ACM Data Science Task Force Course Example

Software Structure
Harbin Institute of Technology, Harbin
Zhongjie Wang

Knowledge Areas that contain competencies (knowledge, skills, and dispositions) covered in the course

Knowledge Area	Total Number of Contact Hours
Software design and development, Software testing	64
Programming, Data structures	16

Where does the course fit in your undergraduate Data Science curriculum?

"Software Construction" is an important professional basic course in the computer category/software engineering category. This course is a professional core course, and most students who take this course are engineering subjects. The majority of the students who take this course are engineering students in the middle grade, with slightly more boys. Learners of this course need to have certain computer knowledge such as data structures and algorithms, C language programming, and Java language programming.

Is this course from or used in other curricula/majors?

Part of this course refers to the MIT 6.031 course, but the teaching content of MIT is not exactly the same as that of HIT. Most of the content of HIT chapters 4-9 is not covered by MIT. Some content of MIT exceeds the scope of HIT teaching.

What is covered in the course?

This course enables students to understand the quality standards and goals of software construction on the basis of high-level language programming, learn the basic process of software construction, in-depth study of abstract data types and object-oriented programming (OOP), and have a preliminary grasp of key quality goals (may be Comprehensibility, maintainability, reusability, robustness, performance) software construction basic technology, understand software code reconstruction and advanced construction technology for more complex software systems, so as to have complex software construction methods and quality goals ability.

What is the format of the course?

The total hours of this course is 80 hours, including 56 hours of classroom teaching and 24 hours of experiments. Understand which quality objectives should be considered in the software development process, master the basic software construction technology for key quality objectives, form a software development thinking mode oriented to quality objectives, and have the ability to use programming language features and tools for program design and implementation; The core idea of "abstract programming", master the basic process of object-oriented software development, be able to abstract and model practical application problems, have the ability to use models to effectively express and communicate with developers and users, and have the ability to use OO for coding System design and implementation capabilities. From focusing on the conversion of a single development link to focusing on the entire development process, we have the ability to analyze and evaluate the whole life cycle system

according to the quality characteristics expected by users, find defects in system design and make optimizations and improvements; understand that complex software systems are relatively simple The essential difference of the program, a preliminary grasp of the use of various software development tools for coding, testing and quality assurance, and the ability to use modern software construction tools for high-quality and efficient software development.

Set up 6 experiments covering each key knowledge point, set up 4 hours of laboratory guidance for each experiment, and the teacher and TA will answer and guide on-site. Set up 6 experiments, including:

1. Java programming foundation/testing/building foundation
2. ADT/OOP design and coding
3. Design and development for code maintainability/scalability/reusability
4. Code analysis and optimization for robustness
5. Code static/dynamic review and optimization
6. Multithreaded programming

Each experiment is completed by the students individually. The teacher/TA evaluates whether the students have the ability to design and implement programs (Experiment 1, Experiment 2, Experiment 6), system design and implementation capabilities (Experiment 2, Experiment 6) based on the quality of the code submitted by the students), system analysis and evaluation ability (experiment 3, experiment 4, experiment 5, experiment 6), expression and communication ability (experiment 2, experiment 6), modern tool use ability (all experiments).

How are students assessed?

Total score (100 points) = experiment (40%) + final exam (60%).
The question types of the final exam include:
Multiple-choice questions: to assess the understanding of basic concepts
Short answer and design questions: Give the requirements and the basic code of ADT. Carry out design and code: drawing/modeling, design, code modification, writing new code (not emphasizing syntax), writing notes (AF/RI/Spec/Testing) Strategy/Thread Safety Argument), design test cases, improve/optimize various quality indicators, etc.

Course tools and materials

[1] The Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3. IEEE Computer Society, 2014.

[2] S. McConnell. Code Encyclopedia (Second Edition), Publishing House of Electronics Industry, 2006.

[3] A. Oram, G. Wilson. The Beauty of Code, Mechanical Industry Press, 2009.

[4] J. Bloch. Effective Java Chinese Edition (2nd Edition), Machinery Industry Press, 2009.

[5] B. Meyer. Object-Oriented Software Construction (2nd edition), Prentice Hall, 1997.

Why do you teach the course this way?

This course is mainly in the form of classroom face-to-face, interspersed with small class discussions, and experiments. Classroom face-to-face teaching is mainly taught by teachers, as the most common form of teaching, can be more comprehensive and systematic transfer of the main knowledge points to everyone. Doing experiments can enhance the practical ability to operate, put the knowledge learned into use, not only on paper, general talk, the understanding of knowledge points more in place, more thorough.

Body of Knowledge coverage

KA	Sub-domain	Competencies Covered	Hours
SDM	Software design and development, Software testing	1. Software construction foundation 2. Software construction process 3. Reusability-oriented construction 4. Maintainability-oriented structure 5. Robust-oriented construction 6. Constructs for intelligibility 7. Performance-oriented construction technology 8. Parallel, distributed, GUI	64
PDA	Programming, Data structures	1. ADT+OOP 2. Refactoring	16