Introduction to Software Engineering

Min-Yuh Day, Ph.D, Associate Professor

Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

1102SE01
MBA, IM, NTPU (M5010) (Spring 2022)
Wed 2, 3, 4 (9:10-12:00) (B8F40)

https://meet.google.com/ish-gzmy-pmo
Min-Yuh Day, Ph.D.

Associate Professor, Information Management, NTPU
Visiting Scholar, IIS, Academia Sinica
Ph.D., Information Management, NTU
Director, Intelligent Financial Innovation Technology, IFIT Lab, IM, NTPU

Artificial Intelligence, Financial Technology, Big Data Analytics, Data Mining and Text Mining, Electronic Commerce
Course Syllabus
National Taipei University
Academic Year 110, 2\textsuperscript{nd} Semester (Spring 2022)

• Course Title: \textit{Software Engineering}
• Instructor: Min-Yuh Day
• Course Class: MBA, IM, NTPU (3 Credits, Elective)
• Details
  • In-Person and Distance Learning EMI Course
    (3 Credits, Elective, One Semester) (M5010)
• Time & Place: Wed, 2, 3, 4, (9:10-12:00) (B8F40)
• Google Meet: [https://meet.google.com/ish-gzmy-pmo](https://meet.google.com/ish-gzmy-pmo)
Course Objectives

1. Understand the fundamental concepts and research issues of software engineering.
2. Equip with Hands-on practices of software engineering.
3. Conduct information systems research in the context of software engineering.
Course Outline

• This course introduces the fundamental concepts, research issues, and hands-on practices of software engineering.

• Topics include:
  1. Introduction to Software Engineering
  2. Software Products and Project Management: Software product management and prototyping
  3. Agile Software Engineering: Agile methods, Scrum, and Extreme Programming
  4. Features, Scenarios, and Stories
  5. Software Architecture: Architectural design, System decomposition, and Distribution architecture
  6. Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service
  7. Cloud Computing and Cloud Software Architecture
  8. Microservices Architecture, RESTful services, Service deployment
  9. Security and Privacy; Reliable Programming
  10. Testing: Functional testing, Test automation, Test-driven development, and Code reviews
  11. DevOps and Code Management: Code management and DevOps automation
  12. Case Study on Software Engineering
Core Competence

• Exploring new knowledge in information technology, system development and application 80%

• Internet marketing planning ability 10%

• Thesis writing and independent research skills 10%
Four Fundamental Qualities

• Professionalism
  • Creative thinking and Problem-solving 30 %
  • Comprehensive Integration 30 %

• Interpersonal Relationship
  • Communication and Coordination 10 %
  • Teamwork 10 %

• Ethics
  • Honesty and Integrity 5 %
  • Self-Esteem and Self-reflection 5 %

• International Vision
  • Caring for Diversity 5 %
  • Interdisciplinary Vision 5 %
College Learning Goals

• Ethics/Corporate Social Responsibility
• Global Knowledge/Awareness
• Communication
• Analytical and Critical Thinking
Department Learning Goals

• Information Technologies and System Development Capabilities

• Internet Marketing Management Capabilities

• Research capabilities
# Syllabus

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2022/02/23</td>
<td>Introduction to Software Engineering</td>
</tr>
<tr>
<td>2</td>
<td>2022/03/02</td>
<td>Software Products and Project Management:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Software product management and prototyping</td>
</tr>
<tr>
<td>3</td>
<td>2022/03/09</td>
<td>Agile Software Engineering:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Agile methods, Scrum, and Extreme Programming</td>
</tr>
<tr>
<td>4</td>
<td>2022/03/16</td>
<td>Features, Scenarios, and Stories</td>
</tr>
<tr>
<td>5</td>
<td>2022/03/23</td>
<td>Case Study on Software Engineering I</td>
</tr>
<tr>
<td>6</td>
<td>2022/03/30</td>
<td>Software Architecture: Architectural design,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>System decomposition, and Distribution architecture</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>2022/04/06</td>
<td>Make-up holiday (No Classes)</td>
</tr>
<tr>
<td>8</td>
<td>2022/04/13</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>9</td>
<td>2022/04/20</td>
<td>Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service</td>
</tr>
<tr>
<td>10</td>
<td>2022/04/27</td>
<td>Cloud Computing and Cloud Software Architecture</td>
</tr>
<tr>
<td>11</td>
<td>2022/05/04</td>
<td>Microservices Architecture, RESTful services, Service deployment</td>
</tr>
<tr>
<td>12</td>
<td>2022/05/11</td>
<td>Industry Practices of Software Engineering</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>13</td>
<td>2022/05/18</td>
<td>Case Study on Software Engineering II</td>
</tr>
<tr>
<td>14</td>
<td>2022/05/25</td>
<td>Security and Privacy; Reliable Programming; Testing: Test-driven development, and Code reviews; DevOps and Code Management: DevOps automation</td>
</tr>
<tr>
<td>15</td>
<td>2022/06/01</td>
<td>Final Project Report I</td>
</tr>
<tr>
<td>16</td>
<td>2022/06/08</td>
<td>Final Project Report II</td>
</tr>
<tr>
<td>17</td>
<td>2022/06/15</td>
<td>Self-learning</td>
</tr>
<tr>
<td>18</td>
<td>2022/06/22</td>
<td>Self-learning</td>
</tr>
</tbody>
</table>
Teaching Methods and Activities

• Lecture
• Discussion
• Practicum
Evaluation Methods

• Individual Presentation 60%
• Group Presentation 10%
• Case Report 10%
• Class Participation 10%
• Assignment 10%
Required Texts

Reference Books


• Titus Winters, Tom Manshreck, and Hyrum Wright (2020), Software Engineering at Google: Lessons Learned from Programming Over Time, O'Reilly Media.

• Project Management Institute (2017), Agile Practice Guide, PMI

Ian Sommerville (2019),


Ian Sommerville (2015),
Software Engineering,

Source: https://www.amazon.com/Software-Engineering-10th-Ian-Sommerville/dp/0133943038
Titus Winters, Tom Manshreck, and Hyrum Wright (2020), Software Engineering at Google: Lessons Learned from Programming Over Time, O'Reilly Media.

Project Management Institute (2021),

A Guide to the
Project Management Body of Knowledge
(PMBOK Guide) –
Seventh Edition and The Standard for Project Management

Software Engineering
Software Engineering and Project Management

Analyze
Requirements definition

Design
System and Software design

Build
Implementation and unit testing

Test
Integration and system testing

Deliver
Operation and maintenance

Project Management
Information Management

Management

Information Systems (MIS)

Information Systems
Information Management (MIS)
Information Systems

Fundamental MIS Concepts

Project-based software engineering

Project-based software engineering

• The starting point for the software development is a set of ‘software requirements’ that are owned by an external client and which set out what they want a software system to do to support their business processes.

• The software is developed by a software company (the contractor) who design and implement a system that delivers functionality to meet the requirements.

• The customer may change the requirements at any time in response to business changes (they usually do). The contractor must change the software to reflect these requirements changes.

• Custom software usually has a long-lifetime (10 years or more) and it must be supported over that lifetime.

Product software engineering

1. Opportunity

- inspires
- realizes
- implemented-by

Product features

Software

DEVELOPER

The starting point for product development is a **business opportunity** that is identified by individuals or a company. They develop a software product to take advantage of this opportunity and sell this to customers.

- The company who identified the opportunity **design and implement a set of software features** that realize the opportunity and that will be useful to customers.

- The software development company are responsible for deciding on the development timescale, what features to include and when the product should change.

- Rapid delivery of software products is essential to capture the market for that type of product.

Software execution models

Stand-alone execution
- User’s computer
  - User interface
  - Product functionality
  - User data
  - Product updates
  - Vendor’s servers

Hybrid execution
- User’s computer
  - User interface
  - Partial functionality
  - User data
  - Additional functionality
  - User data backups
  - Product updates
  - Vendor’s servers

Software as a service
- User’s computer
  - User interface
    (browser or app)
  - Product functionality
  - User data
  - Vendor’s servers

Technical interactions of product managers

Product manager

- Product vision management
- Product backlog management
- Acceptance testing
- User stories and scenarios
- Customer testing
- User interface design

Software Development Life Cycle (SDLC)
The waterfall model

Requirements definition

System and Software design

Implementation and unit testing

Integration and system testing

Operation and maintenance

Plan-based and Agile development

Plan-based development

- Requirements engineering
- Requirements specification
- Design and implementation

Requirements change requests

Agile development

- Requirements engineering
- Design and implementation

The Continuum of Life Cycles

- **Incremental**: Low Degree of Change, High Frequency of Delivery
- **Agile**: High Degree of Change, High Frequency of Delivery
- **Predictive**: Low Degree of Change, Low Frequency of Delivery
- **Iterative**: High Degree of Change, Low Frequency of Delivery

Iterative Life Cycle

Analyze → Analyze Design → Build Test → Deliver

Prototype

Refine

A Life Cycle of Varying-Sized Increments

Iteration-Based and Flow-Based Agile Life Cycles

### Iteration-Based Agile

<table>
<thead>
<tr>
<th>Requirements Analysis</th>
<th>Design</th>
<th>Build</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
<td>Design</td>
<td>Build</td>
<td>Test</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>Design</td>
<td>Build</td>
<td>Test</td>
</tr>
<tr>
<td>Repeat as needed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>Design</td>
<td>Build</td>
<td>Test</td>
</tr>
</tbody>
</table>

### Flow-Based Agile

<table>
<thead>
<tr>
<th>Requirements Analysis</th>
<th>Design</th>
<th>Build</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
<td>Design</td>
<td>Build</td>
<td>Test</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>Design</td>
<td>Build</td>
<td>Test</td>
</tr>
<tr>
<td>Repeat as needed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>Design</td>
<td>Build</td>
<td>Test</td>
</tr>
</tbody>
</table>

From personas to features

1. **Personas**
   - A way of representing users

2. **Scenarios**
   - Natural language descriptions of a user interacting with a software product

3. **Stories**
   - Natural language descriptions of something that is needed or wanted by users

4. **Features**
   - Fragments of product functionality

Multi-tier client-server architecture

Service-oriented Architecture

Everything as a service

- **Software as a service (SaaS)**
- **Platform as a service (PaaS)**
- **Infrastructure as a service (IaaS)**

- Photo editing
- Cloud management
- Monitoring
- Storage
- Network

- Logistics management
- Database software development
- Computing virtualization

Software as a service

Software customers

Software provider

Cloud provider

Microservices architecture – key design questions

- What are the microservices that make up the system?
- How should data be distributed and shared?
- How should the microservices in the system be coordinated?
- How should microservices communicate with each other?
- How should service failure be detected, reported and managed?
Types of security threat

**Availability threats**
- An attacker attempts to deny access to the system for legitimate users
- Distributed denial of service (DDoS) attack

**Integrity threats**
- An attacker attempts to damage the system or its data

**Confidentiality threats**
- An attacker tries to gain access to private information held by the system
- Data theft

**SOFTWARE PRODUCT**
- PROGRAM
- DATA

**Virus**
- Ransomware

Software product quality attributes

1. Reliability
2. Availability
3. Resilience
4. Maintainability
5. Responsiveness
6. Usability
7. Security

A refactoring process

1. Identify code ‘smell’
2. Identify refactoring strategy
3. Make small improvement until strategy completed
4. Run automated code tests

Functional testing

1. Unit Testing
2. Feature Testing
3. System Testing
4. Release Testing

Start

**Test-driven development (TDD)**

1. Identify new functionality
2. Identify partial implementation of functionality
3. Write code stub that will fail test
4. Run all automated test
5. Implement code that should cause failing test to pass
6. Run all automated test
7. Refactor code if required

Functionality incomplete → All tests pass → Functionality complete

DevOps

Development
Deployment
Support

Multi-skilled DevOps team

Code management and DevOps

DevOps automation

- Continuous integration
- Continuous deployment
- Continuous delivery
- Infrastructure as code

Code management system

- Branching and merging
- Code repository
  - Recover version information
  - Save and retrieve versions
- Transfer code to/from developer’s filestore

DevOps measurement

- Data collection
- Data analysis
- Report generation

Marketing
Marketing

“Meeting needs profitably”

“Marketing is an organizational function and a set of processes for creating, communicating, and delivering value to customers and for managing customer relationships in ways that benefit the organization and its stakeholders.”

Marketing Management
Marketing Management

“Marketing management is the art and science of choosing target markets and getting, keeping, and growing customers through creating, delivering, and communicating superior customer value.”

Marketing Management

1. Understanding Marketing Management
2. Capturing Marketing Insights
3. Connecting with Customers
4. Building Strong Brands
5. Creating Value
6. Delivering Value
7. Communicating Value
8. Conducting Marketing Responsibly for Long-term Success

Software Engineering
and
Project Management

Project Management

Analyze
- Requirements definition

Design
- System and Software design

Build
- Implementation and unit testing

Test
- Integration and system testing

Deliver
- Operation and maintenance
Summary

• This course introduces the fundamental concepts, research issues, and hands-on practices of software engineering.

• Topics include:

  1. Introduction to Software Engineering
  2. Software Products and Project Management: Software product management and prototyping
  3. Agile Software Engineering: Agile methods, Scrum, and Extreme Programming
  4. Features, Scenarios, and Stories
  5. Software Architecture: Architectural design, System decomposition, and Distribution architecture
  6. Cloud-Based Software: Virtualization and containers, Everything as a service, Software as a service
  7. Cloud Computing and Cloud Software Architecture
  8. Microservices Architecture, RESTful services, Service deployment
  9. Security and Privacy; Reliable Programming
 10. Testing: Functional testing, Test automation, Test-driven development, and Code reviews
 11. DevOps and Code Management: Code management and DevOps automation
 12. Case Study on Software Engineering
Software Engineering

Contact Information

Min-Yuh Day, Ph.D.
Associate Professor

Institute of Information Management, National Taipei University

Tel: 02-86741111 ext. 66873
Office: B8F12
Address: 151, University Rd., San Shia District, New Taipei City, 23741 Taiwan
Email: myday@gm.ntpu.edu.tw
Web: http://web.ntpu.edu.tw/~myday/