CS 430 - Lecture 01 - Introduction to Software Engineering

Collin Roberts

September 14, 2023

Outline

Introduction to CS 430 - Course Outline

Look up the Course Outline on the unsecured course website.

Introduction to CS 430 - Course Outline

1. This component of the evaluation for the course is still fairly new this term (used once before):
    1. Case Studies on Kritik

Introduction to CS 430 - Course Outline

1. The Case Studies will be marked via a peer evaluation tool called Kritik. The use of Kritik is motivated by my desire to develop your critical thinking skills more than the usual approach of having you hand in assignments which are marked solely by Teaching Assistants does. Since this will be my second offering using Kritik, I am continuing with a blended approach between Kritik and traditional assignments.
2. Kritik is not yet supported across the University of Waterloo, so there will be a small subscription fee of $24 (I think, and will confirm soon) for you to use Kritik this term. If the subscription charge is prohibitive for you, then please reach out to me.
3. More details about Kritik will be posted on the unsecured course website in advance of the release of Case Studies #1.

Introduction to CS 430 - Course Outline

1. I will announce suggested pre-reading from the text for all the future lectures, by email.
2. **Clickers**
   1. iClicker Remote
      1. Register your iClicker using the instructions on LEARN.
      2. If your Participation grade still shows as 0 on LEARN after a few lectures, then please contact the instructor to correct the registration of your iClicker.
   2. Paper Submit one piece of paper to me at the end of each lecture, indicating
      1. student number / login ID / both, and
      2. one line per CQ, numbered my way, indicating which of options A–E you chose.

Introduction to the Scope of Software Engineering

1. **Software Engineering:** the idea of applying Engineering Principles to the building of **big** software products.

Historical Aspects

Examples:

1. On November 9, 1979, U.S. Strategic Air Command had an **alert scramble** when the worldwide military command and control system (WWMCCS) computer network reported that the Soviet Union had launched missiles aimed at the U.S.A. A simulated attack was misinterpreted as the real thing. See the text for full details.
   Moral: Software faults can have disastrous real world consequences!
2. A Standish study of 9236 software projects completed in 2006 revealed (see text Fig 1.1) that
   1. only 35% were delivered successfully,
   2. 19% were cancelled and
   3. 46% were late, over budget, and/or had features missing.

Historical Aspects

Remarks:

1. Bridges occasionally collapse, and power generators occasionally fail, but not nearly as often as operating systems crash or billing systems produce bills for incorrect amounts.
2. Even when software is delivered fault-free, it is often late, over budget, and/or fails to meet all the user's requirements.
3. This motivates the following key definition.

Historical Aspects

Definition 1

The **software crisis** *(or* **software depression***) is the phenomenon whereby, all too often, software is delivered:*

1. *with faults,*
2. *late,*
3. *over budget, and/or*
4. *not meeting all the user's requirements.*

Historical Aspects

Remarks:

1. Applying the same principles that traditional engineers use can help improve the delivery of software.

Historical Aspects

Definition 2

Software engineering *is a discipline whose aim is the production of software that*

1. *is fault-free,*
2. *is delivered on time,*
3. *is delivered within budget, and*
4. *satisfies the client's needs.*

Furthermore the software must be easy to modify when the user's needs change.

Historical Aspects

Remarks:

1. The name software engineering indicates that software developers will have better success if they use the same principles as traditional engineers.
2. Software engineering is new field with a broad scope - math, CS, science, engineering, management, etc.
3. Software Engineering is a response to the Software Crisis.

Economic Aspects

Question: If a new coding method becomes available which is 10% faster than the current method, then should we adopt it immediately?

- CQ 3

Pros	**Cons**
-short term: lower development costs -longer term: compound short term savings -possible improved security features	-higher maintenance costs with a blended system -possible need to rewrite existing code -possible compatibility problems -new environment unproven, possibly unstable -no benefit with respect to maintenance costs -might affect user experience in unexpected ways -training / learning curve -new code could be less robust -might require hardware changes -cost of purchasing the new development studio is not stated -possible issues with stability, performance -new code might be of lower quality

Economic Aspects

Moral: This is not a clear yes/no answer. More analysis is still required.

Remarks:

1. The "Pro"s touch the development phase; the "Con"s reveal impacts in other phases.
2. It turns out that historically, maintenance costs have grown faster than development costs. **Moral:** reducing maintenance costs is a bigger win.

Economic Aspects

This is Coding example. Coding = 10-15% software development effort. Similar principles apply to all aspects of software development