




Цена на софтуера

Оценяване на ресурсите,
необходими за
разработката на софтуер



Теми

- Продуктивност
- Техники на оценяване
- Algorithmic cost modelling
- Продължителност на проекта и подбор на персонала



Основни въпроси при оценяване

- Какви усилия са необходими за завършване на определена дейност?
- Какво време и необходимо за завършване на определена дейност?
- Какви са цялостните разходи за дейността?
- Обща оценка и разписание на проекта
- Определяне на дейностите по управлението

Компоненти на цената на софтуера

- Разходи за хардуер и софтуер
- Разходи за пътувания и обучение
- Разходи за усилията за създаване на софтуер (основен фактор в повечето проекти)
 - Заплати на специалистите от проекта
 - Поддържащ персонал
 - Социални разходи, здравни и пенсионни застраховки
 - Разходи за сгради, осветление, отопление
 - Мрежова среда и комуникации
 - Разходи за общи услуги (библиотеки, ресторант за екипа...)



Разходи и оценки

- Оценката се прави за определяне на цената, която трябва да плати разработчика за разработвания продукт
- Връзката между разходите за разработка и цената за клиента не е еднопосочна
- Широк спектър от организационни, икономически, политически и бизнес съображения влияят върху определянето на цената

Фактори, влияещи върху оценката

Състояние на пазара	Предлагане на по-ниска цена с цел развиване на нов сегмент от пазара
Съмнение в реалността на оценката	Завишаване на цената за осигуровка срещу непредвидени случаи
Особености на контракта	Клиентът е съгласен сорс кодът да остане за разработчика с цел reuse при други проекти – цената се намалява
Неуточненост на изискванията	Ако се очаква изменение в изискванията може да се предложи по-ниска цена, за да се спечели контракта. При промяна в изискванията се променя и цената
Финансови причини	При финансови затруднения могат да намалят цената, за да вземат договора

Продуктивност на програмистите

- Мярка за степента на участие на отделните програмисти в разработката на софтуера и съпътстващата го документация
- Не е качествено ориентирана, макар че гарантирането на качеството е фактор в оценката на продуктивността
- Оценка на полезната функционалност, произведена за единица време

Мерки за продуктивността

- Мерки (мерни единици), свързани с размера на кода, като резултат (изход) от софтуерния процес - редове сорс код, инструкции обектен код и др.
- Мерки, свързани с функционалността на разработения софтуер.
 - Функционалните точки са най-доброто при този тип измерване

Проблеми на измерването

- Пресмятане на размера (обема) на измерването
- Оценка на общия брой месеци за програмиране
- Оценка на продуктивността на доставчика (например екипа по документирането) и вмъкване (отчитане) на тази оценка в общата цена

Редове код

- Какво е ред код?
 - Свързана с началните периоди на програмирането, когато програмите се въвеждат на ПК (1 ред на 1 ПК)
 - Как това съответства на командите на Java, които могат да се простират върху няколко реда или където няколко команди могат да се запишат на 1 ред?
- Кой програми трябва да се разглеждат като част от системата?
- Предполага линейна зависимост между обема на системата и документацията

Сравнение на производителността

- При по-ниско ниво на езика се изисква по-голяма продуктивност от програмиста
 - ▣ При езици от по-ниско ниво се произвежда по-голямо количество код, отколкото при езици от по-високо ниво
- Програмисти, създаващи по-многословен код са по-продуктивни?
 - ▣ Мерките за продуктивност, базирани на редове код предполагат, че програмистите, които пишат по-многословен код са по-продуктивни, от тези, които пишат компактен код

High and low level languages

Low-level language



High-level language



Метод на функционалните ТОЧКИ

- МФТ – усилията за разработка на софтуер и => цената му се определят от неговата функционалност, която се измерва на базата на функционални точки
- Разработен върху информационни системи
- Цели на метода:
 - Използване на външни характеристики на софтуера
 - Третиране на средства, важни за потребителя
 - Прилагане в ранните фази
 - Независим от редовете първичен код
 - Свързване с икономически оценки

5 функционални типа,

3 нива на сложност

- Външен входен тип
 - входен управляващ поток или поток от данни (входни екрани)
- Външен изходен тип
 - изходен управляващ поток или поток от данни (съобщения за потребителя или изходни отчети)
- Вътрешен логически файлов тип
 - Потребителски данни или управляваща информация, която се генерира или използва от приложението
- Външен интерфейсен файлов тип
 - Файл, който се предава или използва от 2 или повече п-ния
- Външен справочен тип
 - комбинация вход/изход – входът предизвиква изход (заявка и отговорът ѝ в ИС)

Метод на функционалните ТОЧКИ

- Базиран на комбинация от програмни характеристики
 - Външен вход и изход
 - Взаимодействие с потребителя
 - Външен интерфейс
 - Файлове, използвани от системата
- Според нивото на сложност към всеки от горните елементи се асоциира определено тегло
- Общата оценка се получава чрез сумиране на отделните елементи, умножени по съответните тегла
$$\text{БФТ} = \sum (\text{брой елементи от даден тип}) * (\text{тегло})$$

Функционални точки

- Функционалните точки могат да се модифицират според сложността на проекта
 - $БФТ = БФТ * КК$ (КК – коригиращ коефициент)
- FPs могат да се използват за преценка на LOC (Lines Of Code) в зависимост от средния брой LOC за FP за даден език
 - $LOC = AVC * \text{брой на функционалните точки}$
 - AVC is a language-dependent factor varying from 200-300 for assemble language to 2-40 for a 4GL
- FPs са много субективни - зависят от оценителя.
 - Не могат да се пресметнат автоматично

Обектни точки

- Обектните точки са алтернативна, функционално-обвързана метрика на функционалните точки при 4GL или подобни езици, използвани за разработка
- Обектните точки НЕ са обектни класове
- Броят на обектните точки в една програма се определя от :
 - Брой различни екрани, които се визуализират (1,2,3)
 - Брой на отчетите, създадени от с-та (2,5,8)
 - Брой на 3GL модули, които трябва да се разработят, за да допълнят 4GL код
- Всяка обектна точка се класифицира като: обикновена, средна, трудна

Сравнение

- Обектните точки се изчисляват по-лесно от спецификациите, отколкото функционалните точки, тъй като се отнасят до екранни форми, отчети и 3GL модули
- Обектните точки могат да бъдат преценени на ранните етапи от процеса на разработка. Тогава е трудно да се прецени броя на редовете код
- Use cases ???

Оценка на продуктивността

- Зависи от редица фактори
 - Опит в приложната област
 - Качество на процеса
 - Размер на проекта
 - Прилагане на технологии
 - Работна среда
- Индивидуалните умения – най-значим фактор
 - Продуктивността на 1 програмист може да е няколко пъти по-голяма от тази на друг

Фактори, влияещи в/у производителността

Опит в приложната област	Знанията за приложната област са най-важни за продуктивността. Тези, които познават областта са по-продуктивни
Качество на процеса	Процесът на разработка оказва значително влияние върху производителността
Размер на проекта	По-големите проекти изискват повече време
Прилагане на технологии	Прилагането на съвременни технологии(CASE средства) може да подобри производителността
Работна среда	Спокойната работна среда със самостоятелни работни места позволява подобряване на производителността

Оценка на продуктивността

- Системи за работа в реално време - 40-160 LOC/P-month
- Системни програми , 150-400 LOC/P-month
- Комерсиални приложения, 200-800 LOC/P-month
- В обектни точки, продуктивността се измерва между 4 и 50 обектни точки/месец в зависимост от туловете и възможностите на разработчика

Качество и продуктивност

- Всички метрики, базирани върху обем/единици са несъвършени, защото не вземат предвид качеството
- Продуктивността може да бъде увеличена за сметка на качеството
- Не е изяснена напълно връзката между метриците за качеството и производителността
- Ако непрекъснато се правят промени, подходът, базиран върху брой редове на кода не е съдържателен (подходящ)

Техники за оценяване

- Не съществува лесен начин за намиране на точна оценка на необходимите усилия за разработка на софтуерна система
 - ▣ Първоначалните изчисления се базират на неадекватна информация в термините на потребителските изисквания
 - ▣ Софтуерът може да се разработва на непознати компютри или да използва нови технологии
 - ▣ Участниците в проекта могат да бъдат непознати
- Оценката на проекта може да е водеща



Техники за оценяване

- Algorithmic cost modelling
- Експертна оценка
- Оценка по аналогия
- Закон на Паркинсон
- Pricing to win – Оценка с цел победа
- Top-down и bottom-up подходи

Top-down & bottom-up оценяване

- Всеки от посочените подходи може да се прилага в 2 посоки - top-down или bottom-up
- Top-down
 - Започва на ниво система и детайлно разработва осигуряването на цялата функционалност на системата и как тя се осигурява чрез подсистемите
- Bottom-up
 - Започва се на ниво компонент и се оценяват индивидуално усилията за всеки компонент. Получените усилия се сумират за крайната оценка

Методи за оценка

- Всеки метод има предимства и недостатъци
- Добре е оценката да се прави по няколко метода
- Ако различните методи не дават приблизително един и същ резултат - първоначалната информация не е достатъчна
- Необходимо е да се предприемат допълнителни действия за получаване на по-точно оценка

Алгоритмична оценка

Най-систематичният подход за оценка на софтуера е алгоритмичната оценка.

Повечето алгоритмични модели имат експоненциална компонента, което означава, че цената не е линейна функция на размера на проекта.

С увеличение на обема на проекта разходите нарастват поради необходимостта от по-голям екип, по-сложна организация на управлението и др.

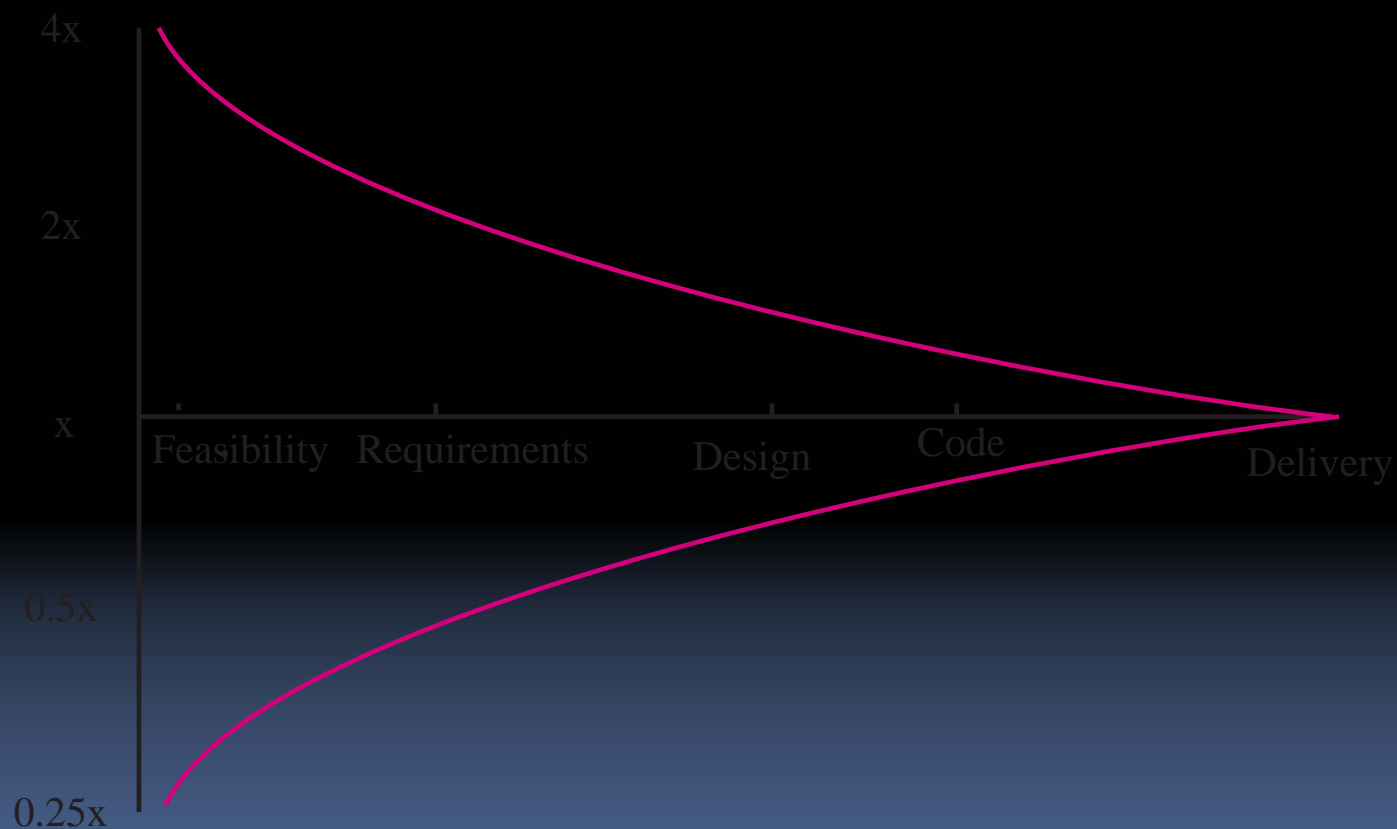
Algorithmic cost modelling

- Стойността (разходите) се пресмятат като функция на атрибути на продукта, проекта и процеса, чиито стойности се оценяват от ръководителя на проекта
 - $\text{Effort} = A \times \text{Size}^B \times M$
 - A е константа, свързана с организационните аспекти, B отразява допълнителните усилия при големи проекти и M е множител, отразяващ атрибути на продукта, процеса и хората,
 - Най-често използваният атрибут при тези оценки е размерът на кода
 - Повечето модели предлагат подобен начин на оценяване при различни стойности на A , B и M

Точност на оценката

- Точният размер на системата се знае след завършването ѝ
- Фактори, които влияят върху крайния размер
 - Използване на COTS и компоненти
 - Език за програмиране
 - Разпределението на системата
- С развитието на процеса на разработка оценката на обема на кода става по-точна

Променливост на оценката



The COCOMO model

- Един от най-известните модели за оценка на софтуера от този тип е **COCOMO model** – (constructive cost model)
- Първоначалната идея е свързана с редовете сорс код
- Дълга история от първата версия от 1981 (COCOMO-81) до COCOMO 2
- COCOMO 2 взема под внимание различни подходи за разработка на софтуер, reuse и др.

COCOMO 81

- Разработен на основата на повече от 60 различни проекта
- Формата на COCOMO модела следва формулата
 - $\text{Effort} = A \times \text{Size}^B \times M$
- като предлага различни експоненти в зависимост от сложността на проекта.
- Усъвършенстване на модела чрез въвеждане на 3 нива на подробност и 3 типа разработки

СОСОМО 81 - Нива на подробност

- **Basic** (базов, основен) – оценка, базирана на атрибутите на продукта
- **Intermediate** (междинен) – модификация на основния модел чрез оценяване атрибути на проекта и процеса
- **Advanced** (детайлен) – оценява отделно фазите и частите на проекта

Класове проекти

- *Simple (organic) mode* (разпространен)-малък екип, позната среда, добре дефинирано приложение, без трудни не-функционални изисквания
 - *с-ми за управление на производствени процеси*
- *Moderate (semi-detached) mode* (полунезависим) - екип с различен опит, с-та може да има по-значителни не-функционални изисквания, организацията може да не е добре запозната с приложението
 - *с-ми за обработка на съобщения, нови операционни системи*
- *Embedded* (вграден)- хардуерно/софтуерни системи, силни ограничения, необичайни за екипа умения (HARD)
 - *авиационни системи*

Оценка на времето

Моделът COSOMO включва и формула за оценка на календарното време (Т) необходимо за завършване на проекта. Това зависи от типа на проекта:

- Organic: $TDEV = 2.5 (PM)^{0.38}$
- Semi-detached: $TDEV = 2.5 (PM)^{0.35}$
- Embedded mode: $TDEV = 2.5 (PM)^{0.32}$
- Personnel requirement: $N = PM/TDEV$

Примери

- Разпространен (Organic), 32KLOC
 - $PM = 2.4 (32)^{1.05} = 91$ person months – човеко/месеца
 - $TDEV = 2.5 (91)^{0.38} = 14$ months – време за разработка
 - $N = 91/15 = 6.5$ people – брой разработчици
- Вграден (Embedded), 128KLOC
 - $PM = 3.6 (128)^{1.2} = 1216$ person-months
 - $TDEV = 2.5 (1216)^{0.32} = 24$ months
 - $N = 1216/24 = 51$

COSOMO предположения

- Неявна оценка на продуктивността
 - Organic mode = 16 LOC/day
 - Embedded mode = 4 LOC/day
- Включва се само кода на крайния продукт
- Не се броят коментарни редове
- Не се включват стандартни програми
- Времето е функция на цялостните усилия, НЕ на големината на екипа
- Не се отчита квалификацията на екипа
- Само фазите програмиране, проектиране и оценка

Усъвършенстване на модела

- В **основния** модел $M=1$, т.е. оценката е свързана основно с размера на софтуерния проект.
- **Междинният модел (intermediate)** отчита такива фактори като надеждност на продукта, размера на БД, ограничения на паметта и времето за изпълнение, характеристики на персонала, използване на software tools (помощни средства).
 - ▣ Коефициентът M заема стойности от 0.7 до 1.66.

Personnel attributes

- Personnel attributes
 - Analyst capability
 - Virtual machine experience
 - Programmer capability
 - Programming language experience
 - Application experience
- Product attributes
 - Reliability requirement
 - Database size
 - Product complexity



Computer attributes

- Computer attributes

- Execution time constraints
- Storage constraints
- Virtual machine volatility
- Computer turnaround time

- Project attributes

- Modern programming practices
- Software tools
- Required development schedule

Пример

Сложна компютърна с-ма върху микропроцесорна хардуерна платформа.

- Основен COCOMO 81
 - 45 person-month effort requirement
- Междинен COCOMO 81
 - Attributes = RELY (1.15), STOR (1.21), TIME (1.10), TOOL (1.10)
 - $45 * 1.15 * 1.21 * 1.10 * 1.10 = 76$ person-months.
 - Total cost = $76 * \$7000 = \$532,000$

COCOMO 2

- COCOMO 2 е модел на 3 нива, който позволява доуточняване на оценките по време на развитие на проекта
 - Етап на ранно прототипиране
 - Оценка на основата на обектни точки ; проста формула за оценка на усилията
 - Етап на ранно проектиране
 - Оценка на основата на функционални точки, преобразуване в LOC
 - Post-architecture етап
 - Оценка на основата на редове сорс код



Изисквания към екипа

- Екипът не може да се определи чрез просто разделяне на времето на и разписанието
- Броят на хората, които работят по проекта зависи от конкретната фаза
- Повече участници в проекта изискват повече усилия за разработка

Rayleigh manpower curves

