Програм хангамжийн хөгжүүлэлт (ICSI402)

Lecture1: Introduction to Software Construction

МУИС, ХШУИС, МКУТ-ийн багш

Маг. Довдонгийн Энхзол

Хүн бол энэ ертөнцийн хамгийн төгс төгөлдөр систем юм.
Хүн компьютерийн системийг бүтээсэн тул
түүнтэй холбоотой бүхнийг ойлгож чадна.

Д.Энхзол

Course Information

- **Contact details**
  - Enkhzol Dovdon
  - Өрөө: 3А байр 225 тоот
  - Email: enkhzol@seas.num.edu.mn, assignmentnum@gmail.com
- **Time in Meeting with students**
  - Monday 12:30-14:00
- **Lectures**
  - Fridays at 17:40-19:10
  - Room 200, 3A building
- **Labs**
  - Wednesdays at 07:40-10:05, 10:05-12:30
  - Room 108, 215, 3A building

Course Information

Assignments	40
Project	10
Enrollment	5
Term exam	15
Final exam	30
Total	100

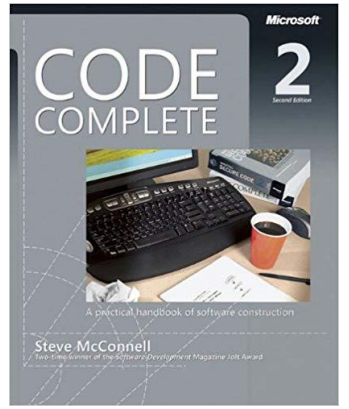
No cheating, No late

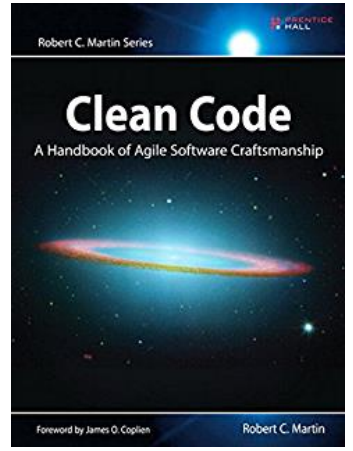
Textbook

- Recommended textbooks:

1. Steve McConnell, Code complete

2: A Practical Handbook of Software construction, O'Reilly, 2009 2. Ian Sommerville, Software Engineering, 9th edition, Addison-Wesley, 2010 АШИГЛАХ НОМ, МАТЕРИАЛ





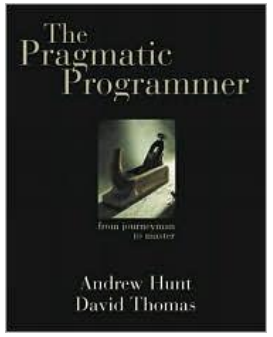
- Other textbooks:

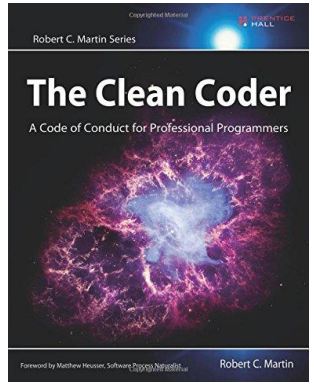
Pragmatic Programmer, Hunt & Thomas *Effective Java* 2nd ed, Bloch

Serious programmers should study these

Decent "Java book" is a wise thing to have

Core Java Vol I, Horstmann





Goals

- ICSI402 will teach you to how to write correct programs

- What does it mean for a program to be correct?
  - Specifications

- What are ways to achieve correctness?
  - Principled design and development
  - Abstraction and modularity
  - Documentation

- What are ways to verify correctness?
  - Testing
  - Reasoning and verification

Main topic: Managing complexity

- Abstraction and specification
  - Procedural, data, and control flow abstractions
  - Why they are useful and how to use them
- Writing, understanding, and reasoning about code
  - Will use Java, but the issues apply in all languages
  - Some focus on object-oriented programming
- Program design and documentation
  - What makes a design good or bad (example: modularity)
  - Design processes and tools
  - Pattern
- Pragmatic considerations
  - Testing
  - Debugging and defensive programming

The goal of system building

- To create a **correctly** functioning artifact
- All other matters are secondary
  - Many of them are ***essential*** to producing a correct system
- We insist that you learn to create correct systems
  - This is hard (but fun and rewarding!)

Related skill: *communication*

- Can you convince yourself and others something is correct via precise, coherent explanations?

Why is building good software hard?

- Large software systems are enormously complex
  - Millions of “moving parts”
- People expect software to be malleable
  - After all, it’s “only software”
- We are always trying to do new things with software
  - Relevant experience often missing

- Software engineering is about:
  - Managing complexity
  - Managing change
  - Coping with potential defects
    - Customers, developers, environment, software

Programming is hard

- It is surprisingly difficult to specify, design, implement, test, debug, and maintain even a simple program
- ICSI402 will challenge you
- If you are having trouble, *think* before you act
  - Then, look for help
- We strive to create assignments that are reasonable if you apply the techniques taught in class…
  … but likely hard to do in a brute-force manner
    … and almost certainly impossible to finish if you
      put them off until a few days before they're due

A Problem

"Complete this method such that it returns the index of the max of the first **`n`** elements of the array **`arr`**."

```
int index_of_max(int[] arr, int n) {
    ...
}
```

What questions do you have about the *specification*?

Given a (better) specification, is there exactly 1 *implementation*?

Moral

- You can all write the code

- More interesting in:
  - What if n is 0?
  - What if n is less than 0?
  - What if n is greater than array length
  - What if there are “ties”?
  - Ways to indicate errors: exceptions, return value, …
  - Weaker versus stronger specifications?
  - Hard to write English specifications (n vs. n-1)

- Software to Software engineering
  - Software is everywhere
  - What is software engineering?

Mathematics is called “the Queen of the Sciences”
Carl Fendrich Gauss(1777–1855)

I think “Nowadays, Computer Science is the King of Applied Sciences.”
Enkhzol Dovdon

Software is everywhere: cars

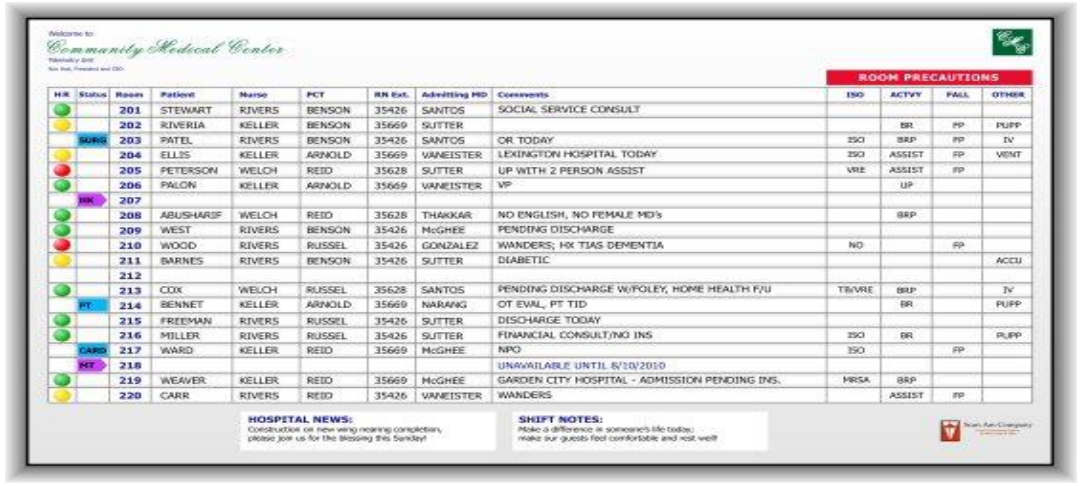
- *http://spectrum.ieee.org/green-tech/advanced-cars/this-car-runs-on-code/*
- “New cars now frequently carry 200 pounds of electronics and more than a mile of wiring”
- “…if you bought a premium-class automobile recently, it probably contains close to 100 million lines of software code…”

Software is everywhere: cars

- “The radio on many cars talks to the automatic transmission over an in-car network”

- “The airbag accelerometer, parking lights, GPS navigation, cell phone, and door locks also network so that in a serious accident, the car calls for emergency aid, sends the GPS coordinates of the accident, unlocks the doors, and flashes the car’s lights”

Software is everywhere: medical systems

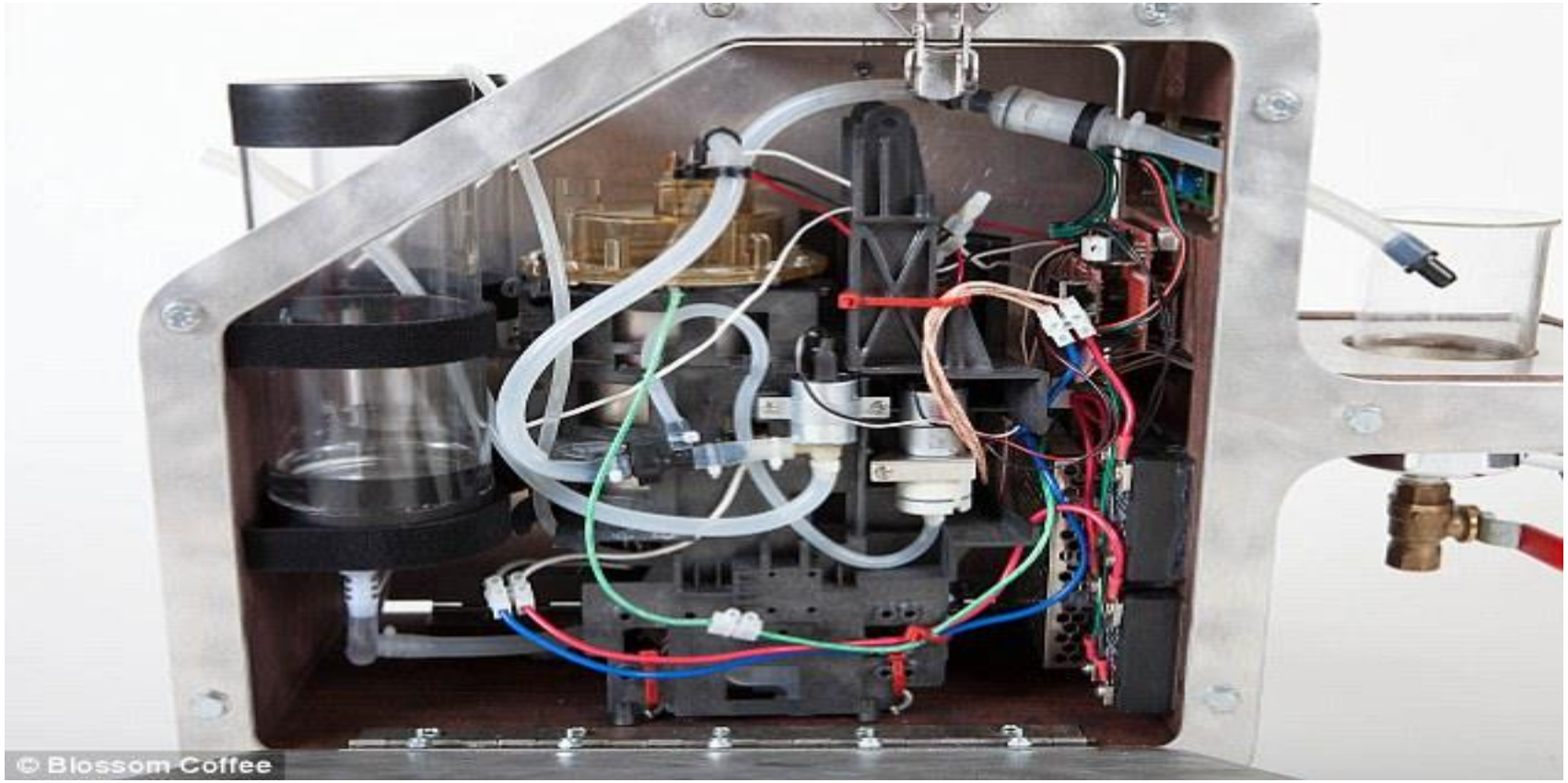


Software is everywhere: games

Software is everywhere: games

Software is everywhere: bonus

Software is everywhere: bonus



The machine also connects to WiFi and has a camera for a QR scanner. They hope that some day, coffee bags will have a QR code that the machine will recognize and brew appropriately.

- Introduction to software engineering
  - Software is everywhere
  - What is software engineering?

What is Software Engineering?

Software –

Engineering -

What is Software Engineering?

Software –

- Code
- Documentation, user manuals
- Designs, specifications
- Test cases
- Plans and schedules

Engineering -

What is Software Engineering?

Software –

- Code
- Documentation, user manuals
- Designs, specifications
- Test cases
- Plans and schedules

Engineering -

- Skill and knowledge
- Application of scientific principles
- Trade-offs, cost / benefit analysis

Software engineering

- *"A broad field that touches upon all aspects of developing and supporting a software system."* [Tsui, Karam, Bernal]

Software engineering

- *"A broad field that touches upon all aspects of developing and supporting a software system."* [Tsui, Karam, Bernal]

Software engineering

- *"A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers."* [Ghezzi, Jazayeri, Mandrioli]

Software engineering

- *"A discipline that deals with the building of software systems which are so large that they are built by a team or teams of engineers."* [Ghezzi, Jazayeri, Mandrioli]

Software engineering

- "*Multi-person construction of multi-version software*." [Parnas]

Software engineering

- "*Multi-person construction of multi-version software*." [Parnas]

Software engineering

- *“A discipline whose aim is the production of fault-free software, delivered on-time and within budget, that satisfies the user’s needs. Furthermore, the software must be easy to modify when the user’s needs change.”* [Schach]

Software engineering

- *"A discipline whose aim is the production of fault-free software, delivered on-time and within budget, that satisfies the user's needs. Furthermore, the software must be easy to modify when the user's needs change."* [Schach]

What is Software Engineering?

Informatics:

- The process of constructing software.
- Phases of development other than programming.
- Principles and qualities of enduring value.

What is Software Engineering?

Informatics:

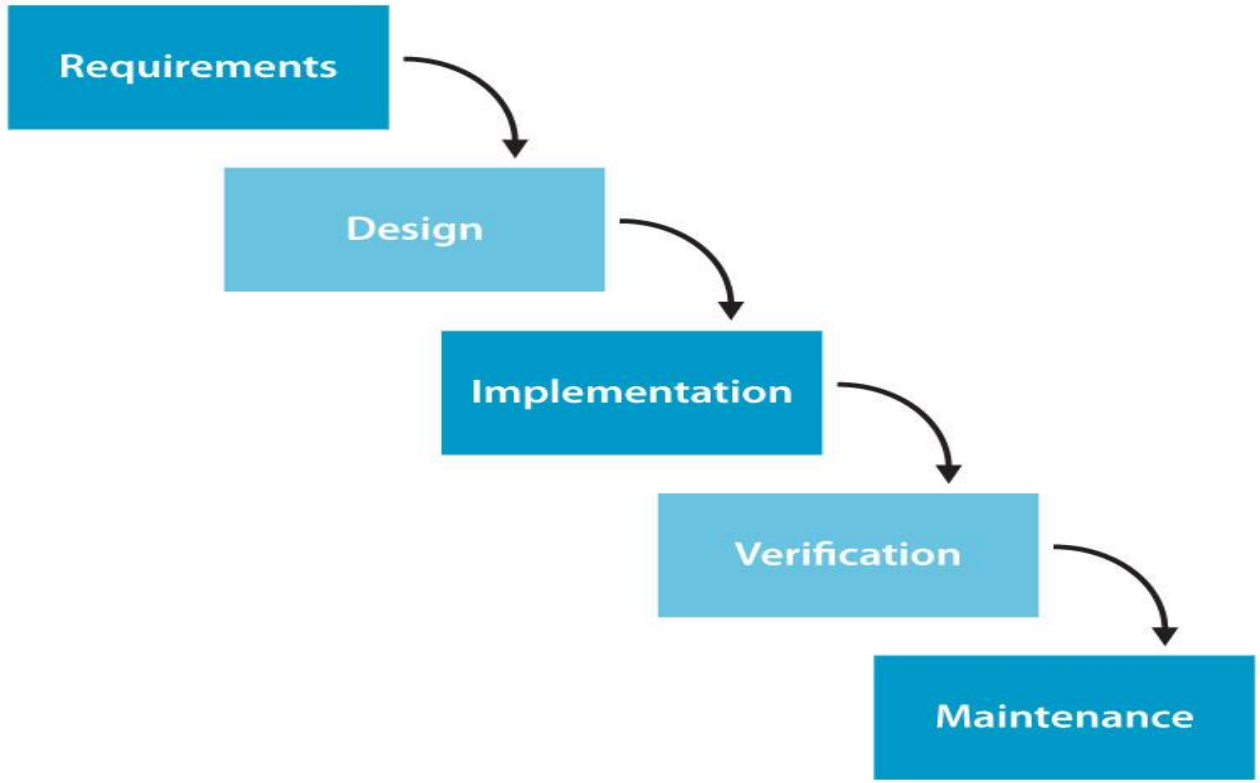
- The process of constructing software.
- Phases of development other than programming.
- Principles and qualities of enduring value.

Also of (lesser) interest (in this course):

- **Managing & scheduling software development teams.**
- **Making money – business models.**
- **Software's impact on users, organizations, and society.**

What is Software Engineering?

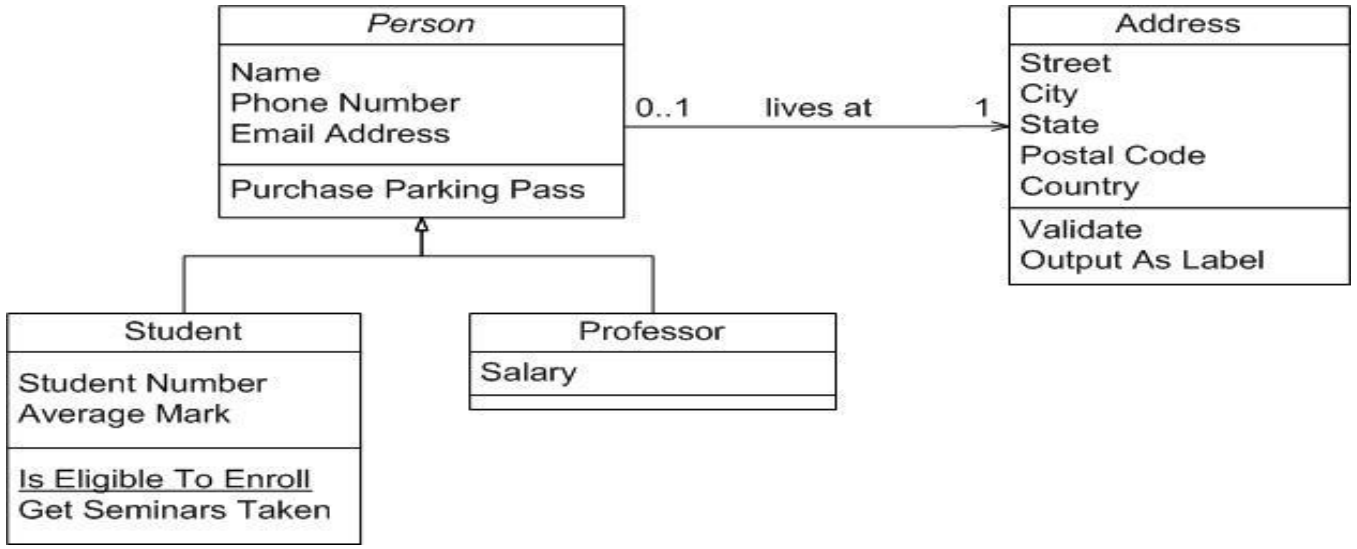
The process of constructing software.



What is Software Engineering?

Phases of development other than programming.

Design:



What is Software Engineering?

Principles and qualities of enduring value.

Correctness!

Reliability

Adaptability

MODULARITY

Consistency

Efficiency

Thank you for your attention ☺

Q & A