Software Engineering
&
Its Career Opportunities

Stephen W. Neville, PhD, PEng (BC)
Associate Prof., Software Engineering
Dept. Electrical & Computer Engineering
University of Victoria

October 21, 2014
My Background & Research:

• Software Engineering for larger-scale software systems
  • System with thousands to millions of users/components
  • My focus is on software Engineering
    • Not Software engineering or software engineering

• Cyber-Security and Privacy

• Quality of Service (QoS) and Quality of Experience (QoE) in wired and wireless networks

• Fault Detection and Diagnosis in larger-scale engineering plants

• Statistical Data Analytics, Big Data, and Data Science
  • Primarily as these relate to Decision Support Systems
• One of 15 founding academic members of NSERC’s cyber-security focused Strategic Network - the Internetworked Systems Security Network (ISSNet)

• Coast-to-Coast initiative involving 8 universities and 22 industry/government partners.

• ISSNet ran from 2007-2014

• Trained over 100+ graduate students in cyber-security

• Founding Director of UVic’s Centre for Advanced Security, Privacy, and Information Systems Research (ASPIRe Centre)

• From 2011-2013
• I primarily do applied engineering work/research around industry identified problems

• This includes very active involvement in the Entrepreneurial Engineering Masters Program (EEMP)

• Were we have co-founded 5 active companies (combined with over 30 technically-focused employees)
• Hunter MacDonald, CEO of Tutela Technologies Inc. received the Top 30 under 30 by BC Business in 2014

• https://www.youtube.com/watch?v=Qlojm5F_u0Y

• All of the companies we have created are software focused

• In part,

  • Software focused companies are more cost-effective to start and grow than other types of enterprises

  • As Hunter said, all you need is a computer (and good software people) to create a company that can then sell products and services internationally
Outline

• *What is Software Engineering:*
  • i.e., versus Computer Engineering, Computer Science, etc.

• *What are some of the interesting problems:*
  • Scalability
  • Cyber-Security and Privacy

• *What are some of the growth areas:*
  • Cloud systems and services
  • eHealth
  • Big Data, Data Science, Data Analytics
  • Machine-to-Machine (M2M) systems and the Internet-of-Things
  • Environmental Engineering
What is Software Engineering?

• More than just being able to code.

• If all you known is how to program then there are a bit less than 1% of the world’s population that you’re competing with for jobs.

  • Discounting sub-Saharan Africa and rural Asia, this leaves a little less than 1% of ~5.5B people.

    • Roughly around ~50M people globally already known how to code.

    • Every university, college, and high school globally is teaching its students some basic ability to write code - so this number is only growing.

• It has become trivial to outsource modest programming tasks globally.
Major software companies are very active in trying to build frameworks to make coding even easier.

- For example,
  - IBM has a framework that lets you build a full small to medium scale eCommerce web site by point-and-click and then it instantiates the full back end system(s), including the data base(s), to IBM’s security best-practices.
  - Basic mobile apps have become relatively trivial to build as what counts is mainly the number of apps on an App Store.
    - i.e., This means that the Googles and Apples of the world are working to make the coding of apps as easy as possible.
    - The low hanging fruit of what used to take substantial skill and knowledge is now well within the purview of those with far less skills.
      - Much of the in-the-wild malware is generated by people with poor coding skills - but who have access to the underground marketplace of good malware development kits.
• One of our PhD students (a very good programmer who is now at Apple) was looking to earn some extra money while completing his PhD

• He looked at a coding task posted to one of these global programming work sites:

  • He estimated would take about him 2 weeks to code

  • The going bid was ~$300 USD

• If all you know is how to do the basics of coding you will be facing global competition for the jobs.

• The low end of coding is becoming commoditized and will go to the lowest cost bidders and countries.
• So “Software” is not the big issue in “Software Engineering” but the “Engineering” is!

• A useful analogy:

“One can build a very nice garden shed:”
• Stacking 100s of garden sheds does not get you the Petronas Twin Towers:

• The engineering of the two structures is *clearly* not the same - Scale matters!
• How does this translate into software systems?

Take a real example of a smaller scale software system designed to take data streaming in from a large suite of Internet data collection points and log this incoming data into a database.

• The rate of data coming in requires a multi-threaded system.

• This back end system is comprised of about 5 servers and cloud-deployed.

• The company we were doing the research for wanted to know how they should set the “block size” parameter to get the best system performance.

• i.e., A common “optimization” type of problem that occurs in engineering.
• We placed this “full system” into an instrumented testbed and emulated its behaviour against various incoming workloads and block sizes.

![Diagram of data collection system](image)

• We then measured the resulting response times.
  • i.e., The time taken from when the data enters the system to when it is finally logged into the data base.

• Question:
  • Should the performance get better or worse as the block size is increased?
  • i.e., Is lots of little things streaming in better than fewer larger things streaming in?
• Results:

• Clearly, block size impacts the system’s behaviour and things get significantly worse as the block size is increased.

• This contradicts the intuition that performance should have improved with increasing block sizes.

• Additionally, it is clear that the variance in the response time is also increasing as block sizes increases.

• Not only is the performance getting worse the variation in the performance is also getting worse (the system is becoming less predictable!).
• If the system is allowed to run longer then:

![Time Evolution of Throughput for a Commercial Distributed System](image)

To reduce the variation over the course of a test run, the architecture of the commercial system was changed to that of the prototype distributed system discussed in Chapter 3. Initially, this new system was tested without any delays to simulate the time spent parsing lines in the commercial system. This produced the results shown in Figure 39, where the processing time is not affected by the number of lines per post and remains fairly consistent across the course of each run, as demonstrated by the error bars.
• Clearly,

• These are not the behaviours one would have guessed existed without actually doing the emulations

• Software systems (even those of only modest sizes) can exhibit complex:
  
  • Non-linear,
  
  • Dynamic,
  
  • Stochastic,
  
  • Discontinuous,
  
  • and, potentially, Emergent Behaviours

  • i.e., Behaviours that arise (or are observable) only as the scale of the system increases.
Software Engineering

- Software Engineering is about how to build larger-scale software systems and solutions that:

  - Behave predictably
    - i.e., Generally in terms of desired quality of service (QoS) and quality of experience (QoE) levels
  - Meet reliability and availability needs
  - Are sufficiently secure
  - Do not leak private information

- We can do this relatively well for the prior generations of smaller-scale systems
- The interesting challenges ahead are how to do this well for the scale and complexities of modern software systems (and networks).
Why care about getting good at the stuff that is harder:

• The Risk Factor: Are Future U.S. Programmers Being Taught to Be Unemployable? (IEEE 24 Jan 2008)

• Two emeritus computer science professors from New York University argue that universities are so desperate to keep computer-science student enrollments up that they are dumbing down the curriculum to attract prospective students.

• This dumbing down, the professors say, is producing software engineers with:

“A set of skills insufficient for today’s software industry (in particular for safety and security purposes), and, unfortunately, matches well what the outsourcing industry can offer. We are training easily replaceable professionals.”
What this means in terms of jobs/employment

• Companies are having a very hard time finding people who have the required higher level skills and knowledge to engineer software systems.

• People who can code are relatively plentiful

• People with the next tiers of skills and knowledge are not

  • There is not going to be less software!

  • Systems are not going to get smaller and less complex!

• Areas where companies/government have strong on-going (and expanding) needs that require higher-levels of skill and knowledge are good career directions as they involve interesting work at significantly higher than average pay rates.

• Software engineering jobs also primarily exist in more urban centres

  • i.e., Where cultural amenities exist, etc.
Software Engineering

• Difference with:

  • Computer Engineering:
    • Primarily focuses on the interfaces/interactions between hardware and software systems
      • e.g. FPGAs, embedded systems, etc.
  
  • Computer Science:
    • Primarily focuses on the underlying theory of computer systems and software
      • e.g. Computational complexity, Turing machines, Graph theory, etc.
  
  • Software Engineering needs to address the building of large-scale software systems and networks that are: reliable, available, behave predictably, are sufficiently secure, etc. and have high degree of complex interactions and interdependencies.
  
    • This is a rapidly emerging field that is only going to get increasingly more important over time!
Interesting Problems

- Scalability:
  - Software systems that must support millions of customers and hundreds of thousands of transactions per hour have become commonplace
  - e.g., Amazon, airline flight reservation systems, ATM transactions, etc.
• Assume that you have a timing bug that occurs only once every one million transactions

• Further assume you are Amazon and this bug crashes your eCommerce system

• Then you would expect one crash per hour if you are processing 1 million transactions per hour

• To get to a usable system you would need to have a software solution that is in the area of parts per billion with respect its rate of serious bugs

  • i.e., One hard crash every 1,000 hours (or 41 days).

• Question:

  • How do you build/test software against a requirement of needing to understand its behaviours down to levels approaching parts per billion?
• These systems do not have simple behaviours:
• Security and Privacy:

• These systems are also subject to attacks by intelligent adversaries (including those backed by nation-states)
  
  • e.g., Flame, Stuxnet, etc.

• and they can leak private information
  
  • e.g., Target (40 million accounts), Sony (77 million credit card numbers), etc.

• Generally, if you lose someone’s credit card number you will need to spend between $100-$300 dollars to address the need to put in place credit checks and other costs

• Multiply this by the millions of credit card numbers lost in the above incidents any you get an idea of the business costs associated with these types of privacy breaches!
• Different geo-political jurisdictions have different privacy regulation, acts, and statutes

• Violations can have direct legal impacts and consequences.

• If you are a company you do not want to have a Privacy Commissioner do an investigation and find you at fault.

• This can have significant effects on your business’s reputation

• Particularly in a climate in which the average public is becoming increasingly more aware of, concerned with, and sensitive to privacy issues.
• Both security and privacy are hard open challenges
  • Particularly for large-scale software systems and solutions

  • Advantage:
    • These problems are unlikely to be solved in the near term
      • i.e., Good career longevity
    • They are becoming increasingly important for industries to address
      • i.e., They pay well
    • And, the jobs generally do not get out-sourced over national-boundaries
      • i.e., The competition is largely local and not global.
Growth Areas

• Cloud system and services:

• Source: http://blogs.forrester.com/stefan_ried/11-04-21-sizing_the_cloud
5 Reasons Businesses Use the Cloud

Every year, more and more businesses are adopting cloud. While traditionally thought of as an IT decision, cloud is increasingly being considered a business decision to enable company functions. Take a look at five reasons why more businesses are adding the cloud to their technology arsenals.

1. It offers better insight and visibility
   Businesses are using cloud technology to support their analytics efforts. Of leading organizations:
   - 54% use analytics extensively to derive insights from big data
   - 59% use cloud to share data seamlessly across applications
   - 59% intend to use cloud to access and manage big data in the future

2. It makes collaboration easy
   Cloud allows work to be accessed from anywhere on multiple devices, making cross-functional collaboration much easier. Here’s what leading organizations—those that are gaining competitive advantage through cloud—cite as popular uses:
   - 58% collaborate across the organization and ecosystem
   - 59% improve integration between development and operations

3. It can support a variety of business needs
   Companies are forging a tighter link between technology and business outcomes. Take a look at the business functions companies have migrated to the cloud:
   - 18% messaging
   - 15% storage
   - 13% office/productivity suites

4. It allows for rapid development of new products and services
   The cloud offers businesses valuable capabilities. Here’s what leading organizations say it enables them to do:
   - 52% use it to innovate products & services rapidly
   - 24% are able to offer additional products & services

5. The results are proven
   From business growth to increased efficiency, businesses using cloud are realizing benefits across the company:
   - 25% of businesses saw a reduction in IT costs
   - 55% saw an increase in efficiency
   - 48% saw improvement in employee mobility

Sources: CDW, IBM Center for Applied Insights
• eHealth systems and services:

Source: http://www.mhealthtalk.com/home/healthcare-statistics/
• Why this matters

![Annual Per Capita Healthcare Costs by Age](image)

• Heath care will need to become far more efficient if the costs are to be contained against a growing elderly population.

• Part of the answer is developing better IT solutions and services

• i.e., personal medical tracking devices are appearing
• These are then tied into data services to track health

• Also a rise of embedded and wearable medical devices and sensors (which then connect into networks)
• Data Science, Data Analytics, Big Data
Data is now everywhere

The low cost of large-scale data storage means it is now easy to collect and store lots of data about everything.

Big Data: This is just the beginning

Volume in Exabytes

Percentage of uncertain data

You are here

© 2013 International Business Machines Corporation
• Analysis of data (particularly at large scales) is a software systems issue.

• But, also requires good backgrounds in areas such as:
  • Probability and statistics, digital filtering, optimization, artificial intelligence, stochastic processes, etc.
• Machine-to-Machine (M2M) systems and the Internet-of-Things (IoT):
  
  • There are currently about 1 Billion computers on the Internet and 2 Billion cell phones
  
  • The estimates are that M2M/IoT will expand this to 50B to 100B network connected devices.
  
  • i.e., Human generated traffic is likely to become a small fraction of the traffic crossing the Internet.
• This leads to:
• Smart Grids:
• Smart Buildings:
• Smart Cities:
• Self-driving cars:
• Smart farming/agriculture:
• Environmental Engineering:

• To reduce our impacts on the environment we can:
  
  • Get better at producing energy from alternative (non-petroleum) sources.
  
  • Get better (more efficient) in using the energy we have.
  
  • Get better at re-using and re-purposing the waste products we produce.
In many cases these issues reduce (in part) to:

- Deploying significant numbers of sensors,
- Using networks to collect the larger volumes of real-time data these sensors produce,
- Analyzing the data to extract knowledge and understanding from it, and then
- Where possible, using the knowledge gained to control actuators and feedback systems.
- Big parts of how to do this rest on the types of larger-scale software systems and solutions discussed above.
Computing and Data Analysis for Environmental Applications

Instructor(s)
Prof. Dennis McLaughlin

MIT Course Number
1.017 / 1.010

As Taught In
Fall 2003

Level
Undergraduate

Transcribed Versions

Course Description
This subject is a computer-oriented introduction to probability and data analysis. It is designed to give students the knowledge and practical experience they need to interpret lab and field data. Basic probability concepts are introduced at the outset because they provide a systematic way to describe uncertainty. They form the basis for the analysis of quantitative data in science and engineering. The MATLAB® programming language is used to perform virtual experiments and to analyze real-world data sets, many downloaded from the web. Programming applications include display and assessment of data sets, investigation of hypotheses, and identification of possible casual relationships between variables. This is the first semester of two courses, Computing and Data Analysis for Environmental Applications (1.017) and Uncertainty in Engineering (1.010), being jointly offered and taught as a single course.
Conclusions

• Computers and software are not going away

• If anything, they are becoming increasingly in-grained into the core of many of the systems and services that modern societies rely on.

• Even if you do not become a Software Engineer you are going to be dealing with lots of software and software systems

  • And you’ll need to be able to competently write you fair share of software.

• As the system scales and complexities increase the core needs move away from just being able to write code to being able to engineer software systems that work, are sufficiently secure, are reliable, available, etc.
• Software Engineering is still an emerging discipline

• But, the skills and knowledge required underly a large number of the directions where modern societies are going.

• Getting competent skills and knowledge in areas of:
  • High and rapidly growing demand
  • Critical important to industry and governments
  • Which have substantial barriers to entry
    • i.e., It takes significant time, effort, and hard work to get the skills
  • Is rarely a bad career move!

*But a lot more is involved than just being able to code!*…
Questions ?