Universiteti i Prishtinës "Hasan Prishtina" Kosovë

Inxhinierimi softverik

Pjesa 10 – Testimi dhe sigurimi i cilësisë

Prof. Asoc. Dr. Ermir Rogova

Objektivat

- Kuptoni cilësinë dhe teknikat bazë për verifikimin dhe vërtetimin e softverit
- Analizoni bazat e testimit të softuerit dhe teknikave të testimit.
- Diskutoni konceptin e procesit të inspektimit .

Njoftim me testimin

- Sigurimi i cilësisë (QA): aktivitetet e projektuara për të *matur* dhe *përmirësuar cilësinë* në një produkt dhe proces.
- Kontrolli i cilësisë (QC): aktivitete të dizajnuara për të *validuar dhe verifikuar cilësinë* e produktit përmes *zbulimit të defekteve dhe "rregullimit" të defekteve .*
- Duhen *teknika*, *procese*, *mjete,* dhe *ekipe të mira*

Çka është "Cilësia"?

- Dy përkufizime tradicionale :
  - *Përputhet me kërkesat.*
  - *I përshtatshëm për t'u përdorur.*
- *Verifikimi : kontrollimi i përshtatshmërisë së* softuerit *me kërkesat e tij* (a *evoluoi softueri nga kërkesat ashtu siç duhet* ; a "funksionon" softueri?)
  - A është sistemi i saktë?
- *Vleresimi* : kontrollimi i softueri se a plotëson kërkesat e përdoruesit (*i përshtatshëm për t'u përdorur*)
  - A po ndërtojmë sistemin e duhur?

Disa teknika të "zbulimit të gabimeve"

- Testimi : ekzekutimi i programit në një mjedis të kontrolluar dhe "verifikimi/validimi" i rezultateve.
- Inspektimet dhe rishikimet .
- Metodat formale (duke vërtetuar saktësinë e softuerit).
- Analiza statike zbulon "kushte të prirura për gabime".

Defektet dhe dështimet

- Gabim : një gabim i bërë nga një programues ose inxhinier softuerësh që shkaktoi defektin, i cili mund të shkaktojë një dështim
- Defekti : gjendje që *mund të* shkaktojë një dështim në sistem
- Dështim : pamundësia e sistemit për të kryer një funksion sipas specifikave të tij për shkak të ndonjë defekti
- Ashpërsia e defektit ose dështimit (bazuar në pasojat)
- Prioritet i defektit ose dështimit (bazuar në rëndësinë e zhvillimit të një rregullimi, i cili nga ana tjetër bazohet në ashpërsinë)

Testimi

- Aktiviteti i kryer për:
  - *Vlerësimin e* cilësisë së produktit
  - *Përmirësimin e produkteve* duke identifikuar defektet dhe duke i rregulluar ato përpara lëshimit të softuerit

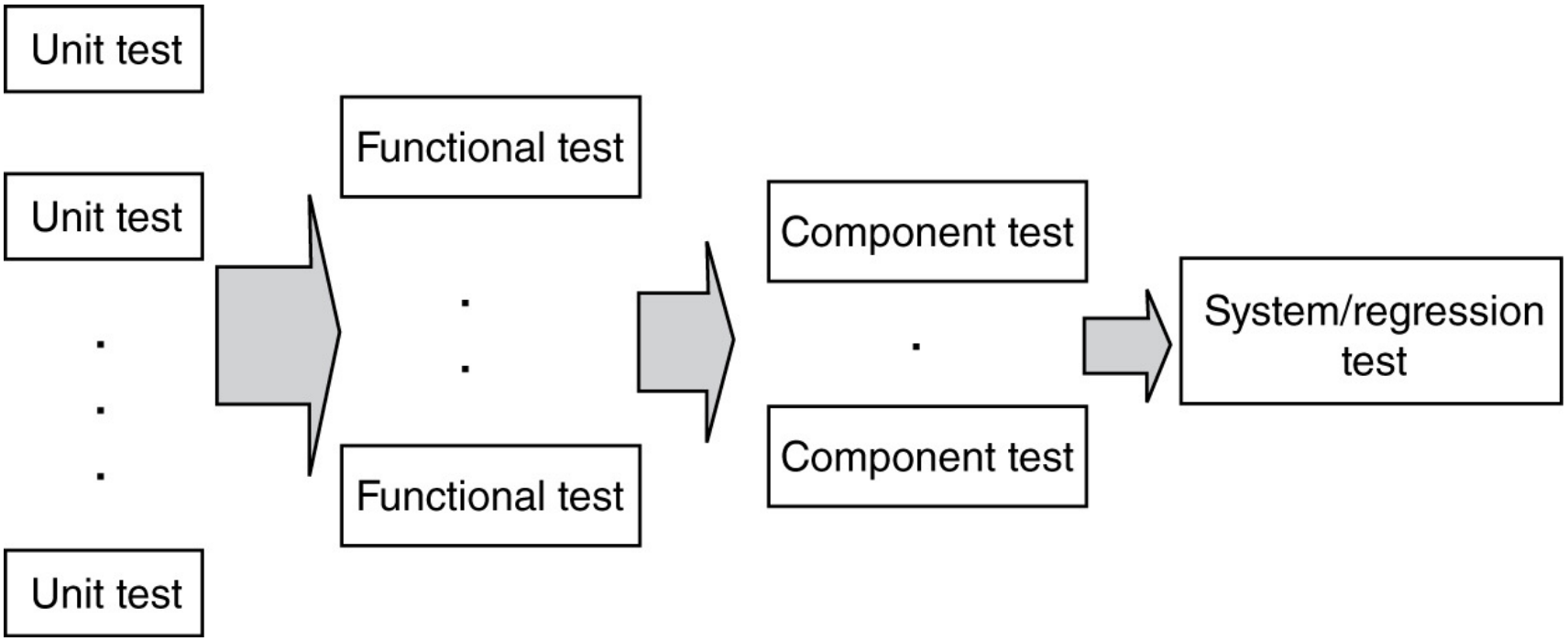
Jo gjithmone ndodh!

- Verifikimi dinamik i sjelljes së programit me një grup të kufizuar rastesh testimesh të zgjedhura nga domeni i ekzekutimit.
- Testimi NUK mund të vërtetojë funksionimin e produktit 100% — edhe pse ne përdorim testimin për të demonstruar se pjesë të softuerit funksionojnë.

Testimi

- Kush teston?
  - Programuesit
  - Testuesit/Analist. kërkesave
  - Përdoruesit
- Çfarë testohet?
  - Testimi i kodit të njësisë
  - Testimi i kodit funksional
  - Testimi i integrimit/sistemit
  - Testimi i ndërfaqes së përdoruesit
- Pse testojmë?
  - Pranimi (klienti)
  - Përputhshmëria (std, ligjet, etj.)
  - Konfigurimi (përdoruesi kundrejt zhv.)
  - Performanca, stresi, siguria, etj.
- Si testojmë?
  - Intuita
  - Bazuar në specifikimet (*kutia e zezë)*
  - Bazuar në kod (*kutia e bardhë)*
  - Rastet ekzistuese (*regresioni*)

Ecuria e Testimit



Ndarja nëpër klasa ekuivalente

- Ndani të dhënat në disa grupe , të konsideruara "ekuivalente" për qëllime të gjetjes së gabimeve.
- Zgjidhni një "përfaqësues" për secilën klasë të përdorur për testim.
- Klasat e ekuivalencës përcaktohen nga specifikimet e kërkesës/dizajnit dhe intuita.

Table 10.1 Equivalence Class Example

Class	Representative
Low	-5
0–12	6
13–19	15
20–35	30
36–120	60
High	160

Analiza e vlerës kufitare (Një teknikë e kutisë së zezë)

- Përvojat e kaluara tregojnë se "kufijtë" janë të prirur për gabime.
- Bëni ndarjen nëpër klasa ekuivalente ; shtoni rastet e testimit për kufijtë (*në kufi* , *jashtë* , *brenda*).
  - Rastet e reduktuara : konsideroni kufirin si një pikë mes numrave.
    - Nëse kufiri është në 12 :
      normale : 11, 12, 13;
    - reduktuar : 12, 13

Kufijtë e vlerave hyrëse

1 <= numri i punonjësve, **n** <= 1,000,000

1 ⟵ **n** ⟶ **1 000 000**

1 <= **mosha** e punonjësit <= 150

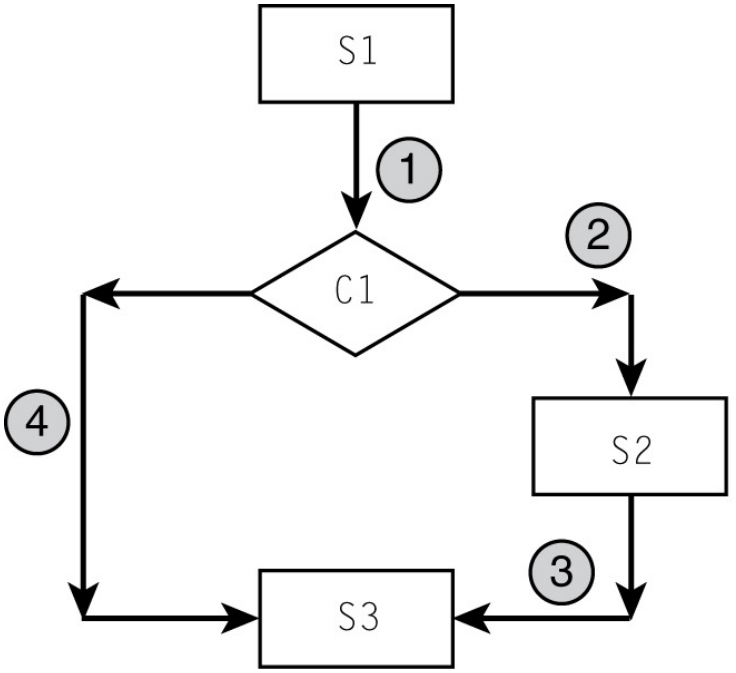
1 ⟵ **mosha** ⟶ **150**

Testimi "bazë" i vlerës kufitare për një vlerë do të përfshijë:

- ***Në kufirin "minimumi".***
- ***Menjëherë mbi minimumin***
- ***Midis minimumit dhe maksimumit (nominal)***
- ***Menjëherë nën maksimum***
- ***Në kufirin "maksimal".***

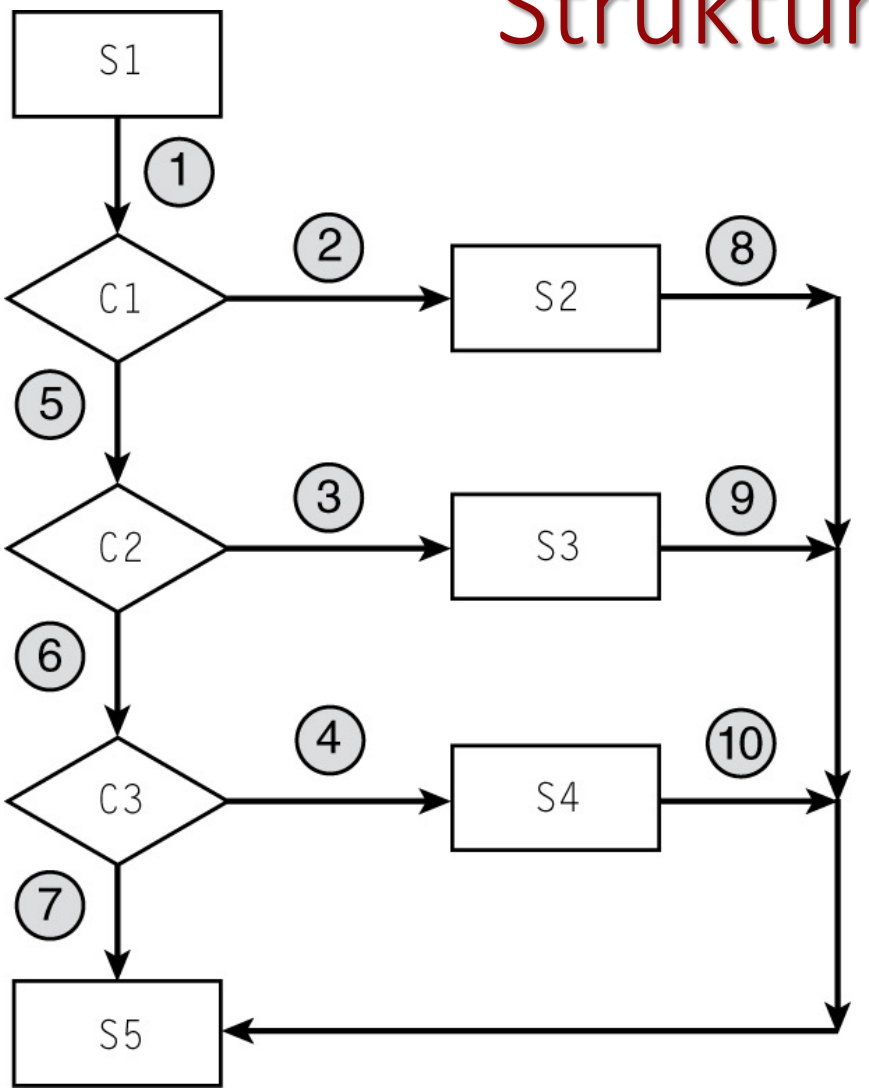
Analiza e shtegut

- Teknikë e kutisë së bardhë
- *Dy detyra*
  1. Analizoni numrin e shtigjeve në program.
  2. Vendosni se cilat të testoni.




```
Path1: S1 - C1 - S3
Path2: S1 - C1 - S2 - S3
    OR
Path1: Segments (1, 4)
Path2: Segments (1, 2, 3)
```

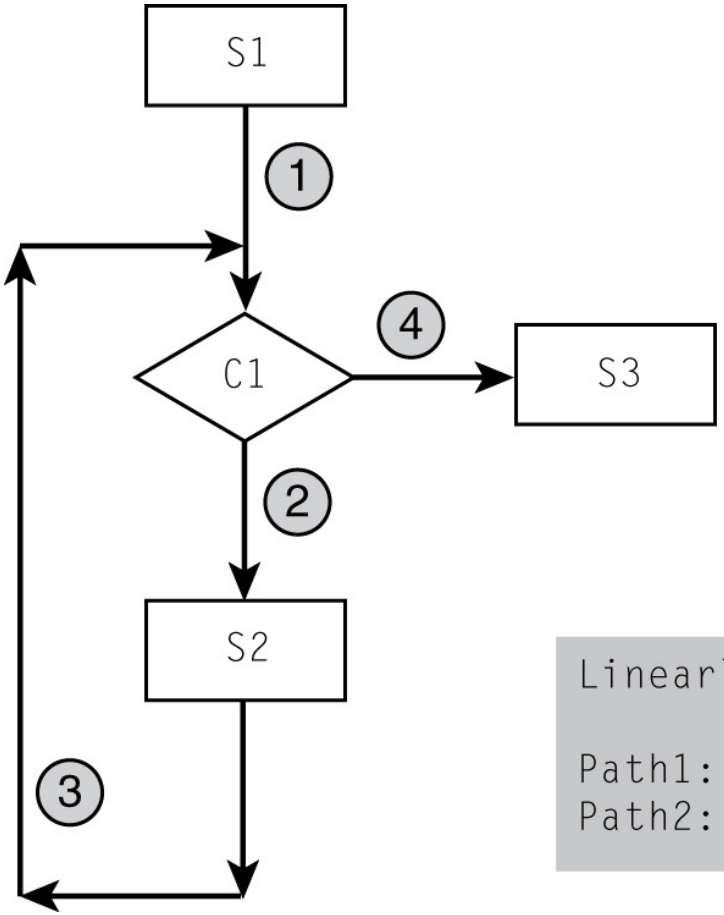
Strukturë CASE




```
The 4 independent paths cover

Path1: includes S1-C1-S2-S5
Path2: includes S1-C1-C2-S3-S5
Path3: includes S1-C1-C2-C3-S4-S5
Path4: includes S1-C1-C2-C3-S5
```

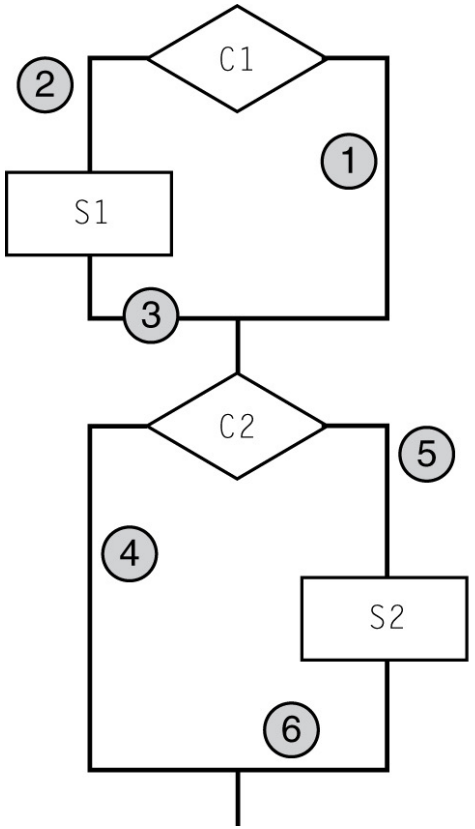
Shembull me një unazë




```
Linearly independent paths are

Path1: S1-C1-S3   (segments 1, 4)
Path2: S1-C1-S2-C1-S3   (segments 1, 2, 3, 4)
```

Grup shtigjesh të pavarura lineare



Consider `Path1`, `Path2`, and `Path3` as the Linearly Independent Set

	1	2	3	4	5	6
Path1	1				1	1
Path2	1			1		
Path3		1	1	1		
Path4		1	1		1	1

Testimi i njësisë

- Testimi i njësisë : Testoni çdo njësi individuale.
- Zakonisht bëhet nga programuesi.
- Testoni çdo njësi ndërsa zhvillohet.
- Ruani rastet/rezultatet e testimit
  - Lejon testimin e regresionit.
  - Lehtëson rifaktorimin.
  - Testet bëhen dokumentacion !!

Zhvillimi i Drejtuar nga Testi

- Shkruani rastet e testimit të njësive **PARA** kodit!
- Rastet e testimit "janë"/"bëhen" kërkesa.
- Detyron zhvillimin me hapa të vegjël.
- Hapat :

1. Shkruani testin dhe kodin.
2. Verifiko (dështon ose funksionon).
3. Ndrysho kodin që të ketë sukses.
4. Riprovo testin.
5. Refaktoro deri në sukses/dëshirë.

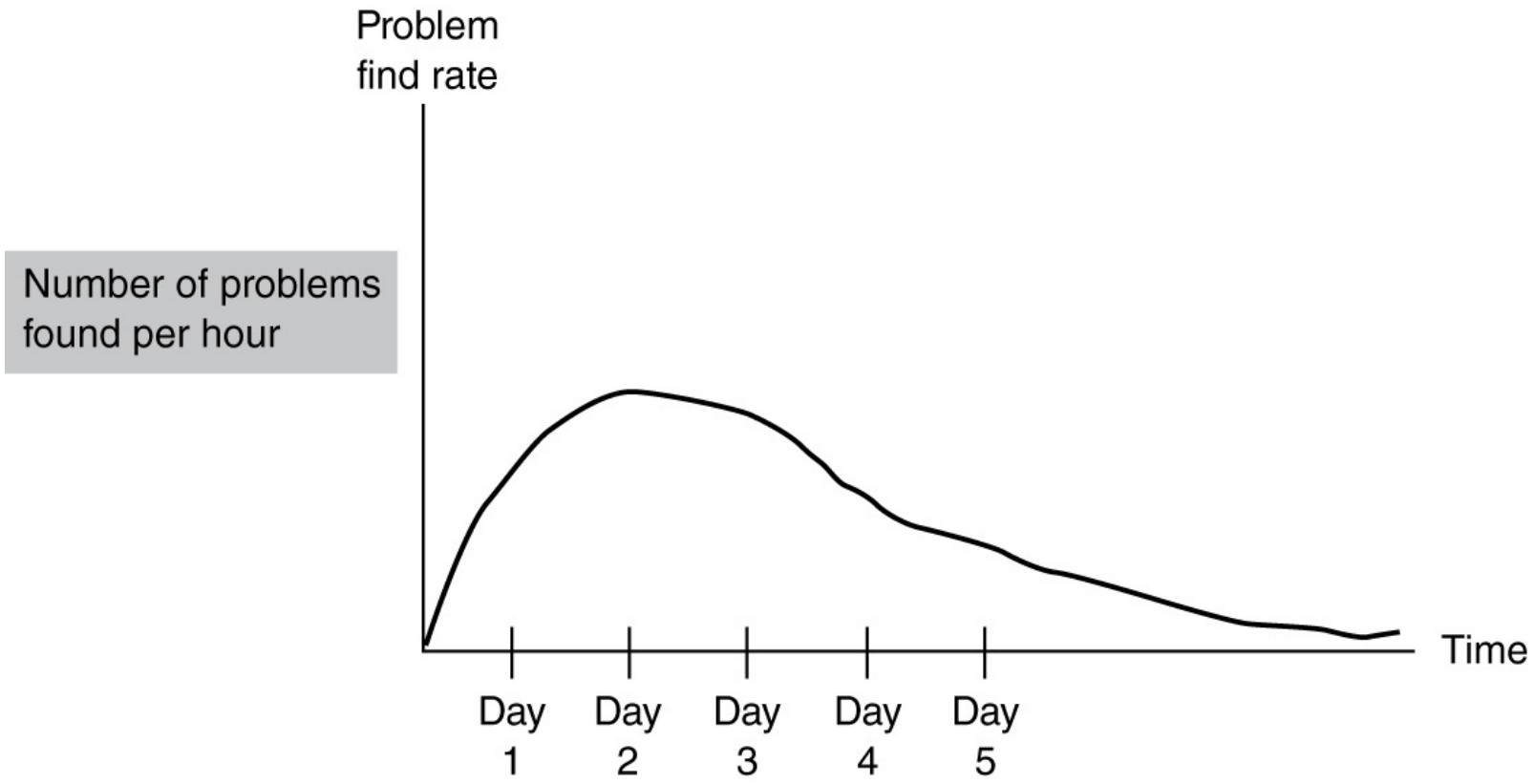
Kur të ndaloni testimin?

- Përgjigje e thjeshtë : ndalo kur
  - Të gjitha rastet e planifikuara të testimit janë ekzekutuar .
  - Të gjitha problemet që janë gjetur janë rregulluar .
- Teknika të tjera:
  - *Ndaloni kur nuk po gjeni më gabime.*
  - *Mbjellja e defektit - testoni derisa të gjenden të gjitha (ose % e) gabimeve të mbjella.*
- JO kur ju mbaroi koha - planifikim i dobët!

Mbjellja e defektit

- Mbjellni defekte në program (komponent):
  - Gjeneroni dhe shpërndani një numër "x" të gabimeve.
  - *Mos u tregoni testuesve* .
  - Caktoni një % (p.sh., 95%) e gabimeve të gjetura si kritere ndaluese
- Supozoni se janë gjetur numri "y" i "x" defekteve të mbjellura
  - Nëse (y/x) > (përqindja e ndalimit), ndaloni testimin.
  - Nëse (y/x) ≤ (përqindja e ndalimit), vazhdoni të testoni.
- Merrni një ide se sa gabime mund të mbeten ende:
  - Ta zëmë se përmes testimit keni zbuluar "u" defekte jo të mbjella.
  - Pasi y/x = u/v; v = (u * x)/y.
  - Dmth ka shumë të ngjarë (v − u) gabime kanë mbetur ende në softuer.

Norma e gjetjes së problemit



Inspektimet dhe rishikimet

- Rishikimi : çdo proces që përfshin testuesit njerëzorë që të lexojnë dhe kuptojnë një dokument dhe më pas e analizojnë atë me qëllim të zbulimit të gabimeve
- Walkthrough : autori i shpjegon dokumentin
- Inspektimi i softuerit : rishikime të hollësishme të punës në vazhdim

Inspektimet e softuerit

- *Hapat* :
  1. Planifikimi
  2. Vështrim i përgjithshëm
  3. Përgatitja
  4. Inspektimi
  5. Ripunoni
  6. Ndiqe

- **I fokusuar** në gjetjen e defekteve
- **Prodhimi** : lista e defekteve
- **Ekipi prej**:
  - 3-6 persona
  - Autori i përfshirë
  - Njerëzit që punojnë në përpjekjet përkatëse
  - Moderator, lexues, shkrues

Inspektimet kundrejt testimeve

- **Inspektimet**
  - Pjesërisht me kosto efektive.
  - Mund të aplikohet në objekte të ndërmjetme.
  - I kap herët defektet.
  - Ndihmon në shpërndarjen e njohurive rreth projektit dhe praktikave më të mira.
- **Testimet**
  - I gjen gabimet më lirë, por korrigjimi i tyre është i shtrenjtë.
  - Mund të aplikohet vetëm në kod.
  - I kap me vonesë defektet (pas zbatimit).
  - E nevojshme për të vlerësuar cilësinë.

Metodat formale

- Teknika matematikore që përdoren për të vërtetuar se një program funksionon.
- Përdoret për specifikimet e kërkesave/dizajnit/algoritmit.
- Vërteton se zbatimi përputhet me specifikacionet
- Para dhe pas kushte
- Problemet :
  - Kërkon njohuri matematikore.
  - Nuk aplikohet për të gjitha programet.
  - Vetëm verifikim, jo validim.
  - Nuk zbatohet për të gjitha aspektet e programit (p.sh., UI ose mirëmbajtje).

Analiza Statike

- Ekzaminimi i strukturave statike të dizajnit/kodit për *zbulimin e kushteve të prirura ndaj gabimeve* (kohezioni — kouplingu).
- Veglat automatike të programit janë më të dobishme.
- Mund të aplikohet për:
  - Dokumentet e ndërmjetme
  - Kodin burimor
  - Skedarët e ekzekutueshëm
- Prodhimi (outputi) duhet të kontrollohet nga programuesi.

Pyetje???