

# CS 4530: Fundamentals of Software Engineering

## Module 1.2: Requirements and User Stories

---

Adeel Bhutta and Mitch Wand  
Khoury College of Computer Sciences

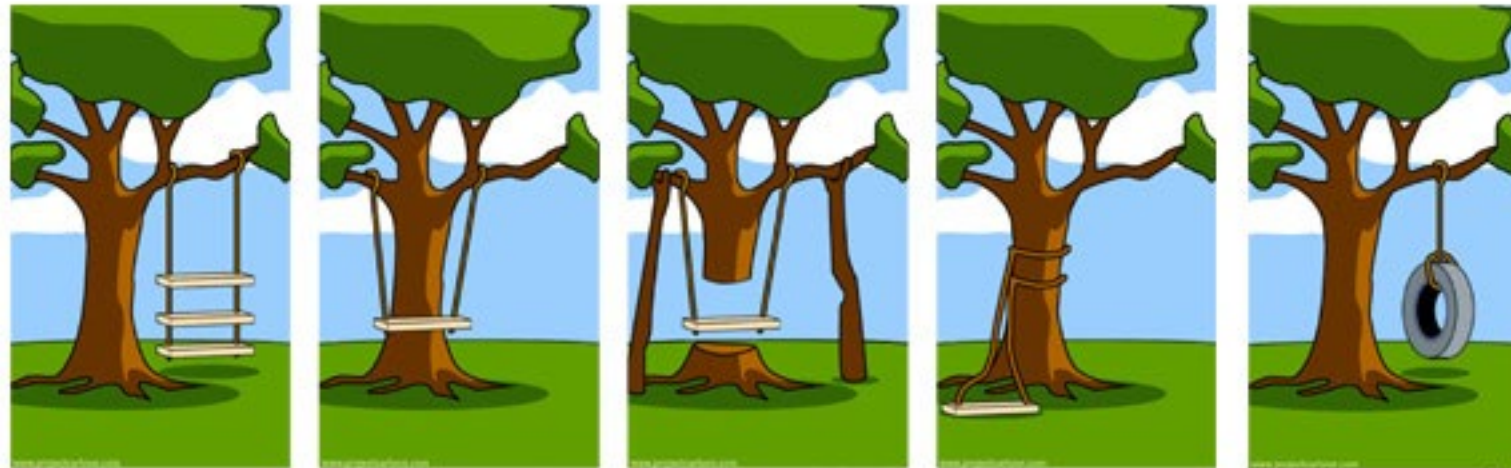
# Learning Goals for this Lesson

---

- 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
  - 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 explained 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 explained 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 <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 desirable, but may not be achievable within the scope of the project.

# Minimum Viable Product

---

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



# Another Example: a Pothole reporting system

- A town wants a system where citizens can report potholes and the town can monitor progress on repairing them.



# User Story #1

---

- As a citizen, I want to be able to report potholes so that the town can do something about them. (E)

# Conditions of Satisfaction

---

- 1.1 I should be able to report a pothole to the system (E)
- 1.2 I should be able to see whether the pothole I report has been repaired (E)
- 1.3 I should be able to see whether someone else has already reported a given pothole (D)
- 1.4 I should be able to see an estimated time when the pothole should be repaired (D)

## User Story #2

---

- As a repair-truck driver, I want the system to display the potholes I should be working on today. (E)

# Conditions of Satisfaction

---

- 2.1 I should be able to see my list of potholes for today (E)
- 2.2 I should be able to report that I repaired a given pothole (E)
- 2.3 I should be able to report that I was unable to repair a given pothole, and to supply a reason (E)
- 2.4 My daily list of potholes should be listed in an order that cuts down the time I spend driving from job to job (D)

## User Story #3

---

- As a maintenance supervisor, I want to be able to control the order in which potholes are repaired (D?)

# Conditions of Satisfaction

---

- 3.1 I should be able to give a higher priority to potholes on a particular street (E)
- 3.2 I should be able to give a higher priority potholes in a particular neighborhood (E)
- 3.3 I should be able to see on a map where there are a lot of potholes (D)
- 3.4 I should be able to see on a map which potholes that have been reported multiple times (D)



# 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:

---

- 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 there 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
  - 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