CS1530 LECTURE 4

INTERACTING WITH STAKEHOLDERS AND ELICITING REQUIREMENTS

SOFTWARE DEVELOPMENT IS NOT

SITTING AT A DESK BY YOURSELF ALL DAY

- Communication is an essential aspect of software development
- In some roles, communication is the majority of all work done
- Yes, email can count as work

R-E-S-P-E-C-T

FIND OUT WHAT IT MEANS TO THE TEACHER OF THE CLASS

- Remember that others have skills that you don't have, as well as different valuable experiences and perspective
- Don't go into conversations looking to mentally intimidate, but to *get* information
- Remember to be polite - ask, don't request unless necessary

ESPERANTO, BUDDY!

DO YOU SPEAK IT?

CARE ENOUGH TO LEARN THEIR LANGUAGE

(WHICH IS PROBABLY NOT ESPERANTO)

- Different stakeholders care about different things, and will have different background
- We don’t have a universal language (sorry, Esperantists), so you will have to learn their language
- They may or may not learn yours!
- To effectively communicate, you’ll need to understand what they need, how they think, and their domain in general

ELICITING REQUIREMENTS

OR, "FIGURING OUT WHAT THEY WANT"

- I prefer the term "eliciting" rather than "gathering" or "determining" because it acknowledges that the difficulties inherent in the task

- People often have vague ideas of what needs to be done - they could not write a detailed specification if they wanted to (and for the most part, people don't want to)

- They may have many assumptions about how the software "should" work that you don't know about

- They may not have thought of, or have good answers for, many questions!

SIGNIFICANT PROBLEMS IN ELICITING REQUIREMENTS

- Non-technical users often may not be aware of “non-functional requirements” (quality attributes), or how to define acceptable values for them
  - Security
  - Performance
  - Scalability
  - Portability

SIGNIFICANT PROBLEMS IN ELICITING REQUIREMENTS

- Customers often do not think to say what the software should NOT do, or what to do in edge cases
  - What do to in case of invalid input?
  - What to do if subsystem, 3rd-party API, etc. goes down?
  - What are acceptable limitations and parameters to use of the system?
  - What happens when a resource is constrained?
  - What are preferred scenarios for graceful degradation?

SIGNIFICANT PROBLEMS IN ELICITING REQUIREMENTS

- Customers often have trouble being precise
- Software Engineers are used to specifying system to an absurd degree ("computational thinking")
- May be qualitative when you need quantitative responses

SOLUTIONS

- Ask questions - ensure that you understand what has been asked of you
- Repeat back what was said in your own words
- Be as clear as possible - avoid pronouns!
  - “It gets the thing, it goes to the other thing, and they see it”
  - “The fetcher downloads the weather data, and sends that data to the parser, which generates a web page and displays it to the user.”

SOLUTIONS

- Paper prototyping
  - “A picture is worth a thousand words”
  - Draw what you think the system should look like, or have them do it
- Have them interact with it on paper, tell them what it did after they interacted - don’t tell them first!
- Can save thousands * of hours of programming UI!

  * Not guaranteed.

SOLUTIONS

- User observation
  - Watch them use similar software or a beta version of your own
  - DO NOT INTERRUPT
  - If you want to ask questions, ask afterwards
  - Allows an independent view of what is easy/difficult/ overcomplicated/etc.

SOLUTIONS

- Be prepared to modify your software based on future communications
- Requirements/user stories/etc. will change if you are interacting with the customer directly to determine what they want
- Develop your software with that in mind
- Avoid over-engineering!