Lecture Notes
September 20, 2018

Team formation assignment due tonight: Everyone will need a team (of 4 students = 2 pairs) to develop your team project. Does anyone not have a team?

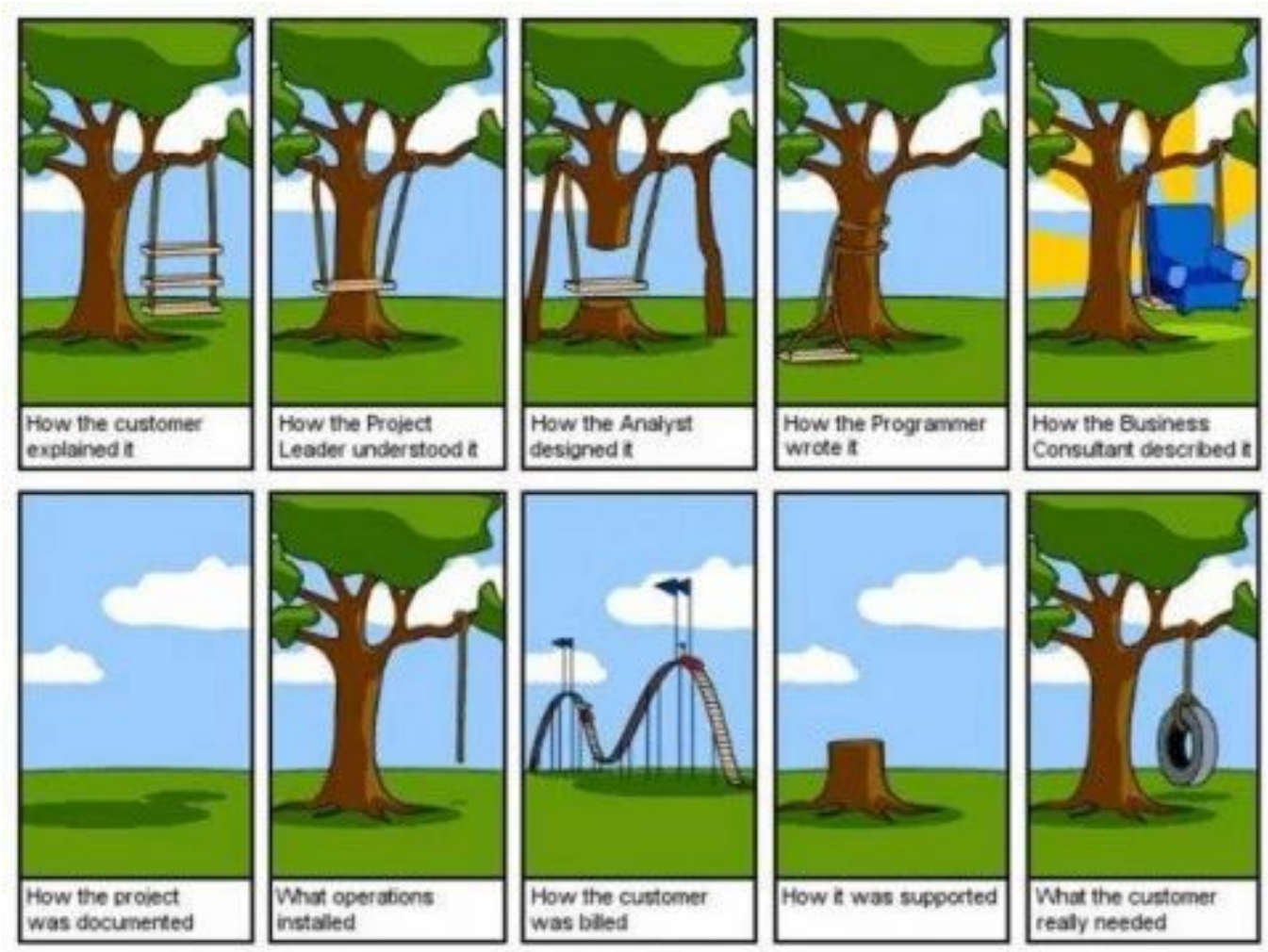
Team Preliminary Project Proposal due next Thursday, September 27, 11:59pm.

To do the preliminary proposal, you will need to understand *user stories* and *conditions of satisfaction*

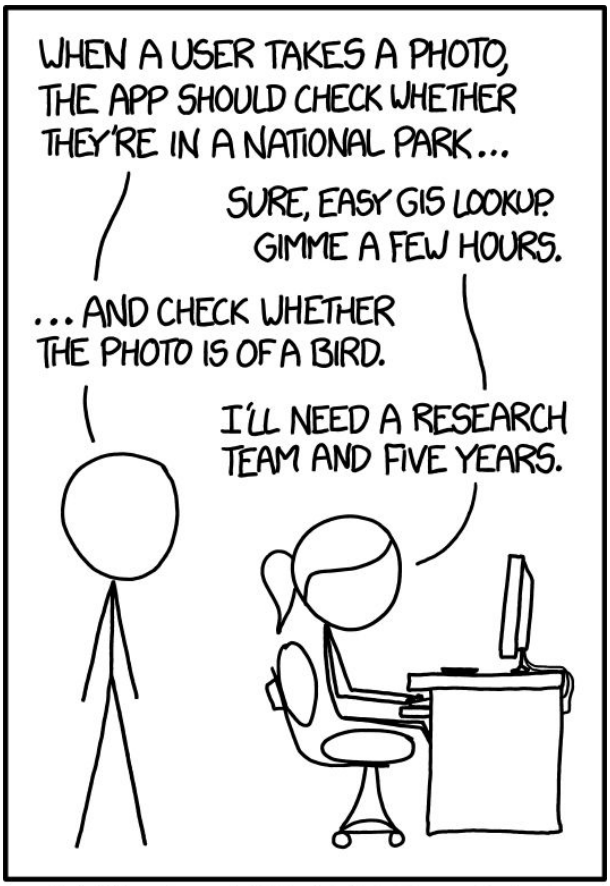
User stories are the agile way to describe customer requirements. Customers have something in mind that they want the software to do, of *value* to them. This *something* is what has traditionally been known as "requirements".

Even for a startup or new product, with no specific customers yet, internal product owners have some idea in mind that they want the software to do, perceived to have value to the market. Again known as "requirements". We will use the term customer even for product owners and in-house users.

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



Customer requirements may not be realistic



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

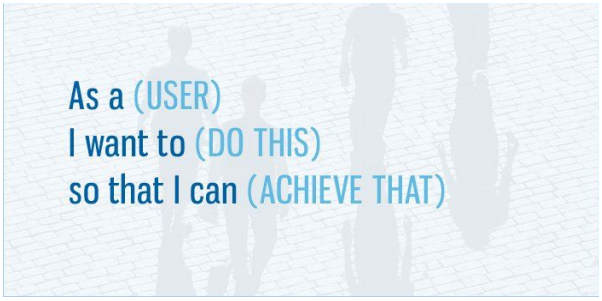
Role playing - pretend to be the software interacting with the user

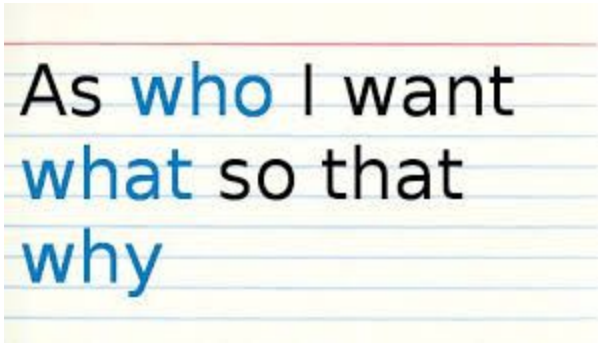
Let's do an example requirements elicitation in class: I need a volunteer customer

I will give the volunteer a list of suggested applications, please pick one.

This customer wants to contract your company to build a new software application. You (the class) need to elicit the requirements by asking questions of the customer, who can answer only "Yes", "No", "Sometimes". "Maybe" or "I don't know".

A user story is a short simple description of ONE feature that can fit on a 3x5 card





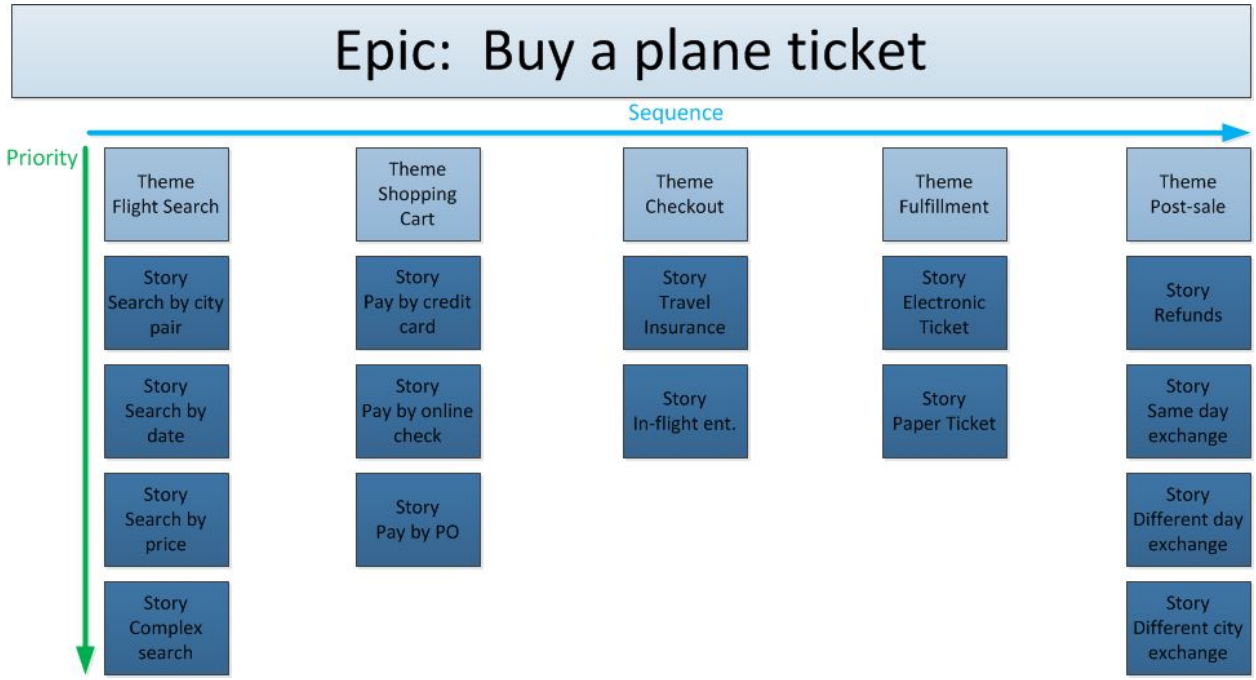
Although a user story provides more information than {yes, no, sometimes, maybe, I don't know}, it is not a *lot* more information - it serves as a reminder that further conversation with the customer is needed

Example: As a traveler, I want to purchase a plane ticket so I can attend a professional meeting

Does this give the developers enough information?

Does this give the developers the right information?

Collections of user stories - big features with many subparts - organized into *epics* and *themes*



For this class, we will use this format

< unique identifier >: As a < role >,
I want < something >
so that < benefit >

Title, label or other unique identifier useful for cross-referencing, e.g., with test plans, bug reports

Role = type of user or persona, e.g., buyer vs. seller, novice user vs. regular user vs. power user

Something = some goal the feature will achieve

Benefit = desired outcome or *value* to the role

Good user stories INVEST

- Independent (of other user stories),
- Negotiable (with customer),
- Valuable (to customer),
- Estimable (by developers),
- Small (doable within a sprint),
- Testable (by both developers and customers)

Recall that a user story represents a "promise of conversation" with the customer. That conversation seeks to extract the conditions of satisfaction (which may or may not fit on the back of the card), and may result in revising or splitting the user story

Conditions of satisfaction (or confirmation, acceptance criteria, etc.) are high-level acceptance tests - all acceptance tests must pass for the customer to accept the software

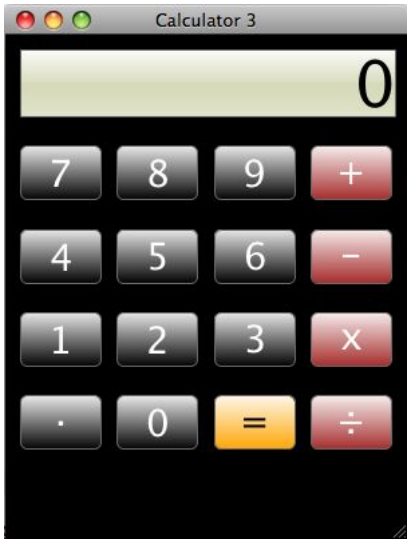
Example user story:

> As a marketing executive, I want to select a holiday season to be used when reviewing the performance of past advertising campaigns so I can identify profitable ones

Example conditions of satisfaction:

> List of holidays: Presidents Day, Memorial Day, Labor Day, …
>
> Holidays can span two calendar years: New Years Eve/New Years Day
>
> Holiday seasons can run from one holiday to the next, such as Thanksgiving through Christmas
>
> Holiday seasons can be set to specific number of days before and specific number of days after the actual holiday
>
> Separates Monday, Tuesday-Thursday, Friday-Sunday portions of each holiday week

How should these conditions of satisfaction be tested?



As a restaurant diner, I want to calculate the tip on my meal so I can leave a tip for the server

Is this a good user story for a general purpose calculator?

What are some testable conditions of satisfaction?

BAD user stories

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

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

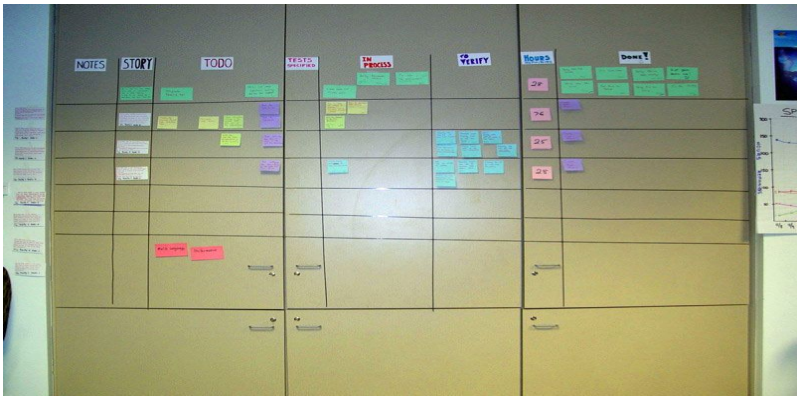
- Any technical jargon should be from customer's domain, e.g., financial services, **not** software engineering (unless the users of your product will be software engineers)
- Examples: cache, third normal form, SSL, websockets, abstract syntax tree, microservices

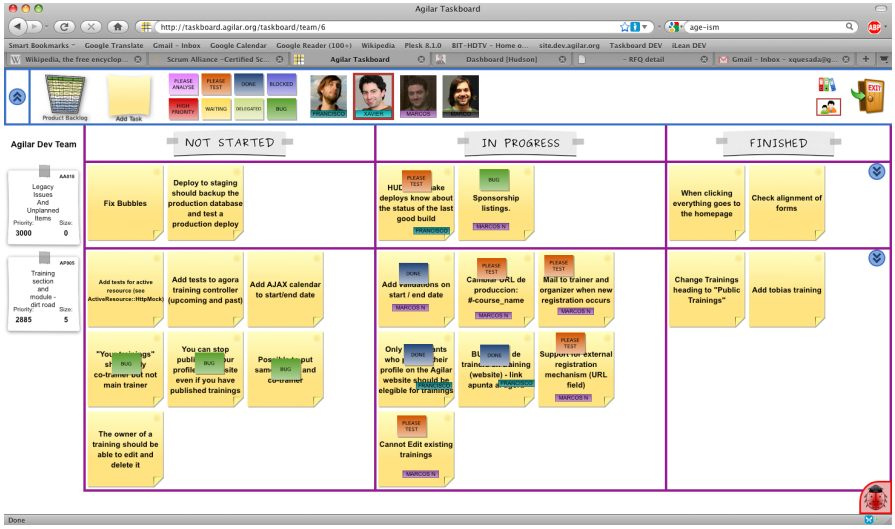
Any specific technology, except as required for interoperability

- The customer probably didn't ask you to use Java 6 on Ubuntu 4.10
  (or Java 10 on Ubuntu 18.04.1)
- But the customer may require that the new application runs on existing hardware, draws data from existing database, and interfaces to existing applications

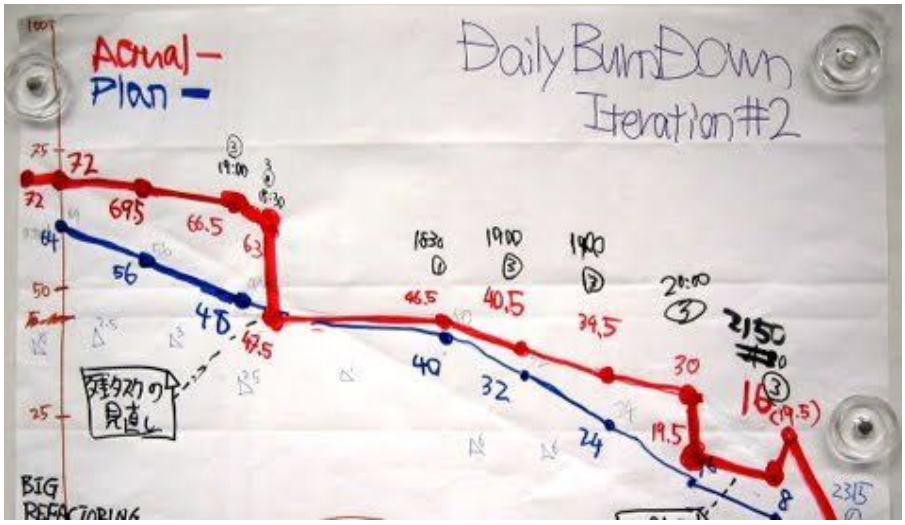
User stories, with their conditions of satisfaction, may constitute the entire "requirements". Or may be elaborated further into Use Cases, which consider workflow steps, alternative choices, what can go wrong - more realistic level of detail for design.

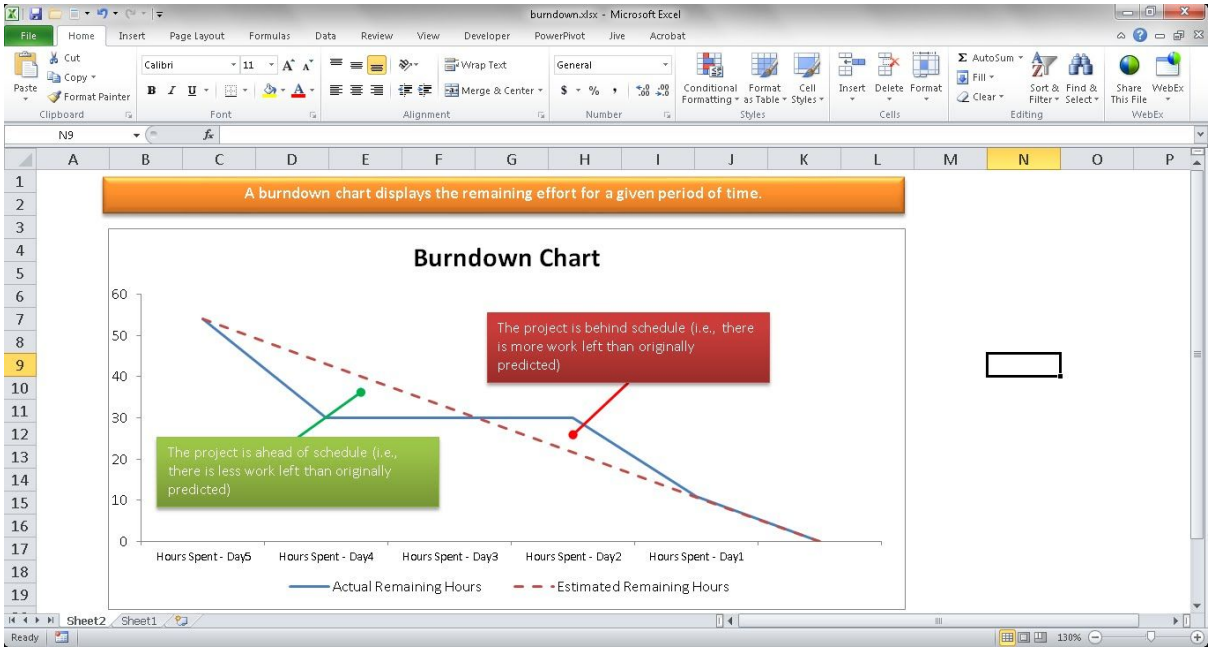
But user stories (usually without the much longer conditions of satisfaction) are great granularity for task boards





And 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 - typically too much for a single iteration, so still need to schedule across multiple iterations before 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

In-class exercise if time permits:

- Group together in teams of ~4 with whoever is sitting near you, does not need to be your project team, get some paper
- Recall the hypothetical drone that flies around the classroom, and uses its ***sensors*** to determine which students are "paying attention" and which are not
- Assume you already have several hardware drones available and they provide an API for accessing their sensors, communicating via wifi, etc. The drones have built-in software that controls flight, obstacle avoidance, and so on.
- Elaborate a set of specific user stories, including *testable* conditions of satisfaction, for the "paying attention" drone app.

Ten minutes

Volunteers present their user stories with conditions of satisfaction

Reading for Tuesday:
textbook, chapter 1 Why Do We Test Software?

Team Preliminary Project Proposal due next Thursday, September 27, 11:59pm

Your team project does **not** need to be a web or mobile app, which was required in recent previous years, and instead I discourage web and mobile apps.

Your project **does** need to be something that you will be able to demo - there needs to be a command line console or GUI.

Your project should also use some kind of persistent and structured data store (e.g., SQL, NoSQL, key-value store) and use some third-party library, framework or API (not just the built-in libraries that come with the programming language).

Some example APIs can be found here