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

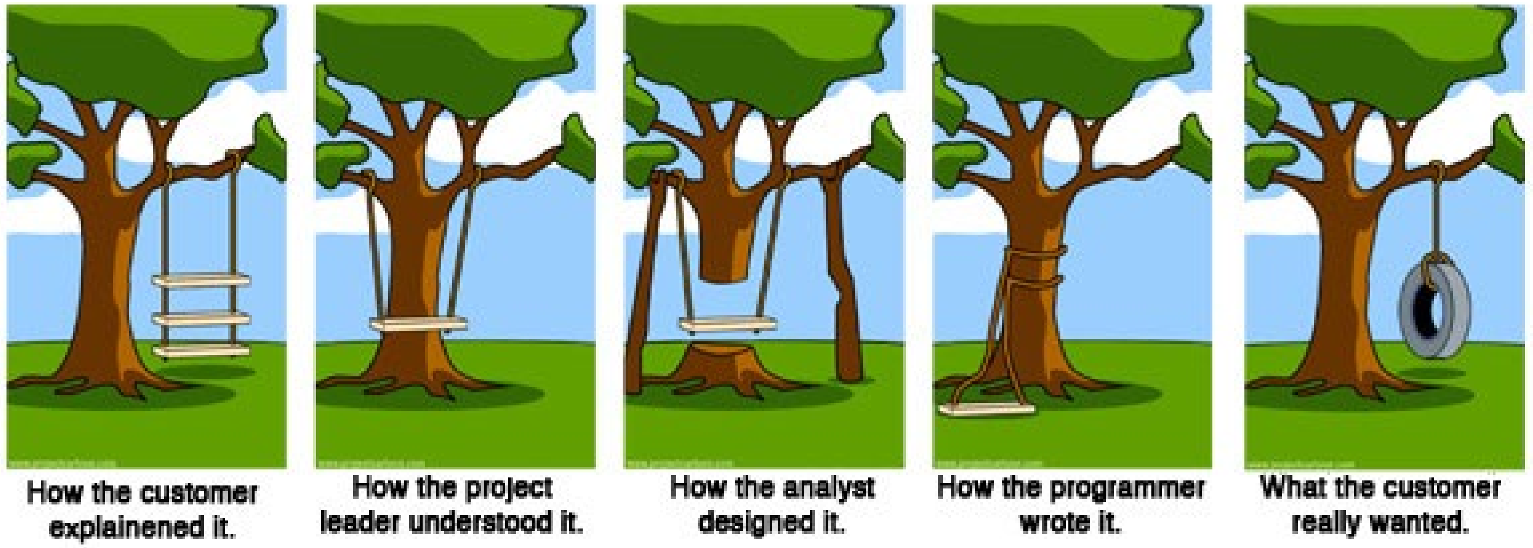
Adeel Bhutta, Joydeep Mitra and Mitch Wand

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.
  - document requirements user stories.
  - track the completion of requirements using conditions of satisfaction.
  - explain the difference between functional and non-functional requirements.
  - Explain how Value Sensitive Design can be used to improve requirement gathering.

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

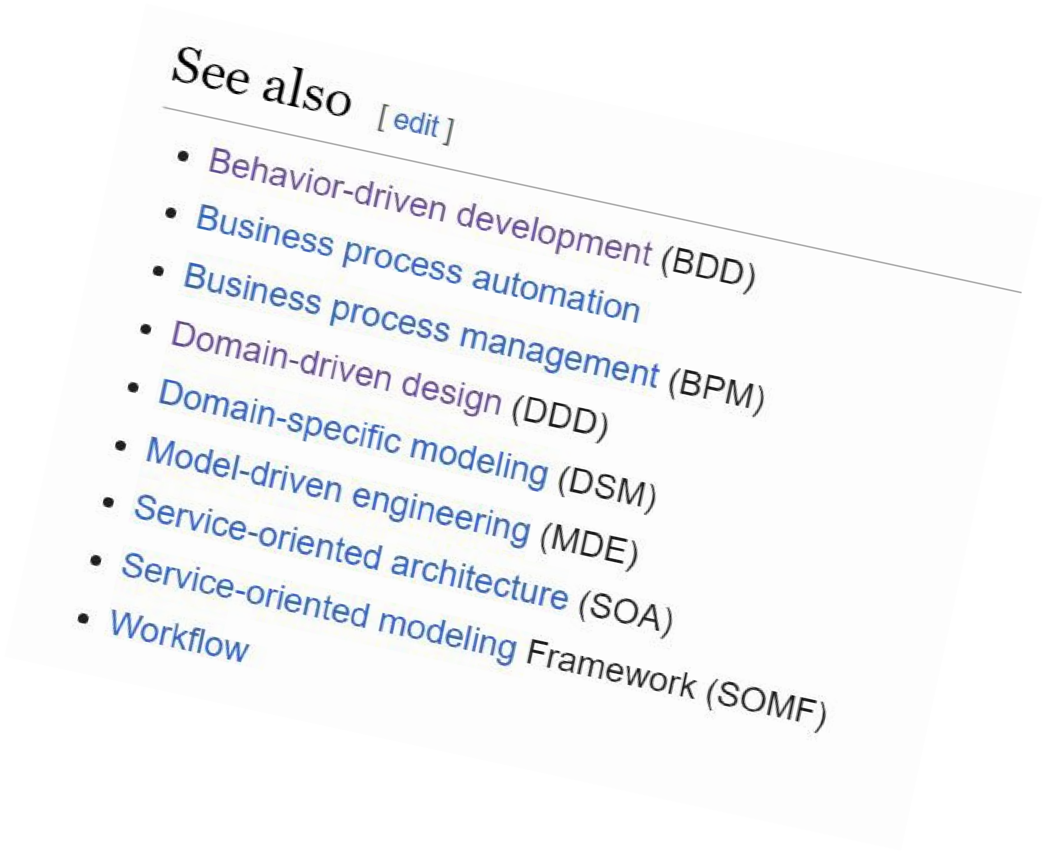
How the customer explainened it.

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 that offer value
4. Document the desired behaviors
5. Iterate and refine!!

Different Methodologies Produce Different Forms of documentation

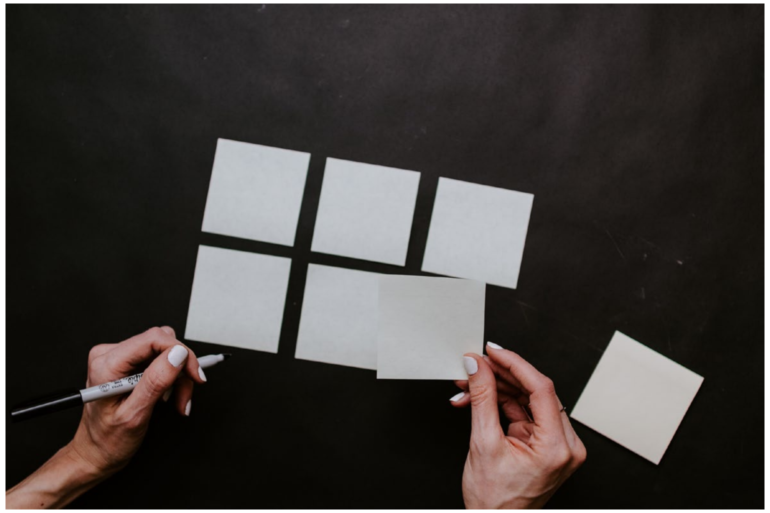
- TDD: executable tests
- BDD: "scenarios"
- DDD: an OO architecture

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" ☺

There are several frameworks that help us in requirement gathering

- One framework recommends paying special attention to **stakeholders** and **human values**
- It allows us to reason about design choices and choose the values that are important
- Considering human values in the design process is considered a **responsible** approach. Some call it ***ethical***
- Its official name is Value Sensitive Design (VSD)

Writing User Stories with VSD (Example: Informed Consent in Stack Overflow)

Stakeholders identified:

- Site users
- Site owners

Values identified:

- Privacy
- Accountability
- Transparency

User Story Examples:

- ❖ As a Stack Overflow user, I want to clearly understand what personal data Stack Overflow collects through cookies and control which cookies are set so that I can make an informed choice about my privacy while still accessing the programming help I need. (Essential)
- ❖ As a Stack Overflow site owner, I want to implement transparent cookie consent processes that comply with regulations while maintaining user engagement so that I can build user trust, avoid legal penalties, and sustain my business model without losing essential functionality. (Essential)

Conditions of Satisfaction and VSD (Example: Informed Consent in Stack Overflow)

User Story and COS:

- As a Stack Overflow user, I want to clearly understand what personal data Stack Overflow collects through cookies and control which cookies are set so that I can make an informed choice about my privacy while still accessing the programming help I need. (Essential)
  - The system should display a clear cookie banner explaining what personal data is collected before any cookies are set (Essential)
  - Users should be able to access granular cookie controls to accept/reject specific categories (functional, analytics, advertising, personalization) (Desirable)
  - The system should provide a “privacy dashboard” showing what data has been collected about the user over time (Extension)

User Story and COS:

- As a Stack Overflow site owner, I want to implement transparent cookie consent processes that comply with regulations while maintaining user engagement so that I can build user trust, avoid legal penalties, and sustain my business model without losing essential functionality. (Essential)
  - Cookie data containing personal information should be encrypted both in transit and at rest to prevent unauthorized access. (Essential)
  - The system should automatically expire cookies based on predefined retention periods. (Essential)
  - The system should allow users to read and search posts even if they reject data collection on cookies. (Desirable)
  - The platform should implement automated data retention policies that permanently delete expired cookie data. (Extension)

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.
  - document requirements user stories.
  - track the completion of requirements using conditions of satisfaction.
  - explain the difference between functional and non-functional requirements.
  - Explain how Value Sensitive Design can be used to improve requirement gathering.