Model based testing cu GraphWalker

Dima Oana 506
Dabu Alexandru 506

GraphWalker

- Tool open-source
- Modelarea și executarea testelor software prin intermediul modelelor grafice (grafuri orientate).
- Generează un **drum** de testare folosind A* sau parcurgerea random.
- https://github.com/GraphWalker
- https://graphwalker.github.io/

GraphWalker

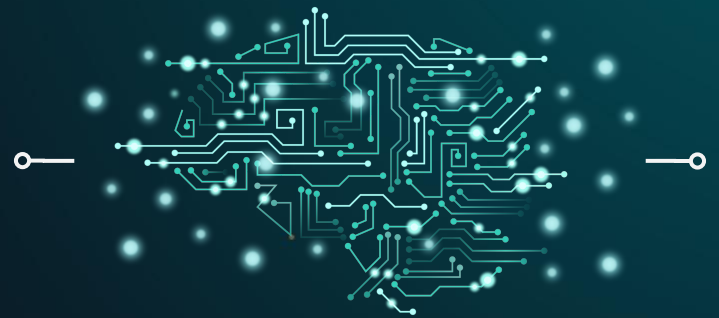
GraphWalker

UTILIZĂRI PRACTICE

Testarea jocurilor video

Testarea securității software-ului

Testarea software-ului embedded



Testarea automată a interfețelor web

Testarea aplicațiilor mobile

Testarea API-urilor și serviciilor web

GraphWalker

Un FSM descrie modul în care sistemul trece prin diferite stări în funcție de intrările primite.

Componente

- **Stări** - stările distincte ale sistemului în care se poate afla.
- **Tranziții** - modul în care sistemul trece de la o stare la alta în funcție de evenimentele care au loc.
- **Intrări** - evenimentele care determină sistemul să efectueze tranziții între stări.
- **Ieșiri** * - efectele generate de sistem în timpul tranzițiilor.

GraphWalker

GraphWalker

Labirint

Labirintul reprezintă aplicația sau sistemul de testat.

GraphWalker creează o hartă a labirintului (**model** = graf) cu locuri (stări = **noduri**) și drumuri (tranziții = **muchii**) posibile. Apoi, călătorul (GraphWalker) urmează aceste drumuri pentru a verifica că nu există capcane sau obstacole (buguri în aplicație).

GraphWalker

Componente de utilizat

GW STUDIO

- Modelarea și rularea testelor
- Generarea codului de test
- Vizualizarea modelului
- Generarea rapoartelor

01

GW PLAYER

Testarea interfețelor de utilizator și a fluxurilor de lucru

02

MAVEN PLUGIN

- Mediu de dezvoltare populare - Java
- Automatizarea testării
- Gestionarea dependințelor
- Portabilitate

03

Când devine necesar design-ul bazat pe model?

- Interacțiunile din cadrul unei aplicații pot crește substanțial pe parcursul dezvoltării.
- Mai multe funcționalități → mai multe cazuri de test.
- Mai multe cazuri de test → creșterea complexității și a timpului necesar definirii testelor.

Când devine necesar design-ul bazat pe model?

- Asigurarea calității produsului — acoperirea **comportamentului de bază**, cât și **a cazurilor limită**.

- **Reprezentarea modulară** este un mod prin care putem defini logic anumite stări ale aplicației și potențiale interacțiuni între ele → putem extinde programatic cazurile de testare prin dezvoltarea modelului aplicației

Avantaje ale testării bazate pe model

R

Reutilizabilă

Scenariile de test definite pot fi folosite și pentru alte zone ale aplicației.

S

Scalabilă

Putem extinde programatic suita de teste pe parcurs ce aplicația se dezvoltă.

A

Adaptabilă

Modelul se poate modifica pentru a reprezenta mai bine starea aplicației.

R

Robustă

Se pot explora ușor atât cazurile de bază, cât și o suită de cazuri limită.

Tipuri de generare a testelor

Algoritmi de parcurgere

Aleator

Parcurgerea stărilor într-un mod aleator cu diverse criterii de acoperire

01

Informat

Parcurgerea stărilor în funcție de reguli definite prin algoritmi de traversare

02

Utilitatea modului de parcurgere aleator

- Cazuri definite manual → Cazuri limitate la nivelul rigurozității celui ce le definește.
- Utilizatorii pot fi foarte imprevizibili → Potențiale cazuri limită neacoperite de teste definite manual.

Utilitatea modului de parcurgere aleator

- Prin parcurgerea aleatorie putem:
  - Explora scenarii de test necunoscute sau improbabile
  - Analiza modul în care răspunde aplicația pentru anumite interacțiuni și modul în care se transformă pentru a răspunde acestor operații
  - Testa rezistența aplicației la o suită de acțiuni neobișnuite și solicitante

Utilitatea modului de parcurgere informat

- Cazuri definite manual → Cazuri ce pot deservi drept garanție pentru calitatea produsului, indiferent de cât de mult se schimba aplicația

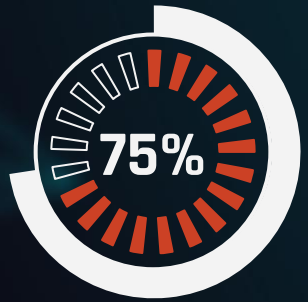
- Utilizatorii pot fi foarte imprevizibili → Se testează cazurile critice și des întâlnite pentru a asigura că utilizatorii pot îndeplinii funcțiile importante

Utilitatea modului de parcurgere informat

- Prin parcurgerea informată (ex: A*) putem:
  - Explora scenarii de test des întâlnite pentru a asigura că aplicația nu a suferit modificări nedorite în timpul dezvoltării
  - Analiza modul în care răspunde aplicația pentru situații cerute, astfel încât să documentăm funcționalitățile
  - Testa rezistența aplicației pentru situațiile cele mai solicitante

Asigurarea calității produsului

Suma tuturor strategiilor



Avem nevoie de ambele modalități de parcurgere (depinde de natura aplicației):

→ parcurgerea **aleatorie** – pentru identificarea cazurilor limită și a situațiilor neprevăzute, sau care contrazic în vreun fel logica aplicației

→ parcurgerea **informată** – pentru asigurarea corectitudinii funcționalităților cunoscute și cerute în cadrul dezvoltării

Multumim!

Dima Oana 506
Dabu Alexandru 506