Software Engineering Introduction

CSCI 5828: Foundations of Software Engineering
Lecture 06

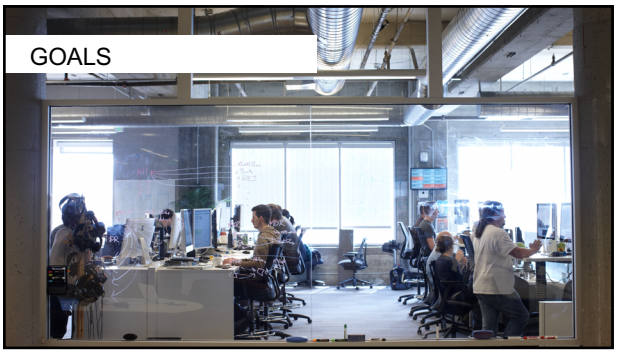
1

Agenda

- Product background
- Goals
  - Product
  - Business
  - Non-goals
- Risks
- Actors/activities
- Stories
- Estimation and prioritization

2

GOALS



3

What does success look like?

Financial success; you're making money, you're saving money

Adoption; people and/or systems are using your software

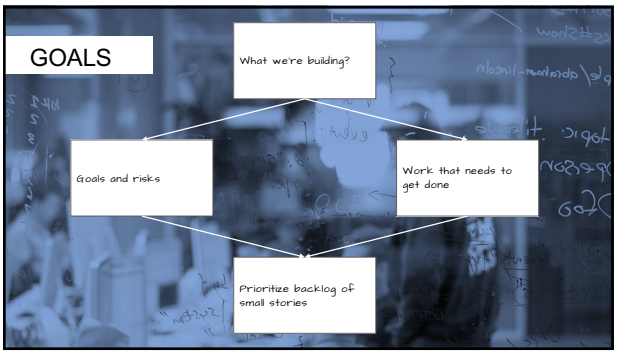
How do we ensure we're building the right software?

- iterate and adjust quickly based on user feedback (one of many techniques)

How do you iterate quickly while adjusting to feedback?

Small list of prioritized stories - but how do you get there?

4



5

Introduction to goals

Common goals

- Business goals
- Product goals
- Non-goals

What does a win look like?

6

Business Goals

Example business goals

- Grow revenue by 15% in one year
- Reduce operational costs by 50%
- 10% increase in user adoption in the first 3 months
- Land 3 new partnerships per quarter

Goals are typically measureable
Did you achieve your goal?

7

Product Goals

Example product goals

- Easy to use, easy to understand
- No product training required (Gmail for example)
- Support 200 concurrent users
- 500 millisecond response time

8

Non-Goals

Example Non-Goals

- Support for multiple languages
- Highly available
- Highly scalable
- Broad browser support

Think first release, and not building too much too early w/out feedback!

9

Why goals are important

Why should I care about goals?

- Helps bring product and engineering together
  - One set of common goals
  - Meet delivery timelines
- Important for engineers to understand what they're building
  - So engineers are focused
  - So that you could better prioritize a backlog of small stories
  - So engineers can push back on work that may be out of scope

10

Goals Exercise

Let's walk through a product description

As you listen add any goals that you can identify to this shared document.

+1 when you see a similar goal

11

Goals Exercise

Let's review

12

AVOIDING RISKS

13

Intro to Risks

What could keep us from accomplishing the goals?

- Competition; just another Twitter feature
- Too big of scope for current timeline and/or team size
- External dependencies; other teams, integrations (single sign-on)
  - Especially those that have not been built yet or dependent on other teams to build
- Choice of technologies; immature language, framework, etc
- Existing technical debt; no tests
- Federal regulations; PCI compliance

14

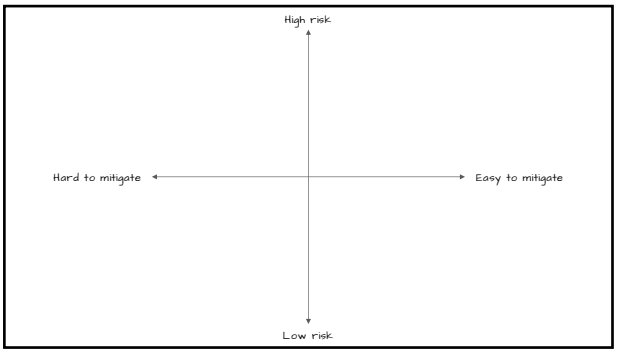
Why are identifying risks are important?

Why should I care about risks?

- May push out timeline
- May increase cost
- May have unrealistic dependency; tools, teams, etc

Because we may prioritize our work based on our ability to mitigate risks!

15



16

Risks Exercise

Write down 2-3 risks

+1 when you see a similar risk

Add any risks that you can identify to this shared document.

17

Tackling risks

How do we mitigate risks?

- Increase budget
- Increase timeline
- Reduce scope and deliver high-value business value first
- Reduce integrations
- Tackle risky areas early
- Create communication patterns to mitigate dependencies

18

19

User stories - lecture goals

Present an introduction to the topic of user stories

- concepts and terminology
- benefits and limitations

Present an introduction to iteration planning

- Estimating User Stories
- Planning a Release
- Planning an Iteration
- Measuring and Monitoring Velocity

20

Credit Where Credit is Due

This material is drawn from the below textbook -

"User Stories Applied" by Mike Cohn
Publisher: Addison-Wesley/Pearson Education ISBN-13: 978-0-321-20568-1

It's a great book for going in depth on the topic of user stories

21

User Stories

User stories are a means to **capture requirements** during the analysis phase of software development

- whenever that phase occurs during your particular software life cycle
- that said -- analysis can really happen at any time -- and it does in practice!!!

They are a **lightweight mechanism** for spreading decision making out across a software development project with respect to individual features

- We know we need feature X but we don't know much about it?
  - name it and put it in a user story
- We learned a little bit more about feature X today?
  - add a short note to the user story (or even better, write a test)

22

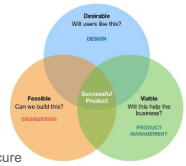
Background (I)

The notion of a user story evolved because capturing software requirements is a communication problem

- Those who want new software need to communicate what they need to those who will build it
- Many stakeholders will provide input to the process
  - customers, users, and domain experts
  - business and marketing
  - developers

23

Background (II)



If any group dominates this discussion, the whole project suffers

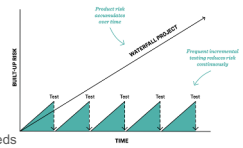
- if business dominates, it may mandate features and schedules with little regard to **feasibility**
- if the developers dominate, a focus on technology may obscure business needs and the developers may miss - is it **desirable** and **viable**

Furthermore, the **goal** is to understand the user's problem and ensure the software meets their needs

- both business and developers will move on, the users have to live with the developed software day in and day out

24

Background (III)

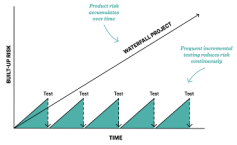


Furthermore, everything up front about the project is slightly impossible -

- We still don't understand **exactly** what the user needs
  - Their domain is complex; they are experts, we are novices
  - We'll get things wrong and need to be corrected
  - We'll get to a certain point and then they will remember things that they forgot to tell us
  - We'll show them prototypes and they'll come up with new ideas
- We don't have enough information to make accurate estimates
  - what we thought would be easy, turns out to be very complex

25

Background (IV)



And, so we have to make decisions based on the information we have

We set our **scope small** (one feature, for instance) and our **development life cycle short** (one week, for instance)

- and then we show the customer what we have

By then, new information will be available and we'll have **feedback on the work we've done so far**

- With that input, we identify the new scope and start a new iteration

We thus **spread out the decision making**
It's not "everything up front" but "a little at a time"

26

User Stories: The Basics (I)

That's where User stories come in; they describe functionality that will be valuable (desirable) to the user and/or customer

- Note the distinction:
  - **user**: the people who actually use the produced software in their work
  - **customer**: a person—not necessarily a user—who is responsible for purchasing the software for a set of users
  - sometimes they are one and the same, but not always

Note also the use of the word "valuable" or "desirable"

- We do NOT implement a feature because it is "cool"
  - we implement features to provide value to users

27

User Stories: The Basics (II)

User stories consist of

- a short written **description** of a feature used for planning and a reminder (140 characters)
- **conversations** about the feature used to flesh out its details
- software **tests** that convey details about functionality and help us determine when the story is completely implemented

Ron Jeffries calls these three aspects **Card, Conversation, and Confirmation**

- He says "card" because traditionally users stories are written on index cards and put up on a wall in the shared space of a development project
  - Using index cards forces you to keep the story brief!

28

User Stories: The Basics (III)

Example users stories for a website that helps a person's job search

- A user can post a resume to the website
- A user can search for jobs
- A company can post new job openings
- Users can restrict access to their resume

INVEST

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Important:

- User stories are written so that **customers value them**
  - This helps maintain a customer perspective within the development team

29

User Stories: The Basics (IV)

So, is this a good user story?

The software will make use of a bloom filter to determine if a desired data element is in our data set before we perform disk I/O to retrieve it

30

Not Really

Is your customer a distributed systems researcher?

- Then, yes, maybe, this might be a good user story

But, in general, technical details like this do NOT make good user stories

- These details may change
  - we need to switch from this framework to this other framework to be compatible on a wider range of devices
- while the fundamental user story does not change
  - **users need to access schedule information**

31

How do we track details?

The users stories for an application can often be written simply at a high level of abstraction (known as **epic user stories** or **epics** for short); for example:

- *A user should be able to **manage projects***

But, you need to specify details at a lower level of abstraction

- how do we do that?

A few places

- in the **conversations** around a user story; we will converge on details
- more users stories!

32

More users stories

You can take an epic like "*A user should be able to **manage projects***" and split it into new stories (aka CRUD)

- Create
- Read (List, Show)
- Update
- Delete

On the **epic**, you note that it's covered by these other stories and then you go work on those stories - the challenge: getting the balance right

- We want to resist the temptation to document everything on a user story
- Our conversations are the key element where details live
  (since the details **WILL change** while the user story remains the same)

33

Tests are integral to User Stories

At the start of a user story, the “tests” might exist as a set of customer expectations written on the back of a card

- Try feature with an **empty project description**
- Try feature with a **really long project description**
- etc.

In this form, the tests can come and go as we learn more about the feature

- As this particular user story is worked on and implemented
  - these expectations are transformed into unit tests and integration tests that tell us when the feature is completely implemented
    - we’re not done until all tests have passed!

34

Benefits

User stories provide the following benefits -

- They emphasize verbal rather than written communication
  - Placeholder for a conversation!
- They are comprehensible by customers and developers
- They are the right size for planning
- They encourage and “work” for iterative development
- They encourage deferring details until you have the best understanding of what you really need to implement a feature

35

Stories Exercise

Let’s write some stories

As you listen add any stories that you can identify to this shared document.

Look for duplicates!

36

Questions?

37

User Stories

INVEST

- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

38

User Stories

Look for activities and actions

- **"Assign an owner"** to a project
  - Add project owner
  - Remove project owner
- **"Assign contributors"** to a project
  - Add user to project
  - Remove user from project

39

User Stories

Typical story flow -

- Product owner **creates** work
- Engineers, designers, etc **start** work
- Engineers, designers, etc **delivery** work to a review environment
- Product owner **accepts** or **rejects** the work
- Story should be **restarted**
- Product owner pushes to production environment

40