CS 361
Software Engineering I

Customer Collaboration

Central Principles

The customer is part of the team.

The customer plays a key role in directing the team.

XP Collaboration with Customers

1. Customers & Engineers: Collect user stories
2. Engineers: Identify tasks
3. Engineers: Estimate effort
4. Customers: Prioritize stories
5. Engineers: Plan work
6. Engineers: Communicate progress
7. Customers & Engineers: Evaluate results

User Story: What It Is

- Fits easily on a 3x5 note card
  - Can also be typed into spreadsheet, but harder to review with customers
- Name + short description

User Story: Example

View eco-friendliness

The e-commerce website users can click on a product and view its eco-friendliness ratings.

Upload eco-friendliness data

Trusted 3rd parties can upload an XML document specifying eco-friendliness ratings for products.

User Story: Principles

- Not written in stone
  - Revision is acceptable and expected
- Fuzzy
  - Further communication is usually needed
- Minimalist
  - If you add stuff that the customer doesn‘t want, then the system will surely cost too much.
- Backed up by acceptance tests
  - Will discuss further during testing lecture

User Story: How to Collect

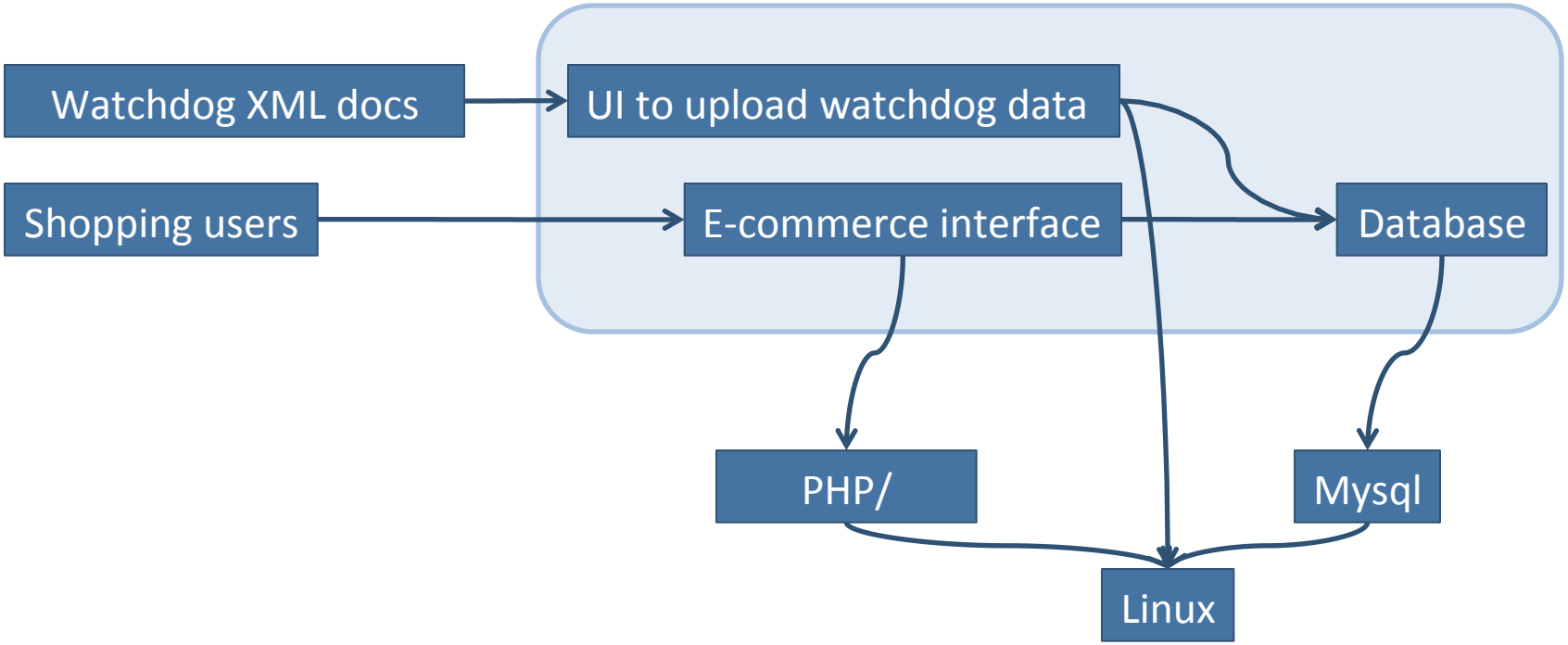
- Gathering user stories
  - Start from a starting point
  - Let the customer walk through the vision
  - Customer may have "artifacts" illustrating vision
  - Chunk the vision into user stories
  - Ask questions to clarify
  - Don't invent stories; let the customer "drive"
- Remind the customer that he can add more stories later

Identifying Tasks

- Systematically break down each user story into the pieces of work that will be required
  - Can often be decomposed using the skills that you already have for decomposing architectures
    - e.g.: object-, process-, feature-oriented decomposition
  - Often stated like, "first we'll implement this change to the system, then this change, then this change… that's 3 tasks"

Identifying Tasks

- Often helpful to sketch an architecture…



Identifying Tasks

Process Oriented Decomposition

View eco-friendliness

- Set up a database to store product data/ratings
- Create web page to show a product's data
- Create web page listing products, so user can click

Upload eco-friendliness data

- Set up a database to store product data/ratings
- Create a web page for uploading XML document
- Write code to read data into database

Effort Estimates: What They are

- Figure out how much effort each task entails
  - All estimates should be done in a nebulous "unit" of your own calibration
  - Typically based on experiences with similar work
  - Do a spike (an experimental implementation) to measure difficulty, if needed
  - Take "bids" from team members for the lowest effort on each task
- Sum up task efforts to compute story effort
- Tweak this estimate if it seems appropriate

Effort Estimates: The Nebulous "Unit"

- A unit is defined in terms of how much work you can get done in the next week's iteration
- To start with, typically defined as something like 1/20th of what the team can accomplish this next week.
  - This is *after* taking into account the fact that all programming occurs in pairs.
  - So an 8-person team has 4 pairs, each of which might be able to get 5 "units" done this week.
  - Better programmers might be able to do more "units" per week.

Estimating Effort: Principles

- Engineers give honest estimates that customers can trust.
- Engineers refine estimates; customers refine expectations.

- It is expected that you will work at a sustainable pace.
  - No heroes, no all-nighters, no super-human feats
  - Either you get the code done like a human being, or you don't

Estimating Effort: Principles

- Estimates are based on each user story individually
  - i.e.: don't assume a certain ordering of the waiting stories
  - Yes, you'll end up *over* estimating effort (that's good)

- "An honest plan means you have made all the difficult decisions you are responsible for making."
  - "He who does the work sets the estimate." – own your time!
  - "Don't theorize, try it." – use spikes, use spikes, use spikes

A Word About Spikes

- Purpose: Discover answers to difficult problems (either design or implementation)
- What it is: A small program that explores and tests possible solutions
- How to do it:
  - Identify the key thing that you need to discover
  - Write a little code to test an idea
  - Tweak the code until you get the info you need
  - Throw the code away, keep the knowledge

Effort Estimates: Examples

- **View eco-friendliness**
  - Set up a database to store product data/ratings *
  - Create web page to show a product's data
  - Create web page listing products, so user can click
- **Upload eco-friendliness data**
  - Set up a database to store product data/ratings *
  - Create a web page for uploading XML document *
    - Write code to read data into database

* = risky… may call for spike!!!

Effort Estimates: Examples

3u - View eco-friendliness

1u - Set up a database to store product data/ratings *

1u - Create web page to show a product's data

1u - Create web page listing products, so user can click

2.5u - Upload eco-friendliness data

1u - Set up a database to store product data/ratings *

.5u - Create a web page for uploading XML document *

1u - Write code to read data into database

- = risky… may call for spike!!!

Prioritizing Stories

- Now that each story has an associated effort, customer gets to choose which stories will be implemented in the next iteration
  - E.g.: "Stories A, B and C each will take 10 units; stories D and E each will take 5 units; we can do 15 units this week. Mr. Customer, what stories should we do?"
  - It's like writing a shopping list using a certain budget.

- The customer gets to make business decisions
- The engineers get to make technical decisions

- Don't "steer" the customer
  - Help the customer to understand consequences of choices
  - Make your case but let the customer decide
  - But negotiation is good and appropriate

Planning Work

- Once the customer has chosen stories, teammates get organized.
  - Get together and decide who will perform each story's tasks.
  - Figure out how to work in pairs for these tasks
    - This is a tentative assignment and will probably evolve
    - Mix it up: don't work in the same pair all the time!
  - Plan how to avoid stepping on each other's toes.
- Finalize your development environment

Planning Work: Examples

Monday: Jim & Peg

1u - Set up a database to store product data/ratings

Load with two sets of test data

Tuesday-Wednesday: Jim & Joe

1u - Create web page to show a product's data

1u - Create web page listing products, so user can click

Use one set of test data for unit tests

Tuesday-Wednesday: Peg & Phil

.5u - Create a web page for uploading XML document

1u - Write code to read data into database

Use the other set of test data for unit tests

Communicating Progress

- Track effort using PSP or similar technique
  - Track velocity… how many units are you doing per day?
  - Refine your understanding of your capabilities
- All news (good or bad) should be communicated early
  - Keep the customer informed of good news & bad.
  - Keep your team informed, too!!!
- Release dates don't slip; functionality slips.
  - It might be necessary to delay user stories
  - Customer prefers 3 fully-working user stories rather than 6 half-working stories!