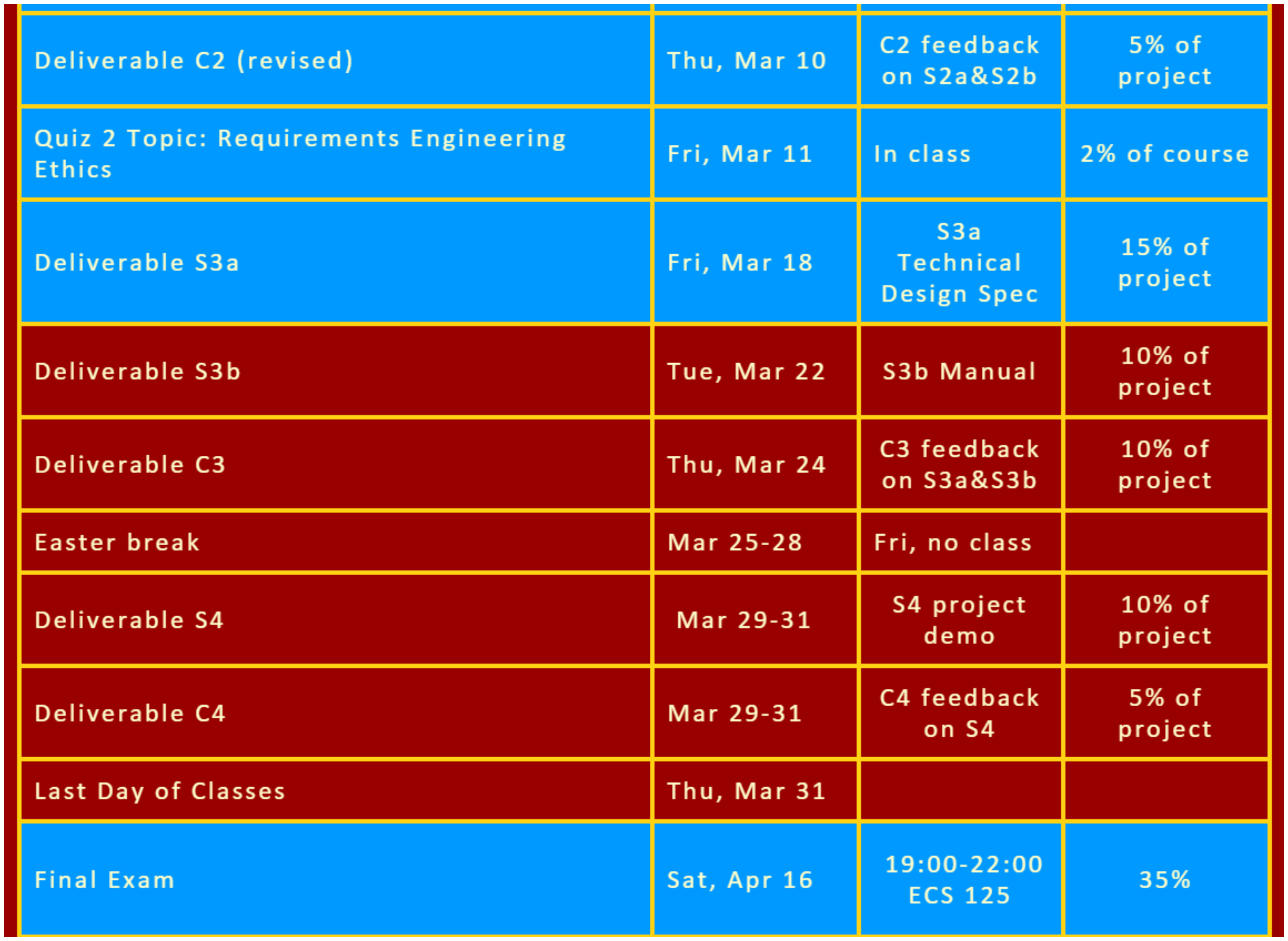
Professor Hausi A. Müller PhD PEng FCAE
Department of Computer Science
Faculty of Engineering
University of Victoria

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

SENG 321 Calendar



Deliverable C2 (revised)	Thu, Mar 10	C2 feedback on S2a&S2b	5% of project
Quiz 2 Topic: Requirements Engineering Ethics	Fri, Mar 11	In class	2% of course
Deliverable S3a	Fri, Mar 18	S3a Technical Design Spec	15% of project
Deliverable S3b	Tue, Mar 22	S3b Manual	10% of project
Deliverable C3	Thu, Mar 24	C3 feedback on S3a&S3b	10% of project
Easter break	Mar 25-28	Fri, no class	
Deliverable S4	Mar 29-31	S4 project demo	10% of project
Deliverable C4	Mar 29-31	C4 feedback on S4	5% of project
Last Day of Classes	Thu, Mar 31		
Final Exam	Sat, Apr 16	19:00-22:00 ECS 125	35%

Announcements

- Thu, March 10
  - C2 due
  - Feedback on S2a & S2b

- Fri, March 18
  - S3a due
  - Detailed technical design
  - Include ethics requirements
  - NOT A MANUAL (!)

- Quiz 2
  - Fri, March 11 (today)
  - In class
  - Requirements engineering ethics

- Final Exam
  - Sat, April 16
  - 19:00-22:00
  - ECS 125

Professional Engineers and Geoscientists of BC

www.apeg.bc.ca

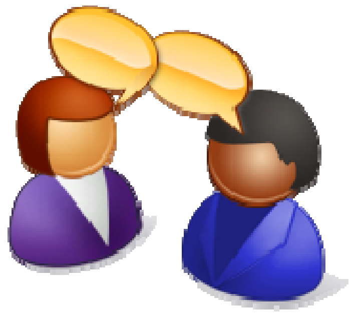
Search for APEGBC Code of Ethics

1) Hold paramount the safety, health and welfare of the public, the protection of the environment and promote health and safety within the workplace;

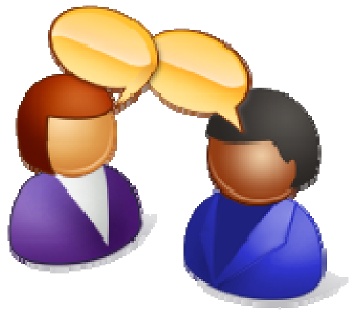
2) Undertake and accept responsibility for professional assignments only when qualified by training or experience;

3) Provide an opinion on a professional subject only when it is founded upon adequate knowledge and honest conviction;

4) Act as faithful agents of their clients or employers, maintain confidentiality and avoid a conflict of interest but, where such conflict arises, fully disclose the circumstances without delay to the employer or client;



5) Uphold the principle of appropriate and adequate compensation for the performance of engineering and geoscience work;



6) Keep themselves informed in order to maintain their competence, strive to advance the body of knowledge within which they practice and provide opportunities for the professional development of their associates;

7) Conduct themselves with fairness, courtesy and good faith towards clients, colleagues and others, give credit where it is due and accept, as well as give, honest and fair professional comment;

8) Present clearly to employers and clients the possible consequences if professional decisions or judgments are overruled or disregarded;

9) Report to their association or other appropriate agencies any hazardous, illegal or unethical professional decisions or practices by members, licensees or others; and

10) Extend public knowledge and appreciation of engineering and geoscience and protect the profession from misrepresentation and misunderstanding.

Software Engineering Code of Ethics and Professional Practice

- Software engineers shall commit themselves to making the analysis, specification, design, development, testing and maintenance of software a beneficial and respected profession. In accordance with their commitment to the health, safety and welfare of the public, software engineers shall adhere to the following Eight Principles:

1. **PUBLIC - Software engineers shall act consistently with the public interest.**
2. **CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.**
3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. **PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.**
7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
8. **SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.**

http://www.acm.org/about/se-code

Review Techniques

- Reading and signing off
- Walkthroughs
- Formal inspections
- Focused inspections
- Active reviews
- Checklists

Inspection Checklists

Checklist for Requirements Specification Reviews

Organization and Completeness

- ❑ Are all internal cross-references to other requirements correct?
- ❑ Are all requirements written at a consistent and appropriate level of detail?
- ❑ Do the requirements provide an adequate basis for design?
- ❑ Is the implementation priority of each requirement included?
- ❑ Are all external hardware, software, and communication interfaces defined?
- ❑ Have algorithms intrinsic to the functional requirements been defined?
- ❑ Does the specification include all of the known customer or system needs?
- ❑ Is the expected behavior documented for all anticipated error conditions?

Correctness

- ❑ Do any requirements conflict with or duplicate other requirements?
- ❑ Is each requirement written in clear, concise, unambiguous language?
- ❑ Is each requirement verifiable by testing, demonstration, review, or analysis?
- ❑ Is each requirement in scope for the project?
- ❑ Is each requirement free from content and grammatical errors?
- ❑ Is any necessary information missing from a requirement? If so, is it identified as TBD?
- ❑ Can all of the requirements be implemented within known constraints?
- ❑ Are any specified error messages unique and meaningful?

Quality Attributes

- ❑ Are all performance objectives properly specified?
- ❑ Are all security and safety considerations properly specified?
- ❑ Are other pertinent quality attribute goals explicitly documented and quantified, with the acceptable tradeoffs specified?

Traceability

- ❑ Is each requirement uniquely and correctly identified?
- ❑ Is each software functional requirement traceable to a higher-level requirement (e.g., system requirement, use case)?

Special Issues

- ❑ Are all requirements actually requirements, not design or implementation solutions?
- ❑ Are all time-critical functions identified, and timing criteria specified for them?
- ❑ Have internationalization issues been adequately addressed?

Inspection Moderator’s Checklist [Wiegers]

- **Things to Bring to the Inspection Meeting**
  - Inspection summary report
    - Inspection identification
    - Work product description
    - Inspector names and roles
    - Pages or lines of code planned for inspection
    - Total overview effort
    - Planning effort filled in
  - Typo list for participants to share
  - Issue log for the recorder
  - Inspection Lessons Learned questionnaire
  - Attention-getting device (e.g., gavel, mallet, whistle)
  - Easel paper and markers for action items and other issues that come up
  - Appropriate work product defect checklist or rule set
  - For a re-inspection, the issues list from the previous inspection

Inspection Moderator’s Checklist [Wiegers]

- **At the start of the inspection meeting**
  - Introductions. Identify the moderator, author, and the individuals performing the reader and recorder roles. Announce the work product being inspected and state the author’s inspection objectives.
  - Author created this product and asked us to help make it better. Please focus your comments on improving the product. Look beneath the superficial minor defects or style issues, to hunt out significant defects. If you aren’t sure, point it out and we’ll decide as a team.
  - Our goal is to identify defects, not devise solutions. In general, permit about 1 minute of discussion on an issue to see if it can be resolved quickly. If not, ask that it be recorded. Typos or small cosmetic problems should be recorded on the typo list, rather than come up in the discussion.

Inspection Moderator’s Checklist [Wiegers]

- **At the start of the inspection meeting**
  - Only one person to speak at a time; no sub-meetings. Explain the attention-getting device. Ask inspectors to respect the moderator’s interruption role.
  - Author to ascertain that everybody has the same version of the document being inspected.
  - At the end of the meeting, decide what our appraisal of this product is: accepted as is, accepted conditionally, re-inspection needed, or inspection not completed. Describe how the group will make the appraisal decision (e.g., 5% rule). Take a few mins to discuss lessons learned from the inspection at the end of the meeting.
  - Record everyone’s preparation time on the inspection summary report and add them up to get the total preparation effort. Judge whether it is sufficient to proceed with the meeting or whether you should reschedule it.
  - Ask for any positive comments they wish to make about the initial deliverable. For any global observations that pertain to the entire document.

Inspection Moderator’s Checklist [Wiegers]

- **At the end of the inspection meeting**
  - Prepare product appraisal and record it on the inspection summary report.
  - If the appraisal was “accepted conditionally”, determine who will peform follow-up
  - Record the actual pages or lines of code inspected.
  - Collect lessons learned from this inspection.
  - Remind inspectors to pass their typo lists to the author before they leave.
  - If a separate action items list was generated, deliver it to the appropriate individual(s).
  - Record the total number of major and minor defects found, and the number of major and minor defects corrected from the author.
  - Enter defect and issue details into inspection database.

Inspection Checklists [Karl Wiegers]

- Completeness
  - Does the document contain all the information called out in the outline for the SRS (e.g., IEEE SRS standard)?
  - Do requirements exhibit a clear distinction between functions and data?
  - Do requirements exhibit a clear distinction between functional and none-functional requirements?
  - Are there sufficient use cases included?
  - Are there areas not addressed in the SRS that need to be?
  - Do the requirements exhibit the different stakeholder groups?
  - Do the requirements exhibit the different domains involved?
  - Have the real-time constraints been specified in sufficient detail?
  - Has the precision and accuracy of calculations been specified?
- User interface
  - Do requirements define all the information to be displayed to users?
  - Can the user specify preferences? Statically, dynamically?
  - Are there sufficient use cases included?
  - Do requirements address system and user response to error conditions and exceptions?

Inspection Checklist [Karl Wiegers]

- Ambiguity and consistency
  - Is each requirement stated clearly, concisely, and unambiguously?
- Validation and verification
  - Is each requirement testable, verifiable, and traceable?
  - Is it possible to develop a thorough set of tests based on the information contained in the SRS? If not, what information is missing?
- Tacit knowledge
  - Are there ambiguous or implied requirements”
  - Have assumptions and dependencies been clearly stated?
- Complexity
  - If the requirements involve complex decision chains, are they expressed in a form that facilitates comprehension (i.e., decision tables or decision trees)?
  - Are there conflicting requirements?
- Adaptation
  - Are there requirements for software upgrades?
  - Are there requirements for dynamic adaptation?
- Unessessary constraints
  - Are there requirements that contain an unnecessary level of design detail?
  - Are there unnecessary “what”, “when”, “implementation” details?

General SRS Checklist

- Is a functional overview of the system provided?
- Are sufficient UML diagrams included?
- Have the software and hardware environments been specified?
- Is there a clear delineation between the system and its environment?
- If assumptions that affect implementation have been made, are they stated?
- Has every acronym, constant, variable, and timeout been defined in the Data Dictionary?
- Are all the requirements, interfaces, constraints, or definitions listed in the appropriate sections?

Structure Check

- Does the specification contain:
  - A number or ID for each requirement for ease of reference
  - Verifiable requirements
  - Purpose of each requirement
  - Use cases
  - Examples of ways to meet requirement
  - Plain-text explanation of diagrams
  - Importance and stability for each requirement
  - Cross refs rather than duplicate information
  - Index
  - An electronic version

Interface Checklist

- Are all inputs to the system specified, including their source, accuracy, range of values, and parameters?
- Are all outputs from the system specified, including their destination, accuracy, range of values, parameters and format?
- Are all screen formats specified?
- Are all report formats specified?
- Are all interface requirements between hardware, software, personnel, and procedures included?
- Are all communication interfaces specified, including handshaking, error-checking, and communication protocols?

Class Diagram Checklist

- Have the multiplicities of all associations been considered?
- Are each object's attributes really data values?
  - Attributes that are objects themselves should be modeled using aggregation.
- Have generalizations of objects been considered?
- Are each object's operations really operations on the object's attributes?
  - Operations invoked by the object's dynamic model are often operations on another object's data.

Statechart Checklist

- Are there any state-transitions that can never occur (because the event never occurs)?
- Are all remote operations (of other objects) invoked as either an action or a message?
- Are all attributes used declared in the object model?
- Are all operations called declared in the object model?
- Do all superstates have an initial state?
- Have special states (e.g., abnormal termination) been considered?

Non-Behavioural Requirements Checklist

- Is the expected response time (from user's point of view) specified for all operations?
- Is the level of security specified?
- Is the reliability specified, including the consequences of software failure, the vital information that needs to be protected from failure, and the strategy for error detection and recovery?
- Is the maximum memory specified?
- Is the maximum storage specified?
- Are planned changes specified (i.e., maintainability)?
- Are acceptable trade-offs between competing non-behavioral properties specified?