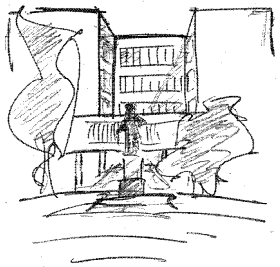
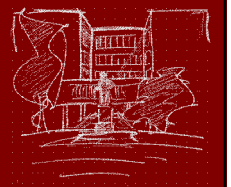
Развој софтвера 12



Саша Малков
Универзитет у Београду
Математички факултет
2023/2024

[P290]
Развој софтвера
Саша Малков



Тема 15

Метрике у развоју софтвера

Шта је метрика у развоју софтвера?

- Метрика је наука о мерама и мерењу
- Метрика у развоју софтвера (или *софтверска метрика*) се бави мерењем различитих карактеристика софтверских решења
- Циљ је описивање стања развојног пројекта
- Нумеричким описивањем различитих аспеката развојног пројекта се стичу упоредива знања о пројекту

Универзитет у Београду - Математички факултет

Типови метрика

- Непосредне (апсолутне) мере
  - нпр. број јединица кода или број линија кода
- Индиректне / изведене мере
  - нпр. количник других мера
- Интервалне мере
  - мера исказана као опсег вредности
- Номиналне мере
  - исказана речима, нпр. “јесте”, “није”, “плаво”,...
- Упоредне мере
  - “већи од” и сл.

Универзитет у Београду - Математички факултет

Врсте метрика

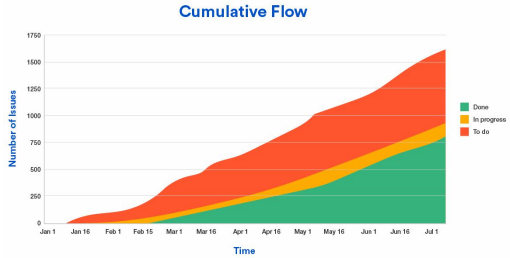
- Метрике праћења развоја софтвера
  - исказују у којој мери ток пројекта прати планове
  - односе се на
    - процес развоја
    - употребљене ресурсе (време, трошкови и сл.)
- Метрике (квалитета) дизајна софтвера
  - исказују неке мерљиве одлике софтвера
  - односе се на софтвер, као производ развојног процеса

Метрике праћења развоја софтвера

- Зову се и *метрике управљања* зато што су важне пре свега управљачком тиму
- Примери:
  - напредак
  - напор
  - трошкови
  - проблеми
  - стабилност захтева
  - стабилност величине
  - стање тестова
  - и многе друге

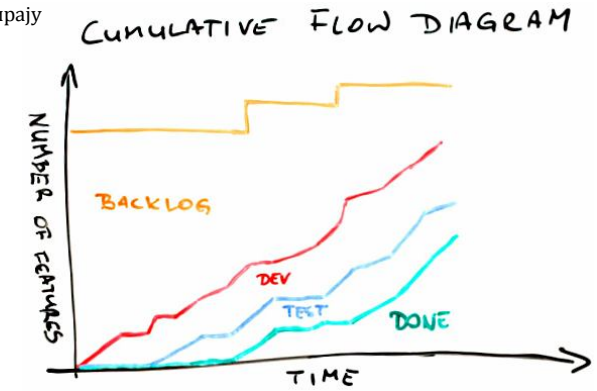
Мера напретка

- Исказује успешност праћења планова
  - укупан (кумулативан) број планираних и остварених послова у односу на време
  - да би дијаграм био реална слика стања, потребно је да величина послова буде релативно уједначена



Напредак по фазама

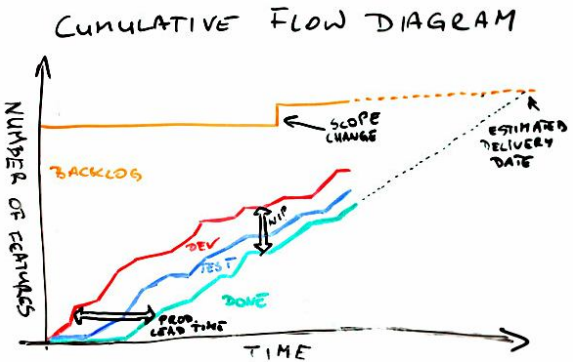
- Уобичајено је да се приказује више параметара
  - препознати послови
  - започети послови
  - послови који се тестирају
  - довршени послови



(http://brodzinski.com/2013/07/cumulative-flow-diagram.html)

Напредак по фазама

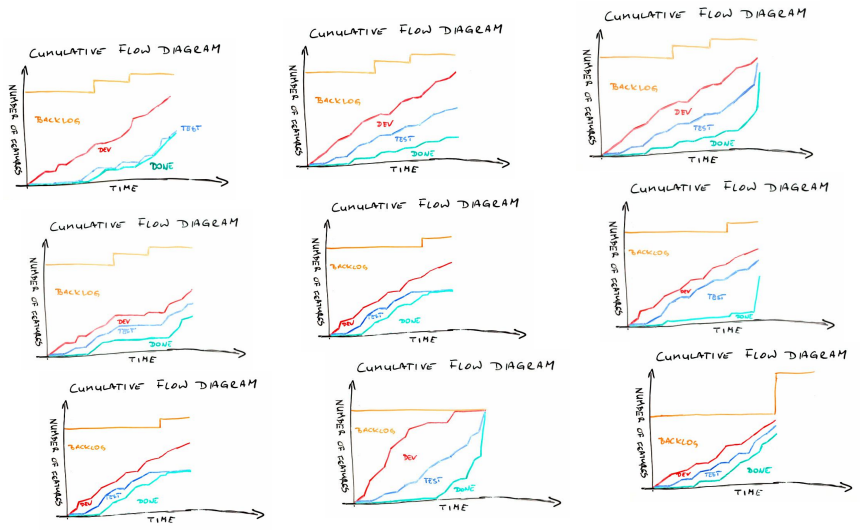
- Уобичајено је да се приказује више параметара
  - препознати послови
  - започети послови
  - послови који се тестираiv
  - довршени послови



(http://brodzinski.com/2013/07/cumulative-flow-diagram.html)

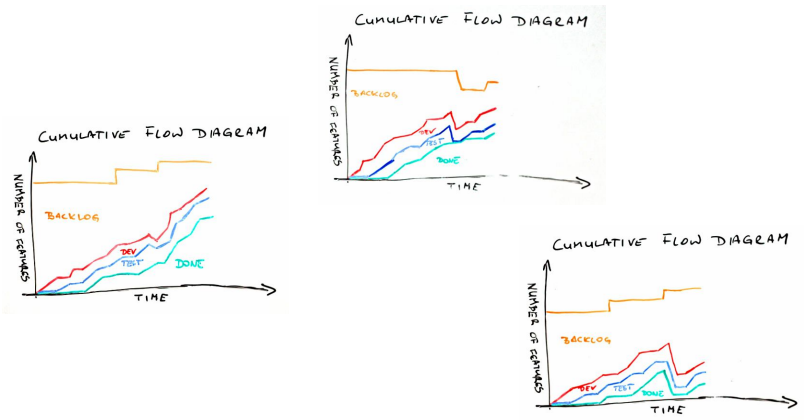
Универзитет у Београду – Математички факултет

Напредак по фазама, могу да се виде неусаглашености



Универзитет у Београду – Математички факултет

"Напредак" по фазама, може чак и да опада (2)

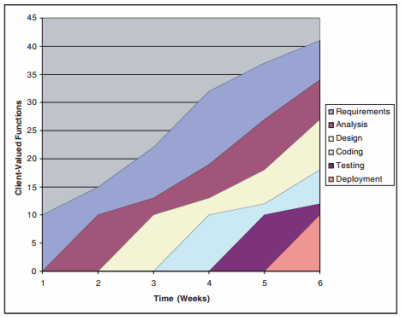


- Назадовање може да значи веће промене у архитектури

Универзитет у Београду – Математички факултет

Мера напретка ...

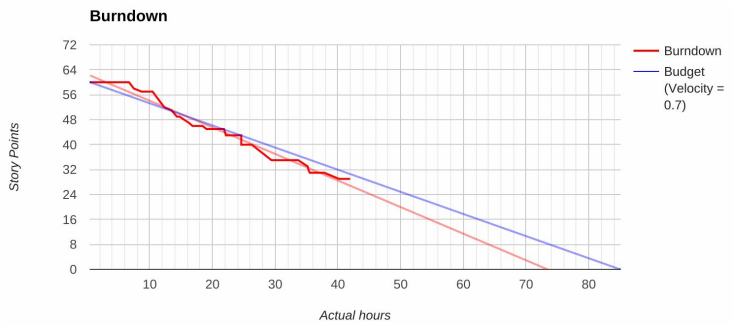
- Мера напретка може да прати различите фазе развоја
  - такав приступ се ређе користи у агилном развоју



Универзитет у Београду – Математички факултет

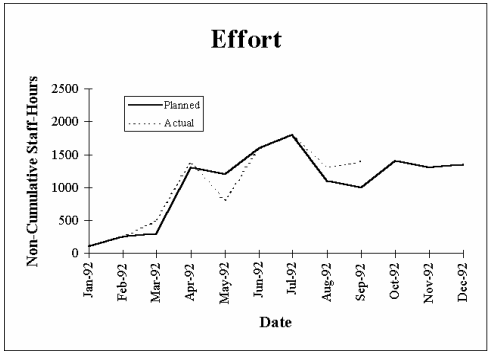
Мера завршавања

- Представља "обрнут" дијаграм у односу на меру напретка
  - уместо остварених представљамо преостале послове у односу на време
  - бар две линије – план и остваривање
  - незгодно када се број послова повећава у току развоја (уобичајено за агилни развој)



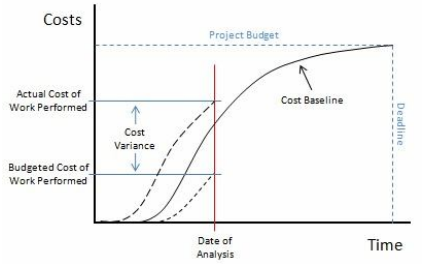
Мера напора

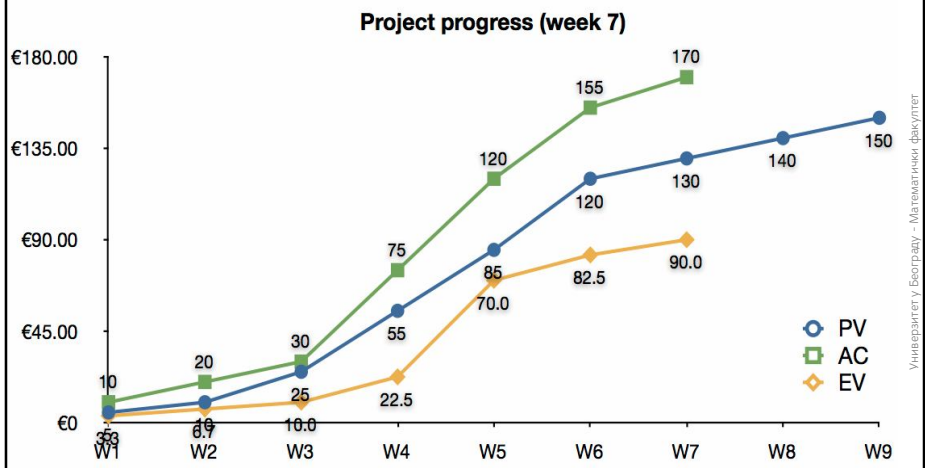
- Планирани и остварени број радних сати у односу на време



Трошкови

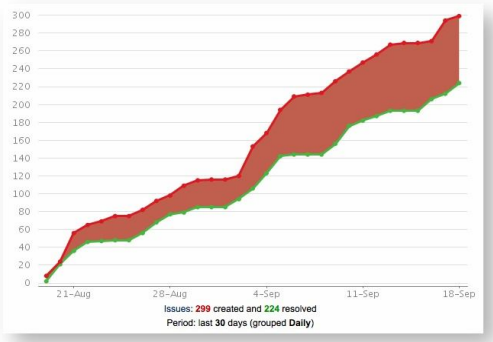
- Планирани и остварени утрошак средстава у односу на време
  - пројектовани укупан буџет (хор.линија при врху) = *BAC*
  - планирана вредност (*planned value*), *PV* = планиран трошак до тада
  - стваран трошак (*actual cost*), *AC* = стваран утрошак до тада
  - добијена вредност (*earned value*), *EV* = %обављеног посла * *BAC*



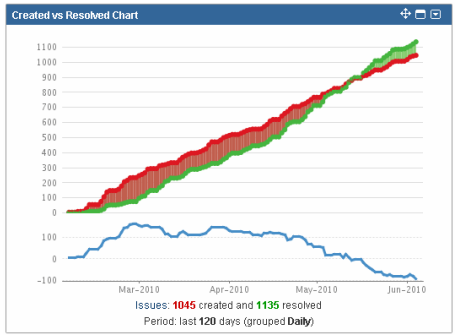


Број проблема

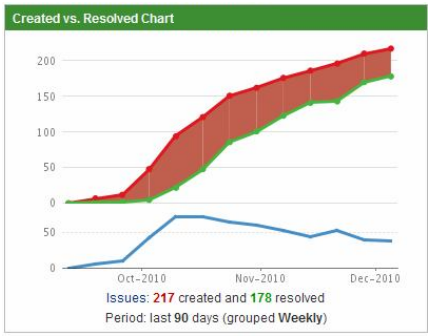
- Број отворених и решених проблема у односу на време
  - по данима
  - или кумулативно (као у примеру)
  - практично се своди на меру напора али без развоја и тестирања



Универзитет у Београду - Математички факултет

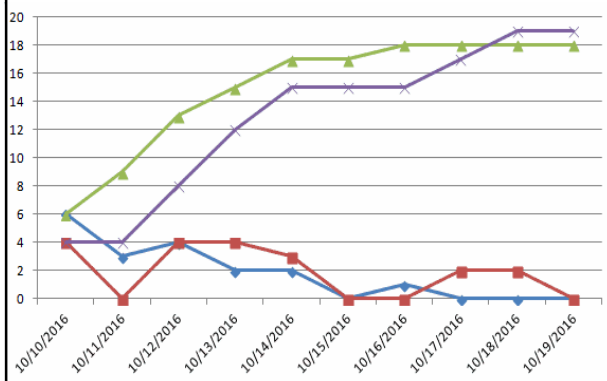


- Додатно приказ по данима



- Додатно кумулативно број отворених

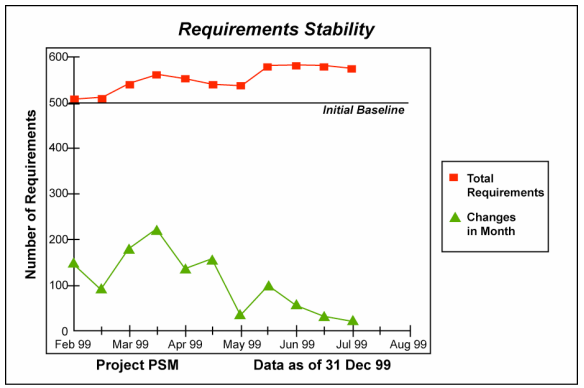
Универзитет у Београду - Математички факултет



Универзитет у Београду - Математички факултет

Стабилност захтева

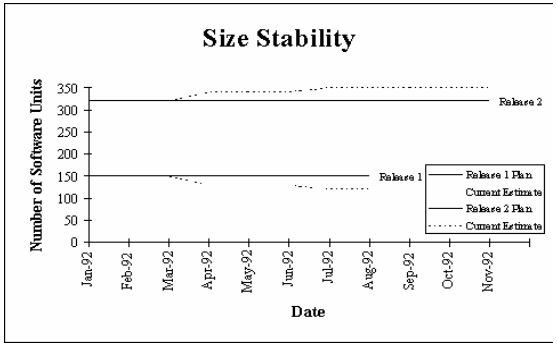
- Исказује број захтева током времена (промена и кумулативно)



Универзитет у Београду - Математички факултет

Стабилност величине

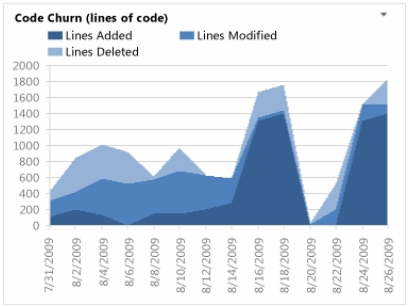
- Величина софтвера, обично у броју јединица кода, током времена (промена и кумулативно)



Универзитет у Београду - Математички факултет

Праћење степена измена у коду

- Ако се много додаје, то је развој
- Ако се много мења и брише, то су или велике промене или је фокус на рефакторисању или оптимизацији



Универзитет у Београду - Математички факултет

Метрике дизајна софтвера

- Исказују различите мерљиве одлике софтвера
- Непосредне мере софтвера су често лако мерљиве величине:
  - број јединица кода
  - број линија кода
  - број функционалних елемената кода
- Изведене мере софтвера могу да описују веома сложене одлике дизајна софтвера:
  - кохезија јединице кода
  - спрегнутост јединице кода
  - стабилност јединице кода
  - апстрактност јединице кода

Универзитет у Београду - Математички факултет

Број јединица кода

- Исказује број целина кода
- Може да представља
  - број програмских датотека
  - број класа
  - број пакета
  - број компоненти
  - број извршних датотека

Универзитет у Београду - Математички факултет

Број линија кода

- Исказује количину написаног кода
- Обично обухвата
  - све непразне линије које нису коментари
  - укључујући извршни код и декларације, дефиниције и друге врсте неизвршног кода

Број функционалних јединица

- Описује број неких функционалних елемената
  - број класа у пакету
  - број метода у класи
  - број метода у интерфејсу пакета
  - ...

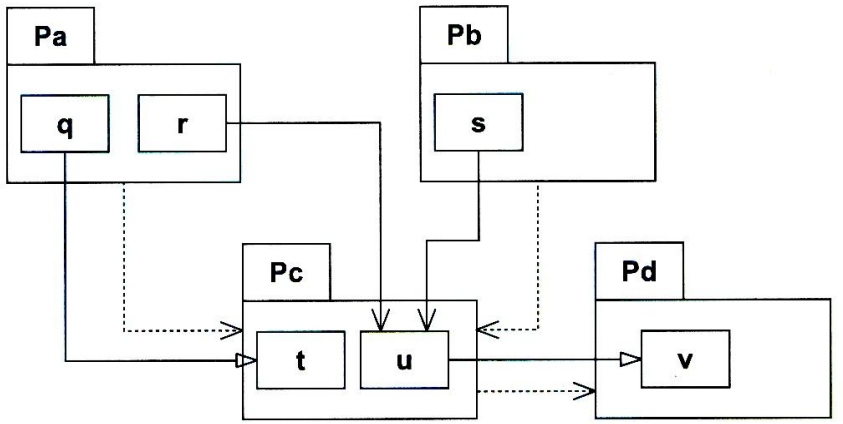
Стабилност пакета

- Стабилност пакета представља његову тенденцију да се не мења током времена
  - C_a = број класа у другим пакетима које зависе од класа у посматраном пакету
    - спрегнутост према пакету (енгл. *afferent coupling*)
    - што их је више, то ће промене пакета бити теже изводиве, а тиме и ређе
  - C_e = број класа у посматраном пакету које зависе од класа у другим пакетима
    - спрегнутост од пакета (енгл. *efferent coupling*)
    - што их је више, то ће његове промене бити чешће потребне
  - I = нестабилност, може да се израчунава као:
    - има опсег [0,1]

$$I = \frac{C_e}{C_a + C_e}$$

Стабилност пакета – пример

Pc = 1 / 4



Стабилност пакета – дискусија

- Није могуће да сви пакети буду стабилни
  - ако би били, то би значило да је читав систем непроменљив
  - уместо тога, желимо да неки пакети буду стабилнији а неки мање стабилни
- Принцип односа зависности и стабилности:
  - Зависност пакета би требало да иде од нестабилних према стабилним пакетима
- Није добро ако постоји зависност стабилног пакета од неког који је релативно флексибилан (нестабилан)

Апстрактност пакета

- Апстрактност пакета је релативна заступљеност апстрактних класа у пакету:
  - N_a = број апстрактних класа у пакету
  - N_c = број класа у пакету
  - A = апстрактност

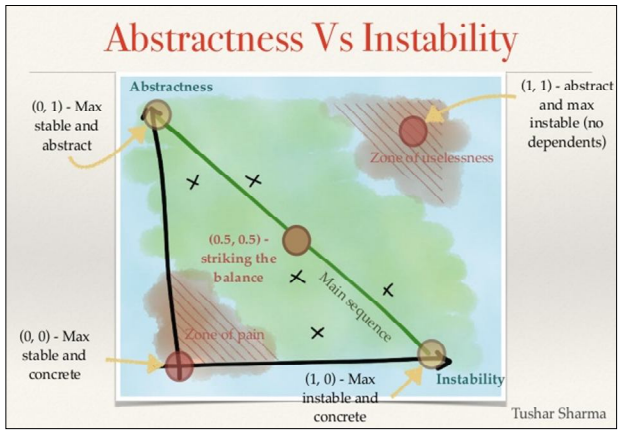
$$A = \frac{N_a}{N_c}$$

  - има опсег [0,1]

Апстрактност пакета – дискусија

- Принцип односа стабилности и апстрактности:
  - Пакет би требало да буде онолико апстрактан колико је стабилан
- Другим речима, мање апстрактни пакети би требало да зависе од апстрактних, а не обрнуто

Дијаграм зависности нестабилности и апстрактности



Апстрактност и стабилност

- Пожељно је да однос апстрактности и стабилности буде што ближе главној секвенци
- Удаљеност D се рачуна као:

$$D = \frac{|A + I - 1|}{\sqrt{2}}$$

или нормализовано (на опсег 0 до 1) као:

$$D = |A + I - 1|$$

Непожељни случајеви

- Зона неупотребљивости
  - висока апстрактност
  - висока нестабилност
  - апстрактан код који се или често мења или ништа од њега не зависи
- Зона бола
  - висока конкретност (ниска апстрактност)
  - висока стабилност
  - такав код се веома тешко мења, било да је потребно да се проширује или прилагођава околностима

Функционална кохезија пакета

- Један начин изражавања функционалне кохезије пакета је у облику количника броја зависности међу класама пакета и броја пакета
- Представља интензитет односа пакета са сопственим класама
  - R = број међусобних зависности међу класама пакета
  - N = број класа у пакету
  - H = релациона кохезија

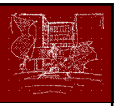
$$H = \frac{R + 1}{N}$$

- Алтернатива је да се свака зависност двеју класа представи бројем метода који је остварују

Мерење сложености кода

- Сложеност програмског кода нијелако изразити, али ипак постоје и различити начини да се она измери
- Могу да почивају на
  - сложености структуре кода
    - пример је цикломатичка сложеност
  - сложености израза у коду
    - пример је Холстедова сложеност
  - и друго...

Цикломатичка сложеност



- Томас Мек-Кејб, *IEEE, SE, 1976.*
  - назива се и Мек-Кејбова сложеност
- Алгоритам се посматра као граф *G*
  - Чворови су полазне и завршне тачке, тачке гранања и тачке спајања
  - Гране су путање између чворова
- Сложеност *V(G)* је број свих међусобно независних линеарних путева кроз граф
- V(G) = E – N + 2P
  - E = број грана
  - N = број чворова
  - P = број повезаних компоненти (подскупова чворова који су међусобно повезани и одвојени од осталих делова графа, обично = број потпрограма)
  - ако су одвојене полазни и излазни чворови
- V(G) = E – N + P
  - ако је свака излазна тачка повезана са улазном
- Мек-Кејб препоручује да буде *V(G) <= 10*

Холстедова сложеност

- Морис Холстед, *1977.*
- Преброје се сви оператори (и типови, заграде,...):
  - N_1 = укупан број оператора
  - n_1 = укупан број различитих оператора
- Преброје се сви операнди:
  - N_2 = укупан број операнада
  - n_2 = укупан број различитих операнада
- Рачунају се:
  - $n = n_1 + n_2$ – речник програма
  - $N = N_1 + N_2$ – дужина програма
  - $L = n_1 \log_2 n_1 + n_2 \log_2 n_2$ – процењена дужина програма
  - $V = L \log_2 n$ – обим (волумен)
  - $D = n1$ x $N2$ $/(2\ n2)$ – тежина програма, за писање и разумевање
    - алтернативно: $D = (n1 + N2 * V)/(2\ n2)$
  - $E = D$ x V – напор, очекивано време писања, отприлике T = E / 18 секунди
  - $B = E^{2/3}$ / 3000 – очекивани број испоручених багова
    - користи се и као V / 3000, што је често веома блиско

Пажљиво !!!

- “Неодговарајућа употреба метрика сложености софтвера може да има велике штетне ефекте због подстицања лоше програмерске праксе и деморалисања добрих програмера. Метрике сложености софтвера морају да се критички оцењују да би се одредио исправан начин њихове примене”
  - *Joseph K. Kearney*

Каталог метрика

- Има више покушаја каталогизирања метрика
- Пример каталога са оврtом на стандарде су сачинили *Rudiger Lincke* и *Welf Lowe*
  - *http://www.arisa.se/compendium, 2007.*

ISO/IEC 9126

- Стандард прописује метрике којима се мери “квалитет” софтвера
- Категоризоване су у 6 група, које се затим деле на подгрупе:
  - Функционалност
  - Поузданост
  - Употребљивост и поновна употребљивост
  - Ефикасност
  - Одрживост
  - Преносивост
- Приметимо да овде има редундантности, ако посматрамо из угла пројектовања, па се неке метрике уклапају у више група

Универзитет у Београду - Математички факултет

Литература за тему

- *Penny Grubb and Armstrong Takang*, **Software Maintenance: Concepts and Practice**, *World Scientific Publishing, 2003*.
- *Scott W. Ambler and Mark Lines*, **Disciplined Agile Delivery (DAD): A Practitioner’s Guide to Agile Software Delivery in the Enterprise**, *IBM Press, 2012*.
- *Rufiger Lincke*, **Compendium of Software Quality Standards and Metrucs**, 2007, *http://www.arisa.se/compendium*

Универзитет у Београду - Математички факултет

Хвала на пажњи!

МАТФ
Универзитет у Београду
Математички факултет