Lecture Notes
October 6, 2020

Final individual project assignment “I3: Saving Program State” due Thursday night.
https://courseworks2.columbia.edu/courses/104335/assignments/482611

First assessment postponed from starting this Friday (October 9-12) to start instead the next Friday (October 16-19).

The team formation assignment is still due before class (9:59am) on Thursday October 15, but the deadline for the team preliminary proposal assignment has been extended.

Addendum to September 29 lecture (second page) - software liability

“Assignment T0: Team Formation” due October 15
https://courseworks2.columbia.edu/courses/104335/assignments/486855

Try to form a team of exactly 4 members. Contact me asap (via piazza, not email) if you’d like to form a team of 5 members. Two or three members is not acceptable.

(Except for CVN students: There are 7 CVN students, so it’s ok in your case to form one team of 4 and one of 3.)

Submit team members (full name and uni), team name, programming language(s) and platform(s), team github repository. If you want to use multiple languages and/or platforms, e.g., front end vs. back end, please explain. Everyone in the team must submit the same thing.

“Assignment T1: Preliminary Project Proposal” due October 27
https://courseworks2.columbia.edu/courses/104335/assignments/486922

Each team proposes their own project, within constraints:

- Must impose authenticated login.
- Must be demoable online (e.g., zoom or discord).
- Must store some application data persistently (database or key-value store).
- Must use some publicly available API beyond those that “come with” the platform. It does not need to be a REST API.

Describe Minimal Viable Product (MVP) both in prose - a few paragraphs - and via 3-5 “user stories”.

< label >: As a < type of user >, I want < some goal > so that < some reason >.

My conditions of satisfaction are < list of common cases and special cases that must work >.

Describe acceptance testing plan that covers these cases.

List the tech you plan to use. If different members of the team plan to use different tools, please explain.

Working in a team: *software requirements*

Specification documents describe requirements in immense detail, e.g., HTTP, URL

Detailed specifications are crucial for interoperability across multiple vendors (so browsers and web servers can talk to each other) and for safety-critical software (so it doesn't kill someone).

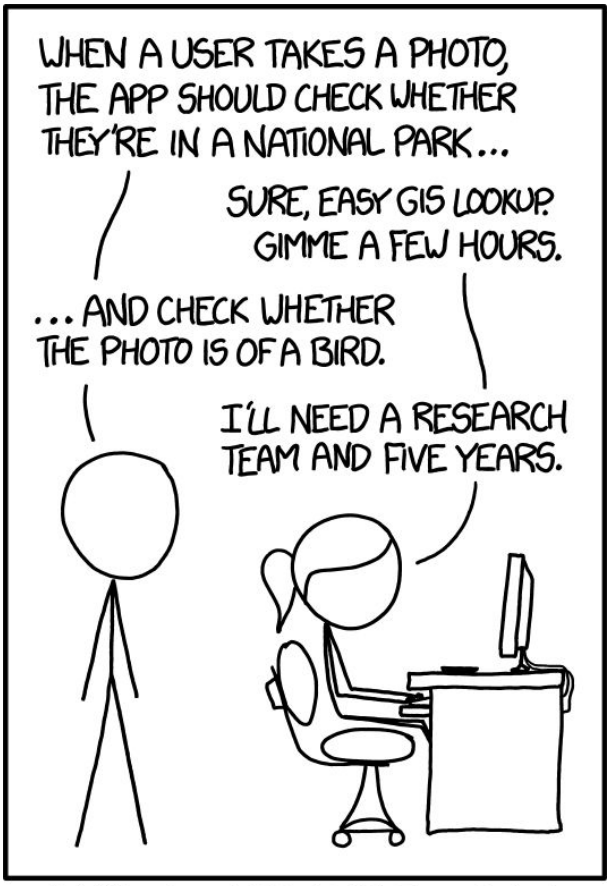
User stories and/or use cases are the agile way to describe customer requirements for consumer and business software. Customers have something in mind that they want the software to do, of *value* to them. This *something* is known as "requirements".

Even in a startup or other new product venture with no customers yet, someone ("product owner") has an *idea* about what will attract customers by providing value - and has a business model as to how this will enable paying for the development effort. Again known as "requirements".

Customer is not necessarily the same as user. If you are developing software to control a self-driving car, your customer might be Waymo or Tesla, but your primary user is the person sitting behind the wheel of the car. That user is your customer's customer.

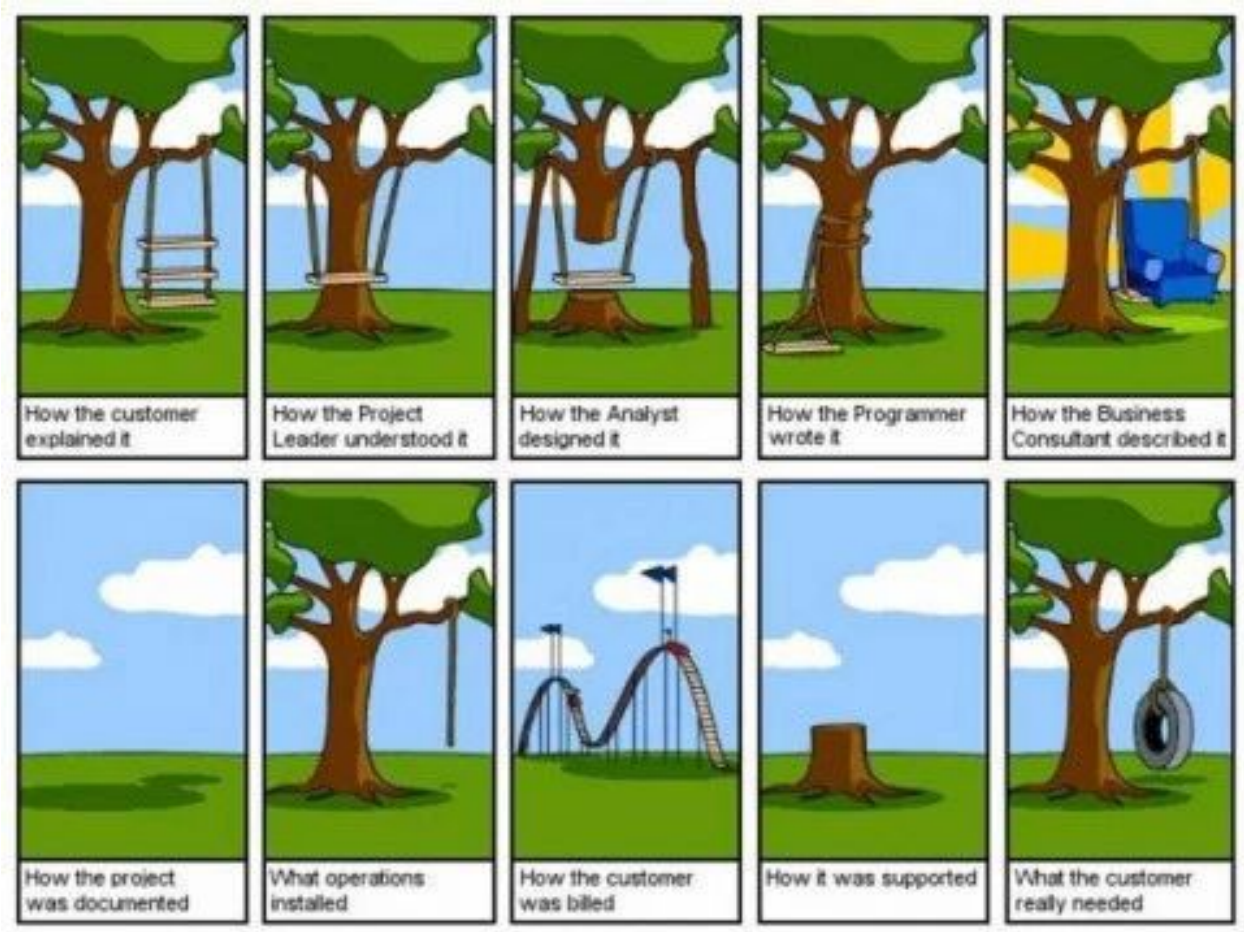
Requirements describe what users will be able to do with the software that they couldn't do without it, or couldn't do as well without it.

Requirements may not be realistic



[This cartoon is out of date]

Customers may not know what they want or how to describe it

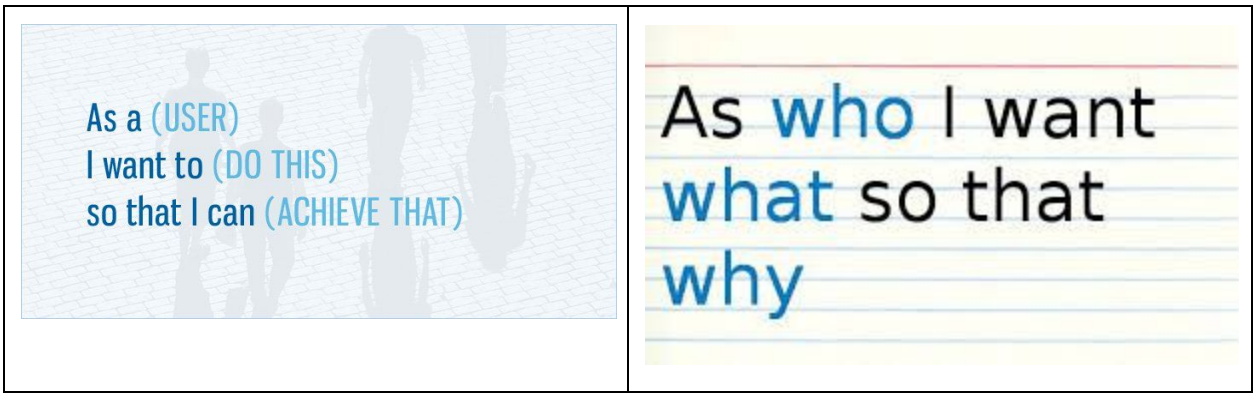


Requirements elicitation can be done in various ways, such as: Brainstorming workshops

Observation and interviews - how the software should fit into business workflows, constraints on how it interacts with pre-existing software or manual procedures

Role playing - pretend to be the software interacting with the user

A user story is a short simple description of ONE feature that can fit on a 3x5 card, written in customer's language



< label >: As a < type of user >, I want < some goal > so that < some reason >.

Label (or title) useful for cross-referencing

Type of user = role or persona

For example, Courseworks has 13 roles, most of them different categories of TAs or students. There's also someone with the role "Course Designer" for this course, turns out that's a CVN employee.

Example: As a PC user, I want to backup my entire hard drive so that my files are safe

Is this enough information to start developing the backup program?

Let’s add more details…

As a typical user, I want to specify folders to backup to make sure my most important files are saved

As a power user, I want to specify subfolders and filetypes not to backup so that my backup store isn’t filled with things I don’t need

Is this enough information to start developing the backup program?

What other information might the developer need? - Another way to think of it is what additional information will the backup program need?

What permissions will the backup program need to access the files its supposed to backup

Where to put the backup copies of the files

What to do if the backup store is not available

What to do if the backup store is full

What to do with the files already in the backup store

More…

This is what “conditions of satisfaction” are for. These usually will not fit on the 3x5 card.

< label >: As a < type of user >, I want < some goal > so that < some reason >.

My conditions of satisfaction are < list of common cases and special cases that must work >.

BAD user stories

Anything that mentions CRUD (create, read, update, delete) - should focus on what user wants to do and why, not the underlying data store

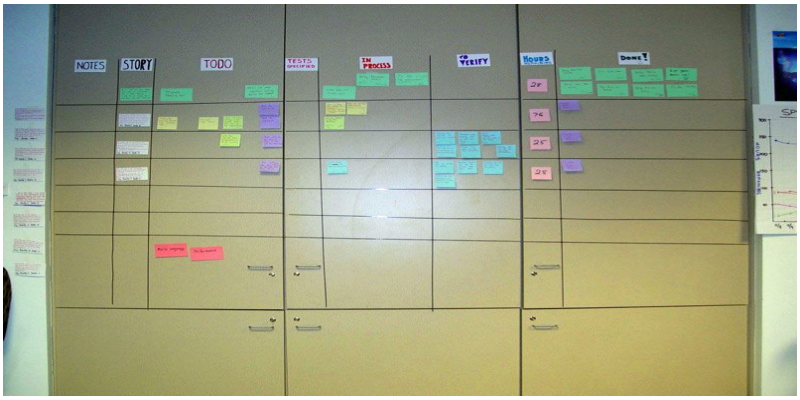
Any terminology the customer does not understand, the customer couldn't have asked for

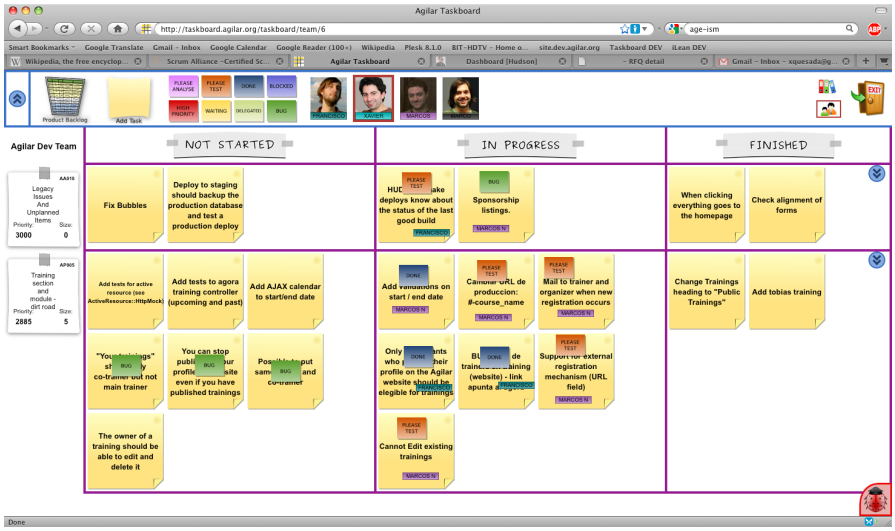
- Any technical jargon should be from customer's domain, e.g., financial services, **not** software engineering (unless the users of your product will be software engineers)
- Examples: MVC, JSON, websocket

Any specific technology, except as required for interoperability

- The customer probably didn't ask you to use Maven, Javalin, Postman
- But the customer may require that the new application runs on existing hardware, draws data from existing database, and interfaces to existing applications

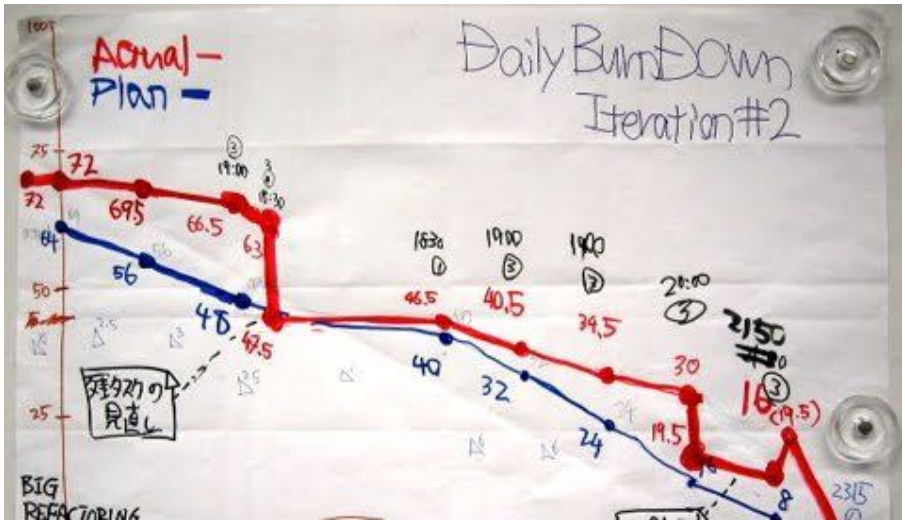
User stories (without the conditions of satisfaction) are great granularity for task boards

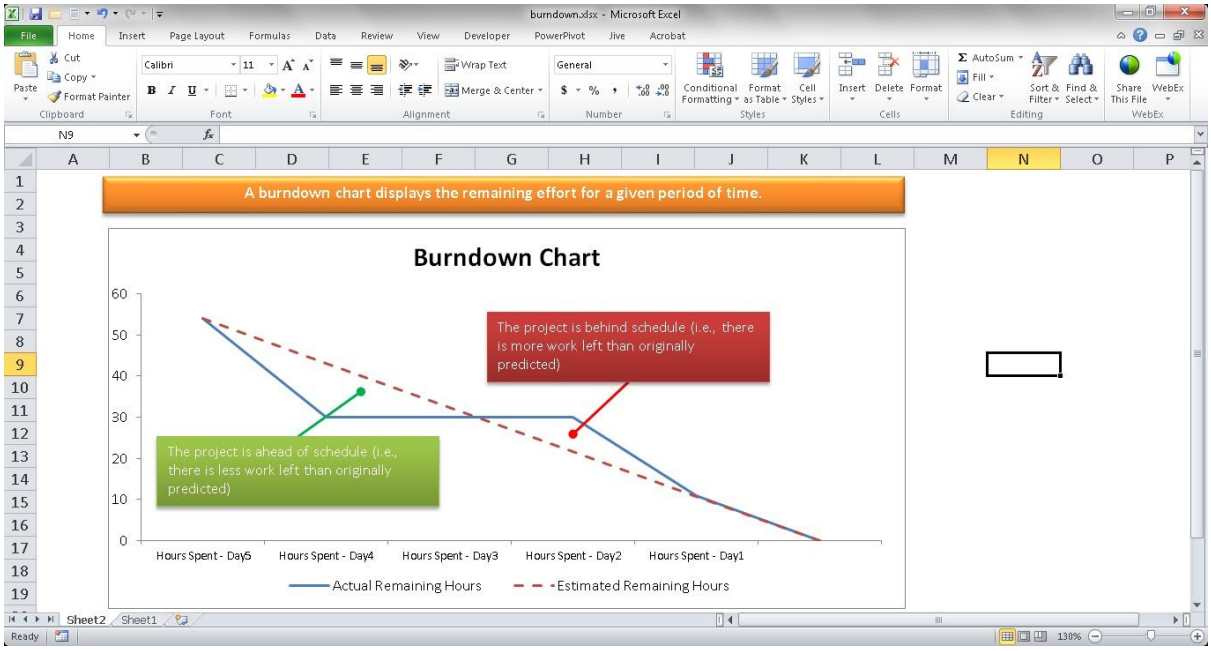




You can set up a task board for your team in github, see https://github.com/Programming-Systems-Lab/4156-PublicAssignment/projects
(or, better, use Jira: go to https://www.atlassian.com/software/jira and scroll down to “free”, connect to github with Github for Jira)

User stories are also a great granularity for burndown charts.





The customer may never see the task board or burndown chart, and probably would not be involved in assigning tasks to developers, but user story *priorities* must come from the customer

Usually start with Minimum Viable Product (MVP) set of user stories for the first release

Thereafter high, medium, low priority for further releases - may change over time

The developers' design may result in dependencies such that some user stories that seem low priority to the customer need to be done before some high priority stories - and some kinds of user stories are hard to retrofit, notably security and privacy

User stories and their conditions of satisfaction may not provide enough detail for time estimation (necessary to determine how many user stories will fit in a time-boxed iteration) or to break down into development tasks (that can be assigned to developers)

Use cases *expand* user stories to describe the step by step details of how the user role interacts with the software to exercise the feature and achieve the goal

Also try to capture anything and everything that *can go wrong*. Some of these may come from customers, some may be from situations that arise in software that developers know about but customers may not

example use case

Elaborate on use cases in next lecture