Idaho State University | Computer Science

CS 3321 | INFO 3307 | INFO 5307 – Fall 2019
Introduction to Software Engineering
Systems Analysis and Design
Intermediate Systems Analysis and Design

Instructor: Isaac Griffith

Office: BA 315
Phone: (208) 282-4876
Email: grifisaa@isu.edu
URL: https://www2.cose.isu.edu/~grifisaa/
Office Hours:

- **TR**: 1430 – 1530
- By appointment scheduled at: https://isaac-griffith.youcanbook.me/

Course Information

Meeting Time: TR: 13:00 – 14:15 **Room:** Pocatello: BA 506, Idaho Falls: TA 286

Final Exam Dates

Section: 01 – Pocatello
Date: Tuesday Dec 10, 2019
Time: 1230 – 1430
Room: BA 506

Section: 02 – Idaho Falls
Date: Tuesday Dec 10, 2019
Time: 1230 – 1430
Room: Idaho Falls Testing Center

Prerequisites:

- CS 3321: CS 2263

- INFO 3307: INFO 1181

Co-requisites:

- INFO 3307: INFO 1182

Textbooks:

- **Head First Software Development, 1st Edition** – Pilone and Miles. ISBN: 978-0596527358
- **UML @ Classroom: An Introduction to Object-Oriented Modeling, 2015 Edition** – Seidl, Schloz, Heumer, and Kappel. ISBN: 978-3319127415
- **Head First Design Patterns: A Brain-Friendly Guide, 1st Edition** – Eric Freeman, Bert Bates, Kathy Sierra, and Elisabeth Robson. ISBN: 978-0296007126

Course Description

Techniques and tools for conceiving, designing, testing, deploying, maintaining, and documenting large software systems with particular focus on the structured analysis and design phases including task analysis, human factors, costs, and project and team management.

Technology

This is a Computer Science course and is heavily focused on programming related activities. As a student you are expected to have at your disposal a computer system capable of running a recent operating system such as MacOS X, Windows 10, or a recent Linux. Additionally, you will need the following tools:

- Git
- A Professional Grade Java IDE such as JetBrains IDEs, Eclipse IDEs, or Microsoft Visual Studio
- Applicable dependency and build management tools for selected language.

Learning Outcomes

The following outcomes will be evaluated via homework assignments and exams

Software Processes

- Describe how software can interact with and participate in various systems including information management, embedded, process control, and communications systems. [Familiarity]
- Describe the relative advantages and disadvantages among several major process models (e.g., waterfall, iterative, and agile). [Familiarity]
- Describe the different practices that are key components of various process models. [Familiarity]

- Differentiate among the phases of software development. [Familiarity]
- Describe how programming in the large differs from individual efforts with respect to understanding a large code base, code reading, understanding builds, and understanding context of changes. [Familiarity]
- Explain the concept of a software lifecycle and provide an example, illustrating its phases including the deliverables that are produced. [Familiarity]
- Compare several common process models with respect to their value for development of particular classes of software systems taking into account issues such as requirement stability, size, and non-functional characteristics [Usage]
- Define software quality and describe the role of quality assurance activities in the software process. [Familiarity]
- Describe the intent and fundamental similarities among process improvement approaches. [Familiarity]
- Compare several process improvement models such as CMM, CMMI, CQI, Plan-Do-Check-Act, or ISO9000. [Assessment]
- Explain the role of process maturity models in process improvement. [Familiarity]
- Describe several process metrics for assessing and controlling a project. [Familiarity]

Requirements Engineering

- List the key components of a use case or similar description of some behavior that is required for a system. [Familiarity]
- Describe how the requirements engineering process supports the elicitation and validation of behavioral requirements. [Familiarity]
- Interpret a given requirements model for a simple software system. [Familiarity]
- Describe the fundamental challenges of and common techniques used for requirements elicitation. [Familiarity]
- List the key components of a data model (e.g., class diagrams or ER diagrams). [Familiarity]
- Conduct a review of a set of software requirements to determine the quality of the requirements with respect to the characteristics of good requirements. [Usage]
- Compare the plan-driven and agile approaches to requirements specification and validation and describe the benefits and risks associated with each. [Familiarity]

Software Design

- Articulate design principles including separation of concerns, information hiding, coupling and cohesion, and encapsulation. [Familiarity]
- Within the context of a single design paradigm, describe one or more design patterns that could be applicable to the design of a simple software system. [Familiarity]
- For the design of a simple software system within the context of a single design paradigm, describe the software architecture of that system. [Familiarity]
- Given a high-level design, identify the software architecture by differentiating among common software architectures such as 3-tier, pipe-and-filter, and client-sever. [Familiarity]
- Describe a form of refactoring and discuss when it may be applicable. [Familiarity]
- Explain how suitable components might need to be adapted for use in the design of a software product [Familiarity]

Software Project Management

- Discuss common behaviors that contribute to the effective functioning of a team. [Familiarity]

- List several examples of software risks. [Familiarity]
- Describe the impact of risk in a software development lifecycle. [Familiarity]
- Describe different categories of risk in software systems. [Familiarity]
- Describe how the choice of process model affects team organizational structures and decision-making processes. [Familiarity]
- Using a particular software process, describe the aspects of a project that need to be planned and monitored, (e.g., estimates of size and effort, a schedule, resource allocation, configuration control, change management, and project risk identification and management). [Familiarity]
- Describe the impact of risk tolerance on the software development process. [Assessment]
- Identify risks and describe approaches to managing risk (avoidance, acceptance, transference, mitigation), and characterize the strengths and shortcomings of each. [Familiarity]
- Apply the basic principles of risk management in a variety of simple scenarios including a security situation. [Usage]

Software Verification and Validation

- Distinguish between program validation and verification. [Familiarity]
- Describe the role that tools can play in the validation of software. [Familiarity]
- Describe and distinguish among the different types and levels of testing (unit, integration, systems, and acceptance). [Familiarity]
- Describe techniques for identifying significant test cases for integration, regression and system testing. [Familiarity]
- Describe how to select good regression tests and automate them. [Familiarity]
- Discuss the limitations of testing in a particular domain. [Familiarity]

Software Evolution

- Identify the principal issues associated with software evolution and explain their impact on the software lifecycle. [Familiarity]
- Discuss the challenges of evolving systems in a changing environment. [Familiarity]
- Outline the process of regression testing and its role in release management. [Familiarity]
- Discuss the advantages and disadvantages of different types of software reuse. [Familiarity]

Software Reliability

- Explain the problems that exist in achieving very high levels of reliability. [Familiarity]
- Describe how software reliability contributes to system reliability. [Familiarity]
- List approaches to minimizing faults that can be applied at each stage of the software lifecycle. [Familiarity]

Information Management Concepts

- Describe how humans gain access to information and data to support their needs. [Familiarity]
- Describe the advantages and disadvantages of central organizational control over data. [Assessment]
- Identify the careers/roles associated with information management (e.g., database administrator, data modeler, application developer, end-user). [Familiarity]
- Compare and contrast information with data and knowledge. [Assessment]
- Identify issues of data persistence for an organization. [Familiarity]

Principles of Secure Design

- Describe the principle of least privilege and isolation as applied to system design. [Familiarity]
- Summarize the principle of fail-safe and deny-by-default. [Familiarity]
- Discuss the implications of relying on open design or the secrecy of design for security. [Familiarity
- Explain the goals of end-to-end data security. [Familiarity]
- Discuss the benefits of having multiple layers of defenses. [Familiarity]
- For each stage in the lifecycle of a product, describe what security considerations should be evaluated. [Familiarity]
- Describe the cost and trade-offs associated with designing security into a product. [Familiarity]

HCI Foundations

- Discuss why human-centered software development is important. [Familiarity]
- Summarize the basic precepts of psychological and social interaction. [Familiarity]
- Create and conduct a simple usability test for an existing software application. [Assessment]

HCI Designing Interaction

- Discuss at least one national or international user interface design standard. [Familiarity]

The following outcomes will be evaluated via the project

Software Processes

- Assess a development effort and recommend potential changes by participating in process improvement (using a model such as PSP) or engaging in a project retrospective. [Usage]
- Use project metrics to describe the current state of a project. [Usage].

Requirements Engineering

- Identify both functional and non-functional requirements in a given requirements specification for a software system. [Usage]
- Apply key elements and common methods for elicitation and analysis to produce a set of software requirements for a medium-sized software system [Usage].
- Use a common, non-formal method to model and specify the requirements for a medium-size software system. [Usage]
- Create a prototype of a software system to mitigate risk in requirements. [Usage]

Software Design

- Use a design paradigm to design a simple software system, and explain how system design principles have been applied in this design. [Usage]
- Construct models of the design of a simple software system that are appropriate for the paradigm used to design it. [Usage]
- For a simple system suitable for a given scenario, discuss and select an appropriate design paradigm. [Usage]
- Create appropriate models for the structure and behavior of software products from their requirements specifications. [Usage]

- Explain the relationships between the requirements for a software product and its design, using appropriate models. [Assessment]
- Investigate the impact of software architectures selection on the design of a simple system. [Assessment]
- Select suitable components for use in the design of a software product. [Usage]
- Design a contract for a typical small software component for use in a given system. [Usage]

Software Project Management

- Create and follow an agenda for a team meeting. [Usage]
- Identify and justify necessary roles in a software development team. [Usage]
- Understand the sources, hazards, and potential benefits of team conflict. [Usage]
- Apply a conflict resolution strategy in a team setting. [Usage]
- Use an ad hoc method to estimate software development effort (e.g., time) and compare to actual effort required. [Usage]
- Demonstrate through involvement in a team project the central elements of team building and team management. [Usage]
- Create a team by identifying appropriate roles and assigning roles to team members. [Usage]
- Assess and provide feedback to teams and individuals on their performance in a team setting. [Usage]
- Track the progress of some stage in a project using appropriate project metrics. [Usage]
- Compare simple software size and cost estimation techniques. [Usage]
- Use a project management tool to assist in the assignment and tracking of tasks in a software development project. [Usage]
- Explain how risk affects decisions in the software development process. [Usage]
- Identify security risks for a software system. [Usage]
- Demonstrate a systematic approach to the task of identifying hazards and risks in a particular situation. [Usage]
- Conduct a cost/benefit analysis for a risk mitigation approach. [Usage]
- Identify and analyze some of the risks for an entire system that arise from aspects other than the software. [Usage]

Software Verification and Validation

- Undertake, as part of a team activity, an inspection of a medium-size code segment. [Usage]
- Create and document a set of tests for a medium-size code segment. [Usage]
- Use a defect tracking tool to manage software defects in a small software project. [Usage]

Software Evolution

- Estimate the impact of a change request to an existing product of medium size. [Usage]
- Use refactoring in the process of modifying a software component. [Usage]

HCI Foundations

- Develop and use a conceptual vocabulary for analyzing human interaction with software: affordance, conceptual model, feedback, and so forth. [Usage]
- Define a user-centered design process that explicitly takes account of the fact that the user is not like the developer of their acquaintances. [Usage]

HCI Designing Interaction

- For an identified user group, undertake and document an analysis of their needs. [Assessment]
- Create a simple application, together with help and documentation, that supports a graphical user interface. [Usage]
- Conduct a quantitative evaluation and discuss/report the results. [Usage]

Student Expectations

The above objectives cannot be met unless you, the student, take an active role in your education. Thus you are expected to:

- **Attend class** on a regular basis and devote your attention to the material presented
- Prepare for each and every class by **reading** the assigned material and completing both **pre- and post-lecture assignments**
- Devote the necessary **time** to preparing assignments and turning them in on time.
  - **Computer Science** is time-intensive
  - You should be prepared to give the time need for each assignment
  - As the class progresses the time required for assignments will increase, be prepared.
- **Time Management** is a requirement. Do not procrastinate, as the amount of time required for any given assignment and any given student cannot be estimated. For this reason you are encourage to begin assignments at the earliest possible date so that you will be able to complete them on time.

This is a 3 credit, as a student, you should expect to put forth on average 6 – 9 hours of additional effort outside the classroom towards this course. Given that, I expect to utilize this completely.

Valid Excuses

This course requires that you attend class and participate in classroom activities (including exams). The student handbook notes the following can be considered viable excuses for class absence, in consultation with your instructor:

- Serious Illness
- Severe Weather
- Religious Holidays
- Approved University Activities (e.g., extracurricular athletics, performance groups, student government)
- Emergency Family Issues

Invalid excuses include anything else, but I will specifically note the following:

- Minor Illness
- Typical Winter Weather Conditions
- Work
- Non-emergency Family Issues
- You or your family Purchased Plane Tickets (non-refundable)

Moodle

Course material including lectures, assignment requirements, handouts, and solutions can be viewed using your Moodle account. Announcement and Help forums will also be available on Moodle. Students are expected to access their Moodle account on a daily basis to keep apprised of course developments.

Attendance

Attendance in the course is mandatory. Missing 9 class meetings (without an acceptable excuse as defined by University Policy) during the course will result in failing the course, and a resulting X grade (IAW CoSE policy). Attendance will be monitored by in class activities, quizzes, etc.

Assignments

Homework assignments are due as assigned on Moodle. Do the homework, it helps. Late homework will be accepted up to the last day of the course, but each day after the due date will incur a linearly increasing penalty. This penalty will result in a 0 grade at midnight of the last day of class.

This course is based upon four components:

- Attendance at lectures and participation during in class exercises.
- Reading assignments
- Project
- Exams

Exams

You will be tested on your mastery of topics listed above as taken from lecture, readings, and assignments. Any information addressed by a component of this class will be considered for exams. There will be two exams, a mid-term and a final. The **final** will be a **comprehensive** exam.

Note: I do not give exams on any date other than the assigned date. The only exceptions to this are to accommodate exceptions noted above. These exceptions will only be accommodated if provided in advance and will require a minimum of **1 week advanced notice** unless an emergency.

Note: If you do not earned a **60% or higher average** on the exams in this course the highest grade you will receive in this course is a **60%** which equates to a **D–**

Grade Distribution

Grades for all assignments, exams, and the final grade will be assigned according to the following table:

Grade	–		+
A	90.00 – 92.99	93.00 – 100.0	
B	80.00 – 82.99	83.00 – 86.99	87.00 – 89.99
C	70.00 – 72.99	73.00 – 76.99	77.00 – 79.99
D	60.00 – 62.99	63.00 – 66.99	67.00 – 69.99
F		00.00 – 59.99	

The final grade calculation for this course will be allocated as follows:

CS 3321 & INFO 3307

Grade Event Type	Percent of Final Grade
Homework	25%
Project	50%
Exams	25%

INFO 5307

Grade Event Type	Percent of Final Grade
Homework	25%
Project	35%
Graduate Project	15%
Exams	25%

Learning Environment

We are all committed to maintaining an inoffensive, non-threatening learning environment for every student. Class members (including the instructor) are thus to treat each other politely–both in word and deed. Offensive humor and aggressive personal advances are specifically forbidden. If you feel uncomfortable with a personal interaction in class, see your instructor for help in solving the problem.

Content

I reserve the right to change the content as needed to fit the flow of the class and experience of the students. Changes will be reflected on Moodle.

Policies & Procedures

Academic Integrity

Academic Integrity is expected at Idaho State University and the College of Science and Engineering. All forms of academic dishonesty, including cheating and plagiarism, are strictly prohibited, the penalties for which range up to permanent expulsion from the university with "Expulsion for Academic Dishonesty" noted on the student's transcript.

Academic Integrity violations are a scourge at any University. I detest them with a passion. Anyone found to be violating the academic integrity code on any assignment or exam will be dealt with with extreme prejudice. The standard remedy, in this course, **for any Academic Integrity Violation will be FAILURE of the course**. Please note that in accordance with the Policy at Idaho State University that attempts to withdraw from the course to avoid such punishment will work to no avail.

Academic dishonesty includes, but is not limited to:

1. Cheating on Exams
2. Plagiarism
3. Collusion
4. Sharing solutions or code on programming assignments

Definitions

Cheating on an examination include:

- Copying from another's exam, any means of communication with another during an exam, giving aid to or receiving aid from another during an exam;
- Using any material during an examination that is unauthorized by the proctor;
- Taking or attempting to take an exam for another student or allowing another student to take or attempt to take an exam for oneself
- Using, obtaining, or attempting to obtain by any means the whole or any part of an administered exam.
- Talking to anyone other than the professor during an exam.

Plagiarism is the unacknowledged incorporation of another student's work into work which the student offers for credit.

- The use of the source code of another person's program, even temporarily, is considered plagiarism. This includes attempting to hide the plagiarism by changing variable names, method names, or class names.
- Copying material from another project (including open source projects) without attributing (citing) that project.

Collusion is the unauthorized collaboration of another in preparing work that a student offers for credit.

- Allowing another person to use your source code, even temporarily, is considered collusion.

Other types of academic dishonesty include:

- Using other student's content from their assignments, disk, etc.

- Performing any act designed to give unfair advantage to a student or the attempt to commit such acts

Exceptions

In this course, the specific exceptions given below are not considered scholastically dishonest acts:

- Discussion of the algorithm and general programming techniques used to solve a problem.
- Giving and receiving aid in debugging.
- Discussion and comparison of program output **(output only not code)**

Student Notification

All students are responsible for checking Moodle and their email on a regular basis, preferably daily, for notification of any class scheduling changes or assignment clarification. Often such notifications will be posted late at night.

Instructor Availability

The instructor will be available during posted office hours, but additional efforts are made to increase accessibility to the students. If the instructor is not available at the telephone number above, the student can leave a detailed voicemail message. However, the instructor's email is checked at a minimum of twice a day and often the student will get an immediate response to questions submitted by email. Email is usually the most reliable means of contact.

Note that I am a working research scientist, thus I may need to attend conferences both with my local colleagues and with my international colleagues. Thus, I am constantly in meetings, both here and abroad. That being said, I work 7 days a week 12 months a year. If you need my help and all other means of scheduling have led to no avail, please do not hesitate to contact me. Note that I am not an emergency service, I will require that you contact me 24 hours prior to an assignment being due, for which you need help. If you do not plan ahead, I cannot help.

Email Etiquette

Email is the best possible method of reaching me outside of my office hours. Note that I have certain expectations for communicating with me via email, as follows:

- DO NOT use chat or SMS shorthand in your messages.
- Use full words, and full sentences.
- Maintain the frame of mind that your are communicating to a professional, and that a professional demeanor is required.

Failure to abide by these requirements will result in your email being deleted and forgotten.

Disability and Special Needs

The Computer Science program at Idaho State University is committed to ensuring that all students achieve their potential. If you have a disability (physical, hearing, vision, psychiatric, or learning disability) that may need a reasonable accommodation, please contact the ADA & Disabilities Resource Center located in the Rendezvous Complex, Room 125, 282-3599, as early as possible.

Closed Week Policy

Information about the ISU Closed Week Policy can be found online. Note that the policy does not prevent the presentation of new material during closed week.

CoSE X Grade Policy

In the College of Science and Engineering, a student who earns a failing grade via course work (exams, homework, etc.) and has unexcused absences that total more than 30% of class meetings will receive a grade of "X".