CS777 - Basics
Spring 2009

Instructor: Dr. Olly Gotel (ogotel@pace.edu; Room: 212)

Remember there is no class on Feb 16 due to Presidents' Day

Part A : Reading

- Read Chapter 1 of Lyu (the online text, link on my website) in preparation for the next reliability session. Draft short answers to questions 1.2, 1.3, 1.5 and 1.7. Bring a hard copy of your answers to class.
- Download the Software Quality chapter of the SWEBOK from my website – skim it.

Part B : Watching

- Watch the 50-minute film "When Engineering Fails" by Henry Petroski – I will email you the link to your Pace email (download Realplayer)
- The film is about one particular field of engineering – civil engineering. You will see that these guys take their work very seriously … they even look the part! As you will see - they can have a huge price to pay if they don't
- I want you to think very long and hard about the message this is telling you. We say software *engineering*… but in what way is what we do when we develop large complex software systems actually a form of engineering? Is it in any way comparable? Are our tools and techniques as apt? Could we do what we do better?
- As you watch this, I want you to think about and jot down lessons – i.e. things that you hear discussed that you believe can be equally applied to the development of software systems. Perhaps add this to the list of good practices we created after going through all the forensics examples in class. Bring your list to the next class.

Part C : Investigating

Start looking around and start asking questions….

- Look at how quality characteristics are specified in documents at your work – find an example. What qualities are specified? Can you see any requirements written for reliability? How are they written? Are they written in terms of reliability targets or are they somewhat vague? Can you find out how such requirements are actually tested for and measured in the products your organisation develops? If you see no reliability requirements or targets anywhere – exactly **how reliable** is the stuff you build? Any ideas? How reliable would you expect it to be to be serviceable? How would you quantify this?
- Examine some shrink-wrap software. Take a look at the warranty. What is the vendor telling you about the product's quality and reliability? For open source fans, what does your favourite open source software have to say about its quality or reliability? Find an example and bring it to class.

The above are all small tasks to keep you moving along and on your toes. Please do not forget to look at the term paper specification and get thinking on that – this is to be a more detailed, longer and ongoing piece of work. Email me about what you want to do.