CMPT 371 – Software Management

Course Outline

Meeting Times

Lectures: Tuesday/Thursday 1-2:30pm in Thorvaldson 205A
Tutorials: Tuesdays, 4-4:50pm in Thorvaldson 205A

Instructors

Class Instructor: Nathaniel Osgood, Thorvaldson 280.6, 966-6102, *osgood@cs.usask.ca*
Office Hours: Thursday 2:30 – 3:30pm (starting week of January 9) and by appointment.
Teaching Assistant: Allen McLean (*jam532@mail.usask.ca*)

Texts

Because of the diversity of material and topics covered, there are texts that are suggested as useful for CMPT 371. Each of the books listed is an good reference for future work, but neither is required.

McConnell, S. (1998). *Software Project Survival Guide*. Redmond, Wash, Microsoft Press. ISBN 1-57231-621-7. *Available through the U of S library*.

DeMarco, T. and Lister, T. *Peopleware: productive projects and teams*. Dorset House, 1999, 2nd edition. ISBN: 0932633439. *Available through the U of S library.*

Course Contents

The course follows up but greatly expands upon the software engineering issues discussed in CMPT 370. While there will be continued and refined emphasis on the technical side of software engineering, the focus for this course is on the critical process and human side of software engineering: coordination and management of the software development process, project planning and estimation, risks and risk analysis, team building, contract design and negotiation, testing and software quality assurance, productivity, behavioral considerations, software configuration management, deployment and maintenance, training and help, software process standards, and software process improvement.

The completion of a significant group project, and the management and documentation of this project, and the staged delivery of quality products, are an essential part of the class. Assignments and class participation will also contribute significantly to students' grades.

All students must be properly registered in order to attend lectures and receive credit for this course.

Project

As described more fully in the final section of this document, the project will be carried out in large project groups. There will be several stages in the development of the project (note that several of the stages are on-going throughout the project and some may be done in parallel):

- Planning new iterations in the development of the project, including adding new requirements. These iterations should be risk-driven, taken into account value delivered to users and salient uncertainties. The students required to offer incremental deliverables at least 5 times during the term; this is beyond the infrastructure update and description required two weeks following project initiation. To help keep

you on course, we recommend (but do not require) more frequent internal deliverables, associated with sprints of 1 or 2 weeks in duration overseen by the project manager.

- Submitting a fully documented, rigorously tested and critically reviewed version of the system *for each incremental deliverable*. The feature sets associated with each such deliverable are less important than its reliability and conformance to stated requirements.
- Handing in project documentation at each of the 5 incremental deliverables required to provide information to the instructor and tutor as to the state of the project.
- Clear *documentation of requirements*
- Simple *prototypes f*or the UI (mostly on paper) and for areas of particular risk
- Initial and ongoing contact with clients/users to validate the requirements and receive feedback on product progress. A documented form of this contact should take place at *least one time per incremental deliverable*. Where this is not possible due to continued unavailability of or lack of communication from a client for more than one iteration of the project, please notify the instructor immediately.
- Meeting multiple times during the term with "senior management" (instructor) in a series of briefings to interact with them on the state of the project (please note that a possible consequence of such meetings could be changed or new requirements added during the project)
- Aggressive up-front and ongoing *risk management*, including through an accountable risk officer. This should include risk scanning on a periodic basis (at least once for each deliverable).
- Creation and updating of the identity and status of clear binary "*mini-milestones*" for the product. Each such mini-milestone should indicate that a task is *either complete or not complete*.
- Use of the *git version control* system for use by all team members. It is essential that this system be used in a way that fosters not just individual version control, but team-wide coordination. The version control system will be available for ad hoc review by the instructor & tutor, and will be used for the following:
    a. Code Check-ins. Such check-ins will refer to design specification and the defect database
    b. an updated "Top 10 Risks" enumeration
    c. the project schedule
    d. task assignments to individuals
    e. mini-milestone list and status
    f. build/smoke test status
- Maintenance of a *continuous build regimen*, including *automated smoke testing*. There should be a
    a. clearly defined build team (may rotate)
    b. an infrastructure (such as with Maven, TeamCity, CruiseControl, Jenkins, Hudson, etc.) to provide feedback to the team on the state of the build and the broader state of the project.
    c. Access of the instructor & TA to the status reports
- Appointment of a **dedicated** *project manager* to help coordinate the project. Please note that for success of the project and project evaluation, important that this individual *not* seek to contribute to the

codebase. Penalties will be applied to projects in which the project manager seeks to play a role as a lead developer (formal or otherwise).

- Use of a *project wiki* (Atlassian confluence is available at https://wiki.usask.ca). Github and other systems will often maintain their own wiki functionality.
- Maintenance of a *defect database* (e.g., JIRA [http://www.atlassian.com/software/jira], Trac, Fossil, Redmine, Google Code) and/or issue tracking system (e.g., trac)
- A *logging framework* (e.g., log4j, log4js,Java Logging API).
- Optional but recommended: Use of a style-checking system (such as Checkstyle, http://checkstyle.sourceforge.net/) to be run together with (continuous integration) project builds.
- A series of *peer deskchecks and inspections on artifacts*.
    a. *Each person in the team must have at least one artifact that is the subject of an inspection; documentation on such inspections must be turned in at semester end. At least one formal inspection must take place and be documented per deliverable (see below).* Note that inspections are not limited to code, and can begin almost immediately on a project, including one the following:
        i. Risk inventory
        ii. Requirements documents
        iii. Testing plans
        iv. Test cases
        v. Design documents
- A strong investment in rigorous and extensive testing of each incremental deliverables. This includes the following:
    a. Consistent, ongoing and documented use of a form of **T*est-Driven Development***, in which *specifications and tests* are created for software prior to implementation of that software.
        i. A substantial component of the mark for the deliverable will be based on the presence and quality of the specifications and tests provided.
        ii. Beyond contributing a large portion of the course mark, the specifications will be of absolutely key value for writing tests, assertions, mocks, writing code working with classes or methods that are not yet written (but are specified in this form). What is expected is regular, reliable use of *english* specifications associated with classes and methods of that class that should be clear to an English speaker. These are specifications that aspire to be clear and minimally ambiguous, but are generally not sufficiently precise and structured to be understandable by a computer and therefore cannot be reasoned about automatically or subject to automatic theorem proving. These English specifications focus on *what* is being performed by the method and not *how*. Such specifications have several pieces.

The most important is a statement of the *purpose* of the construct (method or class) – *what* the construct does (abstracting away from *how* it does it [the implementation]). The specifications for a given construct will also include structured information that varies according to the type of construct being considered. For methods, this would regularly include *pre-conditions and post-conditions* (including return values and any changes caused from the initial state). For classes, *invariants* (e.g., this.size() always >= 1) and history properties (e.g., always stays the same – e.g., that an instance of this class is immutable, or if *a*.size() at time *t1* is *x* and *a*.size() at time *t2* is *y*, $t_1 <= t_2$ and $x <= y$) should be stated. It would be very desirable to see these stated for every method (even if the answer is "none"). When thinking about the *public* methods of a class (part of its interface, or contract), it's admirable and best to state these specifications up front, before the class is written – after all, by test-driven development, these pre-conditions and post-conditions and purposes will be a core part of the reasoning for writing the tests for methods to be added before those methods are written (hence shouldn't add hugely to the required work – you are just writing down many of the things you are thinking when writing the tests). Often when implementing the public methods of a class, you have to go and create other (private or public) methods of a class, and you can write the specification for those methods either immediately before you add that method in, or well ahead of time.

b. Widespread use of *assertions* in your code to operationally check the specifications above and other programmer assumptions.

c. Clear *planning and documentation of tests*, ideally prior to implementation, including an appointed testing team. This should include

   i. use of a *test matrix*

   ii. proactive investments in *testability* – planning not just the testing steps but also the interfaces (“hooks”) and scaffolding (“harnesses”) needed to drive the system through key elements of its functionality. Systems that “cannot be tested” at an integration, system or acceptance level because of technical challenges will be viewed highly unfavourably.

   iii. where possible, diagrams showing path coverage – whether at the level of testing higher-level features or functional blocks in the entire system or at the code level.

d. Demonstration of a close working relationship between testers and developers. Strongly recommended is a system where each tester serves to “buddy” with a developer, and serves to test the contributions of that developer throughout much of the lifecycle. The *developer will retain responsibility for providing unit tests and associated specifications*.

e. Demonstration of *ongoing delivery of code* to testers (and particularly “buddy” testers) throughout development of an incremental deliverable. Teams in which code is only made available to testers within the last day or two of an incremental deliverable will be subject to penalty.

f. *Rigorously testing* the product, and documentation of that testing, in the form of a series of documented, maintained, peer reviewed tests. While not all of these tests need to be automated (indeed, some should not be), they all need to be described and those that are automated need to have results checked. Students are encouraged but not required to develop most tests prior to creation of the software itself. Use of a testing framework, such as jUnit, nUnit, etc. is *required.*

g. For at least the final deliverable

    i. *an estimate of the fraction of undiagnosed defects* (using one of the testing methodologies discussed in class), and corresponding refinements to the QA process

    ii. Use of a *code coverage* testing framework (such as cobertura, Emma).

a. Given the different skillsets involved, we require that teams *maintain a set of dedicated testers*, who will devote themselves to testing projects. These would include, but are not limited to, the following:

    i. Planning test strategies & test cases
    ii. Selecting and familiarizing themselves with testing tools
    iii. Identifying test platforms
    iv. Ensure that Test-Driven development is being practiced
    v. Establishing appropriate logging
    vi. Helping maintain the continuous integration test suite

b. Organizing *bug parties* for social testing. Holding such parties with two different subsets of the team form a natural place to using the "pooling" method (covered in class) to estimate the count of undiagnosed defects.

c. Use of a *mocking framework* (e.g., JMock, Mockito)

- An *activity log* should be kept throughout the project to help provide the necessary information to compile the estimates of individual effort.
- Adherence to a documented process for governing *requirements changes.*
- Use of a "three strike" system that involves the use of (virtual) "Yellow" and "Red" cards to indicate escalated levels of warnings for individuals. Individuals with Red cards must talk to the instructor *immediately* to avoid high marking penalties.
- *Preparing for the deployment and maintenance of the software*, thorough user documentation, and an approach to user training.

Workload

The anticipated due dates and weighting for submissions for Project Related Deadlines and Other Assignments are as below; all dates indicate a deadline of *midnight* at the close of the specified day.

Project-Related Deadlines

Deliverable	Detail	% Mark	Due Date
Infrastructure Presentation	Description of chosen project, process components: issue tracking, wiki pages, continuous integration, smoke test and status reporting mechanisms, version control structure, tools (e.g., for mocking, GUI testing, logging, persistence, etc.)	0	Jan 24
Incremental Deliverable 1	Working minimal version of system with initial requirements, test cases and matrix. Reflects work planned for Incremental Deliverable 2.	7	Feb 5
ID1 Presentation	Presentation on Incremental Deliverable 1. Reflects work accomplished since Incremental Deliverable 1, and work planned for Incremental Deliverable 3.	0	Feb 7
Incremental Deliverable 2	Fully working version of system with modifications to support best practices and plans for Incremental Deliverable 2. Identifies work to be accomplished by Incremental Deliverable 3	7	Feb 17
Incremental Deliverable 3	Fully working version of system with added features.	7	Mar 5
ID3 Presentation	Update on Incremental Deliverable 3. Reflects work accomplished since Incremental Deliverable 2, and no need to work planned for Incremental Deliverable 4.	0	Mar 7
Incremental Deliverable 4	Fully working version of system incorporating added features	7	Mar 19
ID4 Presentation	Final Status: State of project near the end and deployment plan; should summarize progress over work over entire	0	Mar 21
Incremental Deliverable 5	Fully working version of system incorporating all features	7	Apr 6
ID5 Presentation	Final Status: State of project near the end and deployment plan; should summarize progress over work over entire	0	Apr 4
Post-Mortem Report		15	To be submitted as element of final exam

Other Deadlines

Deliverable	Detail	% Mark	Due Date
Pop Quizzes	Throughout term, in class	15	
Final Exam	Closed book	30	TBD

Participation

Classroom Participation

In addition to the above deliverables, a significant fraction (5%) of students' grades will be based on participation. In recognition of differences in communication styles and interests among students, this participation score will reflect interaction in lecture, tutorials, in project meetings at the beginning of class, and in office hours.

While it is understood that occasional absence from lectures is unfortunately sometimes necessary for health or other extraordinary considerations, students who are absent from class will be missing both course material of importance to functioning productively in the project and the team meetings to be held in the first 10 minutes of class (please see below). Students who are chronically absent from class sessions will naturally be at a great disadvantage when it comes to learning (and the marks that reflect it), but can expect both to receive very poor scores for participation, and will also be viewed as contributing less to the team project.

Project Participation

Participation in the team project is mandatory for all students. Students deemed to have inadequately contributed to the mandatory course project will receive an escalating pair of warnings, beginning with the receipt of a yellow (virtual) warning card from the instructor. It is the obligation of a student who receives a yellow warning card to make immediate efforts to arrange for a meeting with the professor or teaching assistant to discuss and resolve their situation. Failure by the student to adequately resolve the underlying concerns regarding participation can be expected to lead to the issuance of a red warning card from the instructor. Students receiving such a red card are at imminent risk of failure of the course, and it is incumbent open them to act immediately to make a meeting with the instructor to address these concerns, and to make all efforts to contribute to the project. Failure of a student to respond to either of these warnings will be taken as an indication that they agree that they are failing to contribute adequately to the project.

Peer Review of Student Artifacts

Students must have at least one artifact undergo a peer deskcheck review (either with or without the author present) for each deliverable. These artifacts may be pieces of code, but could also be other substantial components for the project, including design elements (including class diagrams), requirements documents, a risk management plan, tests, a testing plan, etc. The reviewer and author must sign off on a form that indicates the results of the review (including brief description of defects found or rework to be performed) when the review is complete. These reports must be submitted with the deliverable.

In addition, each team is repsonsible for conducting at least one formal inspection during each deliverable; each student is required to be present as a reviewer or other participant in at least two such inspections during the term:These inspections must be documented and signed off on by the inspection team. *Students whose artifacts are not inspected for the term or who do not participate in at least two peer reviews may have signifcant marks deducted, as may projects that fail to have at least one formal inspection per deliverable.* We suggest the use of the doodle scheduler (www.doodle.ch) for scheduling peer reviews.

Please note that some of the most critical of these meetings must be held prior to the first milestone, with the results reflected in that milestone and milestone presentation:

- Management plan review
- Risk management review
- Test Plan

- Design changes

Finally, there may be additional management meetings, to be held at a mutual agreeable time for students, instructor and TAs.

Topic Plan

Lecture slides will be provided via the course website when possible but are not guaranteed for all classes. There will also be some guest lectures scattered through the term, and frequent meetings with project management. There will also be tutorials as particular topics warrant (see below). A substantial portion of the class grade will reflect student participation, so, students should come to tutorial (as well as lecture) prepared to discuss the material recently presented that will be being presented in that day.

A preliminary lecture schedule is included below. The schedule has been designed to provide students with knowledge and skills that will enhance project success. Students are strongly advised to apply the techniques covered within class within their projects.

Please note that the schedule below is tentative and we expect that it will be revised throughout the term. Updated versions will be posted on the moodle site; in cases of larger changes, the class may be notified via email.

During some sessions of this semester's class in which the instructor's online videos are readily available, *we will be experimenting with a "flipped classroom" style of instruction* that has been positively used by the instructor in other classes. For such sessions, students will be *required* to review relevant material (including a *video*) prior to each session, and most classroom time will be devoted to activities that take key advantage of the shared classroom experience. These include – but are not limited to – assisting students with applying and adapting the ideas captured in the videos to their projects, providing feedback on project requirements, design, code and other issues, and addressing student questions , and assisting students with exercises, such as in refactoring, test and testability planning.

The instructor's videos from the the 2016 version of CMPT 371 may be particularly helpful.; a larger set of earlier recent lectures are also available from the 2014 version of CMPT 371.

Date	Topic
Jan 5	Introduction and Overview, Best Practices: Greatest Hits 1
Jan 10	Best Practices: Risk Management 1, Risk-Driven Incremental Delivery with mini-Milestones (Video) & Team Meetings
Jan 12	Best Practices: Daily Build & Smoke Test & Continuous integration
Jan 17	Specifications and separating interface from implementation
Jan 19	Essential Testing: Test Driven Development, Test Matrices, Testability, overview of test coverage estimating undiagnosed defects,
Jan 24	Best Practices: Assertions, Exceptions, Return Codes and (Monadic) Optional Values
Jan 26	Best Practices: QA Processes , Defect reporting, Traceability & triage, and testing status reports
Jan 31	Best Practices: Estimation: Decomposition , and Estimation: Final Comments
Feb 2	Best Practices: Scheduling
Feb 7	Best Practices: Meetings: Reviews & Inspections And Audits
Feb 9	Best Practices: Clean Coding 1: Value objects & side effects,
Feb 11	Test Case Design
Feb 16	Test Design Tips
Feb 21	Vacation
Feb 23	
Feb 28	Guest Lecture or video
Mar 2	Test Design 1
Mar 7	Debugging 1
Mar 9	Debugging 2
Mar 14	Guest Lecture
Mar 16	Testing 3
Mar 21	Project Control
Mar 23	Human Capital & Team Building
Mar 28	Managing teams & behavioral considerations 1
Mar 30	Managing teams & behavioral considerations 2
Mar 31	Systems thinking & project dynamics
Apr 4	Pressure, Efficiency and Effectiveness
Apr 6	Wrap-Up

Tutorials

Tutorial periods are to be used for team meetings, project presentations, and formal inspections.

Date	Topic
Jan 10	Team acquaintance. • The inventory of skills in your team • The identifying individuals for the accountable positions in the team • The projects which team members feel would be most interesting • A discussion of the technologies that you would anticipate using for such projects • Identifying times that the team can meeting regularly

	• Getting or obtaining contact information as required for ongoing team collaboration (e.g., google accounts, usernames, etc.) • Discussion of resources that you may wish to use.
Jan 17	Formal inspection (inform instructor & TA of material & location by previous day)
Jan 24	Infrastructure presentation
Jan 31	Team Meeting
Feb 7	ID 1 Presentation
Feb 14	Formal inspection (inform instructor & TA of material & location by previous day)
Feb 28	Team Meeting
Mar 7	ID3 Presentation
Mar 14	Formal inspection (inform instructor of location for each team, in case he wants to attend)
Mar 21	ID4 Presentation
Mar 28	Formal inspection (inform instructor of location for each team, in case he wants to attend)
Apr 4	ID5 Presentation

Marking

The term project, pop quizzes, class participation and final exam collectively account for 100% of the grade.

The grade of the course will be assigned on an individual and team basis. For the term project, the work is done by a set of students working together as a development team. The team grade of the term project is obtained from the term project documents. From the term project grade, each member will get an individual term project grade depending on her/his efforts and contributions as evaluated by her/his peers in the group. Each individual will receive a grade equal to the term project grade times a multiplier. This multiplier can be lower or greater than one. The average of the individual grades will be equal to the team grade. Specifically, to establish individual contributions, a peer evaluation is performed during the final project phase. This evaluations will ask each team member to distribute a financial "bonus" among the team members, provide a recommendation for each member of the team, evaluate the effect of each member to the morale of the team, evaluate the contribution of the team member to the term project and assign a "title" to each member of the team including the person filling out the form. There is the expectation that such evaluation will remain confidential. It is also expected that team members will behave professionally and honestly while filling the evaluation. No consultation is permitted with other team members when or before filling out the evaluation Working effectively in a team is a precondition to get a good grade, but in the case that circumstances on the team create a difficult environment, individuals who contribute very strongly to the team can still secure a good mark. *Nonetheless, it bears emaphasis that because this coefficient influences the points accrued to the student from across the entire group project, a student's level of involvement in the project may have a substantial impact on their final grade.*

10% per day may be deducted from late term project phases up to a maximum of seven days. Term project phases received more than seven days late may not receive any credit. Under certain circumstances extensions may be granted. Please contact the instructor or tutor prior to the due-date if an extension is required. Failure to complete the assigned course work will result in failure of the course.

In addition to the above, the course includes a final exam. Failure to write the final exam will result in failure of the course. The final exam will include both a take-home and an in-class portion.

Students are expected to adhere to University of Saskatchewan Academic Honesty policy. More information can be found at

http://www.usask.ca/honesty/pdf/dishonesty_info_sheet.pdf

Project Description

Project Overview

CMPT 371 is a course about the management of the software development process. Participation in a real-world project is important in helping students internalize course material.

The project in CMPT 371 will involve the *incremental* design and implementation of one of a specified set of projects with defined stakeholders. The CMPT 371 project will be carried out in randomly selected larger (7-11 member) groups. Effective management of such a large group to achieve stated project objectives will be critical to the success of the project. After the group has been put together, the first order of business will be to decide which project to develop. Deciding this will require consulting with the stakeholder for a project in which you are interested, and working towards an agreement with that stakeholder as to their willingness to work with the team. If several teams are interested in the same project, the stakeholder can choose with which subset of teams (possibly none at all) they wish to work with, based on the professionalism of those teams in the initial meetings, their time availability, etc.

A plan will have to be devised and put into effect to achieve the stated objectives. It is required that the plan will include at least 5 elaboration cycles, *each delivering a robust and stable version of the software*, and incorporating increasingly sophisticated functionality. As will be discussed in class, you will wish to prioritize these iterations such that they prioritize the early delivery of value, the resolution of pronounced risks, build team morale and spread team knowledge, etc. As the project proceeds, the plan will guide activities, but will always be subject to updating as circumstances change, so it will likely be quite specific only for the immediately approaching development cycles and fairly general for later iterations. In the end, the final project will be a high quality working system, fully documented and tested and measured, complete with a deployment plan (although it won't actually have to be deployed!)

The emphasis for this deliverables will be on ***quality*** rather than on quantity of features – each deliverable should be as close to "industrial strength" as you can make it, and include adequate documentation to permit extension by another team if so required. The higher quality that is expected will require extra levels of care and planning. For example, each person in the class will be required to have all significant artifacts peer reviewed (see below), and we expect formal defect tracking and defect removal estimation. *Please scope your projects accordingly.*

In addition to outside arrangements made amongst themselves, students should plan on holding brief time-boxed project "scrum" meetings in the first 10 minutes of each scheduled lecture period, unless otherwise noted. To make best use of these meetings, we suggest that students refer to the following reference:

http://www.martinfowler.com/articles/itsNotJustStandingUp.html

Tutorial periods are to be used for team meetings and for formal inspections (see schedule).

Workload

There will be 4 presentations and four project major milestones (incremental deliverables) during the course, and regular meetings with management. The requirements for each of these are now outlined.

Incremental Deliverable 1

Unless special arrangements have been made with the instructor, the system should be in a working and stable state for this and other milestones. Initial system with source code repository, continuous integration and smoke tests, a mini-milestone list, list of risks, risk report, requirements, documentation of team personnel roles (which may include more than one role per individual), documentation of any reviews conducted so far (e.g., on initial requirements or design). This should also mention meetings with stakeholders. A report of testing results should be included, including a defect reports for defects found. The deliverable should comment on any existing or planned tests, and any testability investments that currently exist or that are planned in the near future. Many of these testability investments could be hooks (mechanisms to allow tests to examine state which would otherwise be difficult to observed or logging mechanisms, efforts to secure separation of concerns, use of specifications & design-by-contract and other elements that foster testability. You should also give a sense as to how the system is to be tested – not just manual mechanism, and support for any mechanisms ("test harnesses") that are currently used to drive the system through key elements of its functionality, or that will be pursued prior to milestone 2.

A statement of requirements to incorporate in the next deliverables should be clearly spelled out. Paper or simple GUI prototypes should be presented to describe user-oriented functionality for that coming deliverable. The deliverable should also include a detailed plan for achieving the next milestone objectives. This plan should include a design and architecture, documentation of anticipated steps (mini-milestones) in carrying out the project and a critical path through this network. The plan may have sub-plans, corresponding to sub-goals established by the various work units into which the project has been sub-divided. An initial estimate of the size of the software for Incremental Deliverable 2 and the effort required to produce it should also be made.

The first milestone should include an activity report summarizing work undertaken by each group member to date, and the activity log (giving time estimates).

Presentation for Incremental Deliverable 1

The first incremental deliverable presentation will provide a project overview to management and other class members of incremental deliverable 1, and should include a demonstration of the system as it is currently present, as well as a demonstration of status report information.

Incremental deliverables 2 - 4

In milestones 2-4 all project artifacts created or modified since the previous milestone should be handed in, including updated planning, analysis, and design artifacts, as well as new artifacts: updated risk analysis (including results of new scanning), the test matrix and reports on integration and system testing conducted to date (including test results), and the latest measurements of software quality (e.g., results of code reviews). Code and specifications and associated unit tests must also be included. In a manner similar to what was done for milestone 1, you should indicate in detail what is planned to accomplish between now and the next milestone.

It is important that for the accompanying presentation that you summarize not only what has been accomplished for the work products, but also for refining the development *process* – what you have learned or changed in the development or management process.

Milestone 5 (Final Milestone)

Milestone 5 is similar to that for milestones 2-4 in contents, except that no summary of work to be accomplished prior to the next milestone is included. Milestone 5 should, however, include detailed discussion of the lessons learned from the project.

Presentation for Milestone 5

This presentation will provide a summary of milestone 5, and a demonstration of the final, completed system. Stakeholders may be present during any milestone presentations, but we particularly encourage having them here for this final milestone.

Project Postmortem

The project postmortem takes place at the completion of the project can be composed either by the individual or by the group. A given individual can also be represented both by a group postmortem and by personal reflections. The most important thing is that the post mortem should be thoughtful and reflective. The content is more important than length, but 4-5 pages should be enough in most cases. But if you have really salient additional points, you don't have to cap it. Please note the there is space to contribute an individual post-mortem as part of the final examination, but students may elect to treat that as a take home aspect part of the exam, and bring an already composed post-mortem to the examination.

The emphasis in the postmortem should be on learning -- what went right, what went wrong, and thoughts on how you might approach things differently to try to handle it better in the future. These reflections can be about process, technical matters, group organization, etc.

I would like you to include a few comments on contributions of others in the group, but these can be limited to at most 1-2 pages (and preferably just 1). Most of the emphasis should be on the other matters noted above.

Inspections

Each member of a team is expected to have at least one significant artifacts they create undergo peer review via a peer deskcheck per milestone. Artifacts to be reviewed include requirements specifications, design specifications, tests, code, bug fixes, etc. This review will require signoff by both parties involved, plus an indication of any issues that were identified that require changes (and subsequent peer review).

Each person must also have at least one artifact inspected via a formal process described in class and in the readings during the course of the term. This process will involve pre-inspection review of documents by team members, rigorous note-taking, and follow-up.

Some Important Issues

Group Work

Your group should divide the tasks in the project appropriately so that each member can work independently or in a smaller sub-group much of the time. However, effective group interactions will still be mandatory to complete the project successfully, as in CMPT 370 – only more so! You are encouraged to communicate electronically using your mechanism of choice in a private area. Each lecture meeting will provide an opportunity to meet during the first 10 minutes of class (12:55-1:05), but you should all also physically meet separately (at least once a week) to make sure that everybody is on the same page. An agenda for each such meeting should be prepared beforehand, and minutes of the meeting should be kept. If group problems (e.g., caused by personality conflicts, slackers, etc.) start to occur, they should be nipped in the bud before they develop into real obstacles to success. You should try to solve the problems yourselves, perhaps at the regular meetings. As a last resort, you should approach the “higher management” (tutor, and – if absolutely necessary – the professor). Remember that managing such a large group has its own special disadvantages (too many cats, all going their own direction) but also its

own special advantages (one problem worker doesn't usually bring down the whole group). While experience suggests that some members of the class may not participate fully at all times throughout the semester, they do so at their own peril, and it is important that the project members welcome those individuals when and if they choose to make an effort to contribute.

As noted below, the final exam will survey individuals in your team as to the roles played and levels of effort expended by yourself and your teammates.

Presentation Timing

Each of the presentations will be scheduled for a full hour. There should be 5 minutes at most for set up, 30-35 minutes for the presentation, and a 5-10 minute question period. Not all members of the team need to take part in any given presentation, but over the term everybody should stand in front of the class at least once.

Hand-in

At each project milestone, and in the final project, all appropriate artifacts should be handed in, using moodle. The project should be appropriately packaged so each component is clear. There should be some sort of overview document and/or index to help the marker to understand the various parts of the project (and in the final project this should also contain instructions for running the system). Critically, these should involve a readme file in the main folder to direct the reader to the appropriate starting point.

If you are providing an electronic deliverable on github, please be sure to provide access through the instructor's github username (*ndo885*). Similarly, documents shared via google docs or similar google services should be shared with the instructor's preferred google account (*nathaniel.d.osgood@gmail.com*). Sharing of deliverables via dropbox is not acceptable for this course.

Marking

The marking of term work in the course will be based upon meeting the objectives set above for each presentation, milestone, and the overall project. We will be looking for realistic outlook, sensible and responsive planning, comprehensible documentation aimed at practical communication for future teams that will extend this project, appropriate (communicational) use of the analysis and design techniques explored in earlier computer science courses (in particular CMPT 370), and in the later presentations and milestones also a solid risk analysis, thorough testing and measurement, good deployment planning (including training and help material). These artifacts should all be continuously updated as the project proceeds. The system developed at each milestone will be judged for its robustness, maintainability, usability, utility, and adherence to requirements.

Although all members of the group will share in the overall group mark for each presentation, at each milestone, and in the final project, it is possible that group members who do not contribute equally will be assigned only a portion of the group mark. Each of you will be asked on the final exam to indicate in both a qualitative and quantitative fashion the contribution of each member of the group contributed during the year. This information will be combined with the activity tables handed in during the course and appropriate adjustments will be made.