

# Цена на софтуера

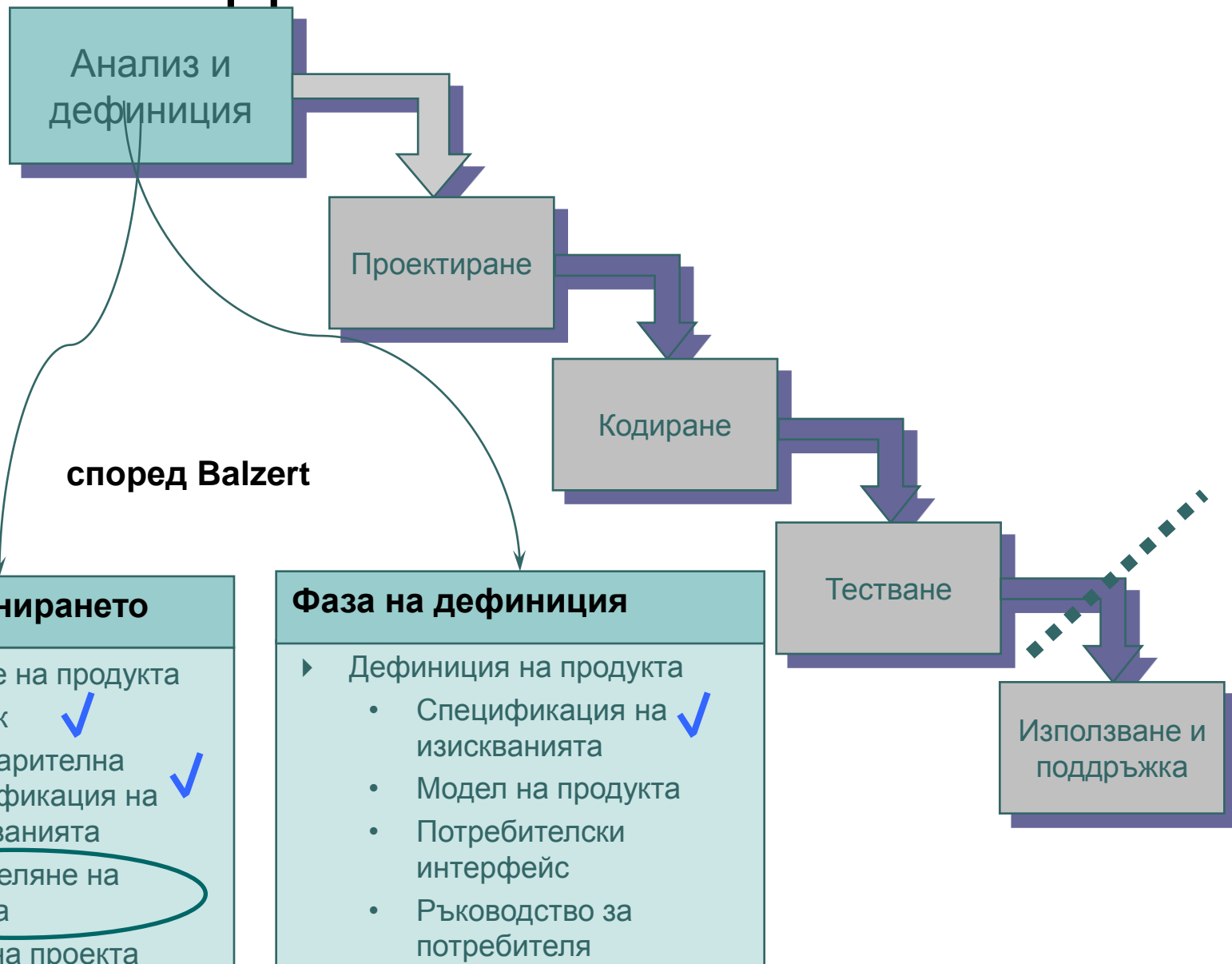
Лекция 4  
гл.ас. д-р Ася Стоянова-Дойчева



# Съдържание

- Въведение
- Критерии на моделите за оценка на цената на софтуер.
- Моделът на Боем – COSOMO
  - същност на модела
  - пример и следствия
  - усъвършенстване на модела
  - редове първичен код
- Метод на функционалните точки
  - Същност на модела
  - Процедура по пресмятане
  - Използване на модела
- Други модели

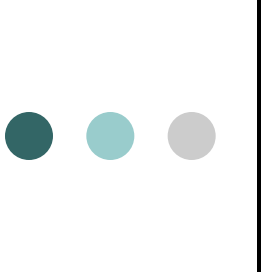
# Въведение





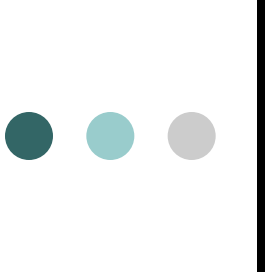
# Въведение

- Първият модел за оценка на цената на софтуер – SDC 1965г.
- 1981г. – Боем COSOMO.
- Процесът на намиране на цената помага също за:
  - Избор на проект за реализация
  - Определяне състава на колектива
  - Оценяване работата на членовете от колектива



# Критерии на моделите за установяване на цената според Боем

- Определеност
- Точност
- Обективност
- Детайлност
- Устойчивост



# Критерии на моделите за установяване на цената според Боем

- Област на приложение
- Конструктивност
- Простота на прилагане
- Предсказуемост
- Икономичност



# Моделът на Боем – COSOMO (Constructive Cost Model)

- Цел на COSOMO модела – за всеки планиран софтуерен проект да се оцени цената и срокът на разработване.
- Основната идея на Боем е използването на броя редове първичен код.



# Моделът на Боем - COCOMO

- Същност на модела – за изчисляване на усилията се използва формулата:

$$\text{ЧМ} = 2.4 \times \text{ХРПК}^{1.05}$$

ЧМ – брой човекомесеци

ХРПК – хиляди реда първичен код

- За оценка на продължителността на разработване на софтуерния проект формулата е:

$$В = 2.5 \times \text{ЧМ}^{0.38},$$

където В е срокът на разработване в месеци.





# Моделът на Боем - COCOMO

- Формулите се прилагат при следните предположения:
  - Редовете първичен код се броят без коментарните редове; принадлежат на крайният продукт; не включват използваните стандартни програми;
  - Включват се фазите проектиране, програмиране и оценка включително усилията по управлението и документирането;
  - Смята се че един човекомесец е от 19 дни или 152 часа;
  - Не се правят сериозни промени в спецификацията на изискванията след одобрението и;



# Моделът на Боем - COCOMO

- Пример:

Нека имаме резултат от предварителна експертиза, че бъдещ софтуерен продукт ще има 32000 реда код =>

$$ЧМ = 2.4 \times 32^{1.05} = 91 \text{ човекомесеца}$$

$$В = 2.5 \times 91^{0.38} = 14 \text{ месеца} \Rightarrow$$

$$\text{Производителността} = 32000 / 91 = 352$$

ЛОС за човекомесеца

$$\text{Екип} = 91 / 14 = 6.5 \text{ човека}$$



# Моделът на Боем - COCOMO

- Промяна на производителността според размера на проекта. Боем предлага следното разделяне:

Тип проект	ХРПК	ЧМ	В	Екип: брой	Производителност
Малък	2	5.0	4.6	1.1	400
Междинен	8	21.3	8.0	2.7	376
Среден	32	91.0	14.0	6.5	352
Голям	128	392.0	16.0	16.0	325



# Моделът на Боем - COCOMO

Първото разширение на модела на Боем е въвеждането на 3 типа софтуерни проекти:

Тип	Характеристика	Примери	Формула
Разпространен	Разработка се от малка група в познатите условия на собствената фирма	Прости системи за управление на запаси или производствени процеси	$ЧМ = 2.4 \times ХРПК^{1.05}$
Полунезависим	Има междинно положение между разпространен и вграден тип	Системи за обработка на съобщения, нови операционни системи	$ЧМ = 3.0 \times ХРПК^{1.12}$
Вграден	Софтуерът работи свързан с апаратура, друг софтуер и изчислителни процеси	Авиационни или радиоелектронни системи	$ЧМ = 3.6 \times ХРПК^{1.20}$



# Моделът на Боем - COCOMO

- Следващо усъвършенстване – редовете първичен код не биха могли да са единствения параметър на установеното уравнение. Преминава се към уравнение в което се отчитат и допълнителни фактори със съответни тегла.

$$ЧМ = k \times ХРПК^p \times A_1 \times A_2 \times \dots \times A_{15},$$

където  $k$  и  $p$  са коефициенти за различните типове софтуер

$A_i$  са теглата на атрибутите

# Моделът на Боем - COCOMO

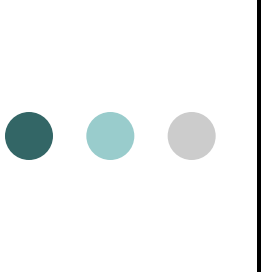
Оценявани атрибути	Много нисък	Нисък	Нормален	Висок	Много висок	Свръх-висок
<b>Атрибути на продукта</b>						
Изисквана надеждност	0.75	0.88	1.00	1.15	1.40	
Размер на базата данни		0.94	1.00	1.08	1.16	
Сложност на продукта	0.70	0.85	1.00	1.15	1.30	1.65
<b>Атрибути на компютъра</b>						
Ограничение по бързина			1.00	1.11	1.30	1.65
Ограничение по оперативна памет			1.00	1.06	1.21	1.65
Изменяемост на виртуалната машина		0.87	1.00	1.15	1.30	
Цикъл на обръщение към компютъра		0.87	1.00	1.07	1.15	
<b>Атрибути на изпълнителя</b>						
Квалификация на проектантите	1.46	1.19	1.00	0.86	0.71	
Опит в приложната област	1.29	1.13	1.00	0.91	0.82	
Квалификация на програмистите	1.42	1.17	1.00	0.86	0.70	
Опит от работа с компютъра	1.21	1.10	1.00	0.90		
Опит с езика за програмиране	1.14	1.07	1.00	0.95		
<b>Атрибути на проекта</b>						
Прилагане на съвременни методи	1.24	1.10	1.00	0.91	0.82	
Ползване инструментални средства	1.24	1.10	1.00	0.91	0.83	
Ограничения в срока за разработка	1.23	1.08	1.00	1.04	1.10	



# Редове първичен код

## Какво е ред първичен код??????

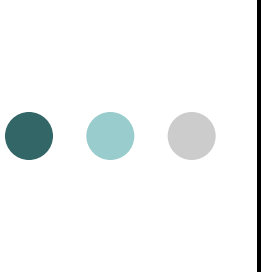
- Физически или логически редове??
- От семантична гледна точка редовете са няколко типа:
  - Изпълними оператори
  - Определения на данни
  - Коментари
  - Празни редове
- Повторно използваемият код.
- Приложения писани на повече от един език за програмиране
- Други – временни редове код, макроси, стил на програмиране и др.



# Метод на функционалните точки

- Предложен е от Олбрихт – 1979г.
- Основа на модела е понятието “функционална точка”. Идеята му е, че усилията в разработката на даден софтуер (следователно и цената) се определят от неговата функционалност, която се измерва с горе споменатите “функционални точки”.





# Метод на функционалните точки

- Цели на метода:
  - Да се използват външните характеристики на софтуера;
  - Да може да се прилага в ранни фази на производствения процес;
  - Да може да се свърже лесно с икономическа оценка;
  - Да има независимост от редовете първичен код.



# Същност на модела

- Основава се на 5 функционални типа. Допълнително се определят по 3 нива на сложност за всеки тип – просто, средно и сложно. Тези типове са следните:



# Същност на модела

- Външен входен тип – всеки потребителски входен управляващ поток или поток от данни. Този тип може да бъде:
  - прост – съдържа малък брой типове данни и те се отразяват върху малък брой вътрешни логически типове;
  - междинен – който не може да се определи еднозначно като прост или сложен;
  - сложен – съдържа значителен брой типове данни и тези данни се отразяват върху голям брой вътрешни логически данни

Пример – входните екрани. Въвежданите чрез тях данни “навлизат” в приложението и засягат един или повече вътрешни логически типа.



# Същност на модела

- Външен изходен тип - всеки потребителски изходен управляващ поток или поток от данни. Примерни такива могат да бъдат различните видове съобщения до потребителя или изходни отчети. Класификацията е като при предходния тип. Допълнително отчетите се класифицират на:
  - прост – съдържа една или две колони, данните почти не се преобразуват при извеждането им;
  - междинен – отчетът съдържа много колони;
  - сложен – резултатите в отчета се получават след сложни преобразования.



# Същност на модела

- Вътрешен логически файлов тип – такава е всяка съществена група от потребителски данни или управляваща информация. Пример са логическите файлове и въобще всяка логическа група от данни. Класифицира се на три нива:
  - прост – малко типове записи, малък брой типове елементи от данни;
  - междинен – типът не може еднозначно да се определи като прост или сложен;
  - сложен – голям брой типове записи, типове елементи от данни.



# Същност на модела

- Външен интерфейсен файлов тип – това е файл, който се предава или се ползва съвместно от две или повече приложения.  
Класификацията се прави според дефиницията за вътрешния логически файлов тип.



# Същност на модела

- Външен справочен тип – всяка комбинация вход/изход, при която входът предизвиква незабавен изходен резултат. Като пример това са заявките. Класификацията се определя така:
  - входната част се класифицира според определенията за външния входен тип;
  - изходната част се класифицира според определенията за външния изходен тип;
  - сложността на външния справочен тип е равна на по-голямата от така определените две сложности.



# Процедура по пресмятане

- Преброяват се елементите на всеки от 5-те типа.
- След това за всеки елемент от всеки функционален тип се определя нивото на сложност.
- С получените данни се попълва таблицата за пресмятане.





# Таблица за пресмятане

Тип	Наименование	прост	междинен	сложен	Общо
IT	Външен входен	X3	X4	X6	...
OT	Външен изходен	X4	X5	X7	...
FT	Вътрешен логически файлове	X7	X10	X15	...
EI	Външен интерфейсен файлове	X5	X7	X10	...
QT	Външен справочен	X3	X4	X6	...
FC	Общо ненастроени функционални точки				...



# Процедура по пресмятане

- Разглеждат се 14 характеристики и се определя степента им на влияние върху приложението. Допустими стойности за всяка от характеристиките:
  - 0 – не е налична или не влияе
  - 1 – незначително влияние
  - 2 – умерено влияние
  - 3 – средно влияние
  - 4 – значително влияние
  - 5 – силно влияние



# Характеристики оказващи влияние на приложението

1. Данните и управляващата информация се изпращат или получават по комуникационни линии;
2. Има разпределена обработка на данни;
3. Важно е достигането на висока ефективност;
4. Експлоатация върху силно натоварена операционна конфигурация – хардуер, софтуер
5. Интензивността на транзакциите е висока;
6. Наличен е интерактивен режим на въвеждане на данните;
7. Цели се ефективност от гледище на потребителя;
8. Наличен е интерактивен режим на актуализирането на данните;
9. Логиката на обработките е сложна;
10. Програмният код трябва да е reusable;
11. Цели се лесно инсталиране;
12. Цели се лесна експлоатация;
13. Може да се използва от разнообразни потребители;
14. Приложението е гъвкаво и лесно се модифицира.

# Процедура по пресмятане

- Определя се величината PC (processing complexity) като сума от стойностите на 14-те характеристики.
- На базата на PC се определя коригиращия коефициент PCA (processing complexity adjustment).

$$PCA = 0.65 + (0.01 \times PC)$$

- Крайният брой FP (функционални точки) се получава по формулата:

$$FP = FC \times PCA$$

Определяне на усилията

$$Effort = 0.6 FP + 0.001FP^2$$



## Други модели

- Doty – наподобява СОСОМО.

Основава се на 2 формули:

$$\text{ЧМ} = 5.288 \times \text{ХРПК}^{1.047} \text{ за } \text{ХРПК} < 10$$

$$\text{ЧМ} = 2.060 \times \text{ХРПК}^{1.047} \times f_1 \times f_2 \times \dots \times f_{14} \\ \text{за } \text{ХРПК} \geq 10$$

Означенията са както при СОСОМО.



## Други модели

- SPQR – подобен на COCOMO – Кейпърс Джоунс Основава се на 20 добре дефинирани и 25 недостатъчно добре дефинирани фактора, определящи цената на софтуера.
- Не е добре документиран.
- Потребителят трябва да отговори на 100 въпроса свързани с проекта, чиито отговори служат за входни параметри. Създателят твърди че метода дава +/- 15% отклонение.



# Други модели

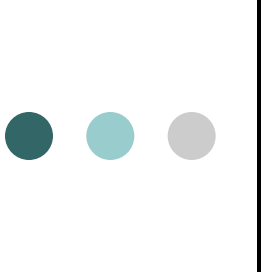
- ESTIMACS – съсредоточава се върху разработването – Рубин – 15% отклонение. Състои се от следните модули:
  - Оценител на усилията за разработване на системата – 25 въпроса за проекта;
  - Оценител на разходите за колектива разработчик – входни данни са получените от предната точка;
  - Оценител на хардуерната конфигурация – като вход се подават изискванията на приложението за ОС, очаквания брой и тип на транзакциите, а като резултат се получава оценка за необходимата хардуерна конфигурация;
  - Оценител на риска. – отговор на 60 въпроса.



## Други модели

- BANG – Де Марко. Принцип на функционалните точки. Цел – системен софтуер и научни приложения. Основните елементи които се оценяват и броят са:
  - Функционални примитиви
  - Модифицирани функционални примитиви
  - Елементи от данни
  - Входни елементи от данни
  - Изходни елементи от данни
  - Елементи от данни в паметта
  - Обекти
  - Връзки
  - Фактори свързани със спецификата на системата в реално време – преходи, състояния на чакане.





# Оценяване при обектна ориентираност

- Модели свързани с броя редове първичен код не могат да се прилагат.
- Моделът на функционалните точки е приемливо приложим.
- Моделът MOOSE (Metric for object-oriented system environment)
- Предложение на Боем – COSOMO 2.0 (обектни точки)