CSE4939W/CSE4940 Course Content, Objectives, and Requirements

This document details the requirements for each of the courses, and common requirements of the new CSE4939W/CSE4940 year long sequence for all CSE and CS majors starting in Fall 2012. In CSE4939W in the fall semester, student teams will spend 10 weeks related to non-programming aspects of the two semester project, to define all of the requirements necessary to prototype the project over the remaining 4 weeks of the first semester and continued into the second semester (CSE4940). CSE4939W has a writing requirement for each student (at least 15 single-spaced pages, edited one time; each student must write a total of 15 pages, the instructor edits it for technical details and writing style/grammatical errors, and the student turns in a revision). This 15 pages is not one single document but represents the various software documentation that the student has to write throughout the semester, including requirements specification, software quality assessment, design documentation, analysis of real-world issues, etc., that are written by each student and spread over the entire semester.

The deliverables for CSE4939W are:

- A detailed written specification for the project that clearly defines the objectives, goals, and scope, and may include software architecture diagram, use cases, GUI mockups, etc..
- Assessment of Software Qualities for the project that analyze your project with respect to its attainment of portability, reliability, evolvability, interoperability, etc.
- A detailed documented design for the project including software design patterns, UML diagrams, and entity relationship diagrams; all diagrams accompanied by written descriptions.
- Project management via the selection of a software process model (spiral, agile, incremental, etc.) and use of a document/source code control (github).
- Incremental deliveries of baseline system components during the first 8 weeks of the semester, with two prototype increments delivered at week 11 and week 14. During the first 8 weeks there will be three proof of concept incremental deliverables (weeks 4, 6, and 8) such as GUI mock-ups, database schemas implemented in a database system, along with various other hardware/software components that have been developed independently by different team members and with limited (or no integration). By the end of the first semester there is minimal integration of software components, with the objective to have all of the necessary infrastructure in place to complete the project in the second semester.
- All of the written documentation will be collected into a single document to be delivered at the end of the semester; this document can be available for the second semester of work, especially if there is movement among for CSE4940 in the spring semester.

Note: An important fact of this is for all of the team members to become familiar with all of the required technologies for the project, which may mean learning new programming languages, new development environments, reviewing and working with existing code, integrating to an external API, cloud, or web services, etc. The intent is by the end of the first semester that all of the preparatory work has been completed and that the team is ready to go for the second semester. This second semester in CSE4940 will concentrate on taking the project to a realistic fruition which will include its final deployment and demonstration at the School of Engineering Senior Design Day held in Gampel pavilion each Spring.
Project Teams and Software Process Models:

**CSE 4939W and CSE4940 Project Teams, Supervision, Sponsorship:**

- **Size:** Teams will range from 4 to 6 members, which will allow the full team dynamics to be explored, and in the second semester (CSE4940) if there is any movement among teams (or a team member takes a coop or internship), the intent with the larger team size is for the majority of the team to stay together as a unit.

- **Supervision:** All teams will be supervised and all projects graded by the primary instructor of the course. Projects proposed by CSE faculty members will have a second supervisor who will make recommendations regarding the grades to be used by the course instructor.

- **Sponsorship:** Some projects may be sponsored by industry and there will be an industry project manager and appropriate contact personnel that will interact with team members.

**CSE 4939W and CSE4940 Project Management Requirements:**

- **Document Version and Source Code Control:** github (http://about.github.uconn.edu/) will be available for use throughout both semesters to allow all documentation associated with the project (specifications, designs, code, documentation, presentations, user manuals, code, etc.) to be maintained electronically. The course instructor will set up a repository with access to team members and the instructor and other supervisors. Please make sure you keep local copies of documents and code on your own computers for backup purposes.

- **Software Development Model (SDM):** Each team is allowed to choose their own SDM (see Chapter 7 of [1]) to use throughout both semesters to take the project from its inception to the final delivery at the end of the second semester. There are many different SDM methodologies that have emerged over the history of computing: the *waterfall model* [1] involve classical phases (requirements, design, implementation, analysis, maintenance, etc.), with the requirement that each phase being completed before moving to the next phase – while obsolete in practice, it is important since it defined the different phase sin use for other modern SDMs; the *iterative model* [1, 2] that introduces feedback loops and cycles among waterfall phases allowing stakeholders to analyze and revise their solutions incrementally; the *spiral model* [1, 3] is a cyclical model of four stages to identify objectives and design alternatives, evaluate alternative and identify/deal with potential risks, develop and verify the next level product, and review results and plan for the next iteration; this process promotes the construction of multiple prototypes in an incremental fashion and brings application stakeholders into the earliest part of the process to contribute with respect to functionality and user interface design; the Unified Process Model [4], a conglomeration of iterative and incremental approaches that employs use case design to focus end user participation, creating multiple architecture and system views (e.g., akin to using various high-level diagrammatic techniques such as design patterns, activity diagrams, collaboration diagrams, etc.), and including the risk assessment process throughout; and, the *agile development lifecycle* [1,5], also available as a unified process model, blends the best aspects of many different process models and expanding the scope and role of stakeholders of all types to participate in a process that truly results in the desired application. Team are strongly encouraged to adopt a software model that promotes incremental development and assessment, such as spiral, agile, on the unified process model.
CSE4939W Specification and Problem Definition and Documentation Requirements:

- **Project Proposal with Objectives and Goals:** This one to two page paper along with a presentation in the second week of the semester is intended to define the problem, goals, objectives, and scope over the two semester period, and include a brief 5-7 slide presentation.
  
  **Software Project Proposal and Assignment and Example at:**
  http://www.engr.uconn.edu/~Esteve/Cse4939W/ProjProp.doc

- **Project Specification (Initial and Revised):** This details an overview of the proposed project organized in a set of three main components that includes:

  - **Purpose, Objectives, Goals:** This written document is intended for both designers and developers that contains a number of sections such as: introduction on the purpose, objectives, and goals of the project; a description of the operating environment such as hardware, software, platforms, unique features, limits, requirements, etc.; one or more UML use-case diagrams to show users and their interactions with the high level use cases for the project; user interfaces (mockups of screen shots) at a high level to demonstrate overall functionality, logical databases or other repositories, etc

  - **High-Level Software Architecture Diagram:** A diagrammatic figure that demonstrates the overall system and its components, also explaining hardware, software, technologies, architecture style (e.g., client server, cloud, web app, embedded, etc.)

  - **User-Based Specification:** This primarily consists of rapidly prototyped graphical user interfaces (via Visio, html, etc.) that can serve as the basis for a dialog between the users and developers in order to extract user requirements and detail required functionality and interactions between various components of the system from the GUI perspective. From a documentation perspective, the screens should be labeled figures organized into a logical order to demonstrate the user interface capabilities. Each screen shot should have an accompanying paragraph description to explain its purpose and usage.

  **Project Specification Document Assignment and Example at:**
  http://www.engr.uconn.edu/~Esteve/Cse4939W/ProjSpec.doc

- **Software Quality Analysis:** From any software engineering textbook (see [1], Chapter 2), there are a number of vital software qualities, namely: Performance, Portability, Understandability, Productivity, Reliability, User Friendliness, Robustness, Repairability, Reusability, Maintainability, Interoperability, and Evolvability. A subset of these qualities should be selected (one per team member) to be analyzed with respect to: The importance and relevance of the quality to your problem/domain; and the attainment and/or incorporation of the quality in your specification and eventually in your design and implementation.

  **Software Quality Analysis Assignment and Example at:**
  http://www.engr.uconn.edu/~Esteve/Cse4939W/ProjSWQ.doc

**Note:** Over the course of the entire semester, new versions of these various documents (using github as a repository) may need to be created to reflect changes in overall project requirements, software architecture, user interfaces and flow, and relevant software qualities.
CSE4939W Design Documentation Requirements: The project design will represent a series of software design artifacts, models, and techniques, accompanied by associated written documentation, that fully describes the software design as a lead in to the development and testing stages. These requirements will evolve through the entire first semester, with the goal of having a relatively complete design by the 10th week of the semester. This design will continue to be updated in the remainder of the semester and during the second semester (CSE4940) as the direction and approach changes based on problems identified, new requirements, etc.

- **Project Design (Initial and Revised):**
  - **Software Design Patterns (SDPs):** Software design patterns typically show *relationships* and *interactions* between classes or objects at a conceptual level, for domain independent structure and behavior. A pattern provides a body of knowledge on a particular structure thereby communicating insight on a portion or component of a solution; the idea is that we can leverage patterns that are generalized from prior solutions and experience to build solutions more effectively. Each project will need to identify, define, explain, and document a set of one or more SDPs including: High-level Patterns such as Model View Controller or MVC with Observer; Creational Patterns such as Abstract Factory, Builder, etc.; Structural Patterns such as Adaptors, Proxy, etc; Behavioral Patterns such as Interpreter, Mediator, etc.; and Concurrency Patterns such as Event-Base Asynchronous, Scheduler, etc.
  - **Detailed Design:** This includes software and database design. For software design, you are to employ a wide range of UML diagrams as appropriate for your particular project and its domain requirements:. Structural UML diagrams include: Class, Component, Object, Profile, Composite Structure, Deployment, and Package. Behavioral UML diagrams include: Activity, State Machine, Sequence, Communication, Interaction, and Timing. For database design, you are required to construct an entity-relationship diagram (*ERD*) that for the conceptual structure of the data and its interrelations to be used to eventually generate a relational database schema. Note that not all projects may have all of these diagrams – but each project should be able to select at least enough different software and database diagrams so that each team member is responsible for one aspect of the software design as realized by a respective diagram.
  - **Combined Design Document:** The SDPs, ERD, and UML diagrams (design artifacts) for a project must be assembled into an overall design must be coherently organized into numbered sections with all diagrams as labeled figures that are explained and referenced within the text of the design. This will include a section that brings the design together by discussion the relationship and dependencies between all of the design artifacts.

Notes:
- A team must have at least one SDP and a subset of the ERD and UML diagrams.
- Each team member is responsible for one of these diagrams.
- Each SPD, ERD, and UML diagram must be accompanied with a textual explanation.
- Over the course of the entire semester, new versions (using github as a repository) of these various design artifacts must be created to reflect changes and modifications over the course of the semester.

Project Design Document and Assignment Example at:
http://www.engr.uconn.edu/~Esteve/Cse4939W/ProjDesign.doc
CSE4939W/4940 - Team Management and Prototyping Requirements

- Prototyping and Management Plan: For each prototype, you must identify the major components of your system that are to be designed/implemented. These components can be listed as bullet items with short explanations. Clearly, later prototypes will include the functionality of earlier prototypes. Estimation is always a difficult task in any software project. When defining the functionality of each component you should avoid being overly optimistic by erring on the side of caution. Remember, it’s usually better to underestimate your capabilities and deliver either intended functionality earlier or more functionality on time. To accomplish this, you must assign final implementation responsibilities among the team members. For each component and sub-component of your project you need to clearly identify the division of responsibility by outlining the components of the system each person is designing and implementing. In addition to identifying individual responsibilities, it is also necessary that you develop a preliminary integration plan to understand the transition between your three prototyping versions. In this case, you are performing a form of risk management by identifying potential problems before they occur. The two facets of the management plan are summarized as follows:

1. Clearly identify each person’s responsibilities in the implementation by indicating exactly which components of the system each person is implementing and the boundary (interfaces) between the components. Include an indication of the components that each member is implementing either as a primary or a backup software engineer.

2. You must design a plan for integration of the individual components of the system. The integration plan must include concrete, ordered steps for the integration as you transition through all prototyping versions. You must identify any problems anticipated during the integration process and include a discussion of the potential problems.

Your plan will need to include three prototypes for the end of the first semester (weeks 9, 11 and 14) and five prototypes for the second semester (CSE4940 – weeks 5, 8, 10, 12, 14) with the last prototype being at least an alpha version (beta preferred).

- Prototype Presentations and Reports: Each prototype will require the following:

  ➢ Presentation and Demonstration: A presentation that reviews your project goals and objectives (including overall architecture, technologies, screen shots, etc.), the functionality you intended to implement (based on your prototyping and management plan), the functional you actually implemented, the changes to the subsequent prototypes for the remainder of the project (revision of the prototyping and management plan), and a demonstration of the prototype and its functionality. The initial prototype may have separate components with minimal interactions while the final prototype should be full tested and integrated.

  ➢ Assessment: This is a team document that is submitted to address the following issues:

    1. Update of the prototyping/management plan that clearly shows your progress.
    2. Impressions of the language/technologies chosen and their ease of use. Include both positive and negative impressions.
    3. Evaluation of languages/technologies against software reuse and software evolution concepts.
    4. Critique of teamwork experiences - what has worked and what hasn’t.

Project Prototype/Management Plan Assignment and Example at: http://www.engr.uconn.edu/~steve/Cse4939W/ProjPTMGMTPlan.doc
CSE4940 Separate Documentation Requirements:

CSE4940 Design Documentation Requirements: In the second part of the course, the teams will submit updated design documentation at weeks 2, 8, and 14. This can be accomplished by reverse engineering from your programming language code (generate UML) or the database (generate Create Statements, Relational Schema, or ER Diagram). You need to update the last design that you submitted in CSE4939W for week 2, and provide subsequent updates in week 8 and 14.

http://www.engr.uconn.edu/~Esteve/Cse4939W/ProjDesign.doc

CSE4940: Assessment of Realistic Issues: This assignment will focus on assessing the issues for your project as related to product development and commercialization. Sample issues include: funding, commercialization, intellectual property, legal/ethical issues, software licensing issues, open source, payment, security issues, etc. Each team member will explore one of these issues relative to their project and its scope.

Assessment of Realistic Assignment Issues and Example at:
http://www.engr.uconn.edu/~Esteve/Cse4939W/ProjRI.doc
Project Deadlines

Project Specification/Documentation:

CSE4939W Documentation
1. Project Proposal:
   - Initial – 3 days after 1st Class
   - Final – 2nd Class; includes a 5-7 slide presentation
2. Project Specification
   - Purpose, Goals, Objectives
   - High-Level Software Architecture Diagram
   - User-Based Specification
     - Week 3: Initial Version
     - Week 5: Revised Version
3. Software Quality Analysis
   - Week 4: Initial Version
   - Week 6: Revised Version
4. Software Design
   - Software Design Patterns
   - Detailed Design (UML/ER)
     - Week 5: Initial Version
     - Week 7: Revised Version
   - Combined Design (UML/ER)
     - Week 8: Only Version
5. Prototyping/Management Plan
   - Week 7: Initial Version
   - Week 8: Revised Version
   - Week 14: Final Version for CSE4940 in Subsequent Semester
6. Final Documentation (Week 14)
   - Updated Spec and Design Docs
   - Final Report and Assessment
   - Initial User Manual

 Prototype Deadlines:

CSE4939W Prototypes
1. Week 9: GUIs, Mockups, etc.
   Update PT/Mgmt Plan
2. Week 11: Partially Integrated
   Update PT/Mgmt Plan
3. Week 14: Adding Functionality
   - Organized Code Repository
   - Set up for Second Semester Implementation
   - Provide PT/Mgmt Plan for 4940

CSE4940 Prototypes
All Prototypes will have a presentation and report on progress, status, and planned work.
1. Week 1: Demo of Project/Adjust Teams
2. Week 2: Revise Design Based on PT Efforts in 4939W Prototype/Management Plan
3. Week 4: Assessment of Realistic Issues
4. Week 5: Increment with Testing Module Testing with JUnit
5. Week 8: Increment with User Interface/GUI Testing Revised Design
6. Week 10: Increment with Testing
   a. Alpha Version
   b. Turned over to another team for Black Box Testing
   c. Changes/Improvements Finalized
7. Week 12: Increment with Testing
   a. General Performance Testing
   b. Beta Version
   c. Final System Testing
8. Week 14: Final Version of Semester
   a. Revised Design
   b. Testing: Demonstrating Software Qualities: maintainability, evolvability, and user friendliness
   c. Presentation at SDP Day
   d. Final Project Report for Semester
Course Requirements for CSE4939W/CSE4940 Year Long Sequence

Final Documentation: CSE4939W
http://www.engr.uconn.edu/~steve/Cse4939W/ProjFinRepUMan.doc

Team Submission: As a team, you are to hand in the following:

1. **Spec and Design Documents:** Updated version of specification and design documents that reflect any significant changes.

2. **Final Report:**
   - A general overview of what the project is and its goals. This is intended to give the general reader the background for your project and integrate all the pieces (different projects) into a consistent story.
   - A summary of all major changes as compared to your original intent. These are essentially changes to your specification which have a direct impact on your final prototype.

3. **Team Assessment:** Each team must answer the following questions in a self contained word-processed document, limited to 4 single-spaced pages:
   - (1 pages) What did you/your team accomplish in this course this semester?
   - (1 pages) What have you/your team learned through your accomplishments listed in response to the previous question?
   - (2 pages) If you/your team had to do the whole project again, what would you do differently or change? Include both the successes (what has gone right so far) and failure (problems).
   - http://www.engr.uconn.edu/~steve/Cse4939W/TeamAssessment.doc

Make sure that you include individual reflections for each student in terms of accomplishments, learning, differences/changes.

4. **Individual and Team Contributions for the Semester:** Please see the link at:
   http://www.engr.uconn.edu/~steve/Cse4939W/TeamAssessment.doc

5. **Individual Self Assessment:** Please see the link at:
   http://www.engr.uconn.edu/~steve/Cse4939W/IndividualAssessment.doc

6. **Developer Documentation:** Complete and detailed instructions on the required technologies and platforms for your system, OS, programming tools/environments, database, server/web technologies, build instructions, etc. All the requirements necessary to set up the development environment and construct the current build.

7. **Initial Users Manual:** An initial user's manual to guide the user through the use of your system.
Final Documentation: CSE4940

Team Submission: As a team, you are to hand in the following:

1. Final Design Documents: Updated the final time for submittal of completed code.
2. Realistic Issues Assessment: Include submittal from 4th week.
3. Final Report:
   - A general overview of what the project is and its goals. This is intended to give the general reader the background for your project and integrate all the pieces (different projects) into a consistent story.
   - A summary of all major changes as compared to your original intent. These are essentially changes to your specification which have a direct impact on your final prototype.

4. Team Assessment: Each team must answer the following questions in a self-contained word-processed document, limited to 4 single-spaced pages:
   - (1 pages) What did you/your team accomplish in this course this semester?
   - (1 pages) What have you/your team learned through your accomplishments listed in response to the previous question?
   - (2 pages) If you/your team had to do the whole project again, what would you do differently or change? Include both the successes (what has gone right so far) and failure (problems).
   - http://www.engr.uconn.edu/~steve/Cse4939W/TeamAssessment.doc

Make sure that you include individual reflections for each student in terms of accomplishments, learning, differences/changes.

5. Individual and Team Contributions for the Semester: Please see the link at:
   http://www.engr.uconn.edu/~steve/Cse4940/TeamAssessment.doc

6. Individual Self Assessment: Please see the link at:
   http://www.engr.uconn.edu/~steve/Cse49OX/IndividualAssessment.doc

7. Developer Documentation: Complete and detailed instructions on the required technologies and platforms for your system, OS, programming tools/environments, database, server/web technologies, build instructions, etc. All the requirements necessary to set up the development environment and construct the final build.

8. Final Users Manual: An initial user's manual to guide the user through the use of your system.
References: