



Лекция 14

Разработване:

принципи, методология, ръководство

DAAD Project
“Joint Course on Software Engineering”

Humboldt University Berlin, University of Novi Sad, University of Plovdiv,
University of Skopje, University of Belgrade, University of Niš, University of Kragujevac

Литература

- ▶ S. Mc Connell: Code Complete, Microsoft Press, 1993 *)
- ▶ W.E. Dijkstra: Notes on Structured Programming, Academic Press, 1972 +)
- ▶ N. Wirth: Program Development by Stepwise Refinement, CACM, 4,1971 +)
- ▶ M. Henricson, E. Nyquist: Programming in C++ - Rules and Recommendations, Ellementel Telecommunications Systems Laboratories, Sweden, 1992
- ▶ L.W. Cannon et al.: Recommended C Style and Coding Standards, University of Toronto, University of Washington, 1989

*) Учебник на Berkeley University

+) Класика в софтуерните технологии

История

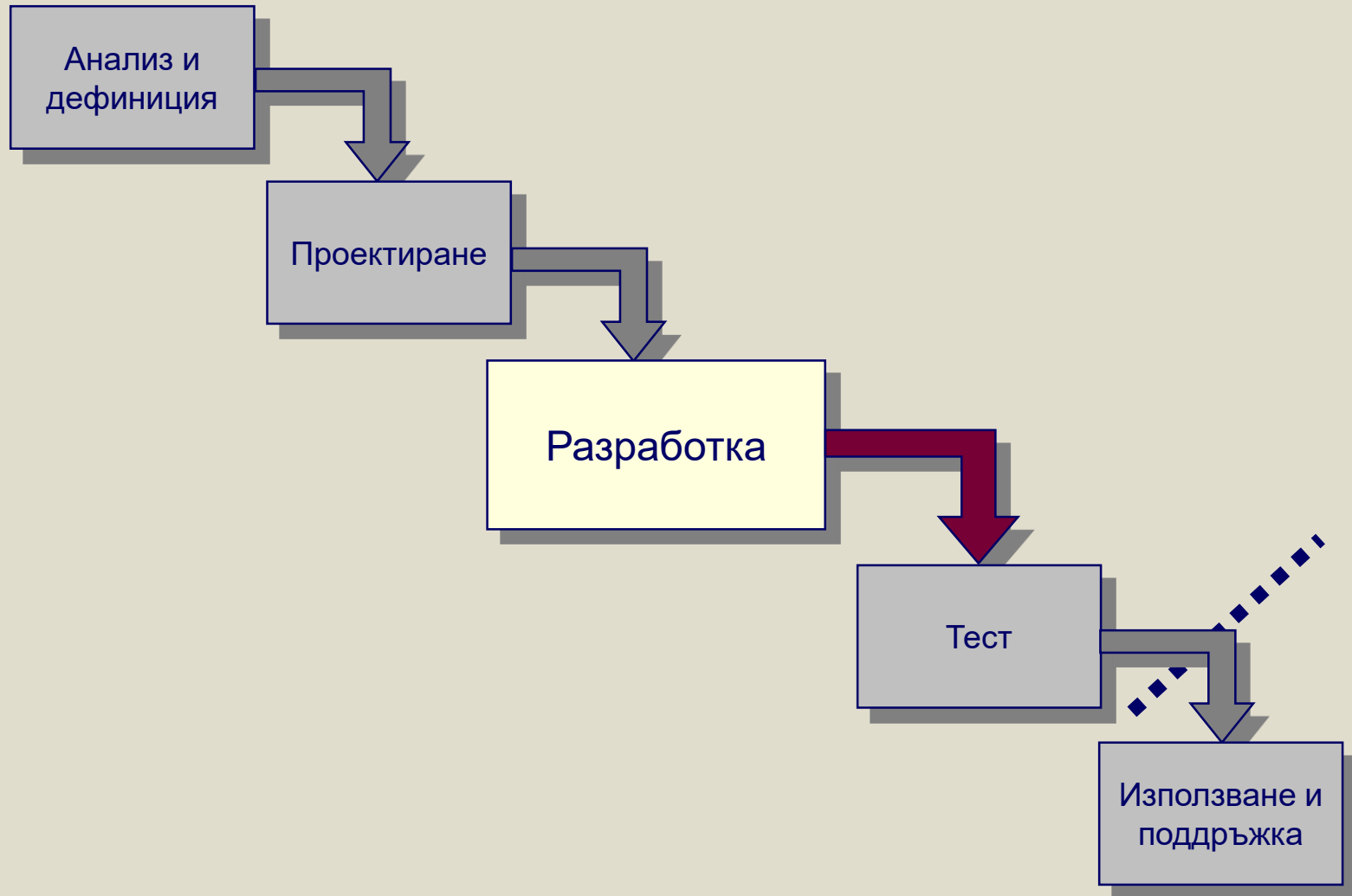


- ▶ **Prof. Dr. Niklaus Wirth**
 - *15.2.1934 в Winterthur, Швейцария
 - Професор ETH Zurich (пенсиониран)
- ▶ Разработчик на програмния език Pascal (1970), Modula-2 (1982) и Oberon (1989)
- ▶ Разработчик на компютърната система Lilith (1980) и Ceres (1986) и техните операционни системи
- ▶ Много добър в структурното програмиране

18. Разработка: принципи, методология, ръководни принципи

- a) Документи във фазата на разработка
- b) Програмен стил и методология
- c) Основни концепции на ориентирания към алгоритми изглед на системата

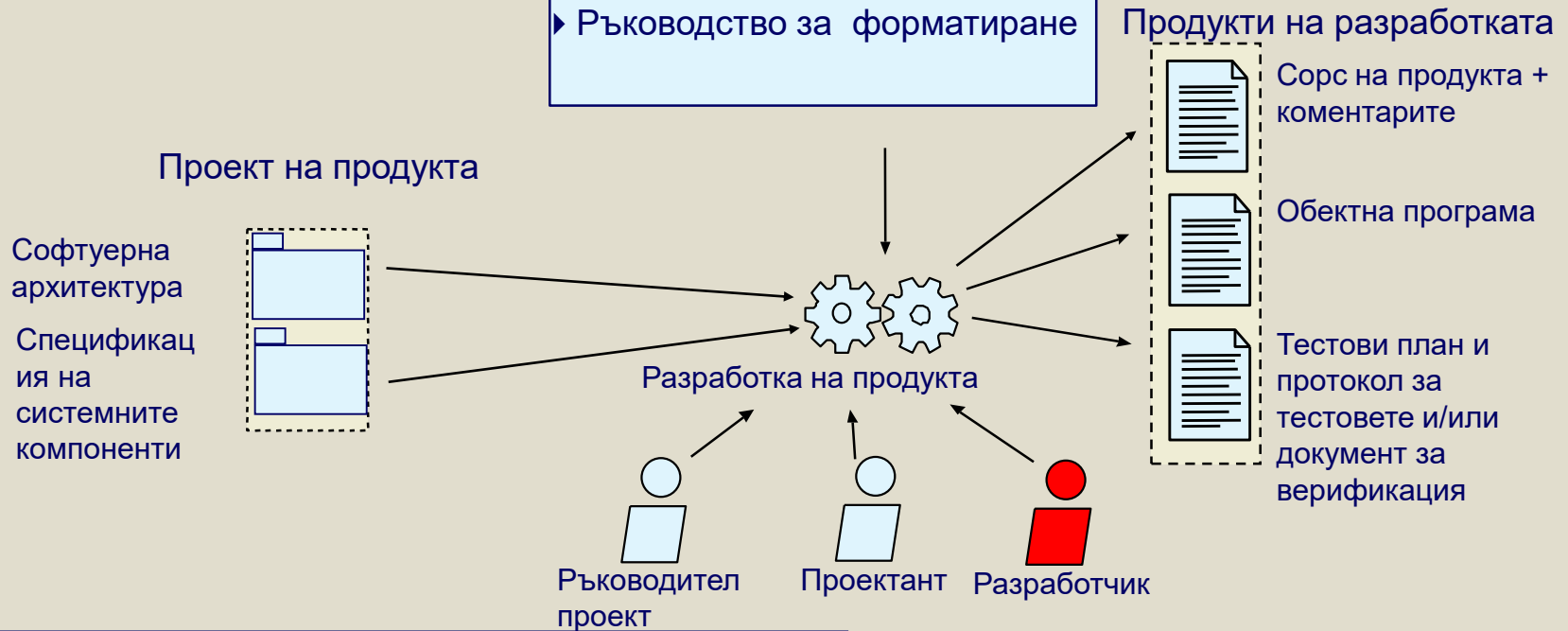
Класическия водопаден модел (1970)



Фазата на разработване в процеса на разработка на софтуер

Принципи

- ▶ Постъпково усъвършенстване
- ▶ Цялостна документация
- ▶ Методология на програмиране
- ▶ Стил на програмиране
- ▶ Ръководство за форматиране



легенда :  действие  модел (артифакт)

 роля  документ (артифакт)

Разработка: принципи, методология, ръководство

► Какво ви е изяснено за програмирането?

- Програмирането вече е въведено в студентската програма, а и много студенти работят в различни софтуерни фирми....
- Какво е важно за вас в програмирането?
- Какво е добър програмен стил?

► Тук:

- Събиране на знанията: форматиране, принципи ...

► SEUN'99, sd & m:

- “Студентите по информатика не знаят как да програмират.”

→ те трябва да научат изискваното от тях в компаниите
за 2 седмици

Пример на C++: нарушаване на програмните принципи

```
void Calculation()  
{  
    float K0, Kn, p;  
    int n;  
    cin >> K0;  
    cin >> n;  
    cin >> p;  
    Kn = K0 * (1+p/100) ^n;  
    cout << Kn;  
}
```



Резултат?

Нарушаване на програмните принципи: проблеми?

Липса на
отстъпи

```
void Calculation ()  
{  
float K0, Kn, p;  
int n;  
cin >> K0;  
cin >> n;  
cin >> p;  
Kn = K0 * (1+p/100) ^n;  
cout << Kn;  
}
```

Грешка

Идентификатори
без значение

Оформяне:
няма нов
ред

Липсват
коментари

Подобрана версия

```
void CompoundInterest()
{
//input data
    float BasicCapital;
    float ProcentValue;
    int Years;
//Outout data
    float EndCapital;

//read values from input
    cin >> BasicCapital;
    cin >> Years;
    cin >> ProcentValue;

//Calculation formula for compound interest
    EndCapital = pow(BasicCapital*(1+ProcentValue/100), Years);

//Output of EndCapital
    cout << EndCapital;
}
```

```
void calculation ()
{
float K0, Kn, p;
int n;
cin >> K0;
cin >> n;
cin >> p;
Kn = K0 * (1+p/100) ^n;
cout << Kn;
}
```

18. Разработка: принципи, методология, ръководни принципи

- а) Документи във фазата на разработка
- б) Програмен стил и методология
- с) Основни концепции на ориентирания към алгоритми изглед на системата

Аспекти на добрия стил на програмиране

организация на програмния код

- ▶ коментари
- ▶ цялостна документация
- ▶ избор на имена
- ▶ форматиране – оформяне на програмата
- ▶ структурно програмиране
- ▶ странични ефекти
- ▶ сигурни програми
- ▶ преносими програми
- ▶ производителност

→ Ръководни принципи специфични за езика
например за C++, Java

→ Ръководни принципи специфични за
компанията

Аспекти на методологията за програмиране: пътят към добрата програма

- ▶ избор на програмен език
- ▶ подредба на компонентите за разработка
- ▶ метод за постъпково усъвършенстване
- ▶ използване на основни концепции
- ▶ оценка на програмните грешки

Коментари

„Без документация, програмата може да бъде разбрана само от оригиналния и разработчик, а след само 2 седмици отпуск и той самия няма да може да разбере продукта, който е създал.“

R.Winder: Разработчик на софтуер на C++, р.3

Ръководни принципи: Какво трябва да се коментира?

- ▶ Започването на нова програмна единица (клас, метод):
 - Задача: обяснение на външното поведение на програмната единица
 - Значение на параметрите
- ▶ Данни: ролята на типовете данни и променливите
- ▶ Започващ код:
 обща задача
- ▶ Промените в програмата
 → коментари за промените !!

Проблеми при прекалено много коментари

- ▶ Софтуер с много коментари не е винаги добре коментиран:
 - Прекалено многото коментари са проблем (липсва прегледност!)
 - Коментарите не трябва да повтарят дума по дума написания код
 - Приоритет пред коментарите имат: програмните структури, избор на имена (програмата сама се документира)

Цялостна документация: административна информация

- ▶ Автори (програмисти)
- ▶ Номер на версия
- ▶ Статус (в работа, завършена, одобрена)
- ▶ Дата
- ▶ Как да извикаме програмните единици?

Ръководни принципи & конвенция за идентификаторите (1)

- 1 Идентификаторите са на естествен език или с имена от проблемната област или разбираеми съкращения на тези имена
- 2 Всеки идентификатор започва с буква
 - Подчертаваща (_) не се използва
- 3 Идентификаторите не съдържат празни символи
 - Изключения
 - UML-нотация, но трябва да се премахне, когато се трансформира в Java програма
- 4 В общия случай трябва да се използват малки и големи букви
- 5 Два идентификатора не трябва да се различават само по големите и малки букви.

Ръководни принципи & конвенция за идентификаторите (2)

6 В имената трябва да използва или английски или локалния език

Изключение: общо одобрените английски имена

- като например *push*

7 Ако идентификатора се състои от много думи, тогава всяка дума започва с голяма буква

- например *WordNumber*
- Подчертаваща не се използва за отделяне на думите.

Ръководни принципи & конвенция за идентификаторите (3)

8 Имена на класове...

- винаги започва с голяма буква
- представят се от съществителни в единствено число и възможност за добавяне на прилагателно
 - например **Seminar, public Advertisement**
- за GUI класове, съдържа суфикса **GUI**

9 Имена на обекти...

- Винаги започва с малка буква
- Обикновено завършва с името на класа
 - например **aCustomer**
- започва (в случаи на анонимни обекти) с **a, an**
например **aPoint, anElevator.**

Ръководни принципи & конвенция за идентификаторите(4)

10 Имена на атрибути

- Започват винаги с малка буква, избягват се съвпадения с името на класа
 - например `hotWaterLevel`, `nameField`, `eyeColor`
- описват се в повече детайли
 - например `lineCounter`, `windVelocity`, `dataState`.

Ръководни принципи & конвенция за идентификаторите (5)

11 Имена на операции

- винаги започват с малка буква
- обикновено започват с глагол следван от съществително
 - например `print`, `change`, `showPicture`, `readAddress`, `shiftRectangle`
- `getAttributeName`, ако извлича стойността само на един атрибут
- `setAttributeName`, само когато се задава стойността на един атрибут
- `isAttributeName`, само когато се извлича стойността на boolean атрибут, например `isEngaged`, `isForbidden`.

Структурно програмиране

- ▶ Използват се структурни изрази:
 - Последователност
 - Условни изрази
 - While цикли

- ▶ Без GOTO !!

Странични ефекти

$x := \text{израз}$



Основна задача:
доставяне на
стойност

Страничен ефект:
Промяна на променливи
 $x = x++ * 5;$
 $x = f(3, 5);$

Цел: избягване на странични ефекти

Сигурно (надеждно) програмиране (Sommerville, p. 384)

► Излишен код:

→ Води до грешки и несъгласуваност

Примерно решение: поддръжка

Преносими програми

▶ Зависим от хардуера код:

→ концентриран в модули:

например:

- Генериране на код в отделен пас на компилатора
- Многослойна архитектура: зависими части на хардуера в най-долния слой

▶ Зависим от системата код:

- Операционна система (UNIX – Windows)
- Компилатор
(например C++: Borland, Microsoft, Gnu, ANSI ...)

Постъпково усъвършенстване

(Top down програмиране)

- Разработка с абстрактни данни (данни само с име) и абстрактни операции
 - обяснени с коментари
- Усъвършенстване на коментарите:
 - Детайлни коментари или код
- Краен продукт:
 - Оригиналните коментари остават

Постъпково усъвършенстване: пример на C++

- Алгоритъма изчислява броя на лихвените дни ако имаме две въведени дати:

```
void calculateInterestDays ()
{
    int date1, date2; // input data in form DDMM
    int days; // output data
    // Condition: dates belong to one year
    // Algorithm -----
    // Input
    // Breakdown to days and month
    // Consideration of special cases
    // Computation of days
    // Output
}
```

1. Усъвършенстване

```
// refinements: 1. refinement:
```

```
// input
```

```
cout << "1. date: "; cin >> date1;
```

```
cout << "2. date: "; cin >> date2; cout << endl;
```

```
// Breakdown to days and months
```

```
// Breakdown to days
```

```
// Breakdown to month
```

```
// Consideration of special cases
```

```
if (day1 == 31) day1 = 30;
```

```
if (day2 == 31) day2 = 30;
```

```
// Computation of days
```

```
days = (month2 - month1) * 30 + day2 - day1;
```

```
// Output
```

```
if (days < 0) cout << "Error: 1. date is before 2.  
date!" << endl;
```

```
else cout << "Number of interest days = " << days;
```

Усъвършенстване
на коментарите

Усъвършенстване
на код

2. усъвършенстване

```
// refinements: 2. refinement:
```

```
// Breakdown to days and months
```

```
// Breakdown to days
```

```
int day1, day2; // help variables
```

```
day1 = date1 / 100;
```

```
day2 = date2 / 100;
```

```
// Breakdown to months
```

```
int month1, month2; // help variables
```

```
month1 = date1 % 100;
```

```
month2 = date2 % 100;
```

```
// end calculateInterestDays ()
```

Усъвършенс
тване на
кода

- + И ядрото на алгоритъма и усъвършенстваните структури са ясно видими
- + Трябва да погледнете само усъвършенстванията, за да видите детайлите.

Заменими усъвършенствания в „ядрото на програмата“ (1)

```
void calculateInterestDays()
{
    int date1, date2; // Input data, Form DDMM
    int days; // Output data
    //Condition: dates belong to one year
    int day1, day2; // help values
    int month1, month2; // help values
    // Input
    cout << "1. Interest date: "; cin >> date1;
    cout << "2. Interest date: "; cin >> date2;
    cout << endl;.
```

Заменими усъвършенствания в „ядрото на програмата“(2)

```
// Breakdown to days and months
// Breakdown to days
    day1 = date1 / 100;
    day2 = date2 / 100;
// Breakdown to months
    month1 = date1 % 100;
    month2 = date2 % 100;
// Consideration of special cases
    if (day1 == 31) day1 = 30;
    if (day2 == 31) day2 = 30;
// Calculation of days
    days = (month2 - month1) * 30 + day2 - day1;
// Output
    if (days < 0) cout << "Error: 1. date is before 2.
date!" << endl;
    else cout << "Number of interest days = " << days;.
```


Принципи на усъвършенстването: предимства

- + Процеса на разработка е документиран в сорс кодовете
- + По-лесно и по-бързо въвеждане на нови хора в програмата
- + По-главните нива на усъвършенстване, могат първо да бъдат описани на естествен език
- + Програмата е структурирана от две дименсии : от контролни структури и нива на усъвършенстване.

18. Разработка: принципи, методология, ръководни принципи

- a) Документи във фазата на разработка
- b) Програмен стил и методология
- c) Основни концепции на ориентирания към алгоритми изглед на системата

Основни концепции на ориентирания към алгоритми изглед на системата

Цели:

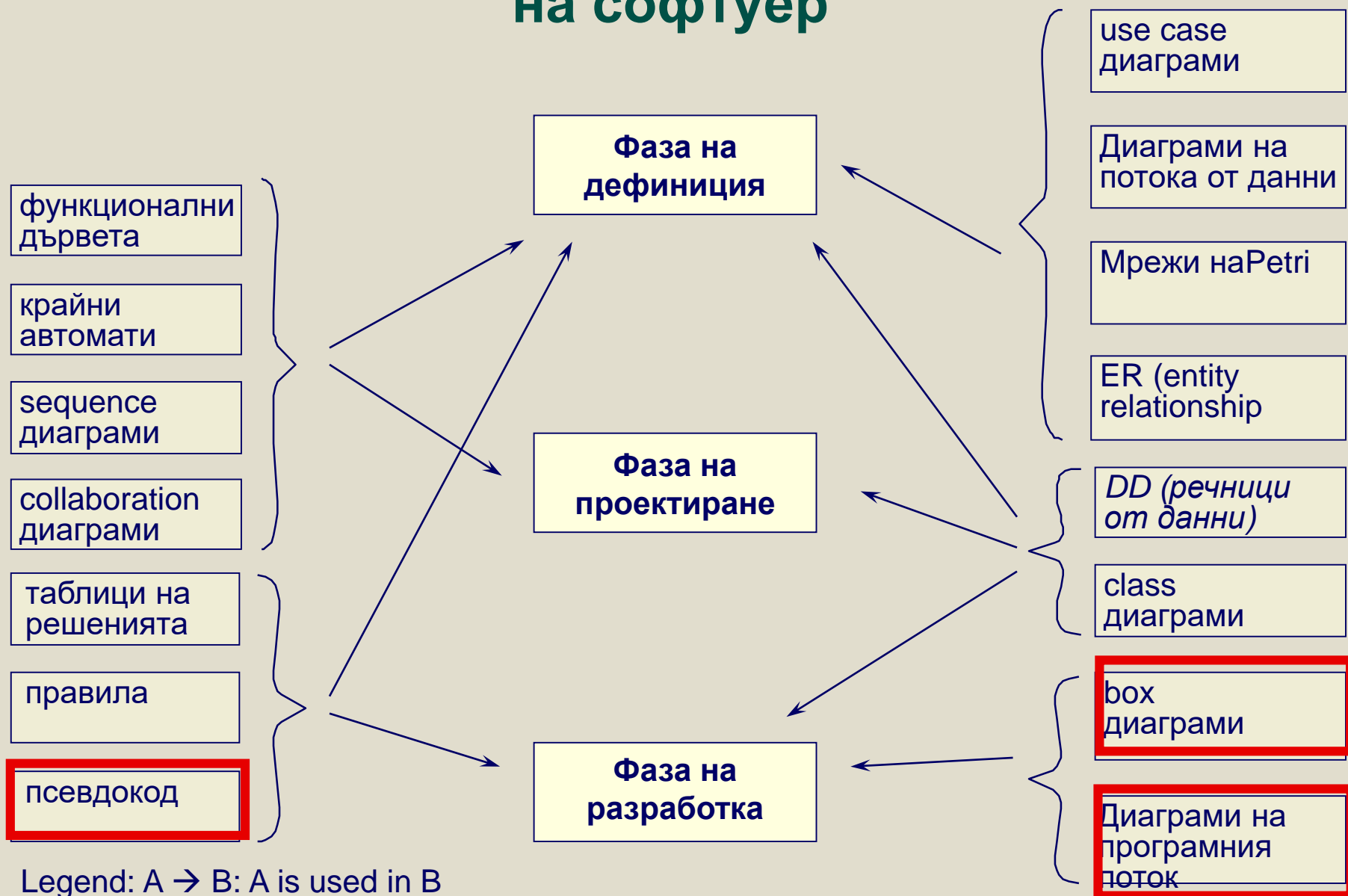
- ▶ Поддръжка на програмната разработка
- ▶ Т.е. процеса на разработка на софтуер е по-сигурен като се използват основни концепции

Основни концепции на процеса на разработка на софтуер

Balzert vol. 1, 2nd edition 2001

Balzert vol. 1, 2nd edition 2001											
<div>Concepts and Views</div> <div>Alternative Notations Often used Rarely used</div>											
Function tree	Use Case Diagram 1987	Data flow diagram 1966	<i>Data-Dictionary</i> 1979	Entity Relationship Model 1976	Class diagram 1980/1990	Box diagram 1973	Program flowchart 1966	Decision tables 1957	Activity diagram 1997		Collaboration diagram
						Pseudo code		Rules	State automaton 1954	Petri Net 1962	Sequence diagram 1987
Functional hierarchy	Business Process	Information Flow	Data Structures	Entity types and relations	Class structures	Control structures	If-Then structures	Finite State Automaton	Concurrent structures	Interaction structures	
Functional View			Data-Oriented View		Object-Oriented View	Algorithmic View	Rule-Based View	State-Oriented View		Scenario-Based View	

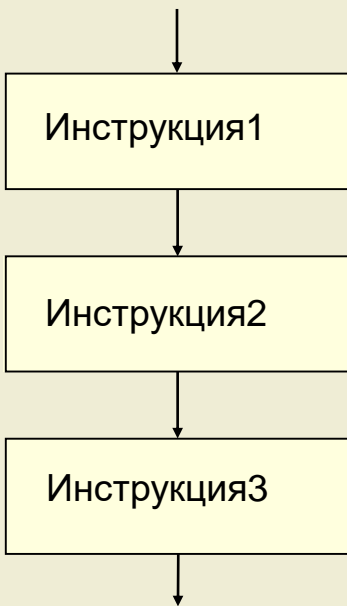
Основните концепции и фазите за разработка на софтуер



Класификация на основните концепции според техните нотации



Последователност

последов.	изглед	описание
Вох диаграма	<div> <div>Инструкция1</div> <div>Инструкция2</div> <div>Инструкция3</div> </div>	Произволно голям четириъгълник затваря с хоризонтални линии всяка инструкция.
Псевдокод (Java)	<div> <div>Инструкция1 ;</div> <div>Инструкция2 ;</div> <div>Инструкция3 ;</div> </div>	Индивидуалните инструкции са отделени с точка и запетая (;).
Диаграма на потока на програмата	 <pre> graph TD Start(()) --> I1[Инструкция1] I1 --> I2[Инструкция2] I2 --> I3[Инструкция3] I3 --> End(()) </pre>	Прости инструкции са представени с четириъгълници, които са свързани с линии на потока

избор	изглед	описание								
Вох диаграми	<table><tr><td colspan="2">израз</td></tr><tr><td>истина</td><td>лъжа</td></tr><tr><td>yes-instruction</td><td>No-instruction (или празно)</td></tr><tr><td colspan="2">инструкции</td></tr></table>	израз		истина	лъжа	yes-instruction	No-instruction (или празно)	инструкции		Ако израза върне истина ще се изпълни yes-instruction. Иначе ще се изпълни no-instruction.
израз										
истина	лъжа									
yes-instruction	No-instruction (или празно)									
инструкции										
Псевдо код (Java)	<pre>if (израз) yes-instruction; else no-instruction; Инструкция (и) ;</pre>	Семантиката е аналогична на box-диаграмите. От една страна else израза ще липсва. Израза трябва да върне булева стойност.								
Диаграми на потока на програмата		Семантиката е аналогична на псевдокода.								

Избор

няколко избора

изглед

описание

Вох диагр.	<div>/ Nassi, Shneiderman 73 /</div>	<div>/ DIN 66261 /</div>
---------------	--------------------------------------	--------------------------

<div>Псевдокод (Java)</div> <pre> switch (expression) { case constExpression1: Instruction(s); break; constExpression2: Instruction(s); break; ... default: Instruction } Instructions(s); </pre>	<div>Израза служи като избор на един от случаите. Ако не е избран подходящ случай инструкциите под 'default' се изпълняват.</div>
---	---

<div>Диаграми на потока в програмата</div>	<div>Няколко избора</div>
--	-------------------------------

Цикли: box диаграми

Повторение	изглед	
<div>description</div> <div>Box диаграми</div>	<div>израз</div> <div>Повтарящи се инструкции</div> <div>инструкции</div>	Повторението е с въпрос преди всяко повторение
	<div>Повтарящи се инструкции</div> <div>израз</div> <div>инструкции</div>	Повторение с въпрос след всяко повторение
	<div>for (start-exp, end-exp, increment)</div> <div>Повтарящи се инструкции</div> <div>инструкции</div>	Повторения с брояч на повторенията (брой на циклите)

Цикли: псевдокод, диаграми на потока на програмата

Псевдокод (Java)

```
while (expression)
{
    RepetitionInstructions;
}
Instruction(s);
```

Повторения с въпрос
преди всяко
повторение

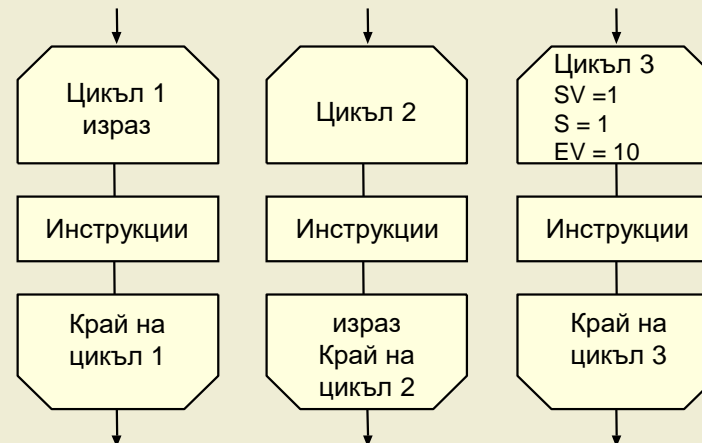
```
do
{
    RepetitionInstructions;
}
while (expression);
Instruction(s);
```

Повторения с въпрос
след всяко повторение

```
for(start-exp, end-exp, increment)
{
    RepetitionInstructions;
}
Instruction(s);
```

Повторения с брояч на
повторенията (брой на
циклите)

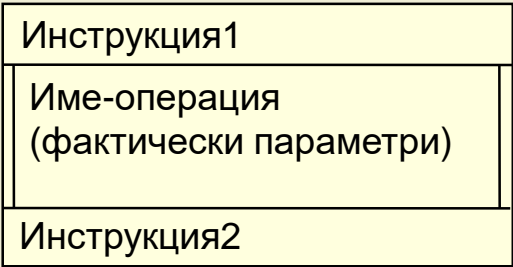
Диаграми на потока на програмата



аналогични

SV = начална
стойност
S = стъпка
EV = крайна
стойност

Извикване

извикване	изглед	описание
Вох диагр.		След изпълнението на извиканата операция, извикващата операция продължава от точката на извикване.
Псевдокод (Java)	<pre> Instruction1; OperationName(actual parameters); Instruction2; </pre>	Извикването става чрез индикация на името на операцията следвано от фактическите параметри
Диаграми на потока на програмата	