1 Instructor

Dr. Tom Nurkkala
Associate Professor, Computer Science and Engineering
Director, Taylor Center for Missions Computing
Office: Euler Science 211
E-Mail: `tnurkkala@cse.taylor.edu`
Phone: 8-5163
Hours: MW 2:00-4:00, R 1:00-2:00, or by appointment

2 Objective

Software Studio trains you as a world-class software engineer. We do this by engaging you in an agile, team-based software engineering organization that develops and delivers non-trivial software for real, external partners. As a result, you will acquire:

- Improved skills in all phases of the software development lifecycle
- Direct experience developing non-trivial software that is of sufficient maturity and quality to be deployed and used by our external partners
- Concrete skills in project management, software configuration management, and project feature and defect tracking
- Personal time management and tracking skills
- A deeper understanding of software engineering from current and historical literature on software engineering, programming, process, management, and related areas.
- Better written and spoken communication skills as you interact with your peers, our partners, and your instructor
- Specific skills in the languages, middleware, databases, and related technologies associated with our project work

In short, I want you to master all the practical and theoretical skills that will make you someone I would be delighted to hire for a challenging and responsible position as a professional software engineer.

3 Culture

Studio inculcates a definite culture. Following are some of Studio's cultural characteristics.

3.1 Software Studio Delivers Working Products

Many of your classes feature "projects" of different size and complexity. Software Studio is not one of them. Instead, our focus is on *products*. Whereas a *project* implies a kind of nebulous open-endedness, a *product* is *working software* that *satisfies its design requirements* and is *delivered* to a *delighted* customer.

3.2 Software Studio is Opinionated

The goal of Software Studio is to train you to be world-class software engineer. To get there, you need to know and wield powerful, world-class software tools.

Studio is a **Unix** shop. Dating to the early 1970's, Unix remains the most powerful operating system in common use. It's true that Windows dominates the world's desktops. But if Windows is the backyard pool behind suburban houses worldwide, Unix is the North Sea: wild and powerful and filled with cargo freighters and battleships. In Studio, "Unix" includes various flavors and derivatives of the original OS, particularly **Linux**, but also **Mac OS**, **Android**, and **iOS**.

Studio uses best-of-breed tools. I particularly recommend **Emacs**. Far more than a mere editor, `emacs` has been called an *extensible computing environment*. Critically, it can be extended by its users to perform a myriad of amazing feats, just a fraction of which you will learn and leverage. Knowing other editors is helpful; I regularly use `vi` for small tasks. But if `vi` is a pontoon boat, `emacs` is a nuclear submarine. That flies. In space.

Other best-of-breed technologies used in Software Studio include:

- **Ubuntu** as our standard Linux distribution
- **Git** and **Github** for revision control
- **Python**, **JavaScript**, **Objective-C** (iOS), **Java** (Android)
- **Django** backed by **SQLite** (development) and **PostgreSQL** (production)
- **Angular** backed by **Node**, **Express**, and **MongoDB**
- **React** and **Redux** for client side components
- **Trello** for project and sprint backlogs, together with **Plus for Trello** (Chrome extension) for time tracking and management.
- **Meteor** for real-time web applications

3.3 Software Studio is Agile

Studio embraces an agile software development methodology based largely on **Scrum**. We are driven by customer requirements, create software that is adaptable to change, and interact with our customers early and often during development. We also incorporate selected ideas from **Extreme Programming**, **Kanban**, and **Pomodoro**.

3.4 Software Studio Emphasizes Service

The products that we build serve a bigger purpose. We focus especially on software that meets the needs of the **global Christian missions community** or other non-profit and relief organizations. The **Taylor Center for Missions Computing** is a key partner in this regard. As a good citizen of the Taylor community, Studio also sometimes develops products for other parts of the university.

3.5 Software Studio is Intensely Practical

Studio teaches you the skills necessary to deliver amazing products to our customers. We focus on two broad categories of skills: *software* skills and *system* skills.

Software skills include requirements elicitation, coding, debugging, unit testing, revision control, continuous integration, continuous deployment, project management, code reviews, and defect tracking.

System skills include, operating system virtualization; installation, configuration, maintenance, and administration of the operating system, network, database management system, and web server; software deployment; and disaster planning

3.6 Software Studio Fosters Teamwork

To foster teamwork, we embrace the time-honored tradition of *apprenticeship*. The medieval guild system classified practitioners into three groups: *apprentice*, *journeyman*, and *master*.

An *apprentice* begun work as a young teen, contracted to a *master* for five to nine years in order to learn the master's trade. The apprentice received no salary, but received room, board, and training in exchange for work done on the master's behalf.

Upon learning the trade to the master's satisfaction, the apprentice was released from the contract to become a *journeyman*. The term, derived from the French word for *day*, indicated that the journeyman was typically paid as a day laborer. For the next few years, the journeyman worked to hone his[1] skills and establish his own business and clientele.

After gaining sufficient experience, the journeyman created a *master piece* as demonstration of his mastery of the trade. Full members of the craft guild—its *masters*—evaluated the piece to determine whether it met the standards of the guild. If so, the journeyman was himself admitted into the guild as a *master*, which bestowed upon him both status and wealth.

Students new to Software Studio are considered *apprentices*. As they gain experience, they advance to become *journeymen* and, finally, *masters*. The following table connects these roles to time spent in Software Studio.

Semester	Course	Level
1	COS 371	Apprentice 1
2	COS 372	Apprentice 2
3	COS 471	Journeyman
4	COS 472	Master

In Software Studio, our expectations for each group of practitioners include the following:

1. *Apprentice*
    - Focus on learning the tools and techniques we employ
    - Contribute meaningfully to the product while learning
    - Seek help from other team members when wedged
    - Shoulder more responsibility throughout your first year
2. *Journeyman*

[1] Almost all participants in the medieval trades were men.

- Focus on growing both the depth and the breadth of your understanding
- Share your knowledge with other team members—especially apprentices
- Learn how to learn on your own
- Know when to ask for help when you find yourself stuck on something new
- Engineer substantial portions of the product
- Prepare to shoulder the responsibilities of a master

3. *Master*

- Focus on leading the team and delivering the product
- Actively seek to provide help to other team members when they're wedged
- Develop journeymen on the team to move them toward mastery—they will be taking your place soon
- Evaluate the performance of team members
- Take on the most challenging aspects of product development
- Continue to hone your understanding of new or advanced tools and techniques
- Interact with customer stakeholders to ensure a high quality product—one that conforms fully to customer requirements

4 Content

Software Studio is about *software* and *scholarship*.

4.1 Software

The majority of your time will be devoted to the design, development, testing, and deployment of production-quality software systems.

We use an agile software process based on the industry standard **Scrum** methodology. The semester is organized into (mostly) three-week sprints, giving us five sprints over the course of each 15-week semester. Except for the first sprint, our class time during each sprint will be spent roughly as illustrated in this table.

Week	Day	First Hour	Second Hour
1	T	Sprint Retrospective	Sprint Planning
	R	Work	Work
2	T	Reading Discussion	Hot Topic/Guest Speaker
	R	Work	Work
3	T	Sys Admin	Work
	R	Team Leads/Work	Work

The first day of a sprint comprises a *sprint retrospective* on the previous sprint and *sprint planning* for the upcoming sprint. Combining these activities on the same class day simplifies meeting with our customer. Sprint planning meetings will proceed as follows:

1. Customer confirms that the top stories on the backlog are indeed the top priority for implementation in the sprint.
2. Senior members of the team (masters, journeymen) are assigned as *team leads* for each story in the sprint.
3. For each story, the team lead will:
   (a) Estimate the story duration and enter the estimate into the task tracker.
   (b) Break the story into tasks.
   (c) Assign a team member as *owner* of each tasks.
4. For each task, the task owner will estimate task duration and enter it into the task tracker.

No work should begin on a story until all these steps are completed. Our goal is to complete all the steps in class on the day of sprint planning.

The single largest activity during class time is doing actual *work* on the project. Other activities during the sprint are as follows.

- *Reading discussion* of a paper or other reading that I will assign at the beginning of the sprint.
- A *hot topic* relevant to the class, the project, or to software development in general. Our speaker may be me, a member of the team, or a guest speaker.
- Because you must know not only how to build a software system, but also how to deploy and administer it, each sprint will include a key topic related to *system administration*.
- On the last class day of each sprint, I will meet with team leads for a face-to-face update.

At the beginning of the semester, we will spend one week on introduction and on-boarding activities, followed by a two-week "mini-sprint" as shown in this table.

Week	Day	First Hour	Second Hour
1	T	Course Introduction	On-boarding
	R	On-boarding	
2	T	Sprint Planning	
	R	Work	Work
3	T	Reading Discussion	Hot Topic/Guest Speaker
	R	Team Leads/Work	Work

During *on-boarding*, the entire team focuses on getting up to speed with the development environment, tools, and processes to be employed during the semester. New students (Apprentice 1) will each be assigned a senior member of the team (Master or Journeyman) as a mentor. The mentor is responsible to ensure that the apprentice has the proper environment and tools available, and that he or she understands all aspects of the development process sufficiently in order to begin contributing meaningfully to the product at the beginning of the mini-sprint.

4.2 Scholarship

The *reading discussion* and *hot topic* activities mentioned above add to your *scholarly understanding* through reading, discussion, and special speakers. Readings for the course can be either classic or current papers, magazine articles, and book chapters that address critical issues in software engineering. Reading topics include requirements, design, construction, testing, maintenance, configuration, quality management, process, methods, and ethics.

5 Mechanics

5.1 Attendance

Physical attendance is required. I will be in class each day, and I expect you to be there also. In general, I am very understanding about students who must miss class due to a sanctioned Taylor activity, job interview, family emergency, and the like. If possible, let me know in advance if you will not be in class. I will work with you to arrange make-up instruction, homework, quizzes, etc.

5.2 Moodle

The Computer Science and Engineering department uses Moodle as our Learning Management System. The URL for Moodle is `https://moodle.cse.taylor.edu`. To sign on to the course site for the first time, you will need an enrollment key. The key for this course is `nerds4christ`.

You are responsible for checking Moodle regularly to keep up with assignment due dates and other announcements posted to the site. For due dates, the Moodle calendar is your friend.

6 Evaluation

Detailed grading criteria for the course will be announced in the next few days.

7 Final Deliverables

Final deliverables are required of all students, as detailed in this section.

7.1 Final Deliverables for COS 371, 372, and 471

Write a paper about your personal experience in Software Studio this term. The goal of this paper is not to critique the class, your project, or your team, but to reflect on your own experience in the class and how you matured as a software engineer.

Your paper should address at least the following questions. As appropriate, include ideas from the reading and in-class presentations, your individual experience, and your personal participation on your project.

- What was the most important knowledge you acquired as it relates to your future as a professional software engineer?

- Similarly, what were the most important skills you acquired or honed?
- What did you learn about yourself as it relates to being a member of a team?
- How did your experience in the class speak to your vocational call as a Christ follower?

These questions are not intended to be exhaustive. You are encouraged to reflect in your paper on any additional insights you gleaned from your experience in the class this term.

Type (double space) your paper. Use good spelling, grammar, punctuation, and structure. Your paper should be between 1,000 and 1,250 words long.

7.2 Final Deliverables for COS 472

As partial demonstration of their mastery of the discipline, *all* CS&E students complete and present a substantial project during their senior year. All students are required to prepare and preset a poster regarding some aspect of their work during their degree program

Most students satisfy this degree requirement through COS 492 (Senior Project). As a Software Studio student, you are not required to take COS 492; instead, you satisfy these requirements as part of Software Studio IV.

An important decision that you should make early in your fourth semester of Software Studio (if not before) is the topic for your paper, presentation, and poster. You should meet with me no later than mid-term to discuss candidate topics. Here are some general guidelines:

- In your first three semesters of Software Studio, you wrote a simple experience paper. Your topic for Software Studio IV, however, must be much weightier.
- You may focus on one or more non-trivial projects that you undertook during your time in Software Studio.
- You may choose as your topic some idea related to Software Studio that was not the direct subject of one of your projects.

You will prepare a poster describing your work and present the poster to visitors who attend a poster session held by the department. Read, study, and evaluate past students' posters (displayed at various locations around the department) to get a better idea of what is expected of yours.

Guidelines for the poster are as follows.

1. Use a clean, clear layout that employs good layout and design, clear fonts, meaningful colors, etc.
2. Employ graphics (photos, illustrations, charts, graphs, figures, etc.) that enhance the poster's message.
3. Make evident the topic of your poster and the contributions that your work in the area has made.
4. Present your ideas logically and clearly so that your poster can be understood by a reader whether you are there to explain it or not.

Observe these guidelines from our system administrators on the preparation of your poster.

- Most students use PowerPoint, although Adobe Illustrator is better designed to do large-format printing. We can print from most apps that can print (Photoshop, Word, Excel, Open Office, etc.), and can enlarge prints from page size to whatever poster size you need.
- We can print from PDFs, although we suggest that you provide the original file format if you are using an app that we support. We can also print from JPEGs if you simply wish to print photos.
- Our paper widths are 24, 36, and 42 inches. The printer is not capable of printing larger than 42 inches. Paper length is variable.
- Avoid using a dark background unless the dark background is important in conveying your message. Dark backgrounds require *large* amounts of ink, can gunk up the print heads, and cost more to print.
- Visit `http://www.swarthmore.edu/NatSci/cpurrin1/posteradvice.htm` for sample templates that you may wish to use.

8 Academic Integrity

As a student at an institution whose goal is to honor Christ in all that it does, I expect you to uphold the strictest standards of academic integrity. You must do your own work, cite others when you present their work, and never misrepresent your academic performance in any way. Violation of these standards stains the reputations of you as a student, Taylor as an institution, and Jesus as our Lord. Such a violation will result in your failing the course and other disciplinary action by the University. Refer to the Taylor catalog for the official statement of these ideas.