ISAT 345—Software Engineering

Fall 2017

Collaboration is the hardest part!!!

What is this course about?

ISAT 345 is designed to teach you the tools and processes that allow medium to large teams to design and build software. Software development is hard enough with two or three people, but it becomes exponentially more difficult to manage the development process the more people you add to the project. In this class you will learn by doing–you will learn software engineering methodology by being part of a large software development team.

What methodology will we use?

There are many different methodologies for software engineering. This class will focus on a subset of those methodologies referred to collectively as “agile.” While the label “agile” was only coined in 2001, agile methodologies have their roots in “lightweight” software development processes invented in the 1990’s, and have grown tremendously in popularity over the past twenty years. As you enter the workforce, you will find that agile practices have now become standard in most domains of software development. Knowing them prior to being hired will be a major selling point to potential employers.

More specifically, we will be focusing on two agile methodologies in particular: scrum and kanban. Scrum and kanban are widely used by companies like IBM and Accenture, and so having experience with these methodologies will improve your chances of employment and ease your transition into a professional software development shop.

What software will we build?

This semester we will be working on building a real system called designed to be launched out into the world. You’ll learn much more about the project as we move forward.

Our goal is to launch a minimum viable product by the end of the semester.

Grades

Because the goal of this class is to teach you how to build software in large teams, ***PARTICIPATION IS ABSOLUTELY CRITICAL!***

In 2016, when I taught this class for the first time, out of the 45 times that we met that semester, I only had 100% attendance on one day. Most days, attendance was closer to 75% or worse. This is completely unacceptable. The entire team relies upon all members to show up all of the time. Since everyone relies upon everyone else, if you are not there, you hold everyone up.

Therefore, this year there will be a new, and very formal grading policy.

1. If you make it to week 13, you will get an A.
2. If you make it to week 10, you will get a B.
3. If you make it to week 7, you will get a C.
4. If you make it to week 4, you will get a D.
5. If you get "fired" before week 4, you will get an F.

What do I mean by "make it?" Essentially, if the team feels that you are not being professional, and not pulling your share of the load, you will be "fired." That being said, we strive to have a friendly and supportive team, so we'll make every effort, as a team, to make sure that everyone survives until the end of the semester. We used this policy last year, in 2017, and everyone had an amazing time.

> *Working in the professional software world is really fun and exciting! This class is an opportunity for you to take the initiative to make it a positive experience. Imagine what kind of place you would like to work in when you graduate. How can you create your perfect world in this class? I know I'm pretty psyched to see what you guys come up with!*

How do you get fired?

There will be a list of "firable offenses." If you commit any of them, you will be put on probation for a specified period of time. If you make any more mistakes while you are on probation, you will be let go. The firable offenses are as follows:

1. If you have 3 unexcused absences, or if the total of excused and unexcused absences reaches 5, you will be put on probation for the rest of the semester. Any more absences after that point will result in your being asked to leave the project.

2. If you harass any of your colleagues, verbally, sexually, or otherwise, you will be put on probation. Your probation will end after you have been through an appropriate training to increase your sensitivity.
3. If you have not satisfactorily completed the independent developer training exercises by the end of week 5, you will be put on probation. If you still have not completed them by the end of week 6, you will be asked to leave.
4. Once we begin sprinting, if you fail to complete your coding tasks in any given sprint, and if the reason is deemed to be due to laziness, or inability to manage your time effectively, you will be put on probation until you have successfully completed the next sprint.
5. You may be put on probation for other types of unprofessional behavior that may arise. If it is not on the list above, you will receive a warning. The second warning will result in probation. The probation will last an amount of time commensurate with the offense.

How NOT to get fired

First of all, you will NOT be fired because of a lack of coding skill. If you can't complete a coding task during a sprint because you honestly don't know how to write the code, that's fine–this is a learning experience after all. That being said, if you know that you are in danger of not completing a sprint, and you have not come to anyone for help or assistance, and if you haven't given the team an appropriate amount of notice that you won't hit your target, you may be put on probation. In other words, ALL coders get stuck, pretty much all the time. However, if you are deemed not to be making an effort to learn what you need to learn to get unstuck, the team will not be happy with you.

Here are some positive things you can do to reduce your team's stress level, as well as your own:

1. Spend enough time socializing or otherwise getting to know your teammates that you will feel comfortable coming to them for help if you need it.
2. Look for opportunities to help other people who might be stuck. If you are proactive about helping other people, the chances that they will want to help you in return go up.
3. Get in the habit of keeping notes or a journal about your efforts to get unstuck. Alternatively, write well-documented questions on Stack Overflow. That way we can tell the difference between honestly stuck, and too lazy or disorganized to get your work done on time.
4. Get in the habit of putting in at least an hour a day. Manage your time well, and don't procrastinate.
5. Communicate openly, honestly, and frequently with your team.
6. Show up to all team meetings on time or early, whether during class or outside of class.

Your Goal This Semester

Your goal this semester is to create a functional, supportive team that is fun to be a part of. We are creating a simulated work environment that will attempt to function in as close to a realistic fashion as possible. So look for ways to boost morale, to put people at ease, to make people laugh, and of course, to develop an amazing software application.

Schedule & Flow

The schedule for the course will roughly follow the following timeline:

Week	What the Class is Doing	What You should be Doing
1-5	Orientation and Groundwork	Completing Required Badges (see below)
6	Pre-Sprint Planning	Planning Poker; Assigning User Stories
7-8	Sprint #1	Development
9-10	Sprint #2	Development
11-12	Sprint #3	Development
13-14	Sprint #4	Development
15	Presentation	Celebration!

Once we reach week 7 we will attempt to have daily standup meetings. On class days these will be held face-to-face, and other days we will use gchat, skype, slack, or some other form of online conferencing system.

Required Badges

By the end of week 5 (i.e. before class on 10/2), everyone must complete the following or be put on probation:

- Socializer x 5–I'm dead serious about everyone getting to know each other!
- Syllabizer–This is mainly insurance for me so nobody can say "I didn't know!" later…
- Logger–The absolute easiest badge
- Intro to Agile–Dive into the process that is at the heart of this class
- Intro to Ionic–The technology we'll be using to build our app

In addition, it would be extremely helpful if you also had the following, but these are not required:

- Git & GitHub
- Subdue Sublime
- The Command Line

The Semester Project

This semester we'll be working on a piece of software called SmartClickr, a web/mobile based audience-response system. The prototype for this system was built in 2014 by a team of 4 ISAT capstone students. The Fall 2015 ISAT 345 students worked on this project and failed miserably. The Fall 2016 ISAT 345 students also worked on this project, and came close to achieving a working prototype, but not quite…

Now it's your turn.

Important Dates and Deadlines

At the behest of the registrar, a list of dates you may wish to take note of:

- Tuesday, September 5th: Last day of add/drop
- Thursday, September 14th: Last day to add a class with Department Head signature
- Friday, September 15th: Last day to withdraw from JMU with charges canceled

So if I scare you off, get out early. Or conversely, if I turn you on, join us soon!

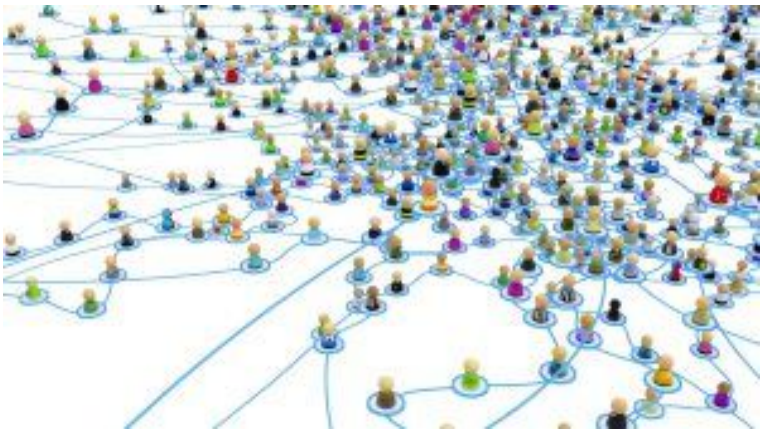
My academic integrity policy is different from JMU's standard policy, but I will adhere to JMU's standard policies listed on the JMU Syllabus Information for Students page with respect to add/drop, disability accommodations, inclement weather and religious accommodations.

What is "Learning"?

Learning is Essentially Relational

The dominant definition of learning in our society is based on psychometrics, or the measurement of psychological properties. Because the amount someone has learned can't be

directly observed, we infer learning using tests or other student-generated artifacts. Although we can never know exactly what a person knows or can do, we believe we can make a pretty good guess by giving them tests and scoring the results. In abstract terms, learning equals the difference in performance of a person over time, or:



$$t_1 - t_0 = learning$$

More concretely, if you give someone a test at the beginning of the course and they get a 62%, then give them a test at the end of the course and they get 92%, then we define learning as:

$$92 - 62 = 30$$

Unfortunately, there are no standardized units of learning, as we have with say money, or weight, or temperature. Therefore we really have no way to interpret such scores in a reliably meaningful way. Fortunately, though, there are other ways to define learning that don't have the same kinds of problems.

A better definition of learning, in the sense that it takes into account the contextualized and systemic aspects of learning, comes from Moss (2003) who said:

> *From a sociocultural perspective, learning is perceived through changing relationships among the learner, the other human participants, and the tools (material and symbolic) available in a given context. Thus learning involves not only acquiring new knowledge and skill, but taking on a new identity and social position within a particular discourse or community of practice. As Wenger puts it, learning "changes who we are by changing our ability to participate, to belong, and to experience our life and the world as meaningful." (p14)*

Learning is, at its core, an artifact of relationship. It's fundamentally a social process because we need the mirror provided by others to help us recognize and apply our gifts and talents. In more concrete terms, as a result of learning, I go from being "that guy who works down the hall" to being "that guy who works down the hall and is a great web developer." As a result of learning, people come to see us differently, and they begin to approach us and rely on us to contribute our gifts and talents back to them.

Personal Integrity Policy

First:

> **If I catch you cheating, or doing anything else dishonest, you will fail the course. Period.**

Second, that being said, I strongly encourage sharing and collaboration in most every aspect of the course. That means that I think it's a smart idea for you to:

- Download code you find on the web (include the URL of where you found it and some notes about how you got there)
- Download your classmates' code and use it, even before an assignment is due
- Pay someone to help you write code
- Get code from upperclassmen or people in previous semesters
- Ask your neighbor to give you a hint on a question on a test that you're stumped on
- Use whatever notes, websites, books, or other materials you need to complete most any assignment or test

You'll note that many of the above behaviors would be considered "cheating" in many or most other courses. Here are some guidelines I'd like you to follow:

- Never EVER copy without attribution
- Even on tests, if someone or something helped you out, acknowledge it. Make notes in your code if you got it from someone or somewhere else. Copying without attribution is stealing and is a breach of integrity. If you got the code off of the web, there should be a URL and some notes about how you found it. If you paid someone to help you write it, say so.
- Never copy without understanding
- The point of the class is to learn and understand stuff. Since you don't get any grades on individual tests or assignments, it's pretty stupid to copy something that you don't understand. Think about it. What point could it possibly serve?
- Be very hesitant to copy an ENTIRE project
- While there's a lot to be gained by incorporating parts of your classmates' code in your own project, copying someone else's entire project doesn't really provide you much of a learning experience and wastes people's time.
- Try to figure it out yourself first
- 90% of writing programs is learning how to write them, and this will stay the same throughout your entire programming career. Being a self-sufficent learner is one of the primary goals of the course.

Code re-use is a HUGE part of hacker culture. What hackers hate more than anything is not understanding stuff. I want you to get a sense for what it's like to be a part of the fun world of professional hackers.

Okay, so what do I consider a breach of integrity worthy of failure?

- Lying about anything to anyone in the class
- It could be as trivial as the reason why you didn't show up for class or do your part of a group assignment. Everybody screws up sometimes. Don't compound the mistake by lying about it. We can forgive mistakes but it's VERY difficult to regain trust once it's broken. Swallow your embarrassment or fear and fess up.
- Stealing anything–this includes copying without attribution
- Stealing is just wrong, and since you have a blanket license to copy most any code you can find, there's no reason not to give people credit for the work they did. Passing someone else's original work off as your own is frankly disgusting.
- Threatening, antagonizing, or intimidating anyone in our learning community
- This is unacceptable behavior and will get you at least fired, if not sued in most every company you'd ever work for.

If you are in doubt about something, please ask your prof. Please feel free to come speak to your prof in confidence about anything in this course that troubles you. So far at JMU I've never had a problem with anyone's integrity (that I know about). Don't be the first group to ruin my perfect record. Thanks!

The Lead Coder

Hi, I'm Morgan. I'll be the lead coder for this journey.

As I've said:

Learning is essentially an artifact of relationship

So, in order to lead by example, here is a brief overview of my relationships, who I know and who knows me, and what roles we've played in each other's lives. But first, here's how you can begin relating to me:

Key Info

Office	EnGeo 2118 (office hours/coffee by appointment)
Email	bentonmc@jmu.edu

Cell	973-495-7736
Social	

Academic Community

Undergrad

I got a BA in Leadership Studies and Sociology (with a minor in Physics) in 1996 from the University of Richmond and the Jepson School of Leadership Studies. (Jepson is kinda like ISAT, but for social science.) For my capstone I went to Capitol Hill and surveyed the IT directors of US Senators to try to figure out if the people who were making and passing laws about technology had actually ever used a computer. I was inspired to do this by an online petition I received related to some of the scarier provisions in the Telecommunications Act of 1996. I was supported in my efforts by my professor Joanne Ciulla, whose unorthodox ideas (see her book The Working Life: The Promise and Betrayal of Modern Work) helped me become an outside-the-box thinker. My capstone partner, Elizabeth MacKenzie Biedell, went on to work for the US State Department where she staffed Colin Powell, the CIA where she briefed the White House, and has become an advocate for information sharing and against the over-classification of intelligence.

As an undergrad I did a lot of community service. I helped Richmond Habitat for Humanity raise close to $50,000 and build four houses. I got schooled in basketball by kids at the Sacred Heart Center while I was volunteering in their after school program. I helped feed meals to homeless men, and learned a lot from them in the process. I left college with a deep commitment to serve, inspired by the words of one of my best friends, the Associate Chaplain David Dorsey, who said: There's nothing very "high" about "higher education" if it doesn't give something back to the community from which it was taken.

Another of my mentors was Richard A. Couto (who insisted that we call him Dick–very hard to do with a straight face if you're my age). I helped Dr. Couto produce An American Challenge: A Report on Economic Trends and Social Issues in Appalachia. My job was to turn US census data into maps–perhaps the first time that I employed computer technology to help effect social change, this time for poor people in the Appalachian region.

These are just a few of the many people with whom I connected as an undergrad, who influenced me, and with whom I continue to connect to solve real problems for real people.

Grad School

I got an MS and PhD in Information Systems from New Jersey Institute of Technology. It took me TEN YEARS (1998-2008)! For my dissertation, I built a web-based system to help teachers write better multiple-choice questions (MCQs). (Spoiler: it didn't really work.) In actuality, I hate

multiple-choice questions and multiple-choice tests. My system was designed as a Trojan horse (figuratively–it was NOT spyware) to convince teachers that MCQs are more trouble than they are worth. This grueling journey solidified my lifelong ambition to radically change education globally.

Along the way, I was lucky to receive guidance from Dylan Wiliam, one of the world's foremost scholars on formative assessment. I met Dr. Wiliam while I was a graduate intern at ETS (you know, where they make the SAT and GRE), and he taught me a painful and enduring lesson–technology is not always the answer. His 1998 meta-analysis of hundreds of studies of classroom assessment has grounded and guided my teaching practice and research ever since.

Starr Roxanne Hiltz and Murray Turoff were the Distinguished Professors at NJIT who gave me my job there and made it possible for me to support my family while getting my PhD. They predicted the age of the Internet way back in 1978 in their book The Network Nation. They were pioneers in online education, and there are still many of their ideas that haven't yet been implemented in modern online learning management systems.
I was teaching four courses per semester throughout my time at NJIT. I may have learned more from my students than from any other source. It was from my students that I learned humility, flexibility, and to see just how vicious the system is that binds us all together. I still have relationships with a number of them, and they continue to keep me on my toes.

Of course, my most important relationship in grad school was with my advisor, Marilyn Tremaine. Marilyn was one of the founding mothers of the field of human-computer interaction (HCI) and helped create one of the most vibrant and successful academic communities (the ACM's SIGCHI) of the last several decades. She let me live in her house. She showed me that even if I've been a total slacker, there's always something productive we can do with this week's meeting.

JMU

I began at JMU in August 2006 and have been here for over a decade. It is here that I've had the opportunity and freedom to put into practice many of the innovations in teaching that I've discovered over the years. The ISAT department deeply values creativity and autonomy, so while my colleagues may have offered (sometimes sharp) criticism over the years at my methods, nobody ever told me to stop, and many encouraged me to keep moving forward. I deeply value this aspect of the ISAT Department–the willingness to take risks to achieve great things.

That being said, it is my students who have had the biggest influence on me. It was Brian Rapp (ISAT '10), who, as my TA, first goaded me into trying my choose-your-own-grade strategy back in the spring of 2009. It was my capstone students who helped me write much of the software that I use in my teaching. It continues to be students like Cyril Thornton ('10), Adam Maas ('12), Chiedo John ('13), and Josh Erney ('14) who regularly ping me about new tech developments

that help keep me at the cutting edge of my field. I feel deep gratitude every time a student opens up to me and allows me to play a part in their story.

I would be remiss if I didn't mention Nicole Radziwill. We teach together, research together, play together, live together. We started the Burning Mind Project together. We've published over half a dozen papers together and applied for over a million dollars in funding together. We've been to Burning Man three times and it is from her that I am constantly challenged to try new things, examine who I am, and continually reinvent myself.

There's plenty more to learn about me, and I strongly encourage everyone to invite me out for coffee or lunch or something! I look forward to getting to know each and every one of you better.