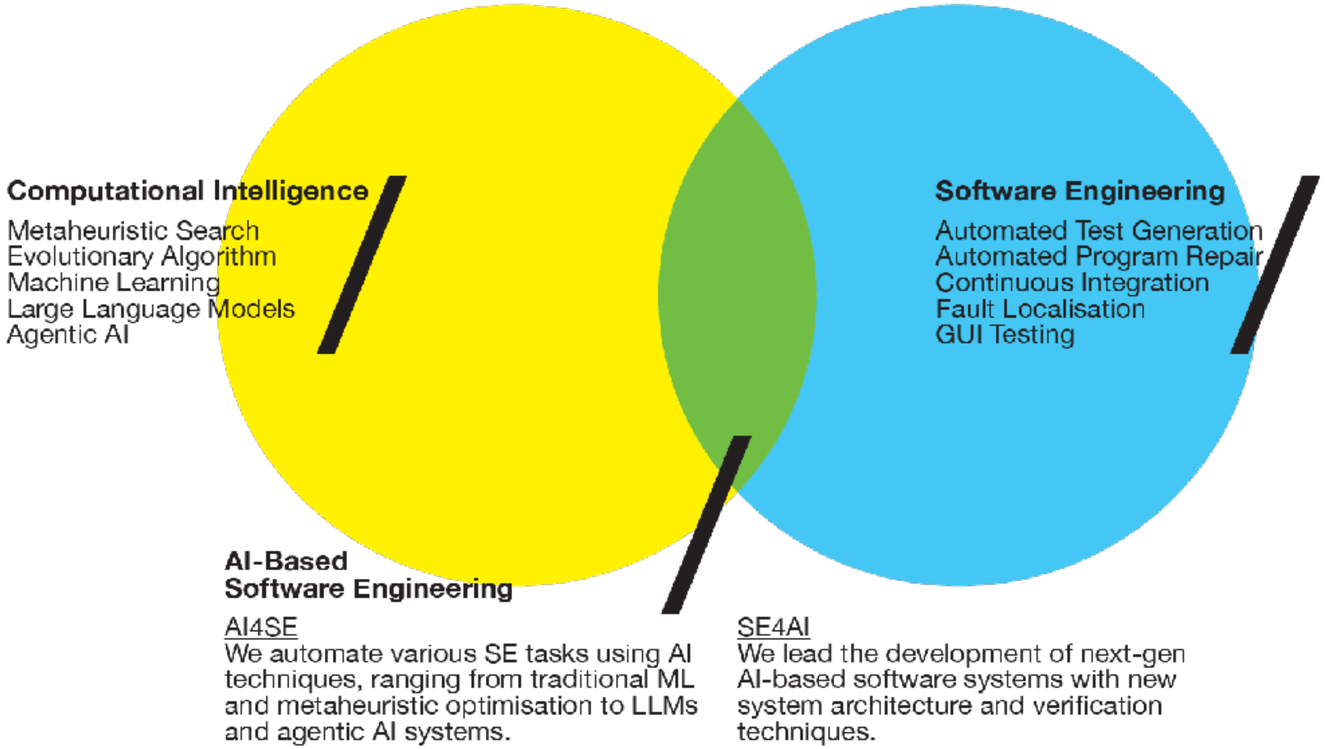
Introduction

CS454 AI-Based Software Engineering

Shin Yoo

Me

- Shin Yoo
  - PhD at King's College London, UK
  - Assistant Professor at University College London, UK
- COINSE (Computational Intelligence for Software Engineering) Lab
- Research interest: SBSE, regression testing, automated debugging, evolutionary computation, information theory, program analysis…
- shin.yoo@kaist.ac.kr

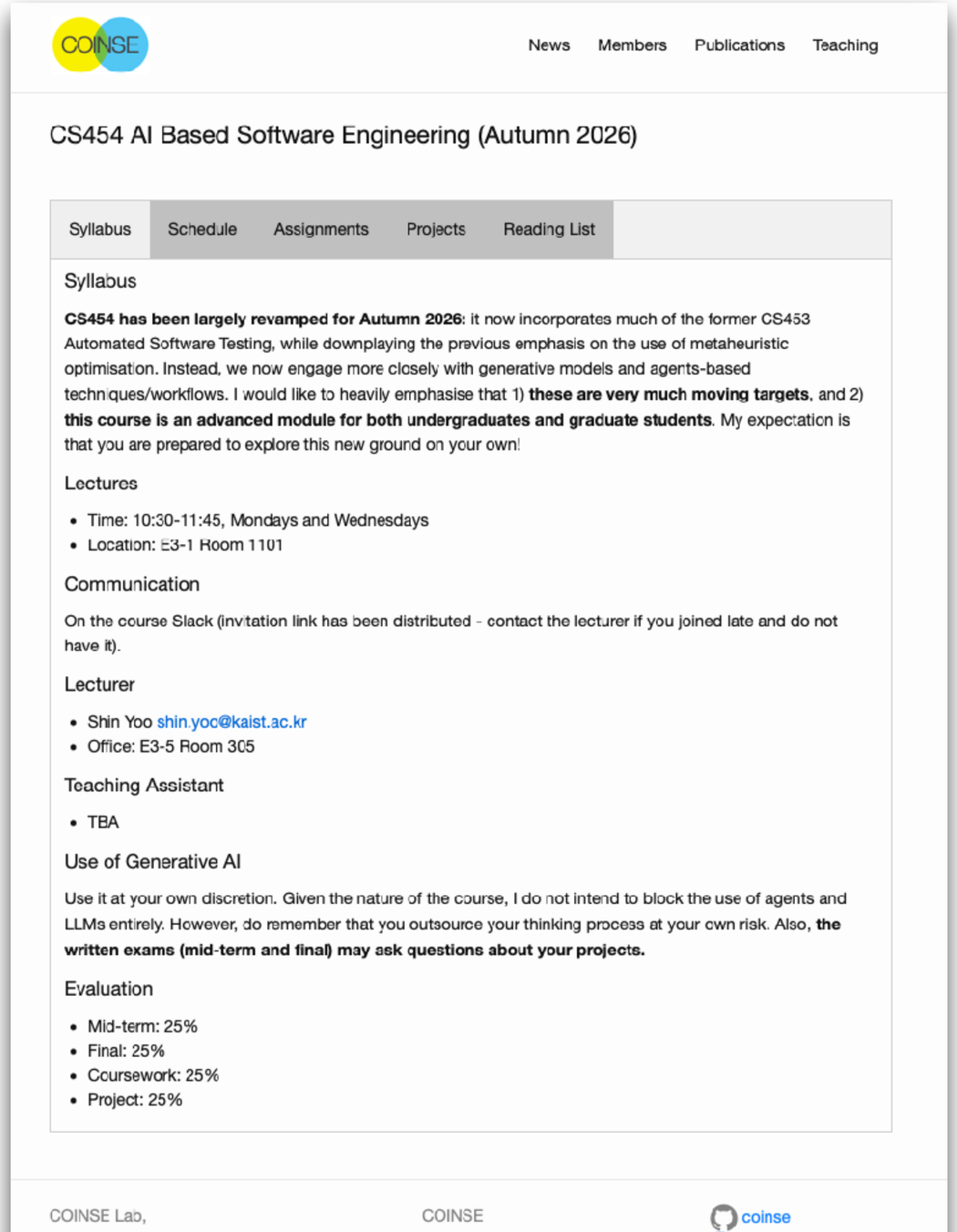


Communication

- We will use Slack for all class communication. Invitations will be sent by email from KLMS, so watch out.
- Join the workspace, no excuse.
- Change your display name to: your full name (your student id)

Course Webpage

- Reading materials will be either linked or provided on the page
- http://coinse.github.io/teaching/2026/cs454/



CS454 Major Update

- Essentially 453 Automated Software Testing is being merged into 454. This was in some sense inevitable because:
  - Automation in SE cannot be discussed without AI/ML
  - So much of the target topics in 454 is testing techniques.
- What we are losing, as a result:
  - Some details about optimisation algorithms
- CS453 as it was will not be offered for the time being; I will focus on CS454 from now on.

Course Evaluation

We are bringing back the written exams (you know why, right?)

- **Mid-term (30%) and Final (30%)**
  - There will be questions about assignment implementations.
- **Assignments (40%)**
  - Four Individual Courseworks, each with 10%
  - Each assignment is a scaled down real SE problem that you need to solve with optimization/AI techniques. We will grade based on the following criteria:
    - Quality and performance of the solution
    - Any potential novelty
  - Participate in Slack discussion + Q&A
  - **Late submission: 0.7 weight if submitted within a week. Submission will not be accepted if it is late more than a week.**

AI Usage Policy

- For programming assignments, you are free to use them.
  - I will set slightly higher goals :)
  - I will ask about actual design choices in the exam.
- Exams are physical written exam, so irrelevant.

Schedules

- I am one of the judges in the national AI Rookie Competition (https://ai-rookie.or.kr/), which requires my presence for some evaluation.
- I am attending IEEE/ACM International Conference on Automated Software Engineering, as well as its steering committee.

Textbook & reading material

- No textbook (we will read up to the bleeding edge)
- Lectures contain strongly recommend reading lists
- Supplementary books:
  - Introduction to Evolutionary Computing, *A. E. Eiben & J. E. Smith*, Springer
  - A Field Guide to Genetic Programming, *Ricardo Poli, William B. Langdon, & Nicholas McPhee* (freely available online at http://www.gp-field-guide.org.uk)

What are we really going to learn here? 🤓

Artificial Intelligence Based Software Engineering Shin Yoo 2020 Fall

Good course! Course teaches you how to approach to optimization problems when you encounter them. Name AI can be misleading, it means traditional AI which solves problems by doing optimization.

Artificial Intelligence Based Software Engineering Yoo, Shin 2025 Fall

강의 내용은 흥미로웠습니다. 흔히 "인공지능"이라고 했을 때 생각 나는 그런 것을 배우고 싶은 기대를 하고 계시다면 전혀 추천하고 싶지는 않습니다.

[강의]
perceptron을 기반으로 한 인공 신경망 등장 이전의 인공지능? 관련 내용을 다룹니다. hill climbing, genetic algorithm 등 metaheuristic에 대해서 폭넓게 배울 수 있던 시간이었습니다. 다시 곱씹어보니 인공지능이라기보단 오히려 알고리즘? 에 좀 더 가까운 느낌입니다. 교수님의 다른 수업(소공개, 테스팅)과 다루는 주제가 비슷합니다.
"인공지능"이 아니라 "소프트웨어 공학"에 초점이 맞춰져 있다고 보시면 됩니다.

The course has been evolving…

- It started as a special topic on Search-Based Software Engineering
  - Application of meta-heuristic optimisation to software engineering
- Then Machine Learning became increasingly important
  - DNN testing became a small but important area in SE
  - Cannot really “teach” the entirety of ML here, so we touch on some of the applications
- Then LLMs exploded
  - We now talk about LLMs a bit, as well as what we can do with them for SE

(Also, to my defense…)

Un-supervised

Supervised

Machine Learning

Reinforcement

Learning

Planning

NLP

Perception

Motion

Reasoning

AGI?

What

How

Entire Machine Intelligence

$\vdash \forall x \neg A(x) \rightarrow \neg A(a)$

$\vdash A(a) \rightarrow \neg \forall x \neg A(x)$

$\vdash A(a) \rightarrow \exists x A(x)$

Symbolic

Statistical

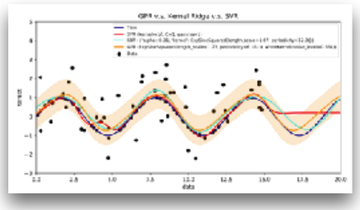
Soft Computing

Artificial Neural Nets

Deep Neural Networks

Evolutionary Computation

Meta-heuristic



AI4SE vs. SE4AI

We will discuss both sides

- AI4SE: using AI as a technology to improve and automate Software Engineering processes
  - Automated Testing, Automated Program Repair, etc…
- SE4AI: using what we know from building traditional software to improve development of AI-centric systems
  - Testing DNNs, Designing AI agents, etc…

One Core Lesson: real SW is messy

- Software is not algorithm, it is NOT even a collection of algorithms.
- Code can be more like clay rather than Lego blocks - **and this can be a good thing**.
- Randomness can be a useful agent for good, provided we can **guide** it.
- Reasoning about correctness of code, without involving human brain, is surprisingly difficult.

Why AI and SE?

(wait, before that…)

What is SE?

Science: intellectual and practical activity encompassing the systemic study of the structure and behaviour of the physical and natural world

Mathematics: abstract science of number, quantity, and space, either as abstract concepts or as applied to other disciplines

Engineering: branch of science and technology concerned with the design, building, and use of engines, machines, and structures

Software as science

- Precise computation and algorithm
- Study of truth
- Simplicity of theory

Software as engineering

- Consideration of adequate cost
- Study of utility
- Scalability of theory

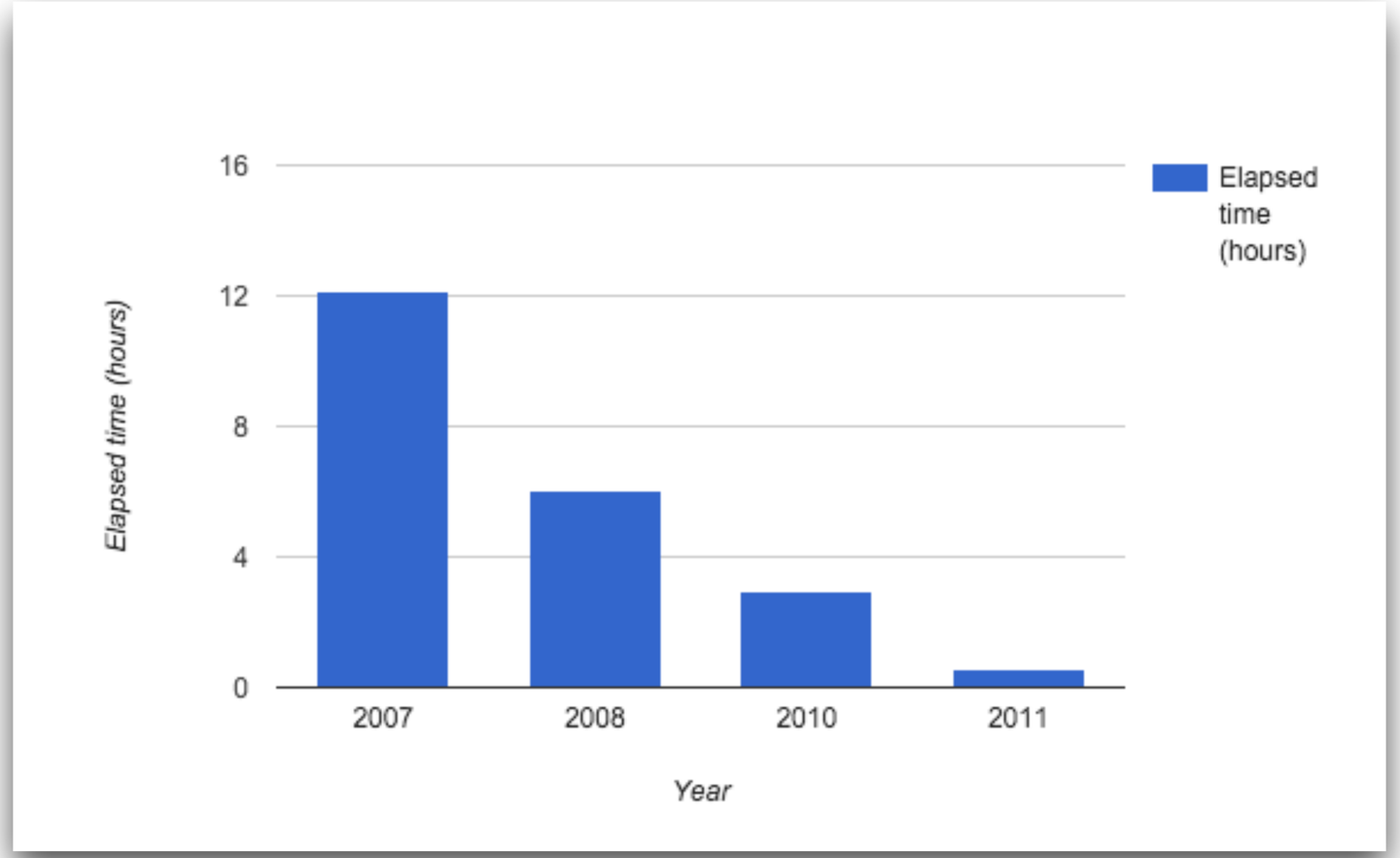
Science of sorting

```python
def mergesort(x):
    """ Function to sort an array using merge sort algorithm """
    if len(x) == 0 or len(x) == 1:
        return x
    else:
        middle = len(x)/2
        a = mergesort(x[:middle])
        b = mergesort(x[middle:])
        return merge(a,b)
```

#precise #theory #O(nlogn) #provable

Engineering the scalability of sorting

from “History of massive-scale sorting experiments at Google”



- Tuning cluster configuration
- Changing the file system entirely + new encoding to reduce write amount
- Hardware tuning (I/O block size + SSD)
- Correctness check only came in 2010 :)

https://cloud.google.com/blog/products/gcp/history-of-massive-scale-sorting-experiments-at-google

Science

- Is the theory correct?

Engineering

- Is the **implementation** correct?

**(On top of everything,
the most important question)**

Is the solution cheap enough?

- How can we maintain quality when team members change?

AI4SE

- AI seen increasingly as a cheap yet highly effective and flexible technique that can handle everything :)
  - But can it really handle everything?
  - What do we exactly mean when we say “AI” nowadays?
  - What are the real cost? Human? Financial? Environmental?

SE4AI

- AI is also increasingly part of larger existing systems.
- Suddenly we have a very powerful blackbox that makes decisions that were not thought to be easily computable.
  - How do we design/implement/test/maintain systems with such components?

Aim of this course

- We will NOT learn how to write programs faster with agents (after all, we did not learn how to use Visual Studio Code, or any other IDEs, right?).
- We will NOT go through details of AI/ML materials; rather, we will try to be informed(?) outside users.
- We will try to connect operational behaviour of various AI/ML/optimisation approaches with topics in software engineering.
- The course will not be easy: sometimes it may feel very high level and abstract, but it is intended that way (note that this is advanced course open for both undergraduates and graduates).