between functional and non-functional requirements, and give examples of each
  - Define the notions of user stories and conditions of satisfaction, and give multiple examples

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


How the customer explainened it.
How the project leader understood it.
How the analyst designed it.
How the programmer wrote it.
What the customer really wanted.

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?

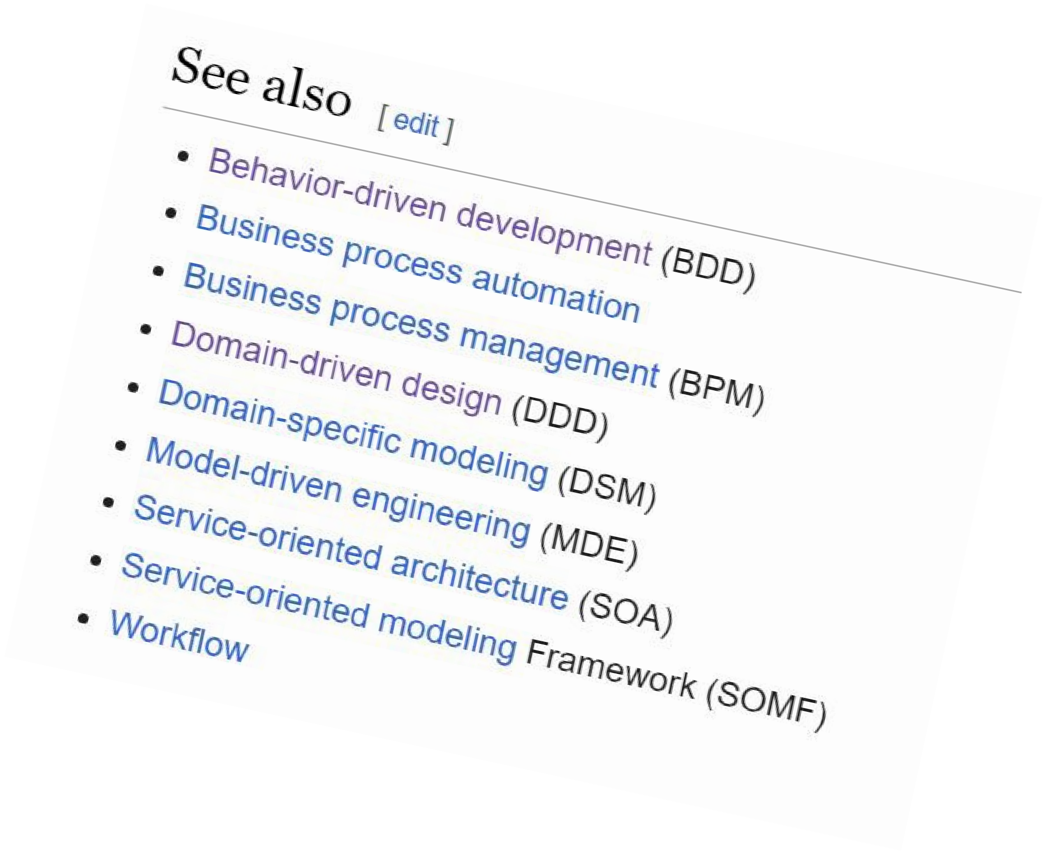
Problems of volatility

Changing requirements over time

How the customer explainened it.
How the project leader understood it.
What the customer really wanted.

How do we capture the requirements?

- There are many methodologies for this.
- Often described as x-Driven Design (for some x)
- They differ in scope & details, but they have many features in common.



See also [edit]

- Behavior-driven development (BDD)
- Business process automation
- Business process management (BPM)
- Domain-driven design (DDD)
- Domain-specific modeling (DSM)
- Model-driven engineering (MDE)
- Service-oriented architecture (SOA)
- Service-oriented modeling Framework (SOMF)
- Workflow

Common Elements

1. Meet with stakeholders
2. Develop a common language
3. Collect desired system behaviors
4. Document the desired behaviors
5. Iterate and refine!!

Different Methodologies Produce Different Forms of documentation

TDD: executable tests

BDD: "scenarios"

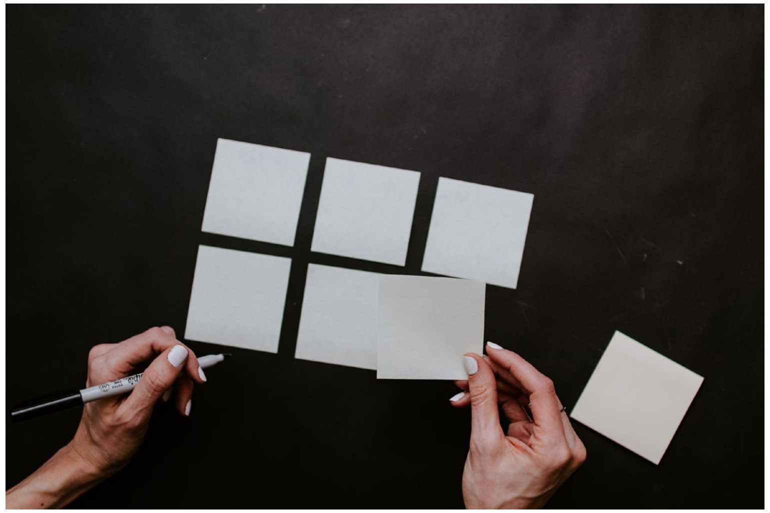
DDD: an OO architecture

We'll use a least-common-denominator approach: user stories

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

As a <role> I want
<some capability>
so that I can <get some benefit>

User stories specify what should happen, for whom, and why



Properties of a user story

- short: fits on a 3x5 card
- may have prerequisites
- has *conditions of satisfaction* that expand on the details
- has a priority
- satisfies the INVEST criteria (more on this later)

Examples:

- As a online blackjack player, I want a game of blackjack implemented so that I can play blackjack with other users. (Essential)
- As a citizen, I want to be able to report potholes so that the town can do something about them. (Essential)
- 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. (Essential)

Conditions of Satisfaction fill in details of the desired behavior

- Each condition of satisfaction
  - Describes a testable behavior, from the user's point of view
  - Must have a priority
  - Should be numbered within its user story

Examples

- 1.1 There should be an accessible blackjack table (Essential)
- 1.2 A user can initiate a game of blackjack (Essential)
- 1.3 Users can enter a blackjack table as a player if no other player is currently occupying the slot (Essential)
- 1.4 Players can successfully hit (take a card) each turn (Essential)
- 1.5 Players can successfully stand (refrain from taking a card) each turn (Essential)
- 1.6 Players successfully win if the dealer goes above 21 before me (Essential)

Priorities

- **Essential** means the project is useless without it.
- **Desirable** means the project is less usable without it, but is still usable.
- **Extension** describes a user story or COS that is may not be achievable within the scope of the project. These might be things you'd want "in the next version".

Minimum Viable Product

- The set of essential user stories constitutes the minimum viable product (MVP)
- A user story is "implemented " when all its essential COSs are implemented.
- Caution: when proposing a project, don't make your MVP too hard to complete (but don't make it too easy, either)

The MVP and Your Project Grade

- On your project, you will get 200 points (out of a total of 400) for code submission:
  - MVP (all essential user stories and their essential COSs delivered): 100 points
  - Extra features (desirable and/or optional features): 50 points
  - Testing: 50 points
- SO: be realistic about what you call "essential" ☺

Value Sensitive Design (VSD) is an ethical Framework to write better User Stories

- The design of technologies **should** reflect and affect human values
- Ignoring values in the requirements and design process is **irresponsible**.
- A Human Value Example: **Informed Consent**
  - Most websites collect vast amounts of information about users, who have no control over: what information is collected/accessed/used/sold/etc.
  - A VSD approach to informed consent would emphasizes **transparency, user control and understanding**.
  - Web *cookies* and *browser security* mechanisms represent solutions to implement the principle of informed consent.
- To consider human values during design, we need to understand ***what*** they means generally and in the specific ***context*** of technology

Value Sensitive Design (VSD) in Brief

VSD is a(n) . . .

Outlook for seeing the values in technology design

Process for making value-based choices within design

- Combines **empirical**, **value**, and **technical** investigations
- Design solutions that incorporate the values held by **stakeholders**
- Considers problems and solutions from **diverse perspectives**

VSD is not . . .

A moral framework or system of ethics

- It does not prescribe decisions to make
- It incorporates values reflections in the choosing process

It is not an algorithm for making decisions

- Often, there are no easy answers
- Takes sustained commitment

Example: The Reddit Case Study

A classifier model that will replace the role of humans in the moderating process. This algorithm will classify new posts, determining whether or not they’re appropriate for Reddit. Those that are flagged inappropriate will be removed.

Empirical Investigation - Development of the Model

Training data has two primary elements:

- Posts that Reddit users have flagged previously
- A dataset of English words that would be flagged as inappropriate

- Reflect on the sources of data
  - Is the dataset representative of the language we want removed?
  - Are there any sources of biases or disparities that in this data that we should be considering?
- Complications
  - Given the contextual nature of offensive speech, what complications or problems can arise from this model?

Value Investigation – Who are the Stakeholders?

1. Relative to the issue of content moderation, who or what are the stakeholders? (i.e. individuals or groups whose interests stand to be impacted by this algorithm?)
2. Relative to the issue of content moderation, what are the interests or values of the different stakeholders?
3. Are there any conflicts of interests or values?
4. Overall, given the different stakeholders, interests and values, do any of them stand out to you as ones we should prioritize? Why?

Value Investigation – What are the value tensions?

1. Why do you think some people might be concerned with Reddit removing user’s posts? What if the posts have misinformation or hate speech?
2. How are users harmed if posts are mistakenly removed by Reddit?
3. How can bias in the moderation algorithms potentially harm users?
4. Given the debate on free speech vs. content moderation what do you think are the strengths and weaknesses of the proposed requirement?

Technical Investigations

- Suppose as a result of the value investigations we came up with two high-level requirements:
  1. The flagging feature gives an explanation to the user as to why the system flagged and removed their content. If, after reading the explanation, the user thought the flagging was in error, they can submit the flag for further review.
  2. The system prompts the user to reconsider (does not prevent) if their post is potentially offensive.
- Which requirement would you prefer based on the *technical feasibility and the values* that you think are important?

Using VSD to Define User Stories

- Formally specify the chosen requirement as user stories.
- Your user stories must capture the following:
  - Who are the stakeholders?
  - What are the values?
  - What value tensions were resolved?
  - What are the conditions of satisfaction?
- Download the detailed instructions of the activity from the course website (add link).

Yet another example: a University Transcript database

User Story

- 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

The database should allow me to:

1. Add a new student to the database
2. Add a new student with the same name as an existing student.
3. Retrieve the transcript for a student
4. Delete a student from the database
5. Add a new grade for an existing student
6. Find out the grade that a student got in a course that they took

Non-Functional Requirements capture the *quality goals* of the system:

- As developers, we often spend most of our time and effort on features (i.e., functional requirements).
- But there is more ….
- 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)

Example:

- “With a 4-core server and 16 GB RAM, the system should be able to service at least 200 simultaneous clients with less than 300ms latency”

Other non-functional requirements

- Accessibility
- Availability
- Capacity
- Efficiency
- Performance
- Privacy
- Response Time
- Security
- Supportability
- Usability

Still more non-functional requirements

- Qualities that reflect the evolution of the system
  - Testability
  - Maintainability
  - Extensibility
  - Scalability

Writing User Stories: INVEST

- Independent
- Negotiable
- Valuable (has value to client)
- Estimable (able to estimate development effort)
- Small
- Testable

As a <role> I want <capability> so that I can <get some benefit>

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
  - Use Value Sensitive Design to uncover requirements based on the different kinds of investigations.
  - Explain the difference between functional and non-functional requirements, and give examples of each
  - Define the notions of user stories and conditions of satisfaction, and give multiple examples