Introduction to C Programming

Lecture 15: Summary

Wenjin Wang
wangwj3@sustech.edu.cn

12-30-2022

Course syllabus

Nr.	Lecture	Date
1	Introduction	2022.9.9
2	Basics	2022.9.16
3	Decision and looping	2022.9.23
4	Array & string	2022.9.30
5	Functions	2022.10.9 (补)
6	Pointer	2022.10.14
7	Self-defined types	2022.10.21
8	I/O	2022.10.28

Nr.	Lecture	Date
9	Head files	2022.11.4
10	Review of lectures I	2022.11.25
11	Review of lectures II	2022.12.2
12	Review of lectures III	2022.12.9
13	AI in C programming	2022.12.16
14	AI in C programming	2022.12.23
15	Summary	2022.12.30

Objective of this lecture

Summarize and prepare for exam

Content

1. Summarize and question review
2. Software engineering
3. Bye

Content

1. **Summarize and question review**
2. **Software engineering**
3. **Bye**

All materials

Introduction

↓

Basics

↓

Decision and looping

↓

Array & string

↓

Functions

↓

Pointer

↓

Self-defined types

↓

I/O

↓

Head files

Chapter 1: Introduction

Introduction

↓

Basics

↓

Decision and looping

↓

Array & string

↓

Functions

↓

Pointer

↓

Self-defined types

↓

I/O

↓

Head files

- Machine and machine intelligence are everywhere. The way to control machine is by **programming**.
- **C + AI (or domain knowledge)** makes you different.
- Programming language allows communication between human and machine. C language is **easy-to-use** for users while still **efficient** for machines.
- C is a high-level language that is **popular and ubiquitous in industry**, especially for edge devices.
- Good ways to learn C is **practicing with projects**, understand the essence instead of memorizing it (Google is your friend).

Chapter 2: Basics

Introduction

↓

Basics

↓

Decision and looping

↓

Array & string

↓

Functions

↓

Pointer

↓

Self-defined types

↓

I/O

↓

Head files

- Machine uses **bit** (0 and 1) and **byte** (group of bits) to store information
- Different **data types** (int, float, double, char) can be used for declaring and initializing variables based on what you need
- Variables can be used for operations/calculations using **pre-defined C operators**
- Five basic operations provided by C: **arithmetic, relational, logical, assignment, Misc**
- Users can interact with the machine using **I/O functions**

Chapter 3: Decision & looping

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- Two major workflow controls provided in C: **decision-making** and **looping**
- Two types of statement for making decisions: **if-else** and **switch**, if-else is more popular, switch is for equality check (sub-case of if-else)
- Two types of statement for looping: **for** loop and **while/do-while** loop, both are essentially the same
- Do-while loop guarantees at least one-time execution of loop
- **Break** and **continue** statements can influence loops, jump out from the loop (break) or skip one iteration (continue)

Chapter 4: Array & string

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- We can use **array** to hold many data for group processing
- Array has the **fixed size** and can only be used to hold data with **same type**
- **Different types of array** can be created, e.g. int array, float array, char array (string)
- **Different dimensional arraies** can be created, from 1D array to ND array
- Array enables the processing of vectors, matrices, strings, etc.
- C provided self-built functions to process strings

Chapter 5: Functions

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- Three steps to create a function: **function declaration, definition, calling**. Function must be declared in front of the place where it is called (e.g. before the main)
- Variable has its scope both in space and time. **Global variable** (outside function) is visible everywhere, **local variable** (inside function) is only visible in the function block. Variable can have **identifiers** (auto, static, extern, register), such as local static variable.
- **Recursion** can be implemented by calling a function itself repeatedly.

Chapter 6: Pointer

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- **Pointer** is a variable that stores the address of another variable.
- We can access the **memory address** directly using the pointer.
- By changing the pointer value, the value stored at the address will be modified, typically useful for **functions** to pass values.
- Pointer can point to arrays, using **arithmetic and logical operations** (++, --, ==, >, <) to scan the memory address.
- We can assign dynamic arrays and control memories using C provided functions in **stdlib.h,** e.g. calloc(), malloc(), relloc(), free().

Chapter 7: self-define types

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- **Struct** can be used to define a new data type for grouping data with different types.
- Struct is very useful and has been commonly used. It can be used with **arrays, pointers, and functions**.
- **Union** is not useful (only need know). **Enum** can be used to assign a sequence of names with integers.
- **Typedef** can define a new type of data by combining existing ones (short int, struct), **#define** can define marcos for pre-processing (values, expressions).

Chapter 8: I/O

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- Three types of I/O provided by C: **console I/O, file I/O, socket I/O**
- **Console I/O**: getchar/putchar, gets/puts, printf/scanf
- **File I/O**: three types of file (txt, csv, bin), bin file can hold C-defined data structures (e.g. struct)
- **Socket I/O:** an established mechanism between client and server, three things are essential for setting up a socket connection (transmission protocol, IP address, port number)
- Two commonly used transmission protocols for socket: **TCP and UDP**

Chapter 9: head files

Introduction
↓
Basics
↓
Decision and looping
↓
Array & string
↓
Functions
↓
Pointer
↓
Self-defined types
↓
I/O
↓
Head files

- **Head files (.h)** can be used for function declaration and macro definition
- Put function declarations in the head file (**.h**), put function definitions in the c file (**.c**)
- Use **#include<XXX.h>** to include C compiled head files, use **#include"XXX.h"** to include self-defined head files (in the same path)
- You can write your own libraries or download third-party libraries (e.g. OpenCV) and use the library functions by including their head files.

Questions

Some questions for practice!

Understand, not memorize!

Questions

1. C program have an entrance function, what is the name of this function?

2. The main function of C must return an integer value to finish the program. Yes/No

3. You must include a head file to use the **printf** and **scanf**, what is the name of this head file?

4. How a C program is executed? ( )
   A. start from the main function and end when main is returned
   B. start from the first function and end at the last function
   C. start from the first function and end at the first function
   D. start from the main function and end at the last function

5. All pre-processor directives in C language begin with the # symbol. Yes/No

Questions

6. One byte has how many bits? ()

A. 2 B. 8 C. 10 D. 16

7. What is the output of below printf? ()

```
int main()
{
    int a[2][3] = { {1,2,3},{4,5,6} };
    printf("%d",*(*a + 1));
}
```

A. 1 B. 2 C. 4 D. 5

Questions

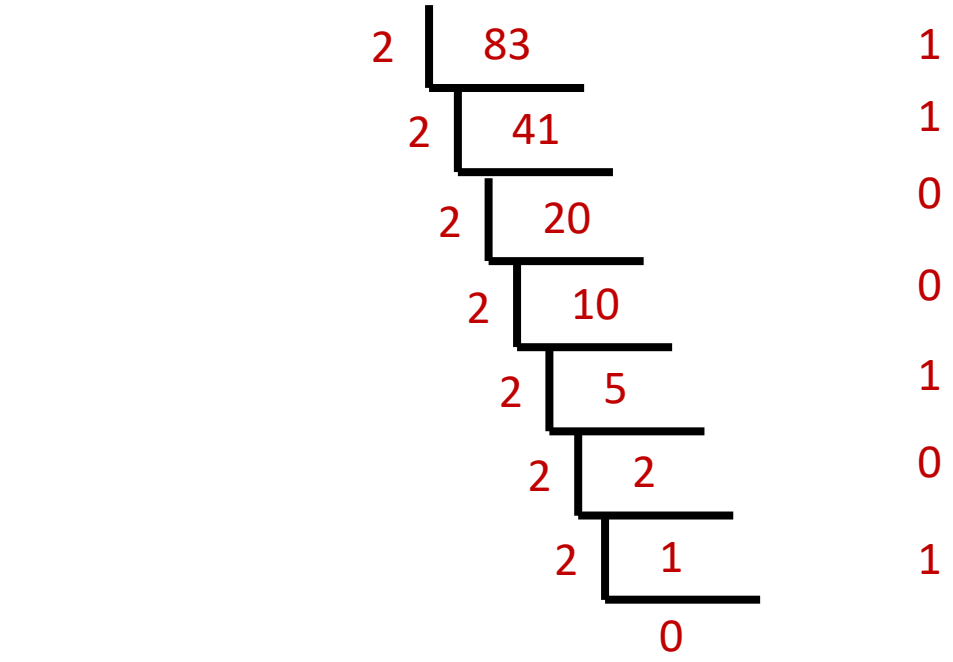
8. In below operators, which one requires the operated variables to be integer? ()
A. / B. ++ C. *= D. %

9. Please convert the 1001011 from binary to decimal, and convert 83 from decimal to binary, and writing down the procedure of conversion!

10. In below operators, which one can get the address of a variable? ()
A. & B. && C. ! D. *

1 0 0 1 0 1 1

$2^6 + 2^3 + 2^1 + 2^0 = 75$



Questions

11. What is the difference between **while** and **do-while**?

12. What is the output of below program? ( )

```
int main() {
	char s1[40] = "country", s2[20] = "side";
	int i = 0, j = 0;
	while (s1[i] != '\0')  i++;
	while (s2[j] != '\0')  s1[i++] = s2[j++];
	s1[i] = 0;
	printf("%s\n", s1);
}
```

A. side　　B. country　　C. sidetry　　D. countryside

Questions

13. **break** statement can be used to skip one iteration in for loop. Yes/No

14. What is the output of below program?

```
int main() {
	for(int k = 0; k < 5; k++)
	{
		if (k == 3) continue;
		printf("%d ", k);
	}
}
```

15. How to use while loop in C? ( )

A. while x < y
B. while (x < y)
C. if x > y while
D. while x < y then

Questions

16. If we use the array name as the argument for the function, what does the argument stand for? ( )

A. The value of the first element in the array
B. The value of all elements in the array
C. The address of the first element in the array
D. The address of all elements in the array

17. How to declare a 1D array? ( )

A. int a(10); B. int a{10}; C. int [10]a; D. int a[10];

18. How to declare a 2D array? ( )

A. int a[3][]; B. float a(3,4); C. double a[3][4]; D. float a(3)(4);

Questions

19. Which of following is the correct for declaring a struct named “Point”? ( )

A. struct {int x; int y; }Point; Point p;

B. struct {int x; int y; }Point; struct Point p;

C. typedef struct {int x; int y; }Point; Point p;

D. typedef struct {int x; int y; }Point; struct Point p;

20. Which statement is correct in checking if string s1 equals to string s2? ( )

A. if(s1 == s2) B. if(s1 = s2) C. if(strcpy(s1,s2)) D. if(strcmp(s1,s2)==0)

Questions

21. You can only output the results from the function by returning a value. Yes | No

22. When input is “1024”, which is output of following function? ( )
A. 4201 B. 7 C. 1024 D. 10

```
int DigitSum(int n)
{
    if(n == 0)
    {
        return 0;
    }
    return n % 10 + DigitSum(n/10);
}
```

23. What are the differences between (i) local variable and global variable, (ii) parameters and arguments?

Questions

24. Which of following statement is the correct for a pointer? ( )

A. float a = 3.14; float *p = &a;

B. char a = ‘a’; char *p = a;

C. int *p = 10;

D. All above are correct

25. Write a function to recursively sum from 1 to 100, in step of 2 (e.g. 1 + 3 + 5 + … 99)

Questions

26. What is the difference between the variable and pointer variable?

27. Which of following is the correct statement for a pointer ( )
A. int a = 5; int *p = &a;
B. char a = ‘a’; char *p = a;
C. int *p = 5;
D. Above are correct

28. Given int a[] = {1,2,3,4,5,6}, assume int *ptr = &a[2]; which of following is true( )
A. *(ptr+2) is 3
B. *(ptr+2) is 4
C. *(ptr+2) is 5
D. *(ptr+2) is 6

Questions

29. K in KNN (K Nearest Neighbor) is the same as the K in Kmeans. Yes | No

30. Assume A = 2, B = 10, write a function to swap the value between A and B (B = 2, A = 10)?

31. Which of following keyword can define a struct? ( )

A. union B. enum C. struct D. typedef

32. Which of following statement is correct to access the member of struct on the right? ( )

A. (*p).data.a B. (*p).a C. *p->a D. p.data

Questions

33. Which of following statement for marco definition? ( )

A. #define MIN(a, b) (a < b ? a : b)
B. #define POW(x) (x * x)
C. #define CHAR "SUSTech"
D. All above are correct

34. What is the value of fri?( )

A. 4 B.5 C. 9 D.10

```
enum week
{
    mon = 5, tue, wed, thr, fri, sat, sun
};
```

Questions

35. Which of following statement is correct for opening a binary file? ( )

A. FILE *f = fwrite("test.bin", "b");
B. FILE *f = fopenb("test.bin", "w");
C. FILE *f = fopen("test.bin", "wb");
D. FILE *f = fwriteb("test.bin");

36. What is the function of following code? ( )

A. Copy file
B. Calculate the number of characters
C. Calculate the number of words
D. Calculate the number of rows

```c
int main() {

    FILE* fp = fopen("fname.dat", "r"));

    int num = 0;
    while (fgetc(fp) != EOF)
    {
       num++;
    }
    printf("num=%d\n", num);

    fclose(fp);

    return 0;
}
```

Questions

37. Which of following file format can be used to store a struct? ( )
A. dat B. txt C. bin D. csv

38. What does a client need to build a socket connection with server? ( )
A. Server's IP address and port number
B. Server's physical location
C. Server's IP address only, port number will be assigned automatically
D. Server's IP address and the format of transmitted data

39. TCP is a data secured transmission protocol, while UDP is more focused on the transmission speed and low cost. Yes | No

Questions

40. Head file can be used to declare functions. Yes | No

41. Which keyword can be used to include a head file? ( )
A. #include B. #define C. typedef D. extern

42. What is the difference between #include “XXX.h” and #include <XXX.h>?

43. A head file cannot include another head file. Yes | No

44. The name of head file and the file to define the functions in the head file must be the same, for example, myfun.h must have myfun.c for its implementation. Yes | No

45. Briefly describe the procedure of Kmeans or KNN.

Content

1. Summarize and question review
2. **Software engineering**
3. Bye

From C to C++/Java

- Since C was so popular, many programming languages were designed based on C
- Two Object-Oriented languages derived from C: C++ and Java
- C++ in 1979 (C with Classes)/1983 (C++)
- Java in 1995

From C to C++/Java

- C is **Procedural Oriented** programming, C++ and Java are **Object Oriented** programming.

- C++ is derived from C and has the features of POP and OOP
- C++ executes the code by using only a compiler. C++ compiler converts the source code into the machine code.
- **C++ is faster but platform-dependent** (compiled on windows does not fit Linux).
- C++ is designed for application and system development (game, medical devices, etc.).

- Java is purely OOP
- At compile time, Java source code is converted to bytecode (.class), at runtime, JVM load .class and converts to machine code with an interpreter.
- **Java is platform-independent but slower**
- Java is popular in IT industry, typically for network applications and web services, Apps.

POP (面向过程) vs OOP (面向对象)

POP (面向过程)

以**足球**为例：

- 门将传给后卫
- 后卫传给边卫
- 边卫传给中场
- 中场带球
- 中场传给前锋
- 前锋突破
- 前锋射门，没进
- 门将传给给后卫
- …

OOP (面向对象)

以**足球**为例：

- 创建2个对象：中国队，阿根廷队

- 中国：门将开球，后卫带球
- 阿根廷：前锋上抢，断球
- 中国：后卫反抢，没抢到
- 阿根廷：前锋带球进禁区，射门
- 中国：大脚解围，球进了！
- 中国：暂时1：0落后
- …

POP (面向过程) vs OOP (面向对象)

POP (面向过程)

以**阳性过程**为例：

- 感觉身体发凉，难受
- 去系办测体温，低烧
- 回宿舍休息，继续难受
- 去系办拿抗原测试，阳性！
- 回宿舍睡觉，硬扛
- 扛不住了，去系办拿退烧药
- 回宿舍吃药，感觉不错
- 退烧了，不难受了，继续造
- …

OOP (面向对象)

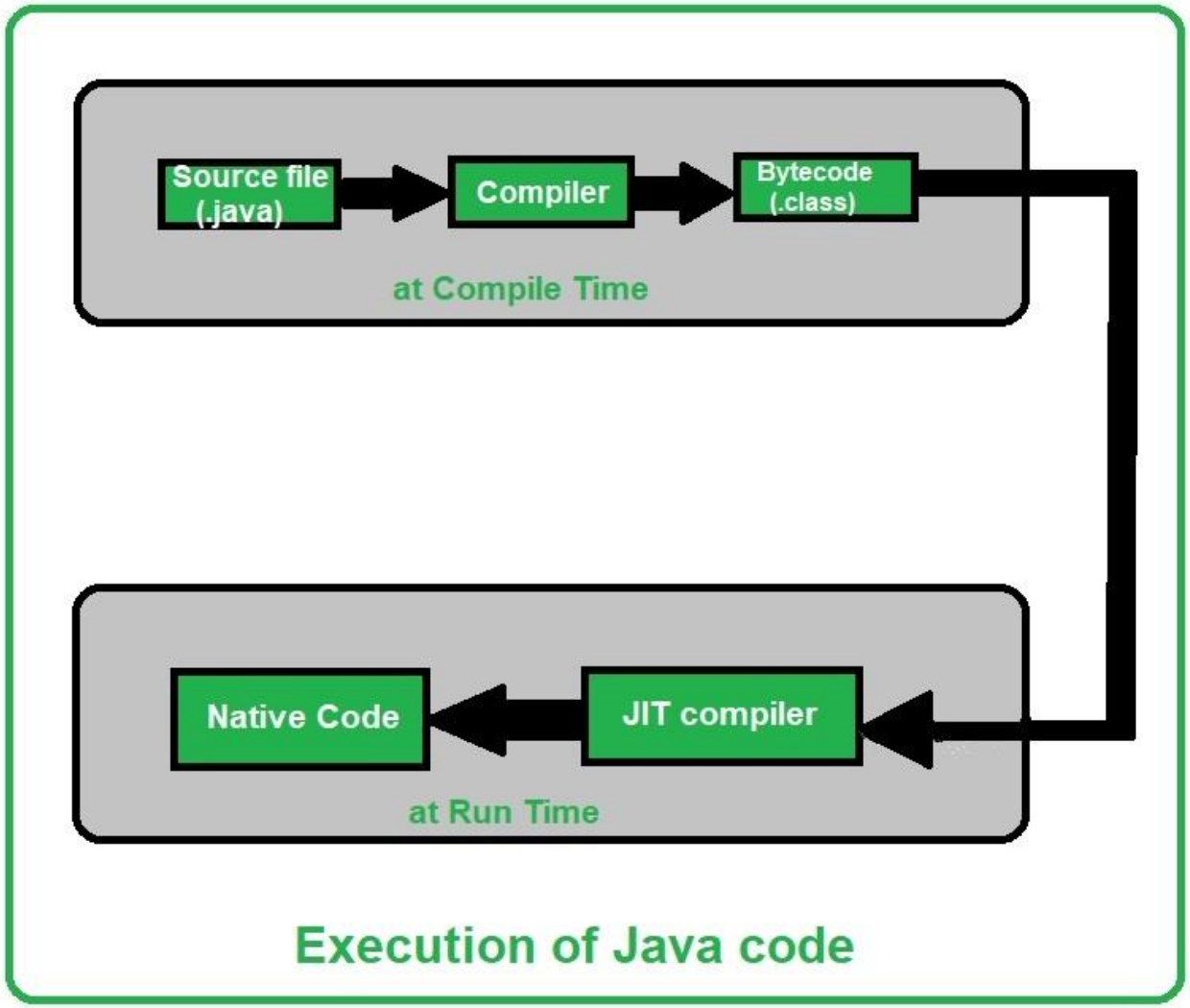
以**阳性过程**为例：

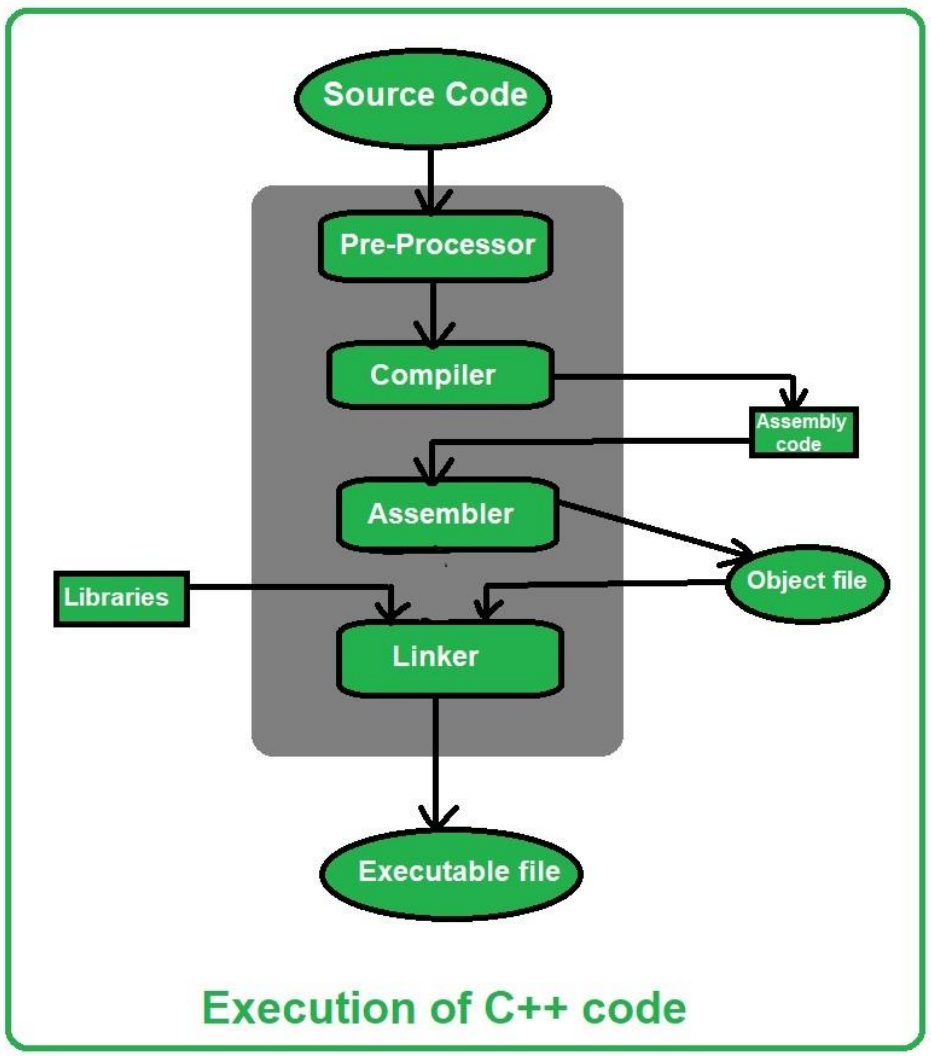
- 创建2个对象：学生，系办

- 学生A：阳了 - 休息 - 睡觉 - 康复
- 学生B：阳了 - 休息 - 睡觉 - 康复
- 学生C：阳了 - 休息 - 睡觉 - 康复
- …

- 系办：给温度计 - 给抗原 - 给退烧药

C++ and Java


Source file
(.java)
Compiler
Bytecode
(.class)
at Compile Time
Native Code
JIT compiler
at Run Time
Execution of Java code


Source Code
Pre-Processor
Compiler
Assembly
code
Assembler
Object file
Libraries
Linker
Executable file
Execution of C++ code

Comparison in Hello World

C

```
#include <stdio.h>
main(){
   printf(“Hello World!”);
}
```

C++

```
#include <iostream>
main(){
   std::cout << “Hello World!” << std::endl;
}
```

Java

```
class HelloWorld {
   static public void main() {
       System.out.println(“Hello World!”);
   }
}
```

Comparison in printing float

C

```
#include <stdio.h>
main(){
   float a = 1.5;
   printf(“a = %d\n”, a);
}
```

C++

```
#include <iostream>
main(){
   float a = 1.5;
   std::cout << “a = ” << a << std::endl;
}
```

Java

```
class HelloWorld {
   static public void main( {
      float a = 1.5;
      System.out.println(“a = ” + a);
   }
}
```

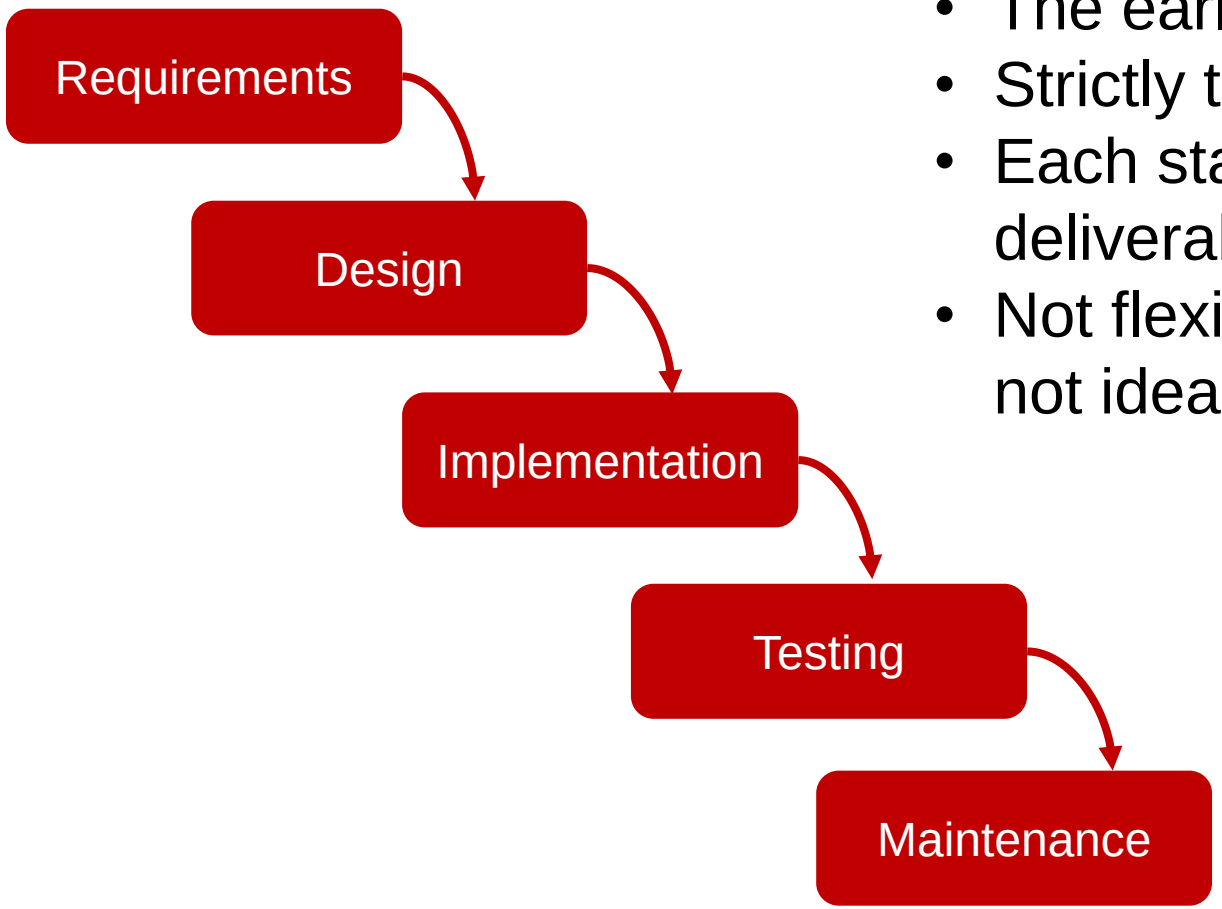
Software engineering models

- Provides a **framework** to build the solidity of the process engineering a product
- Provides a **mechanism** to clarify, create, track and upgrade the product throughout the life cycle
- Provides **guidelines** for project/product management
- Provides a **protocol** for team members to follow, increase compliance, avoid bureaucracy

Software engineering models

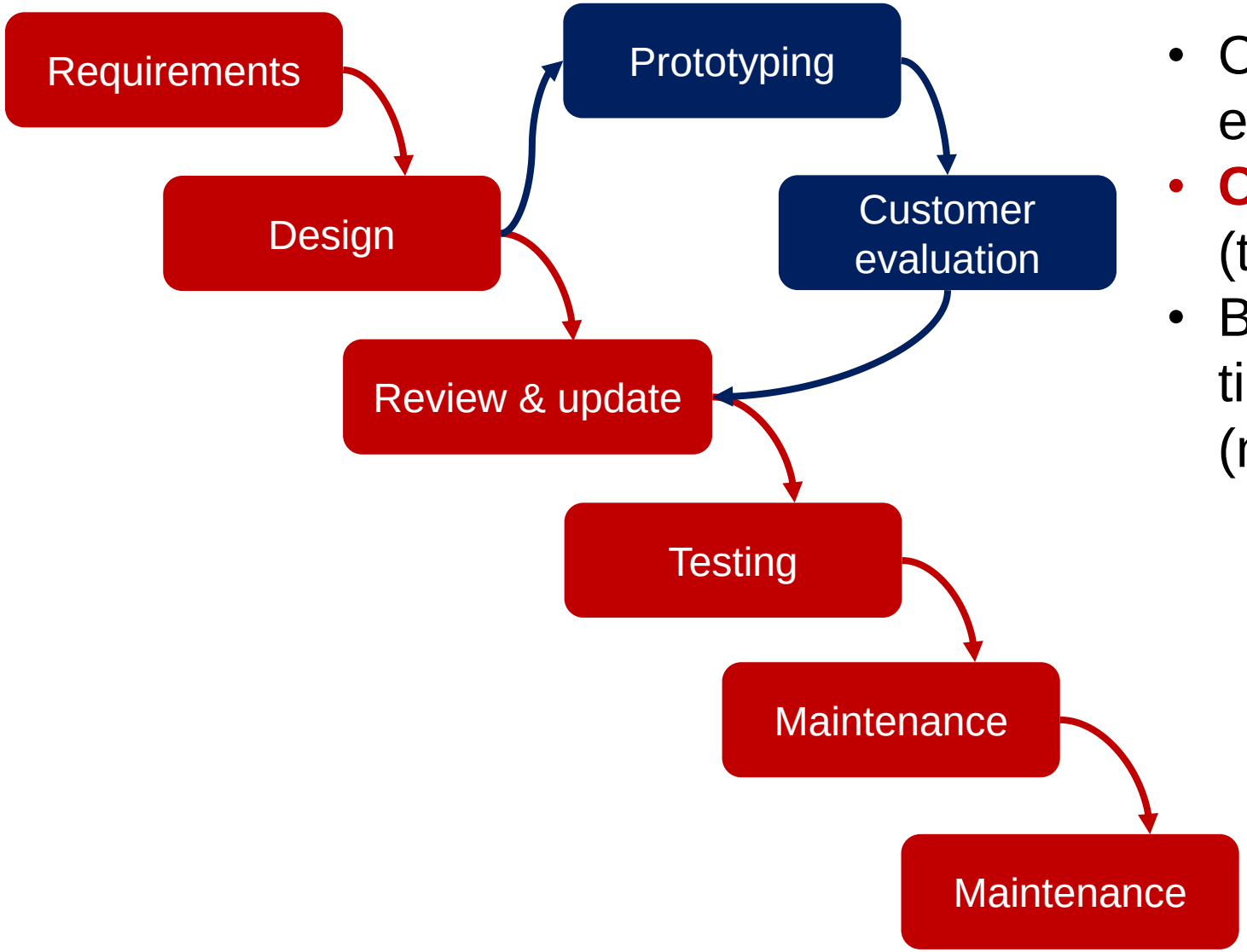
- Waterfall model
- Prototyping model
- V model
- Iterative model
- Scrum model
- Kanban model

Waterfall model



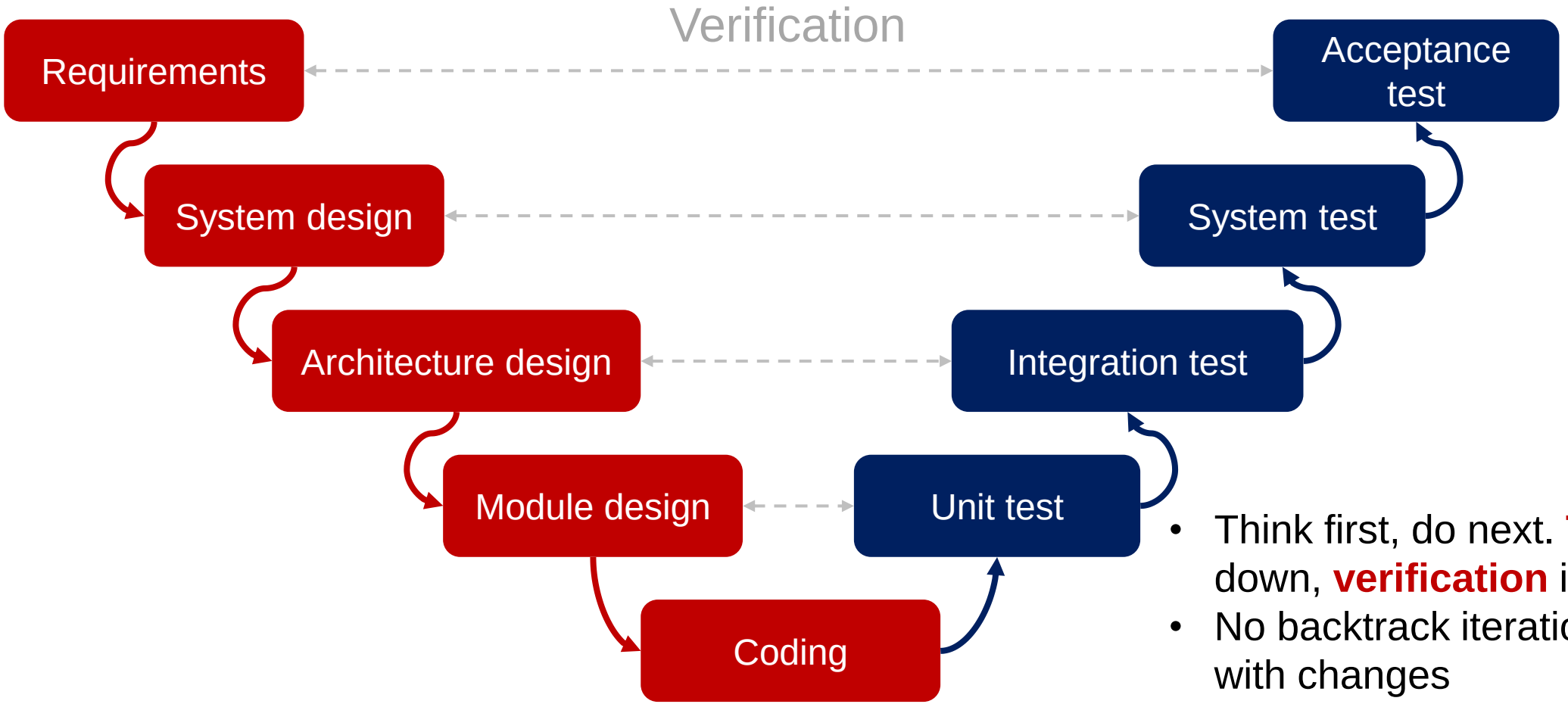
- The earliest model for SW development
- Strictly time **sequential**
- Each stage is clearly defined with deliverables and milestones
- Not flexible, **no back-and-forth iterations**, not ideal for long-term project

Prototype model



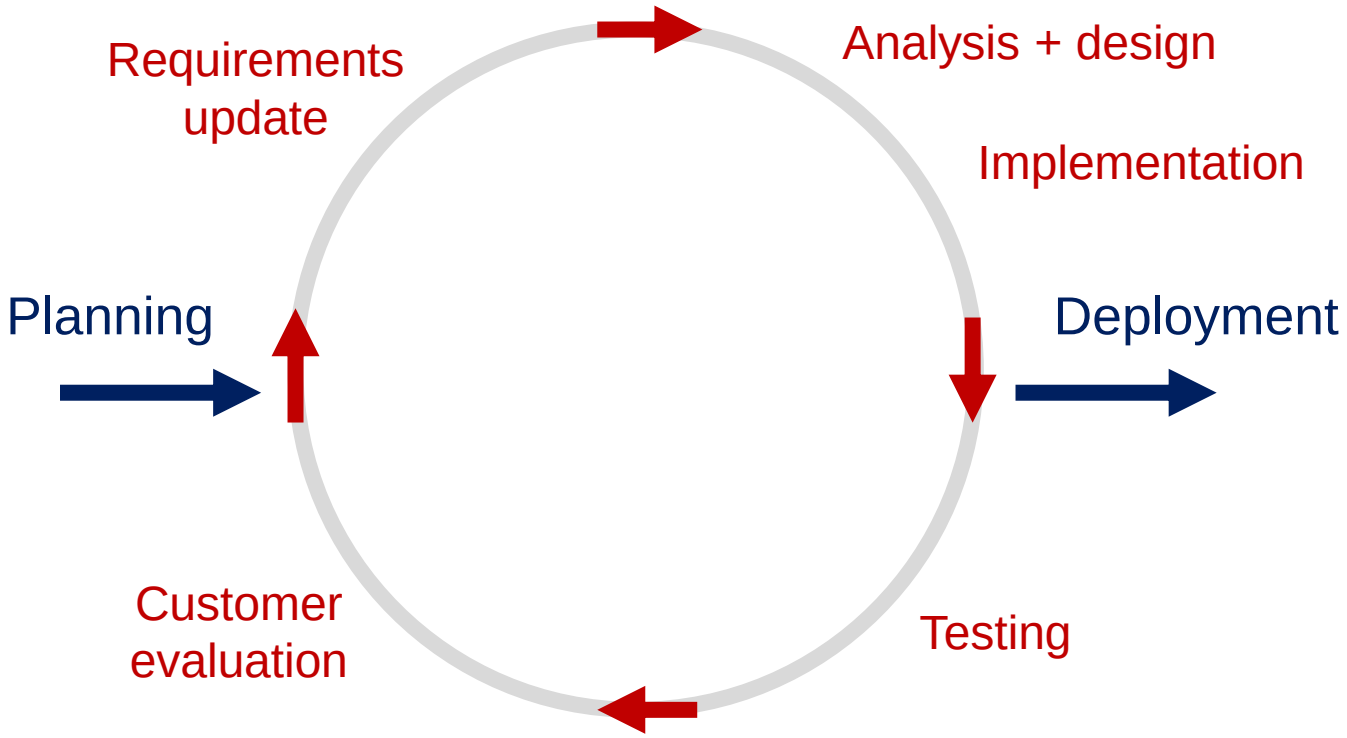
- Create **prototypes** to check expectations and fine tune
- **Customer** centred, not developer (techniques) centred
- Building prototypes take (much) time. You need smart customers (not stupid)

V model



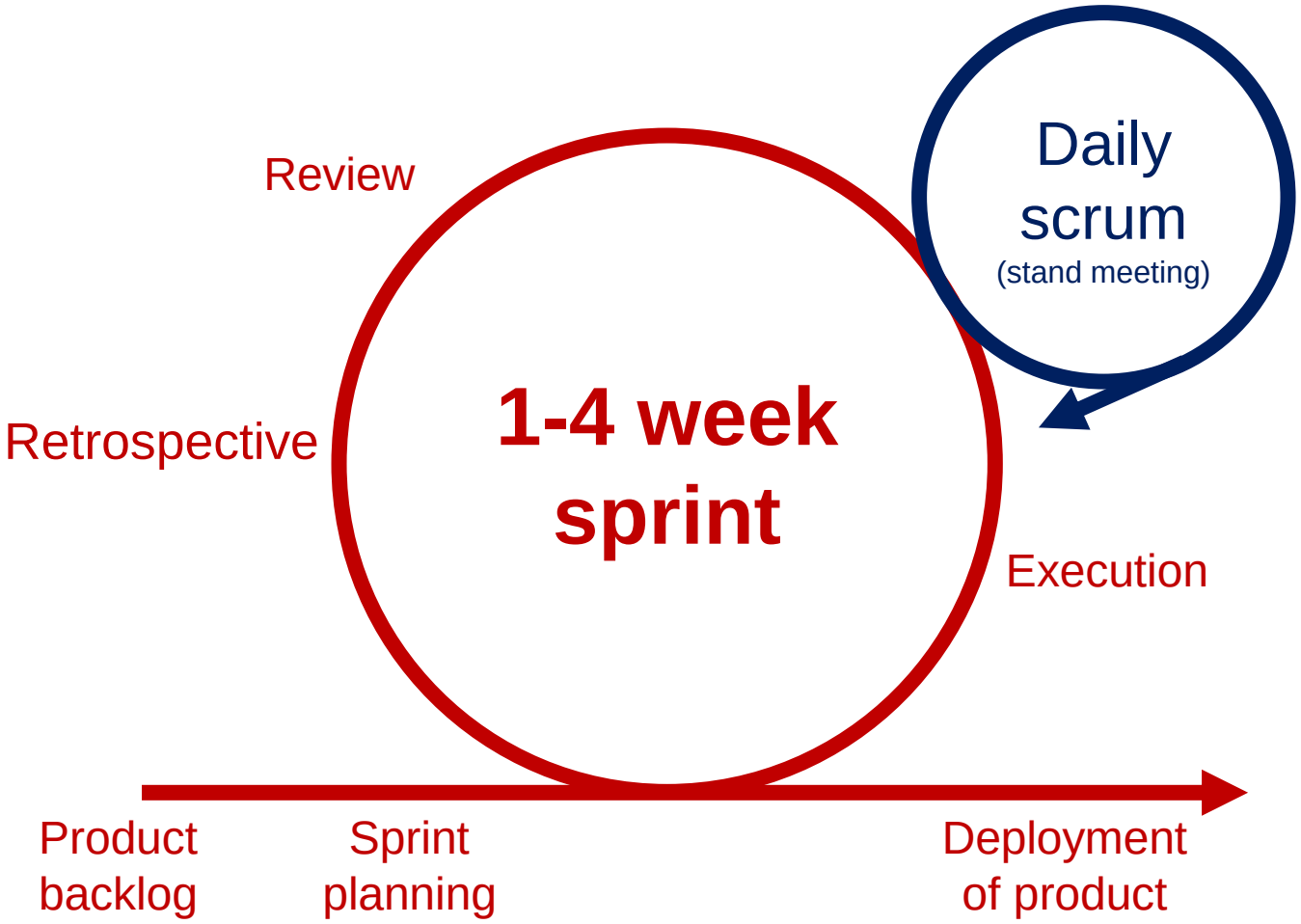
- Think first, do next. **Thinking** is top-down, **verification** is bottom up
- No backtrack iterations, cannot cope with changes
- Intensive verifications
- Used by **short-term small project**

Iterative model



- Use repeated cycle to cope with **changes** in requirements and definitions
- Allow software to evolve and increment during the **iterations**
- Resources can be wasted by repeating the procedure if no adaptions were made

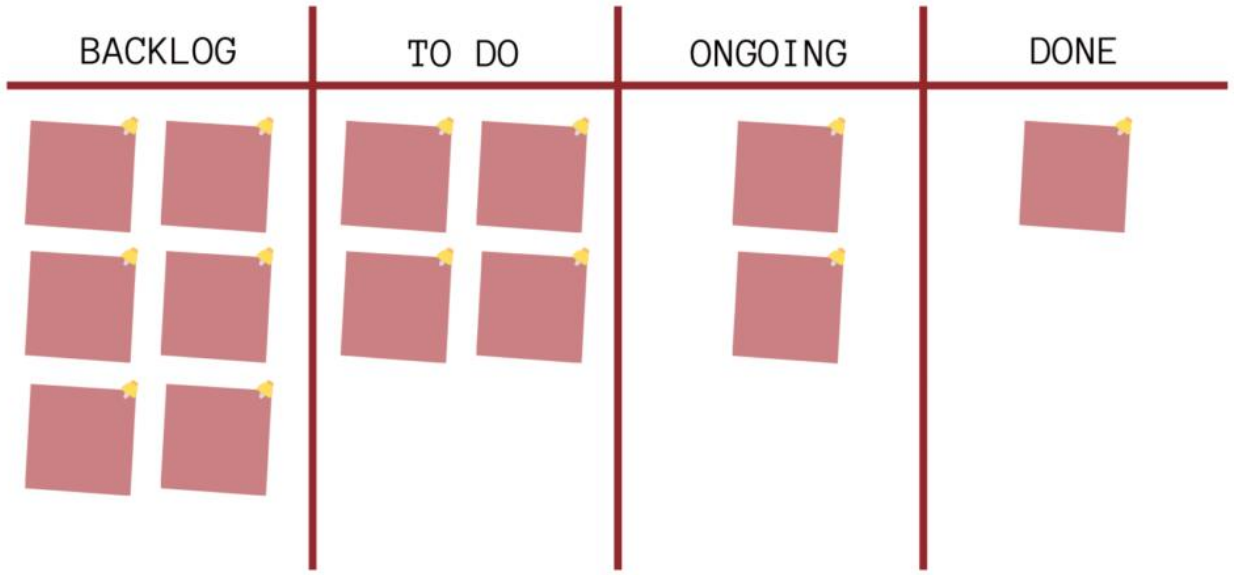
Scrum model



- SW engineering is matured, more **focused on R&D** and advanced features
- The development phase is cut into short **sprints** (1-4 weeks)
- Increase collaborations between cross-function teams, focused on **features**
- Require high-quality **sync** between team members

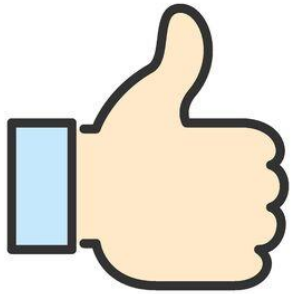
Kanban model

Kanban board



- Focused on **individuals**, no clear rules for team organization
- Suitable for research or content-oriented projects, allows **freedom** to explore
- Require a strong **independency** and initiative of team members and a strong team leader to integrate

Selection of models



- Fast in deliver solutions
- Create competence
- Please end-users

A fast, high-quality, repeatable software development process is “rule of thumb”.

Roles in SW team

Team

Customer

- Define needs & use case
- Clarify scenarios
- Set expectations
- Feedbacks
- Working with team

Manager

- Drive the process
- Organize all parts
- understands each parts
- Foster communications
- Build and motivate team

Developer

- Architecture design
- coding

Tester

- Unit & system test
- Stress test
- Check errors
- Independent code review
- Objective and critical

Definition of your role in a team

Bird

Lion

Bull

Snake

Definition of your role in a team

Teamwork is very important!

My tips

- **Customer centred**: encourage more interactions between customers and developers
- **Think-before-doing**: spend more time in thinking (70%) rather than programming (30%)
- **Clear goals**: rather than writing out long descriptive requirements, write short user stories
- **Core function first**: understand what is the core of project
- **Pair programming**: program in pairs, partners catch each others' mistakes
- **Continually refactor**: re-design the structure and modules of the software to optimize the performance

R&D model

Build models that demonstrate the essential properties of the product

Prototype is faster, cheaper, scalable and flexible than product

R&D procedure

Research oriented

- Define research competence
- (solutions, algorithms, methods)
- Define project goals
- Define development tasks
- Find market/customer needs
- Fit solution to practical cases (hardware, scenarios, price)
- Deliver

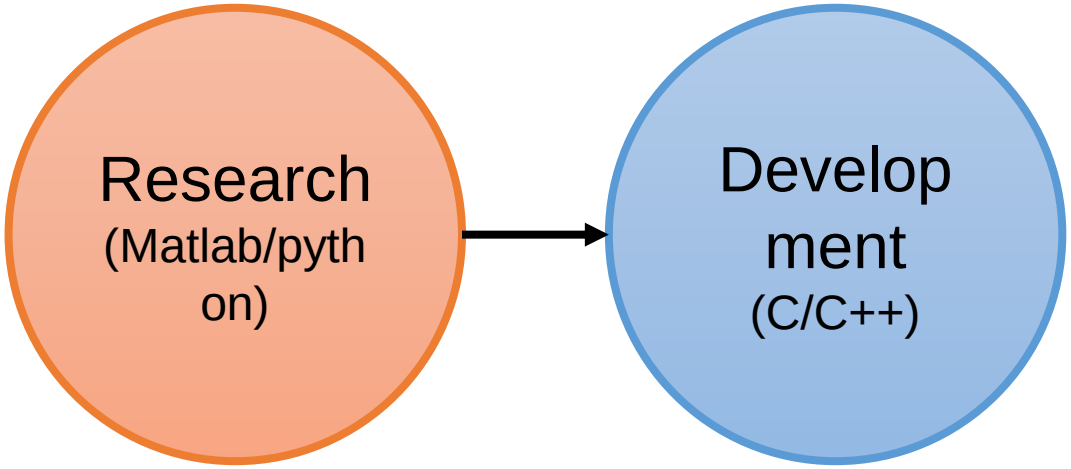
Needs oriented

- Define market/customer needs
- Define boundary conditions
- (hardware, scenarios, price)
- Define project goals
- Define research competence (solutions, algorithms, methods)
- Define development tasks
- Deliver

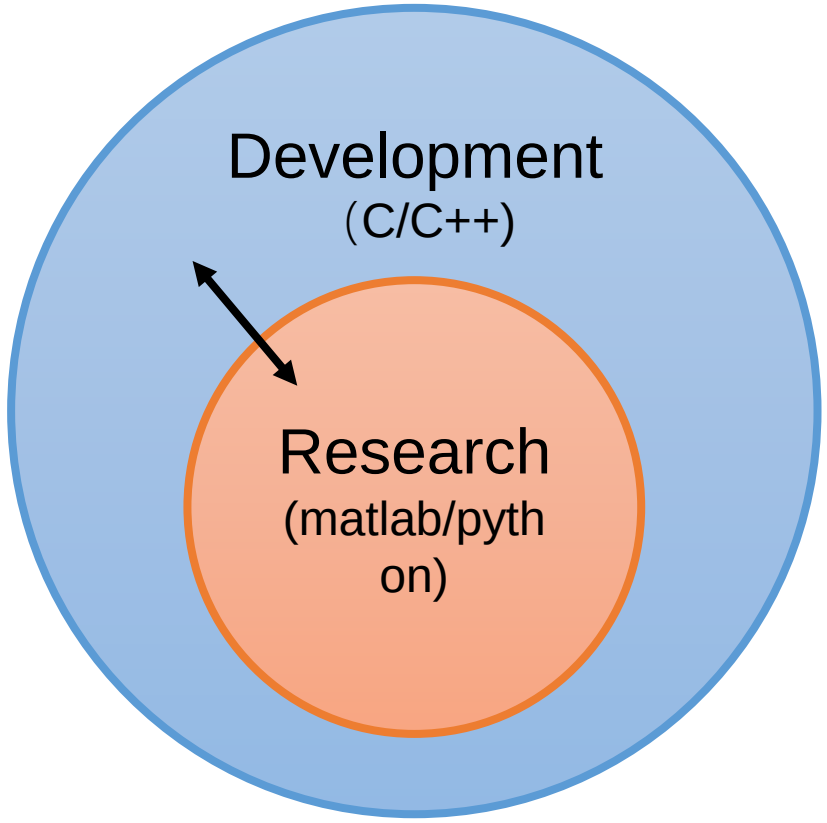
Which one is better?

R&D relationship

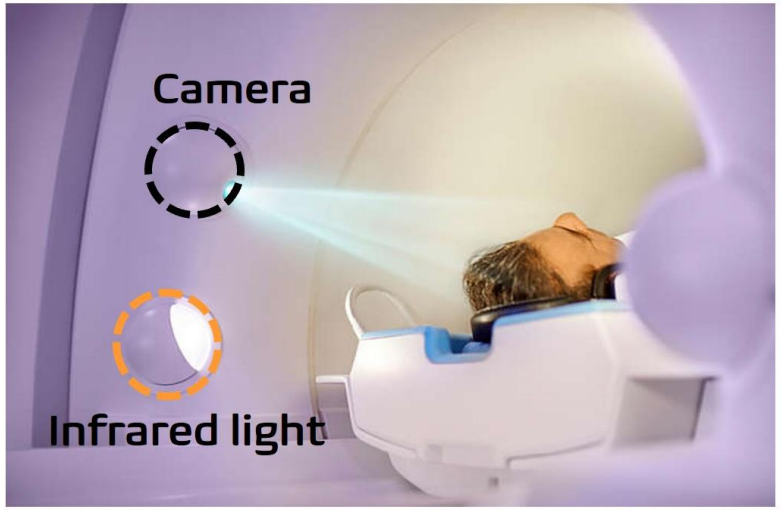
Conventional R&D

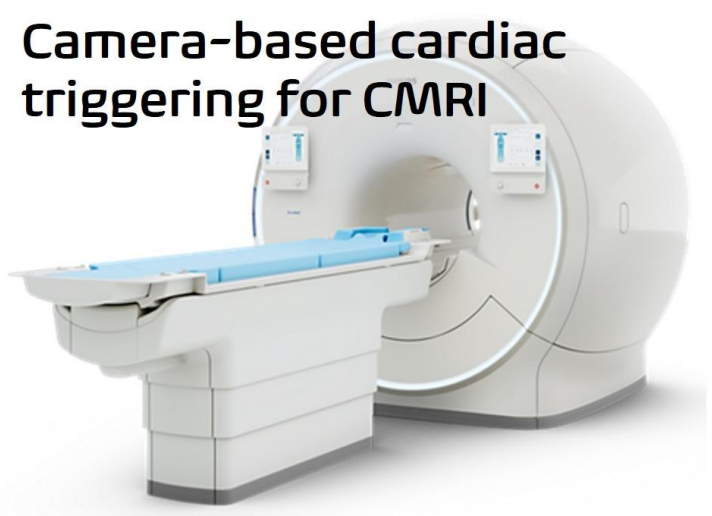


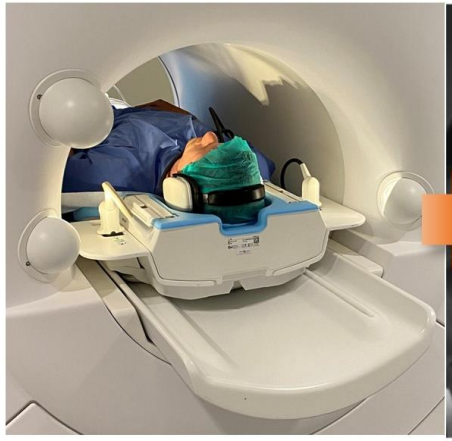
Modern R&D



My R&D model

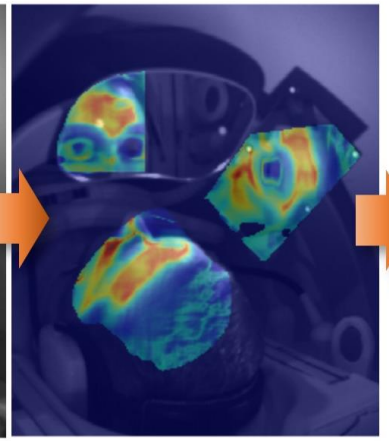




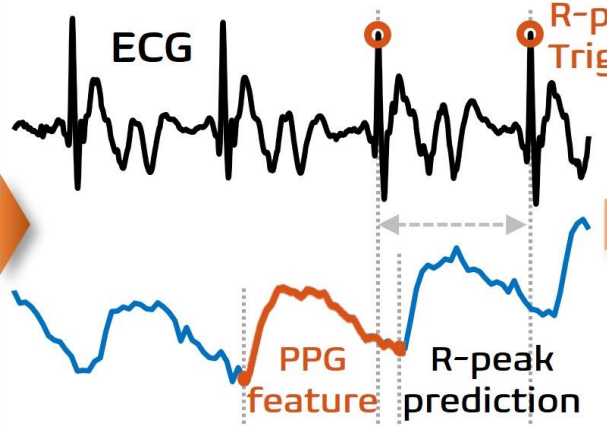


VitalEye camera

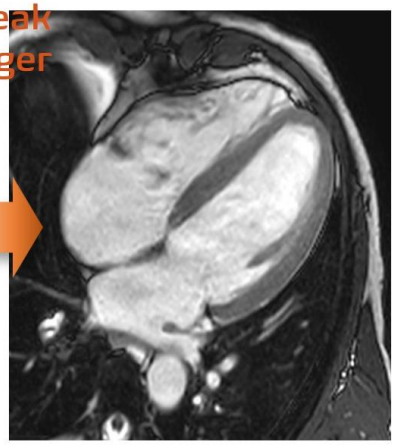
ROI detection



Living-skin



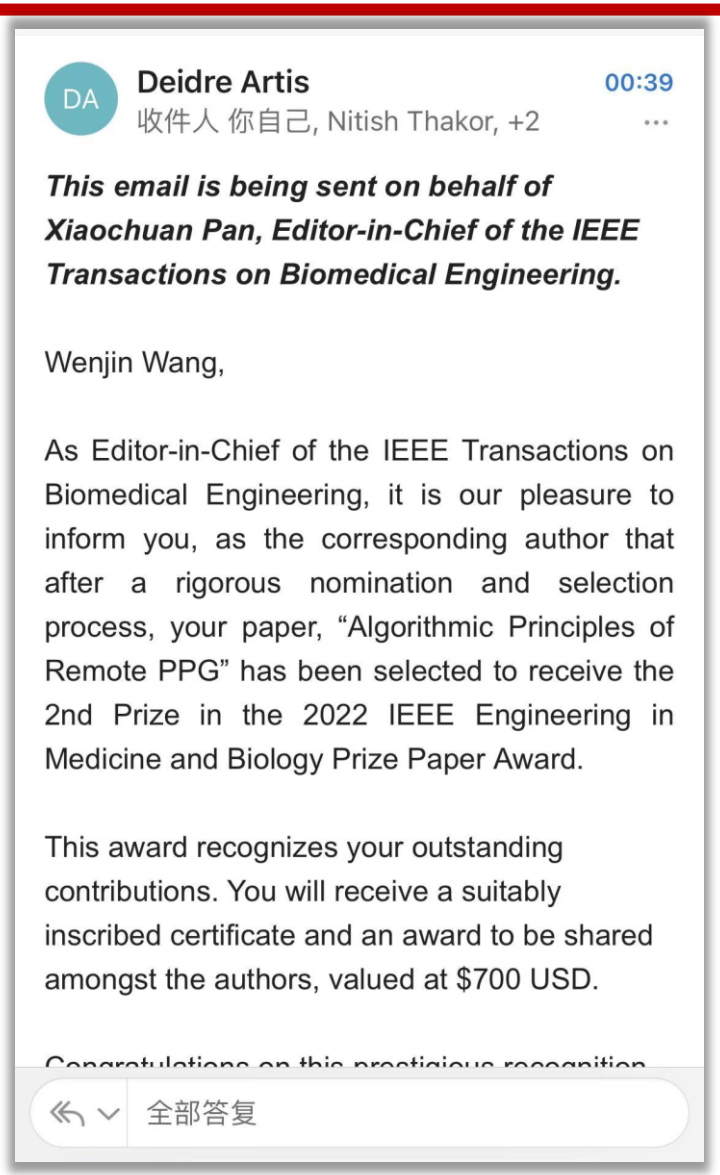
Camera-PPG based trigger

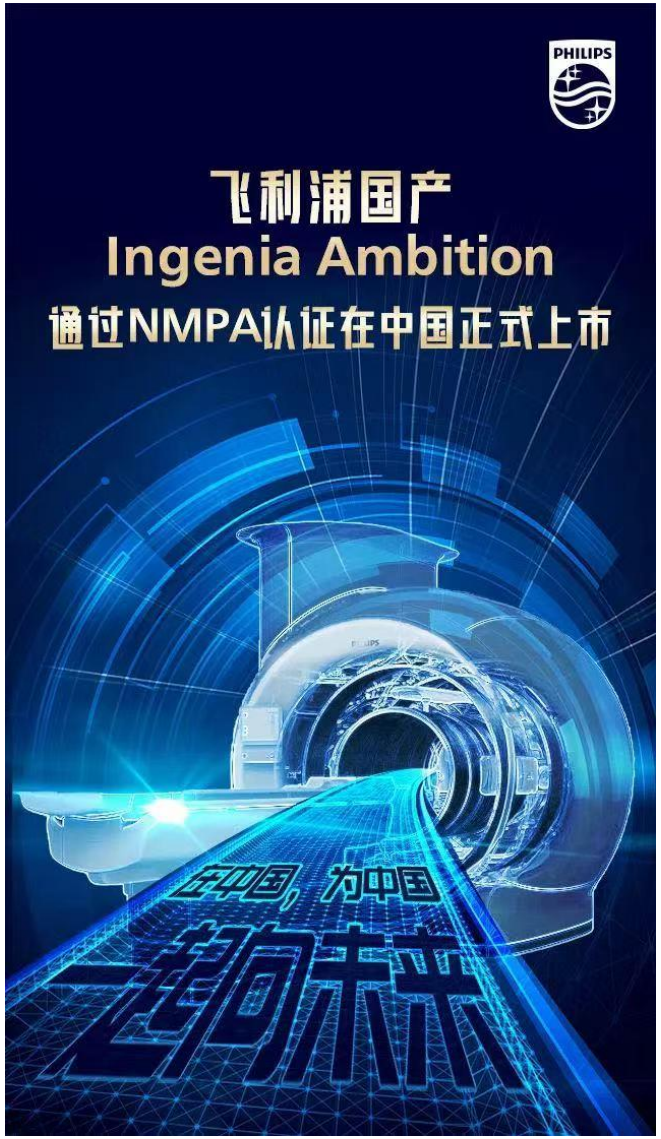


Cardiac MRI

W. Wang, S. Weiss, A. C. Den Brinker, J. H. Wuelbern, A. G. Tormo, I. Pappous, and J. Senegas, “Fundamentals of Camera-PPG based Magnetic Resonance Imaging,” in IEEE Journal of Biomedical and Health Informatics, 26 (9), 4378 – 4389, 2022. (中科院一区, **TOP**)

My R&D model



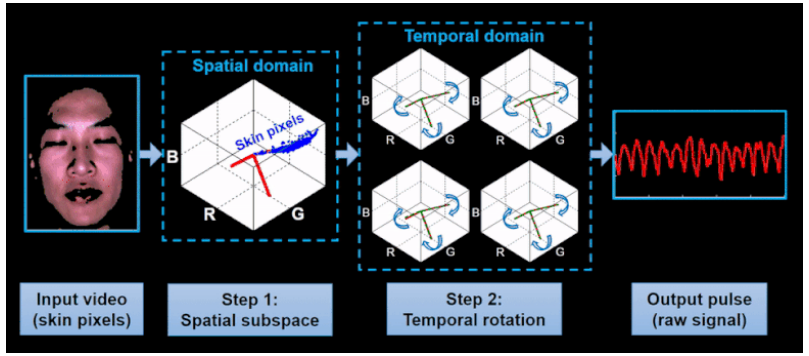


尽量兼顾“产”和“研”：

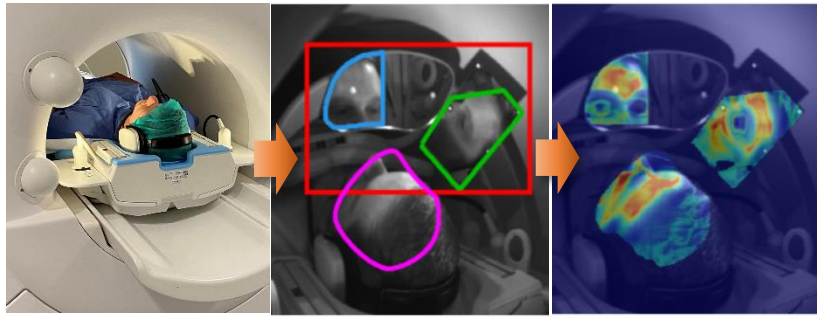
- 获IEEE Engineering in Medicine and Biology Prize Paper Award 2022
- 2022年7月，飞利浦Ingenia Ambition 通过NMPA认证在中国上市

My R&D model

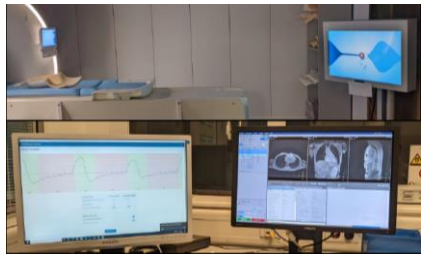
核心



系统



场景

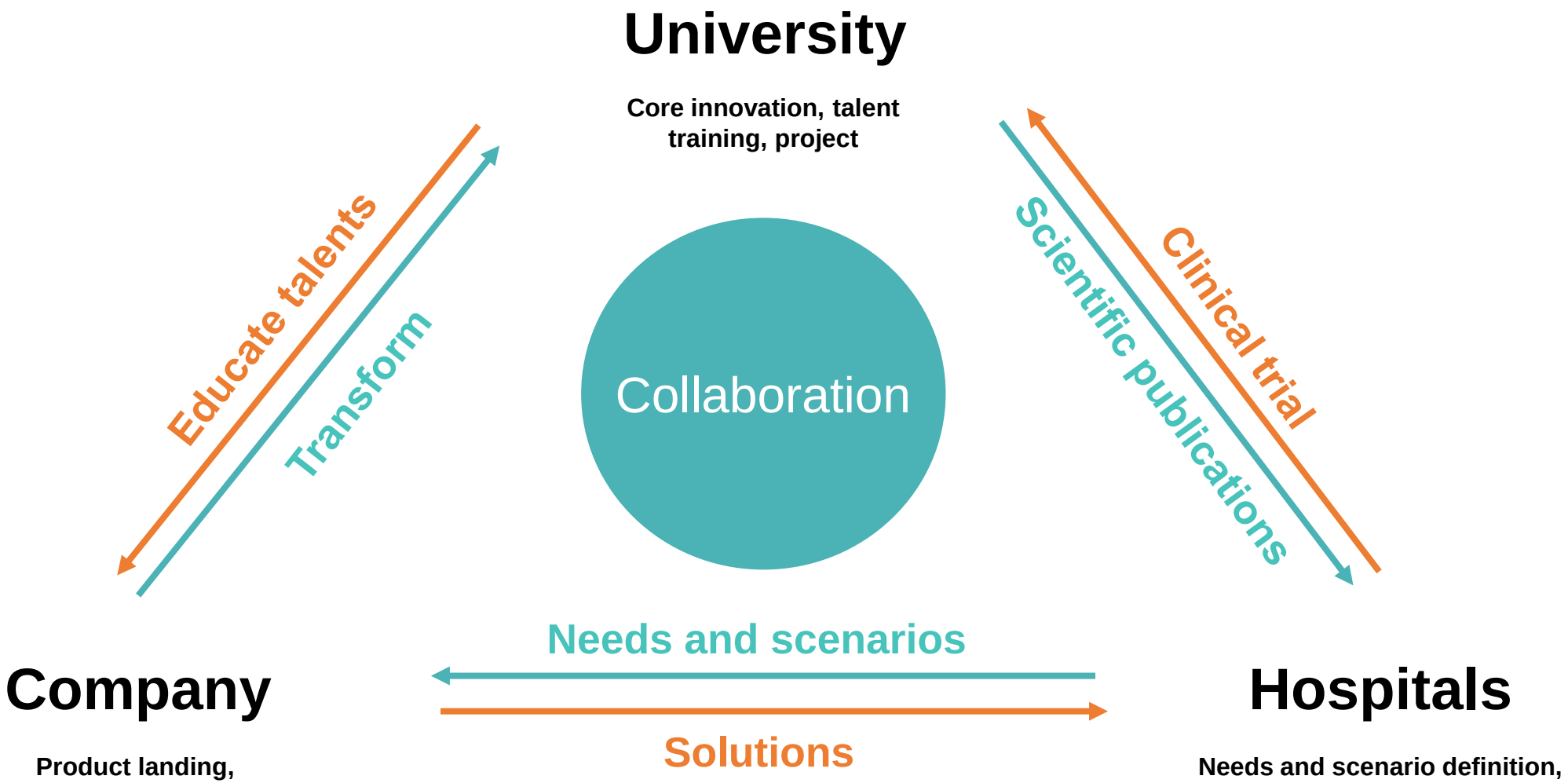


Stage 1: core methods (papers, inventions) → **Prototype!**

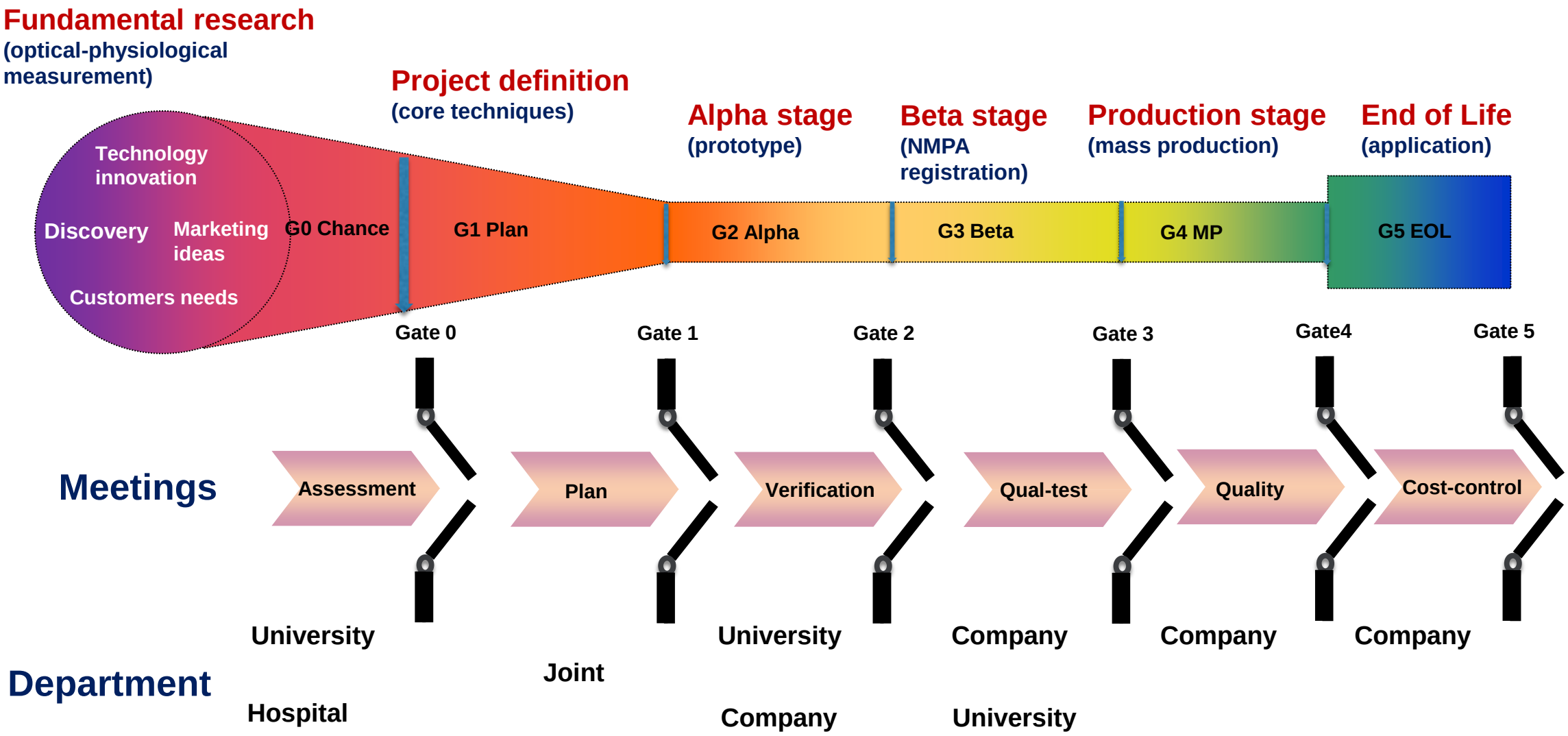
Stage 2: framework (functional systems) → **Prototype!**

Stage 3: Benchmark (trials and tests) → **Product!**

R&D organization



R&D organization



R&D organization

课题组目前和深圳三院共同打造“远程居家ICU抗疫一体化系统”，已报给深圳市委和市长，服务人民！

Content

1. Summarize and question review
2. Software engineering
3. **Bye**

Good initiatives as “teacher” long time ago ...

老俞跟我们说，爱教学才去教学。这是事业，不是工作。

Though the course is not perfect, will keep improving it!

Appreciations for the contributions of

于东方，李丹，罗婷丹

to this course!

Good wishes to your future, anytime want to discuss, you can contact me by wangwj3@sustech.edu.cn

Keep learning and forward!

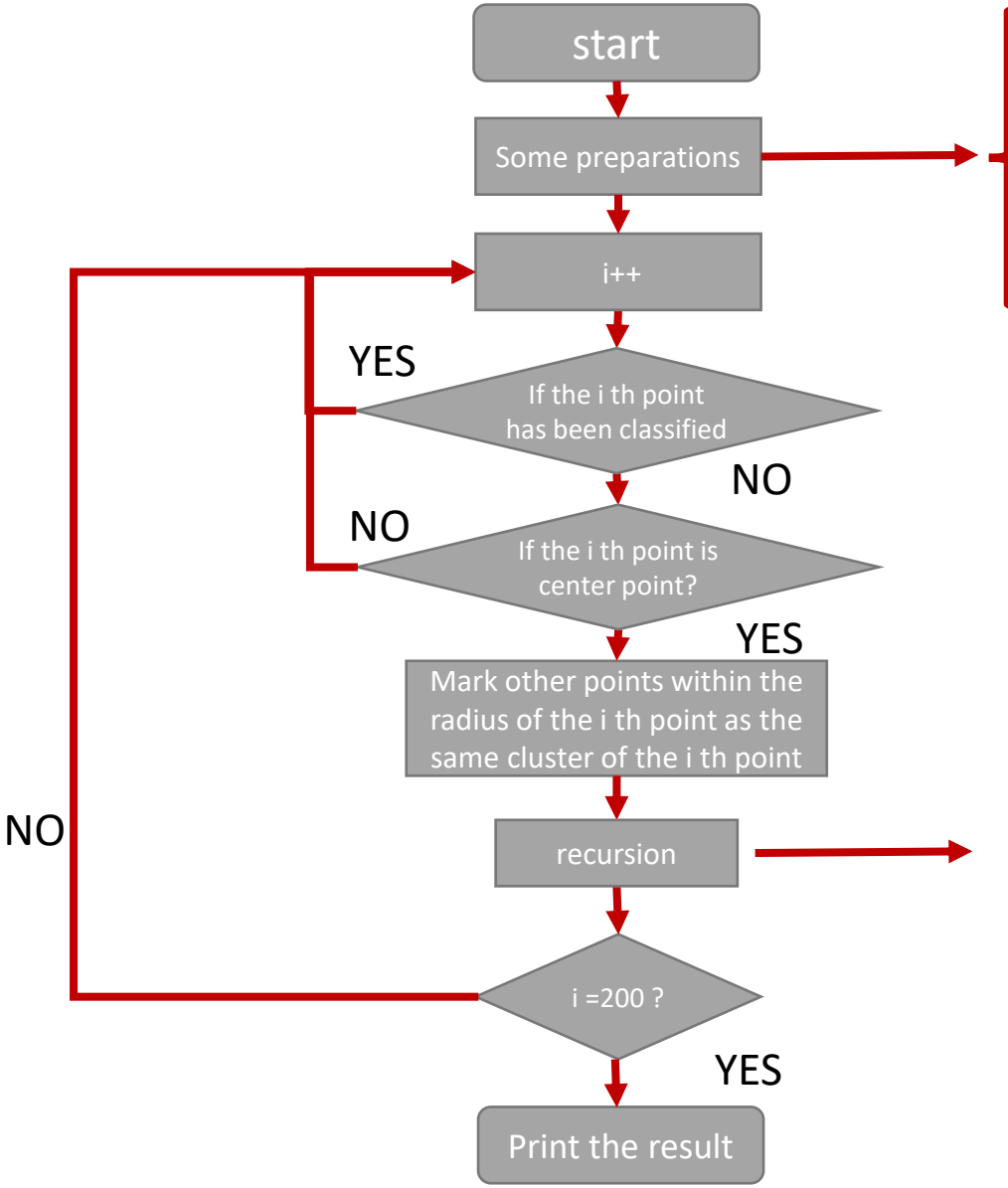
Online project

Use C to implement the DBSCAN clustering algorithm, and print the index of the fish contained in each cluster.

a) The csv file has been uploaded on bb
b) If you don't know how to read a csv file, you can refer to the answer to the previous assignment
c) If you want to plot the result in an image you can install OpenCV.
d) Minpts is 4 and the radius is 2

Note: this is online project, not the assignment! There is no assignment for this week.

Online project



Read the csv file and convert ASCII data into float

Mark the classification of all points as -1(unclassified)

Calculates whether the distance between any two points is less than radius(a total of 200*200 calculations are required)

Int i = 0

Apply the previous three steps to all points within the radius of the i th point

Online project

You can choose to use this function to plot the result

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include <opencv2/opencv.hpp>

#define num_of_points 200
#define dimension 2

typedef struct
{
float coord[dimension];
int cluster;
bool
is_neighbor[num_of_points];
}fish;
```

```
void show_result(fish* fishes)
{
cv::Mat result = cv::Mat(750, 1500, CV_8UC3, cv::Scalar(0, 0, 0));
int first_point_cluster = fishes[0].cluster;
for (int i = 0; i < num_of_points; i++)
{
cv::Point2f fish_point = cv::Point2f(fishes[i].coord[0] * 50,
fishes[i].coord[1] * 50);
if (fishes[i].cluster == first_point_cluster)
{
cv::circle(result, fish_point, 5, cv::Scalar(255, 0, 0), -1);
cv::circle(result, fish_point, 100, cv::Scalar(125, 0, 0), 0);
}
else
{
cv::circle(result, fish_point, 5, cv::Scalar(0, 0, 255), -1);
cv::circle(result, fish_point, 100, cv::Scalar(0, 0, 125), 0);
}
}
imshow("result", result);
cv::waitKey(0);
}
```

Online project

The result looks like this