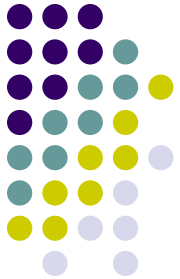

Professor Hausi A. Müller PhD PEng FCAE
Department of Computer Science
Faculty of Engineering
University of Victoria

http://www.engr.uvic.ca/~seng321/
https://courses1.csc.uvic.ca/courses/201/spring/seng/321

# Announcements

- Thu, Feb 18
  - Deliverable C1 due (yesterday)
- Marks for S0 & C0
  - Posted
- S2 & C2
  - Will be posted on Saturday
- Quiz 1
  - ***Wed, Feb 24 in class***
  - Class attendance must increase
- New Midterm Date
  - Wed, March 2
  - In class

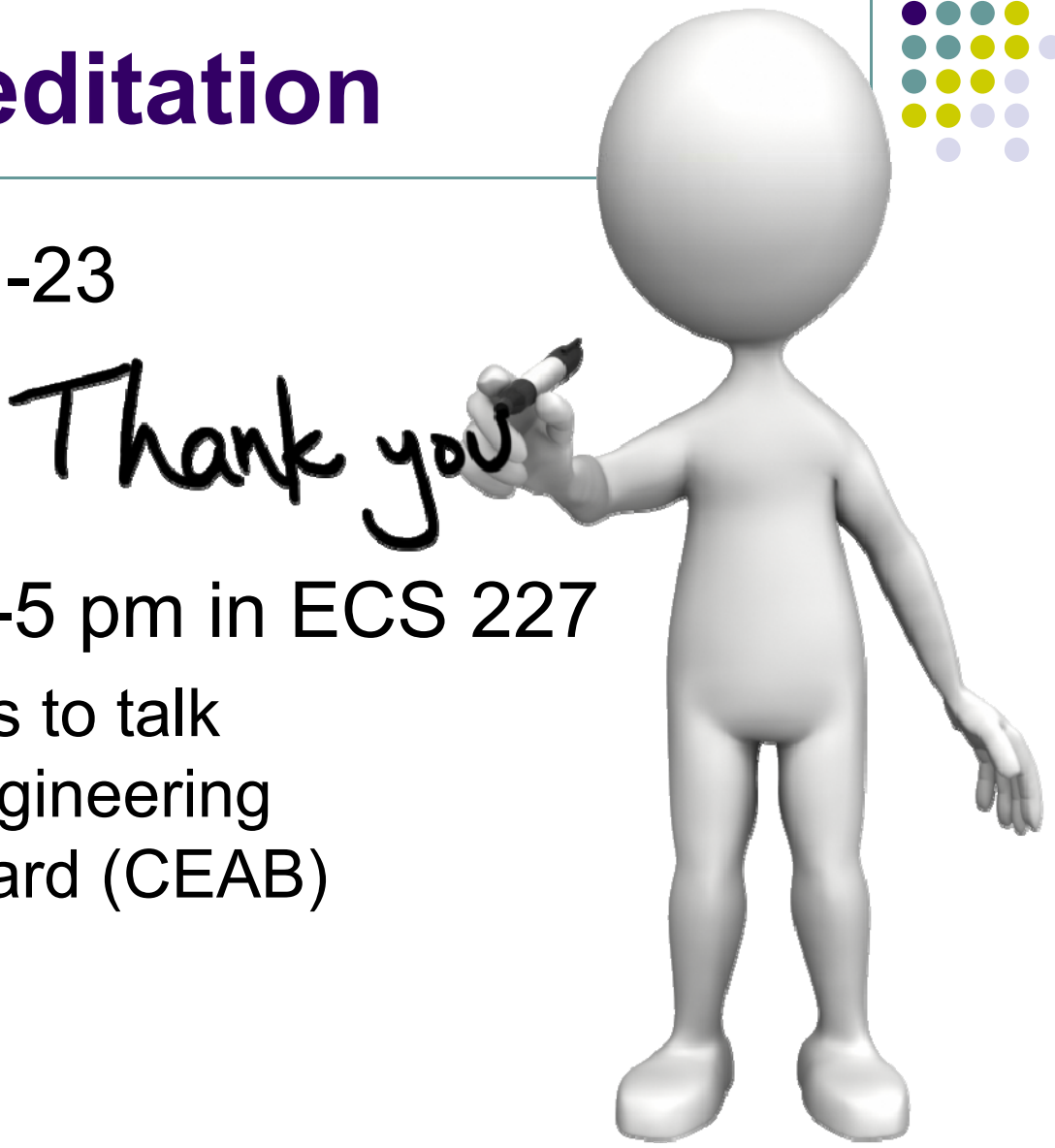
# SENG 321 Calendar



| | | | |
|---|---|---|---|
| Deliverable S1 due | Tue, Feb 16 | S1 formal req spec | 10% of project |
| Deliverable C1 | Thu, Feb 18 | C1 feedback on S1 | 5% of project |
| Midterm | Fri, Feb 26 | In class | 14% of project |
| Deliverable S2a | Tue, Mar 1 | S2a detailed formal req spec | 10% of project |
| Deliverable S2b | Thu, Mar 3 | S2b & demos | 5% of project |
| Deliverable C2 | Tue, Mar 8 | C2 feedback on 23a&b | 5% of project |

# BSENG Accreditation

- Sun-Tue, Feb 21-23



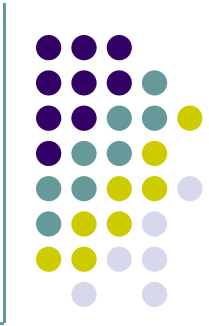

- Mon, Feb 22 – 4-5 pm in ECS 227
  - Need 10 students to talk to Canadian Engineering Accreditation Board (CEAB) site visit team
  - Sign up list

# Requirement Engineering

- Requirements engineering has two main foci
  - Deciding what a software system is supposed to do
  - Communicating that decision between stakeholders (i.e., various people affected)
- RE is difficult since everyone has different needs
  - Users want a useful system
  - Customer, who is paying, wants a low cost system
  - Regulators want a system that is easy to audit

**RE's main goal is**
**to ELICIT requirements**
**not just WRITE requirements !**

# Elicitation



- Lacks a prescriptive solution and depends largely upon the expertise of the elicitors
- Expertise is built upon knowledge of various elicitation techniques and heuristics
- Information exists somewhere but is hidden
  - Latent knowledge: hidden knowledge not readily accessible
  - Tacit knowledge: well known and accessible by one party, but considered to be too obvious to mention
  - Externalising internal head models

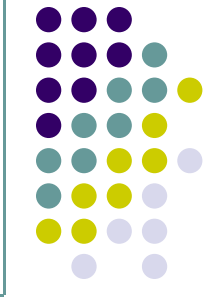
# Elicitation is Hard

- Stakeholders
  - Have difficulty expressing their needs
  - Ask for things that may not address their needs
  - Have conflicting needs
  - Do not know their priorities
  - Have hard time imagining new ways to do things
  - No clear identifiable stakeholders for new products (surveys/questionnaire)
  - Demands change and evolve over time

# Requirement Elicitation

- "to elicit" ➔ "to bring out, to evoke, to call forth"
- "Getting what is in the customers heads without hurting the customers or their heads" ☺
- Elicitation main focus is on ***communication***
- In addition to technical knowledge, elicitation requires
  - Good people skills
  - Common sense
  - "Thinking outside the box" attitude

# Different and Unclear Goals

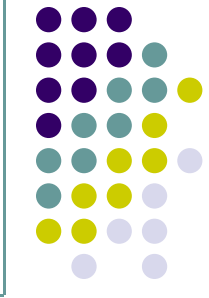


- A child asks dad to make water warmer
- Dad puts hand into the water and finds it warmer than usual?
  - Child wants to make it closer to the temperature that the child calls warm!
- Same for customers their needs and expectations may vary a lot from developers

# Clarifying User Requirements Different Perspectives

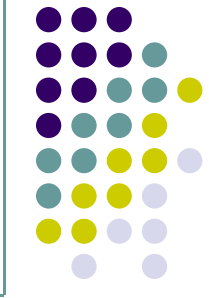
- Nurse: Patient filling in a form should be given a diabetic screening
  - To perform a full medical screening?
  - To fill in a questionnaire?
- 50 Users should be supported with only 2 licenses
  - To cut cost?
  - Used rarely but on many different workstations?
- Slow elevators
  - How do we speed up the elevator?
  - How do we keep people happy in slow elevators?

# Requirement Elicitation



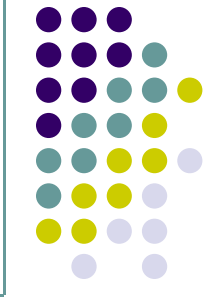
- Robertson & Robertson 1999 use the term Requirement Trawling – Just like fish trawling with a net being dragged behind a boat
- **Trawling** is a method of fishing that involves pulling a large fishing net through the water behind one or more boats. The net that is used for trawling is called a **trawl.**

# Requirement Trawling



1. We get requirements in first pass of fish net then as we dig deeper we uncover other requirements and clarify current requirements
2. Requirements just like fish mature and may die
   - Importance change and even scope as we uncover the complexity of requirements
3. We will never capture all the requirements
4. We are likely to get good requirements and bad bloat requirements (same happens when you trawl you get tires/garbage stuck in you fish net)
5. There are good spots to trawl and there are bad spots:
   - The more experienced you become, the more you will know the good people to talk to and the techniques to use

# Skills Needed by a Requirements Analyst



- A good detective, interviewer and facilitator
  - Identify contexts
  - Spot ambiguities and evasion tactics
  - Get people to open up
  - Instill guilt
  - Encourage people to participate and brainstorm

# Elicitation Techniques

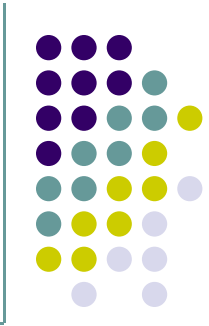
1. Reuse old requirements or existing system
2. Questionnaire
3. Interviews
4. Observation and apprenticeship
5. Ethnographic studies
6. Brainstorming
7. JAD: Joint Application Design
8. Nominal group technique
9. Delphi technique
10. PIECES Approach

# Analyze Existing Systems

- Examine currently deployed system, find out:
  - What is used, what isn't, what's missing?
  - What works well, what doesn't?
  - How is the system is actually used, how was it intended to be used, what new ways do we want it to have?

SENG 371
Software Evolution

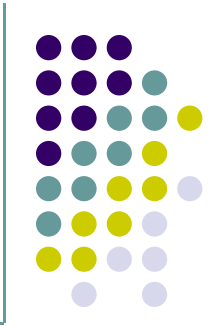
# Analyze Existing Systems



- It is easier to learn from an existing system than starting from scratch; may be a manual, non-computerized system or process

- Risks if you don't
  - Users may not be happy with too much change compared to the old system.
  - May miss real usage patterns, human issues, common activities, relative importance of tasks/features.
  - May miss obvious possible improvements (features that are missing or don't currently work well).
  - May miss which "legacy" features can/can't be left out.
  - May miss latent knowledge

# Analyze Existing Systems



- Example of inadequate analysis before implementation.
  - Hot air hand dryers are neat! Therefore, we need them.
  - They eliminate paper waste, cleaner washrooms, cheaper in long run. So let's install them; they do just the same task as paper towels.
  - But if you had spent a day in a washroom, you might have noticed different usage patterns of paper towels: dry face, clean up spills, dry coffee pots, blow noses, etc.
  - These days, almost all public washrooms have paper towels once again, even if they also have hot air dryers.

# Analyze Existing Systems

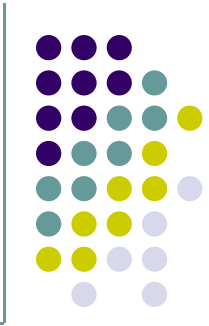
- Review all available documentation
  - If there exists an automated system, review its requirements specifications and user manuals, as well as development documentation, internal memos, change histories, etc.
    - Of course, often these are out of date, poorly written, wrong, *etc.*, but it's a good starting point.
  - If the existing system is a manual system, review any documented procedures that the workers must follow.
  - Watch for business process reengineering opportunities
    - Data centralization to avoid document mismatches (purchase order and invoice)
- The goal is to gain knowledge of the system before imposing upon other people's time, before bothering stakeholders.

# Analyze Existing Systems

- Observation
  - Go out into the field, and observe the “IT specialists in the mist”.
  - Documentation rarely describes a system completely, and it often is not up to date. The current operation of the system may differ significantly from what is described.
  - Besides, no matter how bad a reputation the existing system has for doing the work, the system is not worthless. It contains a lot of useful functionality that should be included in any future system. The objectives of observing the current system is to identify what aspects to keep and to understand the system you are about to change.

So, how many people will gone “into the field” for S1?

# Analyze Existing Systems



- Observation
  - Ideally, you are a silent observer, at least initially. Otherwise, you risk getting a non-standard view of what is usually done.
  - Later on, you can start asking questions OR interview people OR use questionnaires.
  - Perhaps if someone had spent one day in a washroom observing how paper towels were being used, he or she would have discovered the secondary functions that paper towels provide that hot-air dryer do not.
  - When new computer systems are implemented, remnants of the old system often linger on, because the designers of new system overlooked a function provided by the old system.

# Analyze Existing Systems

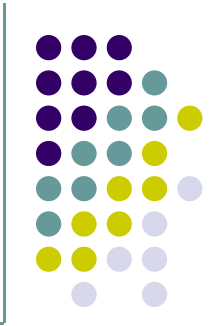
- **Questionnaires**:
  - Most useful when large number of users and you want answers to specific questions.
    - "How often do you use feature XXX?"
    - "What are the three features you would most like to see?"

- **Interviews**
  - At the end, when you have a better idea of what you will be doing and have some good questions that requires detailed answers. Interviewing is very labour intensive. So it is important to wait until the end, so you won't waste your own or other's time
  - [Goguen] – watch for the "say-do" problem - or what is called tacit knowledge – when you know how to do something but can't describe it, *e.g.,* like tying your shoe

# Elicitation Techniques

1. Reuse old requirements or existing system
2. Questionnaire
3. Interviews
4. Observation and apprenticeship
5. Ethnographic studies
6. Brainstorming
7. JAD: Joint Application Design
8. Nominal group technique
9. Delphi technique
10. PIECES Approach

# Questionnaires

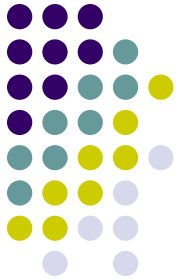


- Good for large groups when you have specific questions
- **NOT** appropriate as the only way, they suffer from being a *one way* communication technique and suffer *time lag*:
  - You cannot do follow up questions
  - You cannot follow users down interesting paths/points that they might raise during their answer

# Good Usages of Questionnaires

- Good uses of questioners include:
  - Survey of currently used features
  - Reasons for not using specific features
- Questionnaire can be used to establish usability requirements and priorities of features
- You should use interviews to create useful questionnaire for example:
  - "What new features would you like to see?" Open ended and needed... but then you can create a questionnaire and send to users for voting

# Common Questionnaires Mistakes



- Bias in sample selection
- Bias in responding users
- Small sample size
- Untested questions that are ambiguous