Software Development Project

Morgan Ericsson
<morgan.ericsson@chalmers.se>

Last Time

- Scrum
- XP
- User Stories

Today

- More on user stories
  - requirements in general
  - user stories in particular
- Planning

Requirements

- Functional Requirements
  - features
- Non-functional Requirements
  - qualities

Requirements

- Functional Requirements
  - describes behavior (functions and/or services)
  - that support users, tasks, or activities

Requirements

- Non-functional Requirements
  - qualities and constraints
    - qualities are important to stakeholders and determine how happy they are with the system
    - constraints limit what solutions are possible

Requirements

- Simple (but not exact) rule
- Functional
  - The system should *do* X
- Non-functional
  - The system should *be* X

Examples

- The product should support several natural languages
- Persistence should be implemented using a relational database
- The system should use Oracle 8i
- The system should have 99.999% availability
- The system should have an online help function

FURPS+

- FURPS
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability

FURPS+

- +
  - Design requirements
  - Implementation requirements
  - Interface requirements
  - Physical requirements

FURPS+

- F
  - Functional requirements
- UPRS+
  - Non-functional requirements
  - UPRS = qualities
  - + = contraints

FURPS+

- Usability
- Reliability
- Performance
- Supportability

Examples

1.	The product should support several natural languages	S
2.	Persistence should be implemented using a relational database	+ (D)
3.	The system should use Oracle 8i	+ (I)
4.	The system should have 99.999% availability	R
5.	The system should have an online help function	F

Finding Non-Functional Requirements

- In general more difficult than functional requirements
  - not as visible
  - stakeholders not aware
  - quality is difficult to define
  - “always” important

Finding Non-Functional Requirements

- However, just as important as functional
  - affects the whole product
  - can be contradictory
  - and not always in an obvious way

Helpful Advice

- Think in reverse
  - start with a list of known qualities
- Consequence analysis
  - questions
  - consequences
  - answers/priority

Example

Requirements: Availability

Question: Are there any requirements wrt MTBF for example?

Consequence: Higher availability, higher development costs

Answer: Availability is a key quality. MTBF should be at least 60 days

Priority: High

User Stories

- Combines functional and non-functional requirements (or can be used to describe both, at least)
- Lists of stories
- Scenarios can help to find stories

User Stories

- Describes functionality that is of value for either users or customers
- Three perspectives / parts
  - card
  - conversation
  - confirmation

Card

- A story is generally written on a small paper card (think size of a post card)
  - the story is short
  - it focuses on a valuable goal
  - is written from the perspective of a role
- *"A user can post her resume to the web site"*

Conversation

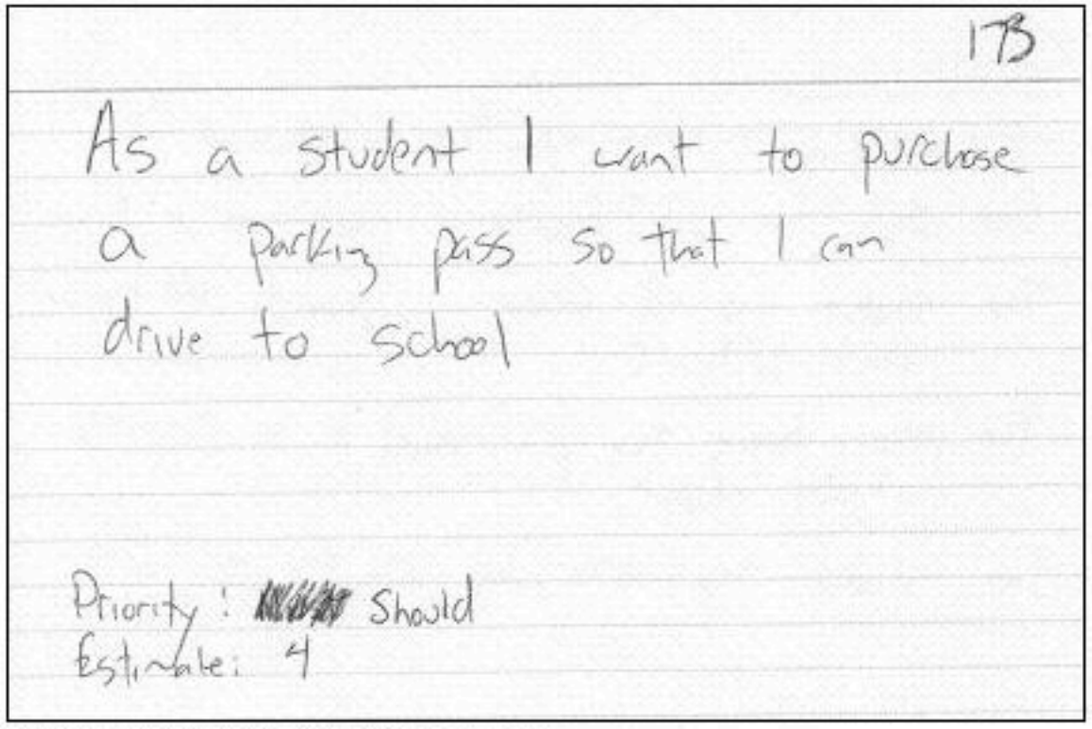
- User stories are short stories, not contracts
  - can be discussed
  - can be negotiated
- Can be used as reminders to discuss something further
- *"A company can pay for a job posting with a credit card"*

Confirmation

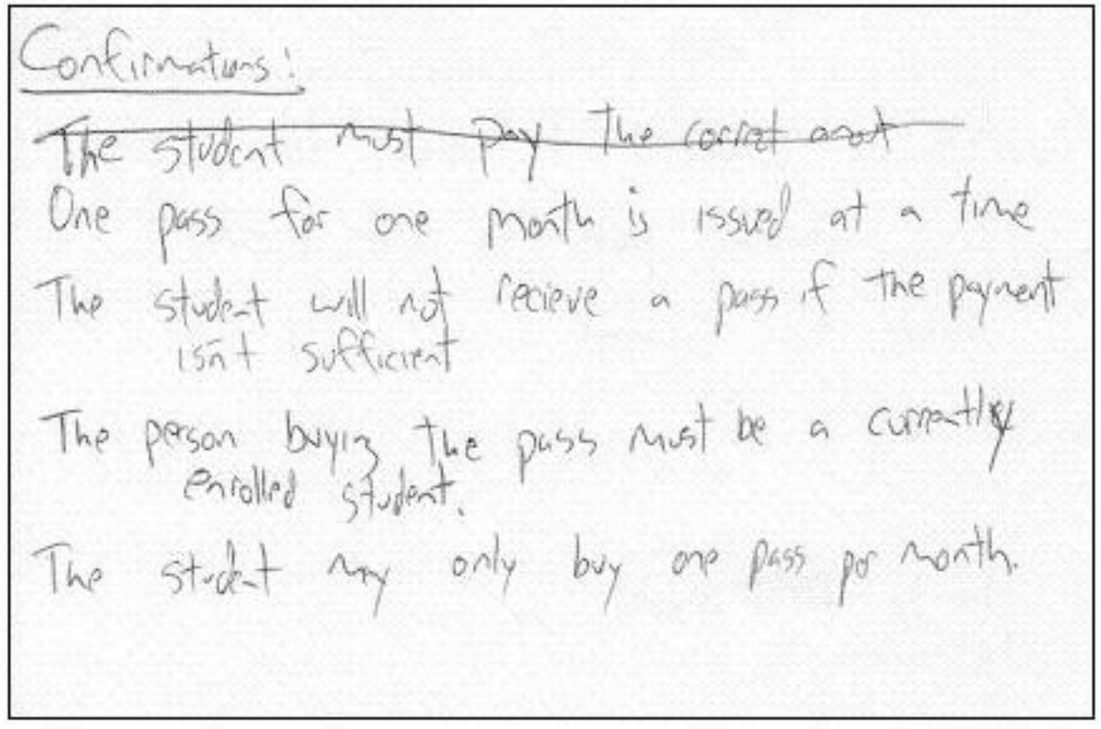
- A story should have a goal that can be tested
  - acceptance test, where the stakeholders can determine if the story is supported in an acceptable way
  - (more about this at a later lecture)

Example

Front of Card



Back of Card



Planning Tool

- User stories are used to guide the development
  - "customer team" prioritizes stories
    - stories are assigned to iterations and releases based on priority
- Stories can be changed, removed, reprioritized or added during the development process

Who Writes?

- Users stories are written by the customer team. Should ensure that the product matches the demands/wishes
  - customers
  - testers
  - end-users
  - designers
  - ...

A Simple Example

- South Coast Nautical Supplies have sold sailing equipment via mail order for more than 30 years
- They want to introduce a web store
- Initially, the web store should only contain books, but they want to include everything if the books work well

Process

- Understanding the customer
- Create roles
- Writing stories
- Estimate the stories
- Make a release plan

Roles

- The following attributes are important for the roles
  - how often do the role use the system?
  - domain expertise (sailing)
  - computer experience
  - experience from internet shopping
  - Why do the role use the system

Example Roles

- Novice Sailor
- Instructor
- Hard Core Sailor
- Experienced Sailor
- Non-Sailing Gift Buyer
- Librarian
- Administrator
- Report Viewer

Roles

- **Instructor**: Expected to use the website frequently, often once a week. Through the company's telephone sales group, an Instructor frequently places similar orders (for example, 20 copies of the same book). Proficient with the website but usually somewhat nervous around computers. Interested in getting the best prices. Not interested in reviews or other "frills."

- **Experienced Sailor**: Proficient with computers. Expected to order once or twice per quarter, perhaps more often during the summer. Knowledgeable about sailing but usually only for local regions. Very interested in what other sailors say are the best products and the best places to sail.

Stories for Experienced Sailors

- A user can search for books by author, title or ISBN number
- A user can view detailed information on a book. For example, number of pages, publication date and a brief description
- A user can put books into a “shopping cart” and buy them when she is done shopping
- A user can remove books from her cart before completing an order

Stories for Experienced Sailors

- To buy a book the user enters her billing address, the shipping address and credit card information
- A user can rate and review books
- A user can establish an account that remembers shipping and billing information
- A user can edit her account information (credit card, shipping address, billing address and so on)
- A user can put books into a "wish list" that is visible to other site visitors

Stories for Instructors

- A user can view a history of all of his past orders.
- A user can easily re-purchase items when viewing past orders.
- The site always tells a shopper what the last 3 (?) items she viewed are and provides links back to them. (This works even between sessions.)

Assessment

- “A user can search for books by author, title or ISBN number”
- What does that mean? Either or, or any possible combination?
- Fix!
- “A user can do a basic simple search that searches for a word or phrase in both the author and title fields”
- “A user can search for books by entering values in any combination of author, title and ISBN”

Estimation

Story	Estimation
A user can do a basic simple search that searches for a word or phrase in both the author and title fields.	1
A user can search for books by entering values in any combination of author, title and ISBN.	1
To buy a book the user enters her billing address, the shipping address and credit card information.	2
A user can write a review of a book. She can preview the review before submitting it. The book does not have to be one the user bought from us.	4

Release Plan

Iteration 1
A user can do a basic simple search that searches for a word or phrase in both the author and title fields.
A user can search for books by entering values in any combination of author, title and ISBN.
To buy a book the user enters her billing address, the shipping address and credit card information.

Non-functional Requirements

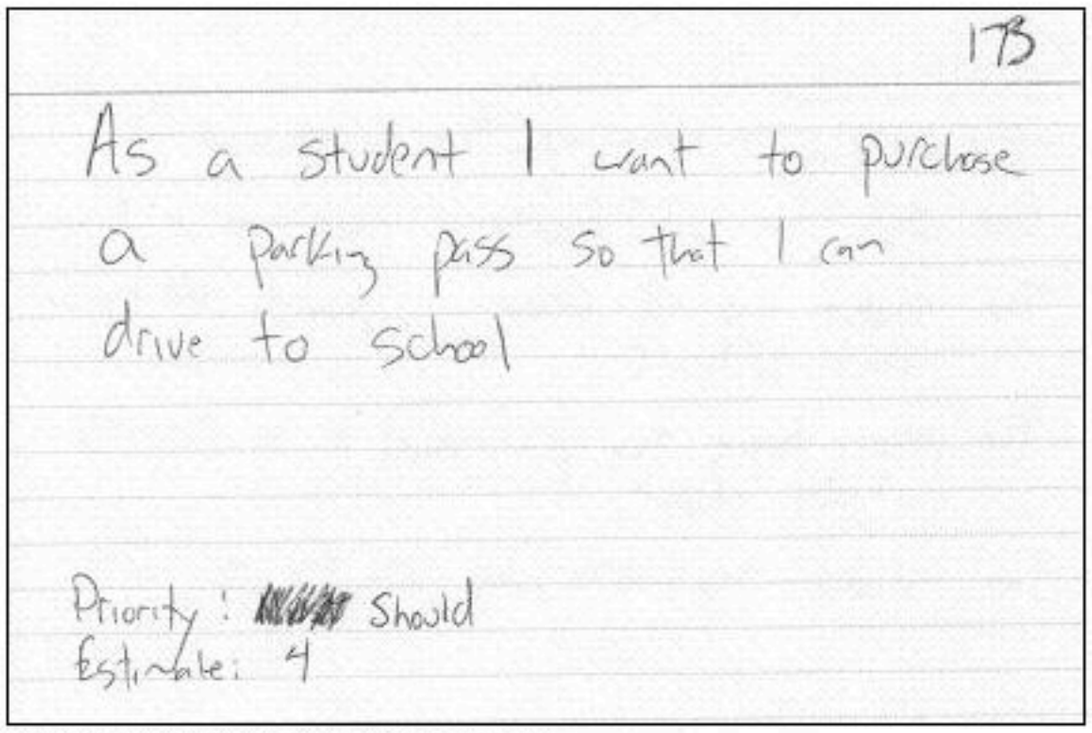
- Stories, often constraints
  - "A repeat customer must be able to find a book and complete an order in less than 90 seconds"
  - "The system must support peak usage of up to 50 concurrent users"

INVEST

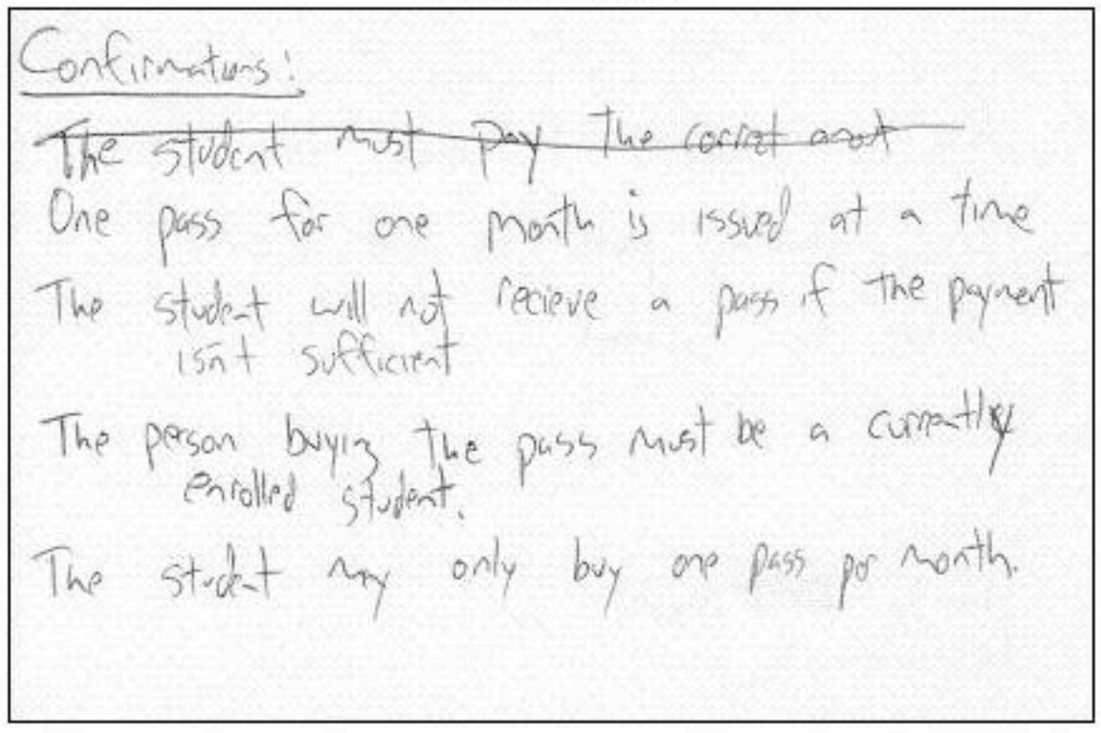
- Independent
- Negotiable
- Valuable
- Estimable
- Small
- Testable

Estimation, again

Front of Card



Back of Card



How much time will this take to implement?

Points, not days!

- The secret is not to deal with estimates that talk about hours/days/weeks
- Instead, we assign points to a user story to indicate relative lengths
  - "This is a short user story" – 1 point
  - "This story will take a while" – 10 points
  - "This story is about three times longer than that short one" – 3 points

Points, not days!

- Team velocity then is a measure of "how many points can we complete in a single iteration"
  - For our first couple of iterations, we simply guess and see what happens
  - Our guess will be wrong but if we stay consistent in assigning points to stories, the velocity we can achieve will soon make itself known

Benefits of Points

- Humans are good at working with relative sizes
  - All we are trying to do is capture the "bigness" of a task
  - Points remind us that our estimates are guesses and we are doing other things (team velocity, iteration planning, etc.) to verify them
  - Relative estimates rarely change (Task A IS twice as big as Task B)
  - This type of system is fast, simple, and easy to learn

Are points necessary?

- No, some teams use “t-shirt sizes”: small, large, XL
- When you do that, you need a way to map that into iteration planning

Triangulation

- Triangulation refers to the fact that once you have sized a few user stories
  - and converted them to working software
  - you'll be in a much better place to size future stories
- "That task is similar to the one we did last week, and that was a 2 point task"

Triangulation

- You can then quickly categorize all remaining user stories or any new ones
  - you can even use triangulation when your team velocity is not yet clear
  - if you've decided that one database-related task is 5 points before your first iteration, you're likely to classify all such tasks as 5 points

Planning Poker

- A popular estimation technique in agile methods
- Addresses the problem in which two or more team members come up with wildly different estimates for a story
  - i.e. when a single user story generates estimates of say "3 points", "10 points", and "100 points" from three different developers
- The underlying cause for these different estimates is assumptions; what did you assume was true or not true about the project to generate the number that you did?

Example

- "Add a comment on a product page"
- One team member might think:
  - "Simple. We need a form, a script to process the form, and a place to store the comment in the database. 3 days."
- Another might think:
  - "Hmm. How do we relate the comment to the product? Do we have one comment table per product in the database? Will I need to change the product class? Maybe there is code from some other place in the system that I can re-use. 2 weeks."

Example

- Finally, another might think:
  - "Ugh. Complete database re-design. No code to re-use (this is the first time we're allowing comments). What user interface should we use? Can the user embed HTML in their comments? How do we handle smileys? How will this impact the product model class? Do we keep the comments forever? Do we need moderation? Can a user edit a past comment? Who gets to delete comments? Yuck!! 3 months!"
- Based on your assumptions, you'll get completely different numbers. How do you get these assumptions to the surface? Planning Poker!

Planning Poker

- Create “deck” of cards. 13 cards per “player”.
    - Each card contains an estimate spanning from “already done” to “wow this is going to take a long time”.
        - 0, .5, 1, 2, 3, 5, 8, 13, 20, 40, 100 days
- One card has a “?” meaning “not enough information”
- One card has a coffee cup meaning “lets take a break”

Planning Poker

- Place a user story in the middle of the table
    - Each team member thinks about the story and forms initial estimate in their heads
    - Each person places the corresponding card face down on the table; note: estimate is for entire user story
    - Everyone then turns over the cards at the same time
    - The dealer marks the spread across the estimates

Planning Poke

- The larger the difference between the estimates, the less confident you are in the estimate, and the more assumptions you need to highlight and discuss
- So, the next step in planning poker is
  - Put assumptions on trial for their lives
- Have each team member list the assumptions they made and then start discussing them
  - You need to criticize the assumption not the person
- Goal is to get agreement on what assumptions truly apply

Planning Poker

- If the assumptions reveal a misunderstanding of the requirements, then go back to the client and get that misunderstanding clarified
- Otherwise, start to eliminate as many assumptions as possible, then have everyone revise their estimates and play planning poker again to see if the spread has decreased
    - Your goal is convergence. Once estimates cluster around a common number, assign that number and move to the next story
- Do this until all user stories have a consensus estimate assigned
    - If any ambiguities remain, consult with the customer and try again

Planning Poker

- Things to watch out for
- Although implied in the previous slides, don't do one card at a time with multiple customer sessions each time
  - Value your customer's time
  - Process each card, identifying assumptions/ misunderstandings that need clarification; THEN meet with customer

Planning Poker

- Big estimates (== bad estimates)
  - they indicate that the story is too big; decompose; try again
  - iterations are typically 20 work days (1 month) or less
  - estimates longer than 15 days are more likely to be wrong than those shorter than 15 days; (others think 7 days is upper limit)

Now What?

- At this point, we have
  - a set of user stories
  - estimates assigned to each story
    - and, importantly, estimates that we have thought carefully about
- The next piece of the puzzle is customer priorities
  - We need to know from the customer what stories are the most important