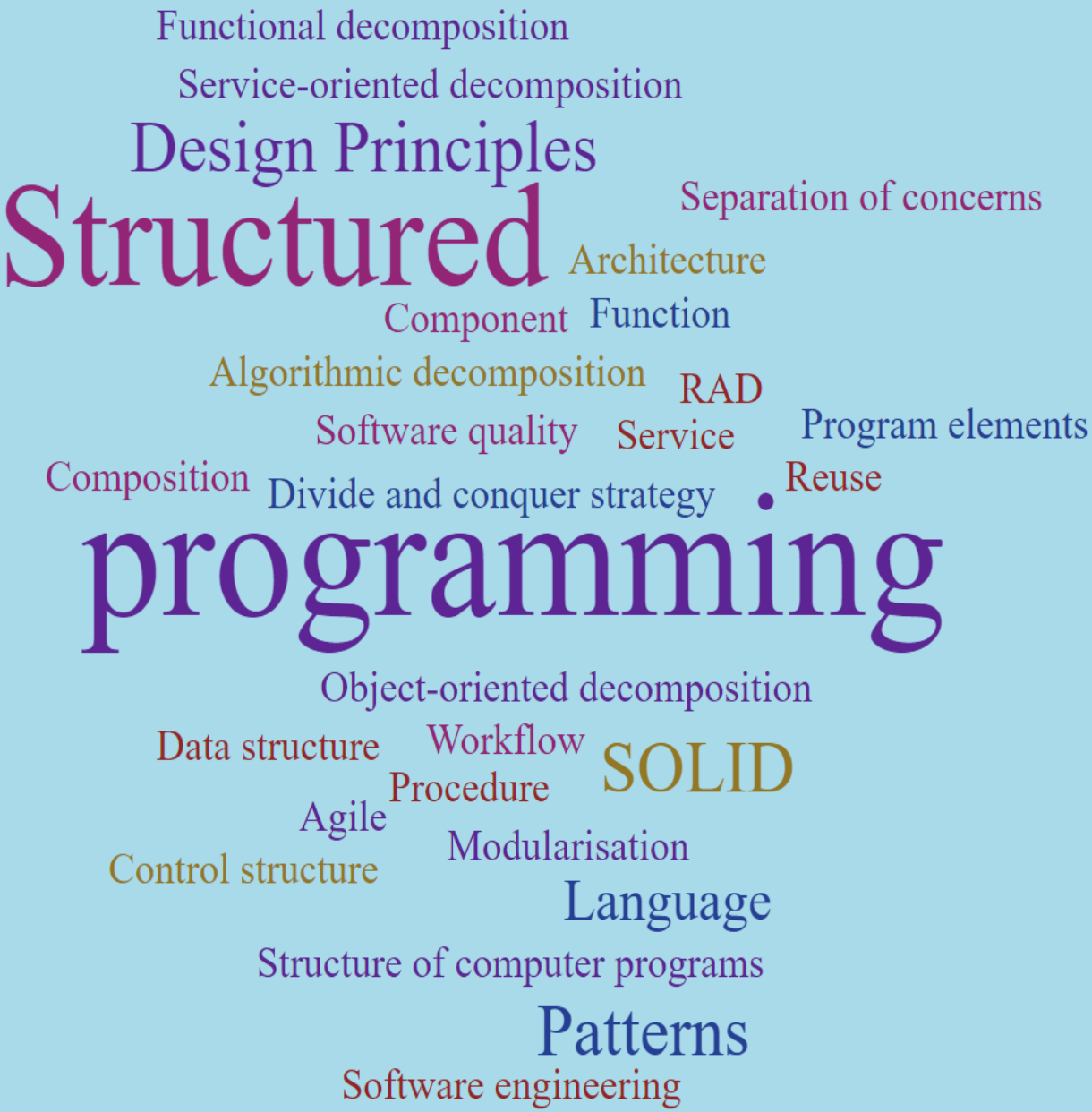

Functional decomposition
Service-oriented decomposition
Design Principles
Structured
Separation of concerns
Architecture
Component
Function
Algorithmic decomposition
RAD
Software quality
Service
Program elements
Composition
Divide and conquer strategy
Reuse
programming
Object-oriented decomposition
Data structure
Workflow
SOLID
Procedure
Agile
Modularisation
Control structure
Language
Structure of computer programs
Patterns
Software engineering

IT4492/IT4492E
Structured Programming

Lecturer:	**VU Thi Huong Giang**
Office:	R.601 B1 building 1A Dai Co Viet street 10000 Hanoi
Phone:	(84 4) 3 8 68 25 95
Email:	giangvth@soict.hust.edu.vn vthgiang@gmail.com
Web site:	http://www.soict.hust.edu.vn/~giangvh
Reception hours:	only by prior arrangement

Information

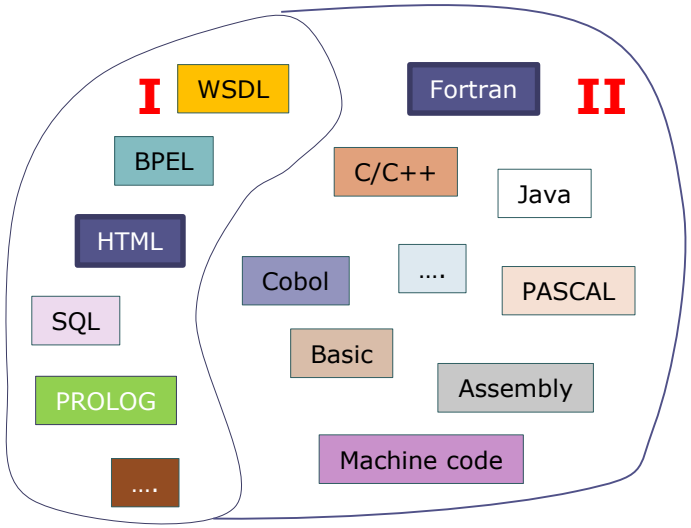
- Class: ICT – K59
  - Location: R.406, D9 building
  - Schedule: Thu. 14h15 - 15h50
    (45 minutes x 2 x 15 weeks)
- Class: AS – K59
  - Location: R.406, D9 building
  - Schedule: Wed. 16h00 – 17h35
    45 minutes x 2 x 15 weeks

Introduction

- Given a problem, how to:
  - Design an algorithm for solving it
  - Implement this algorithm as a computer program

→ Needs of programming languages and paradigms

- Language: express the algorithm to a machine
  - Declarative language (**I**): what to do, what to store
  - Non declarative language (**II**): how to do, how to store

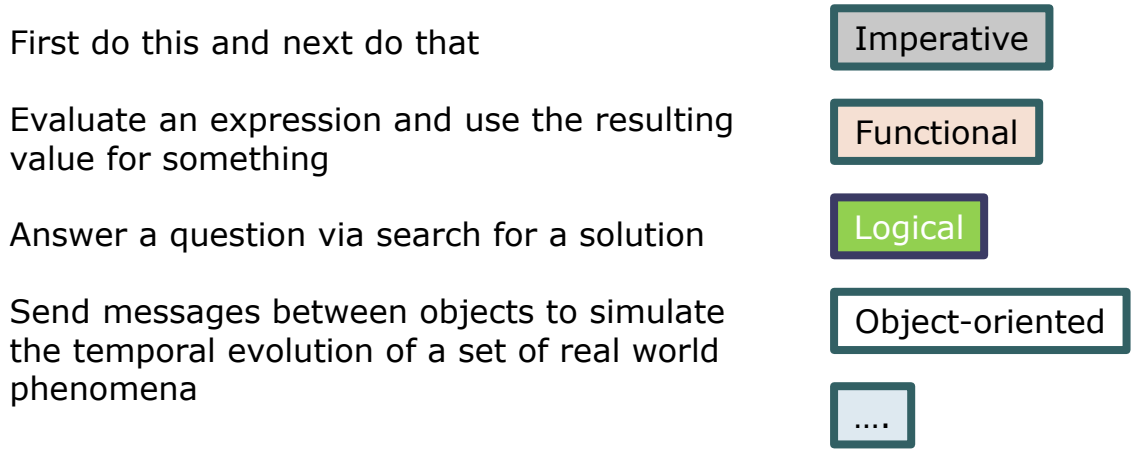


Introduction

- Given a problem, how to:
  - Design an algorithm for solving it
  - Implement this algorithm as a computer program

→ Needs of programming languages and paradigms

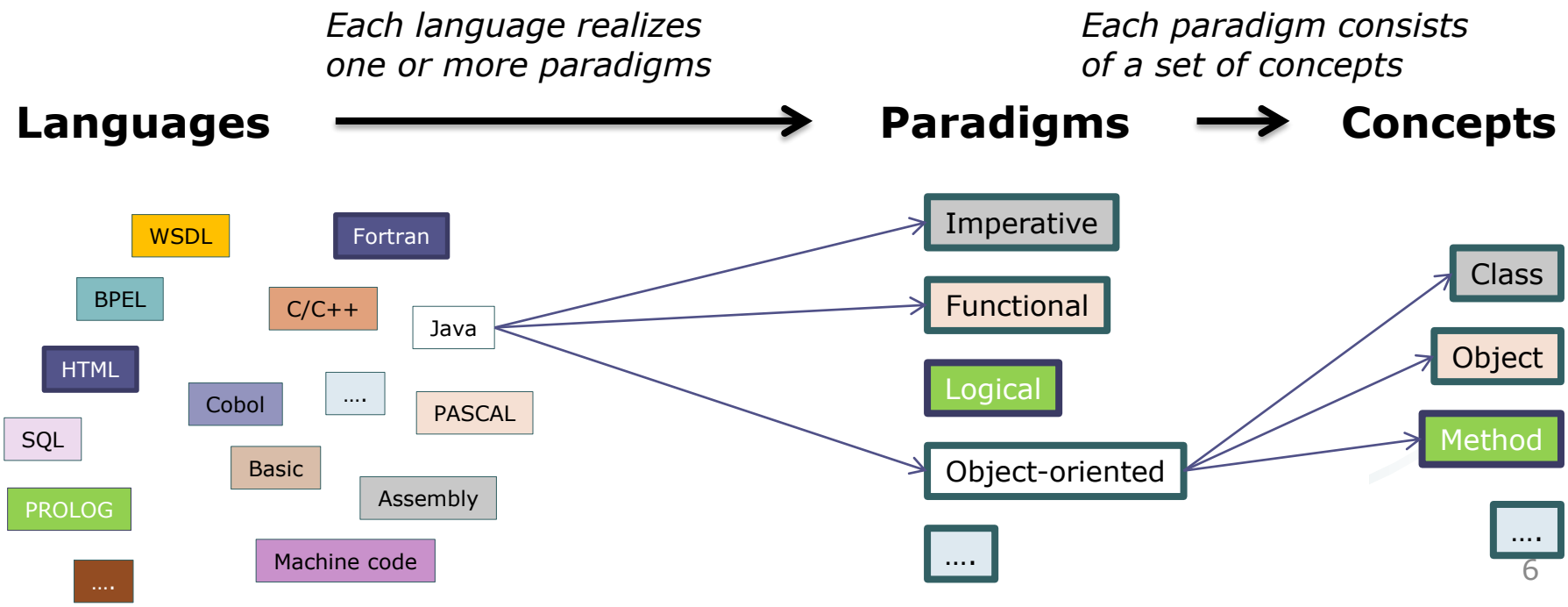
- Paradigm: comprise a set of concepts that are used as patterns for programming



Introduction

- Given a problem, how to:
  - Design an algorithm for solving it
  - Implement this algorithm as a computer program

→ Needs of programming languages and paradigms



Introduction

How do you organize your code ???

- From **bad programming habits** …
  - Foggy idea about what is to be done
  - Write program with no planning: start from the beginning and write to the end
  - No systematic debugging: considered it finished if it works on only one test case

- … to **undesirable results**
  - Do not know how to program
  - Any change of requirement invites rewriting of the entire program again
  - Program with numerous bugs that take extremely long time to debug, or even failure to complete

Introduction

How do you organize your code ???

- Structured programming
  - Becomes popular since the 70's
  - Should have been learnt by students that have taken any programming course
  - Absolutely essential for handling software projects

Course Description

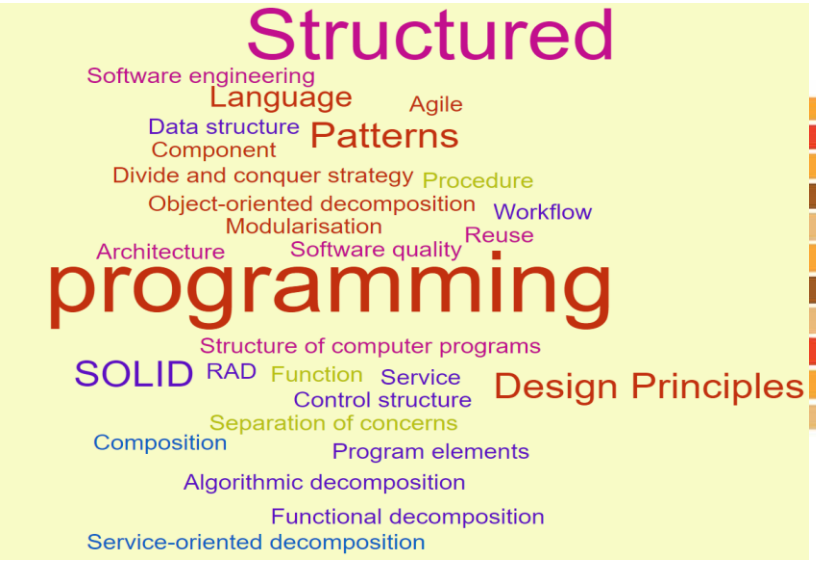
- Structured programming in software projects
- Principles and patterns central to successful software development
- Modern development methods
  - **rapid software development**
  - **software reuse**
  - software as a service
- Well-structured software practices

Objectives

Upon completion of this course, students will be able to:

- Knowledge:
  - Identify the needs for structured programming in software projects
  - Distinguish the program's low level structures that can be implemented by various programming languages
  - Summarize general design principles in software engineering and modern development methods
  - Determine good experiences as software patterns
- Skills:
  - Visualize the program structure in software design and development
  - Recognize, produce and maintain well-structured software
- Attitude:
  - Revise the program structure in software design and development
  - Show off a well-structured programming style

Schedule


Structured
Software engineering
Language
Agile
Data structure
Patterns
Component
Divide and conquer strategy
Procedure
Object-oriented decomposition
Workflow
Modularisation
Reuse
Architecture
Software quality
programming
Structure of computer programs
SOLID
RAD
Function
Service
Design Principles
Control structure
Separation of concerns
Composition
Program elements
Algorithmic decomposition
Functional decomposition
Service-oriented decomposition


1
2
3
...
1
2
3
...
Software
Software

Schedule

Week	Topic	Reading	Report / Assessment
	Software Engineering Revisited	SE1: Chapter 1 : pp.4-19 FSE: Chapter 1-3 : pp.1-66	
	Structured Programming in Software Project	SE2: Chapter 6: pp.254-277	
	Software Design and Architecture	FSE: Chapter 4: pp.67-160 FSE: Chapter 4: pp.146-152 SE1: Chapter 11: pp.241-265	
	Student Presentation		**X**
	Rapid Software Development	SE1: Chapter 17 : pp.391-414	
	Design Principle	ASD: Chapter 10-12: pp.111-145	
	Student Workings		**X**

Schedule

Week	Topic	Reading	Report / Assessment
	Software as a Service /Web service	SE1: Chapter 31: pp. 743-769	X
	Software patterns	ASD: Chapter 13-14: pp.151-172, Chapter 16-17, pp.177-192, Chapter 23-26: pp.293-354, Chapter 28-29: pp.387-441	X
	Student working		**X**
	Wrap up and Advanced topics		
	Student's final presentation		**x**

Method of Evaluation

- Mid-term evaluation (30%):
  - Attendance
  - Continuous assessment:
    - Assigned readings
    - Homework
    - In-class case study
    - Q&A
    - …
  - Project:
    - implement + test + enhance
    - demonstration + report + defense + review
- Final exam (70%):
  - Writing exam

The instructor reserves the right to modify course policies, course calendar, course content, assignment values and due dates, as circumstances require.

How to learn ?

- Self-studying at home
  - Do your homework (no late assignments will be accepted)
  - Read the lecture notes and answer the directive questions (if any) before attending class
  - Use Moodle:

  http://moodle.hust.edu.vn/course/view.php?id=37

- Active learning in class
  - Summarize the lecture content
  - Give answers for assignments