1. Качество на софтуeра. Модели на качеството. Софтуерни метрики.

**Измерването** е процес, при който в съответствие с определени правила на характеристики на изследвания обект се съпоставят стойности.

Ако тези стойности са числови, измерването се нарича **количествено**. **Мярка** е числото, съпоставено при количествено измерване. **Метрика** -1)Друго име за мярката;2)Процедура за намиране и използване на мярката.

Една метрика се нарича **адитивна,** ако резултатите от прилагането й за програмна система могат да се получат чрез сумиране на резултатите от прилагането й върху съставящите я програмни части

Измервайки атрибута, ние получаваме стойност, която може да е в различни скали. Типът на скалата се определя в зависимост от разрешените преобразувания и статистики, които можем да използваме при анализа на метриките. Скалата може да бъде:

* Номинална – изброими множества от категории – стойности, които се присвояват, без да се вземат предвид количествените отношения. Не съществуват формули за превъщане на мерки от една номинална скала в мерки от друга номинална скала. Допустими статистики: мода, медиана, перцентили
* Порядкова – наредено множество от категории, позволяващо подреждане на стойности на оценъчните елементи чрез сравняването им с определени гранични стойност. Преобразуването: трябва да запазва частичната наредба. Допустими статистики: мода, медиана, перцентили, корелации на Спиърмън.
* Интервална – числови стойности, като разликата между всеки две последователни е една и съща. Преобразуване: ах+b. Статистики: мода, медиана, перцентили, корелации на Спиърмън, средно, стандартно отклонение, корелации на Пиърсън.
* Относителна – същата като интервална, но има ясно дефинирана нула. Преобразуване: с умножение по константа. Всички статистики са допустими
* Абсолютна – стойностите са числови и се получават чрез директно преброяване или изчисление. Всички статистики са допустими.

Измерването и получаваните мерки трябва да притежават следните свойства:

**Обективност** — получаваните мерки да не зависят от субекта, извършващ измерването.

**Надежност** — мярката да има необходимата различаваща способност и при повтаряне на измерванията при еднакви условия да се получават еднакви резултати.

**Валидност** — получаваните мерки да отразяват реално свойствата на измервания обект.

Предпоставки за прилагането на дадена софтуерна метрика:

а) можем точно да измерваме някакво свойство на обект в софтуерното производство;

б) съществува връзка между това, което можем да измерим, и това, което бихме искали да знаем за съответното свойство;

в) тази връзка е осъзната, потвърдена и може да бъде изразена чрез формула или в термините на съответен модел.

Следователно основните проблеми при създаване на софтуерни метрики са два:

* да се индентифицират съществени за изследване характеристики на софтуерни обекти и те да се дефинират в изчислими термини;
* да се определят типът на използваната скала, допустимите операции върху метриките, статистическият смисъл на оценяването и ограничаването на грешките при измерване.

Индикатор – за да се означат метричните данни за променящото се състояние на софтуерна разработка или на процес.

Софтуерното измерване обхваща слледните дейности:1. формулиране на метрична система.2. събиране на данни. 3. Анализиране. 4. Интерпретиране. 5. Осъществяване на обратна връзка

Прилагането на метрики в реални проекти е ограничено, т.к.:

* участниците в софтуерните разработки нямат достатъчно добра теоритическа подготовка
* немислимо е използването на метрики в реални сложни софтуерни проекти, без да са налице инструментални средства, автоматизиращи процедурите на измерване

А. Класификация в зависимост от **предназначението**

а) за оценяване на разходите и усилията за разработване на софтуера;

б) за измерване на производителността на разработчиците;

в) за моделиране на качеството на софтуера и оценяването му;

г) за измерване и прогнозиране на надежността на създаваните софтуерни системи;

д) изследване на някои програмни свойства.

Б. Класификация в зависимост от **целта на прилагане на метриката**

а) за характеризиране (to characterize) — да разберем изследвания обект и да установим база за сравнение при бъдещи оценки;

б) за оценяване— регистриране на текущо състояние на изследвания обект;

в) за прогнозиране — предварителна оценка за бъдещо състояние на изследвания обект;

г) за подобряване — натрупване на количествена информация за определяне на факторите, от които зависи качеството на продукта или ефективността на процеса.

В. Класификация в зависимост от **типа на изследвания обект**

а) процеси — дейности, свързани със създаването и използването на софтуера;

б) продукти — резултати от процесите;

в) ресурси — елементи, входни данни за процесите.

Г. Класификация в зависимост от начина **на получаване на информацията, въз основа на която се определят стойностите на измерваните характеристики:**

а) регистрационен метод — получаване на информацията по време на разработване или функциониране на ПП чрез регистриране на отделни събития

б) измерителен метод — информацията се получава чрез прилагане на специално разработени инструментални средства;

в) възприятиен (органолептичен)метод — информация, получена след анализ на зрителни или слухови възприятия

г) изчислителен метод — използване на теоретични или емпирични зависимости, изведени въз основа на статистическите данни, натрупани през различни фази на жизнения цикъл на ПП.

Метрики изследващи програми:

а)в зависимост от начина на изследване — статични(само текстът на програмата) и динамични(изпълнението й).

б)в зависимост от изследваната програмна характеристика

в)в зависимост от нивото на декомпозиране

Примери за софтуерни метрики:

**1. Метрика за размера на програма** –прилага се за програмен модул и за програмна система.

За мярка на размера на програма се избира броят програмни редове в изходния текст на програмата.

L = L1 + L2 (L-общ брой редове;L1 -коментарни и празни редове;L2-същински програмни Приложимост

1. Броят на грешките, усилията за разбиране, поддържане и съпровождане са правопропорционални на размера на програмите.
2. В зависимост от размера (на ниво програмна единица) и на размера и броя модули в програмна система програмите могат да се класифицират като прости, средно сложни, сложни и свръхсложни
3. . L е най-проста мярка за извършената от програмиста работа.

**2. Метрика на Маккейб за структурна (цикломатична) сложност**( програми).Измерва сложността на потока на управление в програмата. Трябва да се построи управляващият граф на програмата, в който върховете са отделните оператори и два върха са свързани с ребро, ако има предаване на управление между съответните оператори. V(G) = е — n + 2 \* р,V(G) е цикломатичната сложност

е -брой ребра;n -брой върхове в управляващия граф;р -брой на свързаните компоненти в графа.

Приложимост

Метриката предлага мярка за вътрешномодулната сложност, която определя леснотата на разбиране и усвояване на програмите и точността на внасяне на изменения в тях.

**3. Метрика на Холстед за текстуална сложност**(за програма, на произволен език за програмиране.

Програмата се разглежда като състояща се от активни елементи (оператори - константи или променливи) и пасивни елементи (операнди).

n1-брой различни оператори;n2-брой различни операнди;N1-общ брой оператори;N2-общ брой операнди.

Чрез тези оценъчни елементи се изчисляват:

Речник: n = n1 + n2Дължина на програмата: N = N1 + N2

Изчислена дължина на програмата:N = n1\* log2 n1 + n2 log2 n2

Обем на програмата: V = (N1 + N2) \* log2 (n1+ n2);потенциален обем: V\*=(2+n2\*)log2(2+n2\*)

Ниво на релийза: L=V\*/V;Усилия за създаване на програмата: Е = V/L

Приложимост

1. Могат да се оценяват усилията за създаване на програмата.

2. Може да се оцени времето за програмиране

3. Усилията за създаване на нова програма са сравними с усилията за разбиране на съществуваща.

4.Предлагат се формули за предварително оценяване на броя на откриваните грешки:

5. Възможността усилията за тестване да се оценяват

**M4. Метрика на Рехенберг за технологична сложност**(за програми или програмни системи, написани на произволен език за програмиране от високо ниво.

СС = SC + EC + DC,(СС е комплексната сложност;SC-операторна сложност;ЕС- сложност на използваните изрази;DC-сложност на данните.)

Въведени са и относителни мерки за сложност(дават възможност да се сравняват програми с различна дължина.). Ако NS е броят на операторите в програмата:

RCC=CC/NS; RSC=SC/NS; REC=EC/NS; RDC = DC / NS

Приложимост -за прогнозиране на усилията за тестване и съпровождане чрез сравняване на мерките на новоразработвана софтуерна система със съществуваща, за която тези усилия са документирани.

**5.Метрика за информационния поток –** мярка за междумодулна сложност на първичния код. Определя се от информационния поток от и към даден модул.

Оценъчни елементи: ini – мярка за входящия информационен поток; outi – мярка за изходящия информационен поток; weighti – теглови коефициенти

HKi=weighti\*(ini\*outi)

Приложимост: за оценяване на междумодулната информационна сложност и да се оценят усилията за съпровождане и за повторно използване на отделните модули и на изследваната програмна система като цяло

**6. Метрика за четимост(**обект на изследване на различни видове документи**)**

F = 0.4 \* (lm + plw ), lm е средна дължина на изречение;plw — относителен дял на дългите думи

брой дълги думи   
plw = \* 100

общ брой думи.

Приложимост - Мярката на четимост F отразява трудността на текста.

**Обектно-ориентирани метрики:**

1. Метрики за ОО програмиране

* Среден брой методи за клас = общ брой методи / общ брой класове за обект
* Средна сложност на метод = сума от цикломатичната сложност на всички методи/ общ брой на методите в приложението

1. Метрики на проектирането: съвкупността MOOD включва:

-Method Hiding Factor(MHF) – частно на сумата от скритостите на всички методи, дефинирани във всички класове и общия брой методи, дефинирани в изследваната програмна система

-Attribute Hiding Factor(AHF) – частно на сумата от скритостите на всички атрибути, дефинирани във всички класове и общия брой методи, дефинирани в изследваната програмна система

-Method Inheritance Factor(MIF) – частно на сумата на наследените методи във всички класове на изследваната система и общия брой налични методи за всички класове

-Attribute Inheritance Factor(AIF) – частно на сумата на наследените атрибути във всички класове на изследваната система и общия брой налични атрибути за всички класове

-Polimorphism Factor(PF) – частно на реалния брой възможни различни полиморфни ситуации за даден клас и максималния брой на възможните различни полиморфни ситуации за същия клас.

-Coupling Factor(CF) – частно на максималния възможен брой на всързванията в системата и реалния брой на свързвания, които не са резултат от наследяване

3) Класически метрики, адаптирани към ОО разработка: Метриките за цикломатична сложност, брой на програмните редове и ниво на коментируемост. Чрез метриките за дълбочина на дървото на наследяване и брой на наследниците в това дърво могат да се изледват характеристиката „наследяване ” в конкретната програмна реализация на ОО приложение.

**Оценяване качеството на софтуера**:

**Качеството** е съвкупността от средства и характеристики на даден продукт или услуга, носители на способността му да отговори на явно или неявно указани нужди.

Модели на качеството на софтуера:

**Модел на Боем:** Моделът на качеството на софтуера на Боем има йерархичен характер, но е сравнително слабо структуриран със своите реални две нива. Качеството на софтуера Боем свързва преди всичко с неговата полезност(зависи от надеждност, ефективност и използваемост от гледна точка на потребителя човек) и възможност за лесно съпровождане(зависи от тестируемост, разбираемост и модифицируемост). Освен това има още една, несвързана с двата аспекта характеристика, наречена портабелност (мобилност). Оценяване на всяко свойство за конкретния програмен продукт чрез някаква обективна мярка - метрика.Недостатъци:не съвсем ясната структурираност, недостатъчната пълнота на множеството от характеристики, съсредоточаване почти изключително върху качеството на програмния код, сравнително тясната експериментална база.

Типичен йерархичен модел:

Проектът 3 основни задачи:

— развиване и утвърждаване на резултатите от предходни проекти на подобна тематика;

* разширяване на рамката на разглеждане на програмния продукт като такъв;
* разработване на методология за определяне и специфициране на изисквания към факторите, определящи качеството на софтуера

Структура:Качеството се разглежда като йерархична структура. То се намира н  
най-високото ниво — 0.

На следващото ниво 1 се намират факторите. Факторът се определя като потребителски ориентирано свойство, представящо даден аспект на качеството на софтуера от гледище на потребителя.

В зависимост от конкретния модел факторите могат да бъдат от 6 до 16.

* Гъвкавост - Лекота на адаптиране към нови функционални условия,
* Коректност - Степен на съответствие на специфицираните алгоритми и други изисквания спрямо обработката на данни,
* Надеждност - Способност на програмния продукт да изпълнява зададените функции, предвидени в програмната документация,
* Съпровождаемост - Възможност за отстраняване на отклонения и грешки в процеса на експлоатация на програмния продукт и за поддържането му в актуално състояние. ,
* удобство на използване - Възможност за усвояване и експлоатация на програмния продукт с минимални усилия,
* Ефективноcт - Пълнота и скорост на изпълнение на специфицираните  
  функции в зададената изчислителна среда.

На ниво 2 са критериите. Критериите са софтуерно ориентирани свойства, представящи характеристики на програмния продукт. Всеки фактор се определя от няколко критерия.

На ниво 3 се позиционират метриките. В йерархичните модели метриките са софтуерно ориентирани детайли на даден критерий.

Последното ниво — 4 — е това на оценъчните елементи.

**Определяне стойностите на оценъчните елементи**:Методите, по които става това, могат да се класифицират по два признака:

А. По **начина на получаване** на информацията за програмния продукт:

- измерителен - състои се в използването на програмни инструментални средства за определяне обема на програмата, на времето на изпълнението на цялата програма или определени нейни клонове, на времето на реакция на програмата на определени входове, както и на други показатели.

- регистрационен - основава на информация, получена по време на изпитания или функциониране на програмния продукт, когато се регистрират определени събития.

- органолептичен - основан на използването на информация, получавана от човека в резултат на анализ на възприятията му чрез сетивните органи и се прилага при определяне на такива показатели, като удобство на използване, коректност и други подобни.

- изчислителен - използва теоретични и емпирични зависимости, статистически данни, натрупвани при изпитанията, експлоатацията и съпровождането на програмния продукт.

Б. По **източника на получаване** на информацията:

- традиционен - определят в специализирани организации от звена по изпитания и изчисления.

- експертен - осъществява се от група експерти, компетентни в решаването на дадения тип задачи, на основа на техния опит и интуиция.

- социологически - основава се на обработката на специално подготвени анкети въпросници

**Стойностите** на оценъчните елементи могат да попадат в следните видо-  
ве скали:

* **интервална** скала, характеризираща се с относителни или абсолютни величини в даден интервал;
* **порядкова** скала, позволяваща ранжиране на стойностите на някои оценъчни елементи чрез сравняването им с определени опорни стойности;
* **номинална** скала, показваща само наличието на конкретното разглеждано свойство в дадения програмен продукт.

**Процедура по оценяване –** Цел - чрез използване на вече дефинираната структура и като се следва точно определена процедура, да се определи за всеки даден програмен продукт число (обикновено дефинирано в интервала [0,l]), което да характеризира качеството.  
Основната идея на процедурата за оценяване на конкретен ПП се състои в следното:

1.Стойностите на всички оценъчни елементи се определят от експерти по един от изброените по-rope методи. Когато се отнася за определяне на стойност чрез измерителен, регистрационен или изчисли-  
телен метод, се посочва точният начин за получаване на стойността.

2. .Приема се също така, че на всяка характеристика на всяко ниво съответства тегло в интервала [0,1]. При това, както е обичайно, сумата от теглата на характеристиките, отнасящи се до една характеристика от по-горно ниво, е 1

3. При започване на оценката на качеството на конкретен ПП от даден тип експертите разполагат с вече готовите процедура за оценяване, указания за намиране стойностите на оценъчните елементи и теглата към всички характеристики на всички нива точно за дадения тип софтуер.

4.Експертите определят стойностите на всички оценъчни елементи

5.Нека получените стойности на оценъчните елементи, определящи метриката М са e,, е2,..., en, a съответните им предварително зададени тегла са w1,w2,...,wn. Тогава стойността на метриката М се изчислява по формулата:М = е1 \* w1 + е2 \* w2 +... + en \* wn

6.Същата схема на пресмятане се прилага за всяко от следващите нива.

С = М1 \* w1 + М2 \* w2 +... + Mn \* wn,където Mi са метриките, определящи критерия С.

**F = C1\*W|+C2\*w2+... + Cn\*wn** , Ci са критериите, които определят фактора F.

Q = F1\*w1 + F2\*w2+... + Fn,\*wn, където Fi са факторите, които определят качеството Q в инт [0,1].

Оценка на модела и метода

Предимства:

1. Простота на модела от структурна гледна точка.
2. Структурата на модела съдържа в себе си и директно следващи възможности за конструктивност.
3. Автоматизация на оценяването
4. Крайният резултат е едно-единствено число

Недостатъци:  
1. Процедурата за оценяване съдържа твърде много елементи на субек-  
тивност.

2.Огромна трудоемкост

3.Всеки тип програмни продукти изисква свое собствено множество от тегла.

**Моделът на Боем СОСОМО**(Constructive Cost Model)

Цел - за всеки планиран софтуерен проект да се оцени цената и срокът на разработване. Основополагащите му идея е използването броя редове първичен код.

Същност на модела:

За оценяване на трудоемкостта: ЧМ = 2.4 х ХРПК1.05,ЧМ - брой човекомесеца, ХРПК - хиляди реда първичен код.

За оценяване продължителността:В=2.5хЧМ0.38,където В е срокът на разработване в месеци.

Формулите се прилагат при следните предположения:

а) редовете първичен код се броят без коментарните редове, принадлежат на крайния продукт, не включват използваните стандартни програми;

б) включват се само фазите проектиране, програмиране и оценка, включително усилията по управлението и документирането по време на тези фази;

в) не се включват обучението, планирането и инсталирането на софтуера при потребителя;

г) включва се трудът на проектантите и програмистите, но не и този на компютърните оператори, висшите ръководители и секретарките;

д) 1ЧМ=19 дни=152 часа;

е) никакви сериозни промени не се правят в продукта след одобряването на изискванията

ж) потребителят и разработчикът са добросъвестни

Усъвършенстване на модела:Първото усъвършенстване е въвеждането на 3 типа софтуерни проекти — разпространен(разработва се от малка група в познатите условия на собствената филма ЧМ=2,4\*ХРПК1,05), полунезависим(има междинно ниво между разпространен и вграден тип ЧМ=3,0\*ХРПК1,12 ) и вграден(софтуерът работи свързан с апаратура, друг софтуер и изчислителни процеси ЧМ=3,6\*ХРПК1,20).

Следващото усъвършенстване е че все пак редовете първичен код не биха могли да са единственият параметър на установената формула. Преминава се към по-сложни модели — междинен u детайлен.

При извършването на оценката експертите следва да се ръководят от тази таблица и за конкретния проект трябва да определят за всеки атрибут съответния рейтинг. ЧМ = к х ХРПКР х A1 x A2... х А15,к и р са коефициенти, които са различни за различните типове софтуер,А са рейтингите на атрибутите

Разликата в междинния и детайлния модел е в начина на получаване на крайната оценка.

**Критика на„редове първичен код"**:Основните критики към СОСОМО се свеждат до следните две:

* Няма нито стандарт, нито единно виждане за това, какво е „ред първичен код".
* Много е трудно дори за експерти с голям опит да предскажат достатъчно точно в ранен етап на разработването броя на редовете първичен код.

По въпроса за редовете първичен код се изтъкват следните проблеми:

1.Какво всъщност да се разглежда — физически(Enter) или логически редове(някакъв ограничител, зависещ от конкретния език за програмиране).

2.Различните от семантична гледна точка типове редове( изпълними оператори, определения на данни, коментари, празни редове

3.Голям проблем е повторно използваемият код.

4.Как да се ползват редовете първичен код при приложения, писани на повече от един език.

5.Допълнителни:Да се включва или изключва временно поставен в програмата код, код от типа макрос, код, свързан с управлението на заданието в рамките на операционната система. Трудности предизвиква и разликата в стила на програмистите.