ISAT 340—Software Development

Fall 2017

Course Overview

Software is a tool for solving scientific and technological problems. This course provides an overview of what constitutes “software” and gives students the opportunity to learn about and demonstrate their capacity to follow the steps and techniques required to develop modern software solutions to scientific and technological problems. Heavy emphasis will be placed on analysis and design, particularly with respect to the social context of the problems targeted by potential software solutions.

Learning Outcomes

As a result of taking this course, learners may:

- Be able to describe what is meant by the word “software” with particular emphasis on modern contexts which are more likely to result in web and/or mobile applications as opposed to desktop applications
- Have developed concrete artifacts to demonstrate that they have experience with, and capacity to perform tasks necessary to develop software, among them:
    - Problem Statement: Articulating a problem to be addressed by software, specifically focusing on why software is the right tool to address this problem, as opposed to other non-software approaches to problem-solving
    - Social Context Analysis and Scenarios: Describing the social context in which the problem is relevant, including an analysis of:
        - the needs/challenges of the specific people, or groups of people, to whom this problem applies
        - the people/groups who may provide resistance to this particular approach
        - a more or less comprehensive history of what other kinds of solutions have been applied to this problem, their outcomes, and how the lessons from those experiences are incorporated into the current design
    - Mockups: Thorough paper-and-pencil mockups of an application design, including rounds of revisions
    - Architecture Diagram (Generic): Analysis and selection of an appropriate software architecture, e.g. client-server, standalone, etc., for the proposed application
    - Entity-Relationship Diagram and Database Schema: Analysis and abstraction of a data model from the screen designs that results in a concrete database design
    - Architecture Diagram (Specific): Analysis and selection of an appropriate set of technologies that will be used to implement software, including:

        - A programming language or languages
        - A database technology
        - A server technology
        - An app development framework
        - A web hosting provider
        - Any other services, e.g. notifications, pub/sub, etc., necessary to implement the solution
        - Unit and Integration testing framework(s)
    - Unit Tests and Integration Tests: Automated software tests that specify at a granular level what the software is supposed to be able to do
    - Source Code: The actual code that implements the software solution, stored in a public repository, e.g. GitHub
    - Database: A concrete implementation of the data model articulated previously
- Have developed ancillary skills required to be effective at software development, and artifacts to demonstrate such skills, including, but not limited to:
    - Setting up a development environment including elements like:
        - Code editor
        - Necessary compilers, debuggers, unit testing frameworks, etc.
        - Proficiency/comfort using CLI tools (command line/terminal/console)
        - Virtual machines
    - Comfort/proficiency using revision control systems, particularly Git/GitHub
    - Ability to ask questions effectively in online forums such as Stack Overflow
    - Ability to identify and select among resources useful to learning how to write code, such as online documentation, tutorials, forums
    - A proactive attitude toward seeking outside help when necessary without undue procrastination
    - A collaborative and supportive attitude towards making oneself a valuable member of the community of developers
    - Choosing the “right” technological foundation for a software project

Grades

Your grade this semester will be based upon a point system. You earn points by earning badges. Each badge has a certain number of points associated with it. The point cutoffs for various grades are in the table below. I will not round or curve. You are responsible for keeping yourself on track. To get an “A”, you need to be earning 8-10 points a week, on average. There should be plenty of opportunities to earn points. If for some reason, you find yourself without anything to do, please take your prof out for coffee (you can earn a point that way), and we can come up with a plan for you. ***Please see the important caveat to this grading system in the attendance policy below.***

Grade	Points
A	129-150
A-	120-128
B+	111-119
B	99-110
B-	90-98
C+	81-89
C	69-80
C-	60-68
D+	51-59
D	39-50
D-	30-38
F	0-29

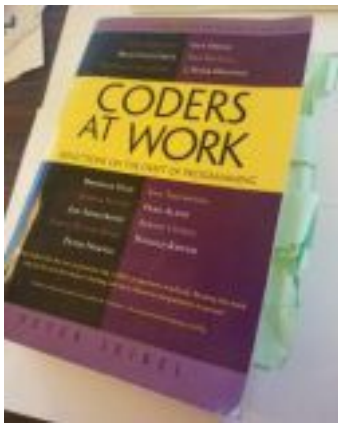
Attendance Policy

Attendance is mandatory. I will take attendance at the beginning of every class period. If you are not there when your name is called, you will not get credit for being there. If you are marked absent more than five (5) times during the course of the semester for any reason, your semester grade will drop by one letter. Woody Allen said, “80% of success is just showing up.” I care if you’re here. **Please note, even an “excused” absence counts toward one of your five absences!!!**

Some Interesting Books

There are NO required textbooks this semester. That being said, if you are really serious about becoming a rockstar programmer, here are some books that I highly recommend. If you do take the time to read a book, I’ll definitely create a badge that gives you the opportunity to earn some points for it. If there’s a book you want to read that’s not on this list, come talk to me beforehand and I’ll let you know if I think it’s worth your time.

Charles Petzold has written a fascinating and accessible introduction to the low-level architecture of computers called Code: The Hidden Language of Computer Hardware and Software. I require this book of my ISAT 252 students, but if you didn't take that class with me, ask someone who did–it's a great read. This book is amazing and one of those ones you'll want to keep forever and read to your kids. (Note, if you already read this one as part of ISAT 252, you can't get points for reading it again in this class.)



Coders at Work: Reflections on the Craft of Programming by Peter Seibel is literally chock full of coding wisdom. The book is a compilation of interviews with famous programmers, such as:

- Douglas Crockford, perhaps the world's foremost JS guru
- Jamie Zawinski, invented Netscape (now Firefox)
- Peter Norvig, basically a god in the AI world
- Don Knuth, maybe the only name almost all coders know

I found this book so fantastic, the picture at the right is my personal copy. You can see all of the sticky notes that I put on the pages everywhere I found some nugget of coding wisdom. This is a book you can read over and over and still get more out of it. It will definitely make you a better programmer.

Getting Real: The smarter, faster, easier way to build a successful web application, is by Jason Fried and the folks at 37 Signals, the people who invented the Ruby on Rails framework that powers many, many websites from big companies. This is a collection of very short essays that really cut through all of the BS that you tend to hear from a lot of business books on coding. They were the first I found to really nail some important concepts like following your own passion, solving your own problems, and putting as few features into your program as possible. You can also read it for free online.

Schedule

The schedule for this class is somewhat flexible and constantly evolving. The first few weeks, however, have been mapped out pretty tightly.

Wk#	Day	Topic	Homework
1	8/28	Course Intro	Socialize, Syllabize , Read What's Your Problem?
	8/30	What makes a good software idea?	Do Idea Generator Badge
	9/1	Present Ideas and Form Teams	Hang out with your Team
2	9/4	Social Context Analysis	Work on SCA, Create a Pitch
	9/6	Scenario Development	Work on Scenarios
	9/8	Presentation #1: What's your problem?	Work on SCA/Scenario Slides
3	9/11	Creating Mockups	Paper and Pencil Mockup
	9/13	Wireframing	Begin learning LucidChart and/or Balsamiq
	9/15	Presentation #2: Social Context Analysis	Convert Mockups to Wireframes
4	9/18	What is a Database?	Relational Database Reading TBD
	9/20	SQL vs. NoSQL	Reading TBD
	9/22	Presentation #3: Interface Mockups	Database Analysis Intro Tutorial/Video
5	9/25	Database Analysis I	ERD
	9/27	Database Analysis II	JSON Schema
	9/29	DB Analysis Lab Day	Work on DB Analysis
6	10/2	DB Analysis Lab Day	Work on DB Analysis
	10/4	Demo of Database Analysis Presentation	Finish DB Analysis Slides
	10/6	Presentation #4: Database Analysis	

7	10/9	Implementation: Test-Driven Development	TDD Lab
	10/11		
	10/13		
8	10/16		
	10/18		
	10/20		
9	10/23		
	10/25	TDD	Continue TDD Lab
	10/27	Intro to Code Review	Code Review
10	10/30	In-Class Code Review	Projects
	11/1	In-Class Code Review	Projects
	11/3	In-Class Code Review	Projects
11	11/6	Continuous Integration	Travis-CI Lab
	11/8	Travis-CI	Continue Travis-CI Lab
	11/10	Code Coverage	Codecov
12	11/13	Projects	Projects
	11/15	Projects	Projects
	11/17	Presentation #5: Practice Final Demonstration	Projects
Thanksgiving Break!			
13	11/27	Practice Project Presentations	Projects
	11/29	Practice Project Presentations	Projects
	12/1	ISAT Senior Symposium	Projects
14	12/4	Project Presentations	None

	12/6	Project Presentations	None
	12/8	Presentation #6: Final Demonstration	
End of Semester Exit Interviews			

Important Dates and Deadlines

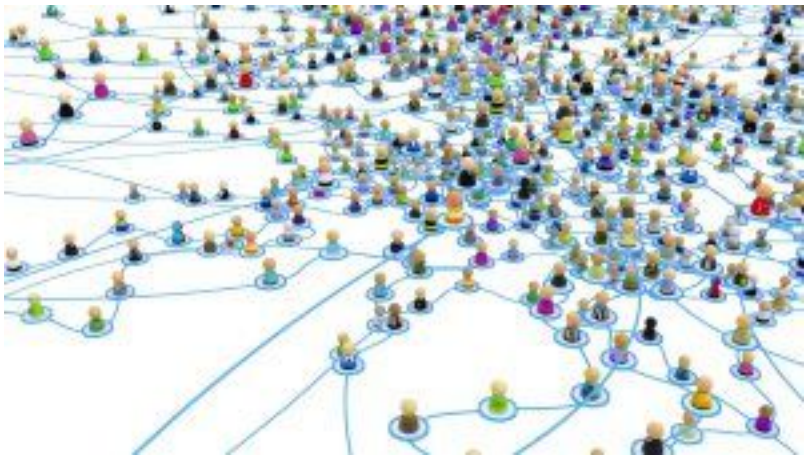
At the behest of the registrar, a list of dates you may wish to take note of:

- 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.