

Raport despre folosirea AI în dezvoltarea și testarea software

În cadrul [proiectului nostru](#) am încercat folosirea inteligenței artificiale în anumite etape muncitorești pe care le-am avut de executat.

În marea majoritate a cazurilor, tool-ul folosit, mai exact [ChatGPT](#) nu ne-a ajutat pe cât de mult am fi crezut, în ciuda faptului că întrebările pe care i le-am pus au fost clare, concise și avea toate detaliile necesare pentru a ne oferi un răspuns cel puțin satisfăcător.

Cazurile concrete în care am încercat folosirea lui ChatGPT sunt următoarele:

1. Am scris Clasele de echivalență și am dorit obținerea celor globale

Concluzii:

Rezultatul a fost unul nesatisfăcător din partea inteligenței artificiale, răspunsul obținut fiind următorul:

Pentru a determina clasele globale pentru acest exemplu, putem urma aceeași logică ca în primul caz. Vom avea astfel:

Clasele de echivalență pentru întregul program (globale) vor fi o combinație a claselor individuale:

1. $NL_1 = 3 \dots 50$; $NL_2 = \{ nl \mid nl < 3 \}$; $NL_3 = \{ nl \mid nl > 50 \}$
2. $DL_1 = 3 \dots 50$; $DL_2 = \{ dl \mid dl < 3 \}$; $DL_3 = \{ dl \mid dl > 50 \}$
3. $S_1 = 1 \dots DMV$; $S_2 = \{ s \mid s < 1 \}$; $S_3 = \{ s \mid s > DMV \}$
4. $P_1 = 1 \dots DMV$; $P_2 = \{ p \mid p < 1 \}$; $P_3 = \{ p \mid p > DMV \}$
5. $D_1 = 0 \dots 100$; $D_2 = \{ d \mid d < 0 \}$; $D_3 = \{ d \mid d > 100 \}$

Domeniul de ieșiri constă în:

- $PC_1 = \{ x \mid x \text{ este valid și creat (CreatedAtActionResult - cod de stare 201)} \}$

- **PC_2 = { x | x este invalid (BadRequestObjectResult - cod de stare 400) }**

Aceste clase de echivalență globale ar putea fi folosite pentru a genera un set de date de test, similar cu cel dat în exemplul anterior.

În final am împărțit noi în clasele globale, astfel:

$F1\ 1\ 1\ 1\ 1\ 1 = \{ (nl, dl, s, p, d, x) \mid nl \in NL_1, dl \in DL_1, s \in S_1, p \in P_1, d \in D_1, x \in PC_1 \}$

$F_{_ _ _ _ 2 _} = \{ (nl, dl, s, p, d, x) \mid d \in D_2 \}$

$F_{_ _ _ _ 3 _} = \{ (nl, dl, s, p, d, x) \mid d \in D_3 \}$

$F_{_ _ _ 2 _ _} = \{ (nl, dl, s, p, d, x) \mid p \in P_2 \}$

$F_{_ _ 2 _ _ _} = \{ (nl, dl, s, p, d, x) \mid s \in S_2 \}$

$F_{_ 2 _ _ _ _} = \{ (nl, dl, s, p, d, x) \mid dl \in DL_2 \}$

$F_{_ 3 _ _ _ _} = \{ (nl, dl, s, p, d, x) \mid dl \in DL_3 \}$

$F2\ _ _ _ _ _ = \{ (nl, dl, s, p, d, x) \mid nl \in NL_2 \}$

$F3\ _ _ _ _ _ = \{ (nl, dl, s, p, d, x) \mid nl \in NL_3 \}$

2. Am încercat să îl facem să ne ofere date de test care să îmbine toate valorile de frontieră dintr-o listă dată:

nl: 2, 3, 50, 51

dl: 2, 3, 50, 51

s: 0, 1, DMV

p: 0, 1, DMV

d: -1, 0, 100, 101

NL_1 = 3...50 ; NL_2 = 2; NL_3 = 51;
DL_1 = 3...50 ; DL_2 = 2; DL_3 = 51;
S_1 = 1...DMV; S_2 = 0;
P_1 = 1...DMV; P_2 = 0;
D_1 = 0...100; D_2 = -1; D_3 = 101;

Concluzii:

Răspunsul a fost nesatisfăcător deoarece ChatGPT ne a oferit o listă infinită (la alegerea opțiunii de a continua generarea) de opțiuni de următoarea formă:

Desigur! Putem genera date de test care să acopere toate valorile de frontieră indicate. Voi crea o listă cu câte un exemplu pentru fiecare combinație posibilă de valori de frontieră:

1. nl = 2, dl = 2, s = 0, p = 0, d = -1
2. nl = 2, dl = 2, s = 1, p = 0, d = -1
3. nl = 2, dl = 2, s = DMV, p = 0, d = -1
4. nl = 2, dl = 2, s = 0, p = 1, d = -1
5. nl = 2, dl = 2, s = 1, p = 1, d = -1
6. nl = 2, dl = 2, s = DMV, p = 1, d = -1
7. nl = 2, dl = 2, s = 0, p = DMV, d = -1
8. nl = 2, dl = 2, s = 1, p = DMV, d = -1
9. nl = 2, dl = 2, s = DMV, p = DMV, d = -1
10. nl = 2, dl = 2, s = 0, p = 0, d = 0
11. nl = 2, dl = 2, s = 1, p = 0, d = 0
12. nl = 2, dl = 2, s = DMV, p = 0, d = 0
13. nl = 2, dl = 2, s = 0, p = 1, d = 0
14. nl = 2, dl = 2, s = 1, p = 1, d = 0
15. nl = 2, dl = 2, s = DMV, p = 1, d = 0

3. Am încercat să creem mutanți cu ajutorul AI

I-am oferit drept input programul normal:

**Am acest cod, si vreau ca pe baza fiecarei functii sa mi generezi cate un mutant. using AutoMapper;
using Moq;
using NUnit.Framework;
using InventoryManagerDataAccess.Entities;**

```
using InventoryManagerBusiness.Services;  
using InventoryManagerDataAccess.Interfaces;  
using InventoryManagerBusiness.DTOs;  
using Assert = NUnit.Framework.Assert;
```

```
namespace InventoryManagerServiceTests  
{  
    [TestFixture]  
    internal class ProductByCategoryTests  
    { .....
```

Concluzii:

Răspunsul a fost nesatisfăcător deoarece AI ne-a îndemnat mai degrabă spre cum ar trebui să facem asta, nu ne-a rezolvat problema efectiv. Practic doar am primit explicații despre ceea ce voiam noi să facă el...

Pentru a genera mutanți pentru fiecare funcție din testele tale, ar trebui să identifici punctele de interes în codul tău și să încerci să modifici acele puncte pentru a vedea cum influențează testele.

Iată câteva sugestii de modificări pe care le poți face pentru a genera mutanți:

Funcția

**GetByCategory_CategoryNotFound_ThrowsKey
NotFoundException**

- Schimbă valoarea lui `categoryId` la o altă valoare, sau înlătură-l complet. Testează și cazul în care `categoryId` este `null`.
- Modifică mesajul de eroare așteptat pentru a vedea dacă testul evaluează mesajul corect.

Funcția

`GetByCategory_NegativeIndex_ThrowsKeyNotFoundException`

- Înlocuiește valoarea negativă a lui `index` cu o valoare pozitivă.
- Încearcă să introduci o valoare `null` pentru `index`.

Funcția

`GetByCategory_NoProductsInCategory_ThrowsKeyNotFoundException`

- Înlocuiește categoria cu o categorie diferită, pentru a verifica dacă testul aruncă excepția corectă.
- Modifică mesajul de eroare așteptat.

Funcția

`GetByCategory_IndexGreaterThanProductCount_ThrowsKeyNotFoundException`

- Înlocuiește `index` cu o valoare mai mică decât numărul de produse disponibile.
- Modifică numărul de produse disponibile pentru a fi mai mare decât `index`.

Funcția

`GetByCategory_Success_ReturnsCorrectProductsAndDiscounts`

- Înlocuiește `categoryId` cu o altă categorie.
- Înlocuiește valorile de `index` și numărul de produse pentru a testa scenarii diferite.
- Modifică prețurile sau reducerile produselor pentru a vedea dacă acestea sunt calculate corect.
- Verifică comportamentul funcției în cazul în care `category` sau `products` sunt `null`.
- Încercați să modificați mesajul de eroare așteptat.

Acestea sunt doar câteva sugestii de modificări pe care le poți face pentru a genera mutanți în codul tău. Este important să te asiguri că testele tale sunt robuste și capabile să detecteze erori și modificări în cod.

4. Am folosit ChatGPT pentru probleme comune de teorie sau pentru a găsi diferite funcții a căror denumire nu o cunoșteam cu exactitate.

Concret, i-am dat mai multe întrebări de genul:

ca sa compar 2 liste in c#, pot zice lista1!=lista2?

Concluzii:

Răspunsurile oferite la astfel de întrebări au fost satisfăcătoare, ne-au ajutat și ne-au scutit de căutatul prin documentația C#.

Nu, pentru a compara două liste în C#, nu poți folosi operatorul != direct pentru a verifica dacă listele sunt egale sau nu. Operatorul != în C# compară referințele, nu valorile efective din listă.

Pentru a verifica dacă două liste sunt egale în conținut, trebuie să folosești metoda `SequenceEqual` din clasa `System.Linq.Enumerable`. Această metodă compară fiecare element din prima listă cu corespondentul său din a doua listă și returnează `true` dacă listele conțin aceleași elemente în aceeași ordine.

Exemplu de utilizare:

csharp

Copy code

```
using System;  
using System.Collections.Generic;  
using System.Linq;  
  
class Program  
{  
    static void Main()
```

```
{  
List<int> lista1 = new List<int> { 1, 2, 3 };  
List<int> lista2 = new List<int> { 1, 2, 3 };  
  
if (lista1.SequenceEqual(lista2))  
{  
    Console.WriteLine("Listele sunt egale.");  
}  
else  
{  
    Console.WriteLine("Listele nu sunt egale.");  
}  
}
```

Această abordare compară conținutul listelor, nu doar referințele către ele. Dacă ordinea elementelor este importantă pentru tine, poți folosi această metodă. Dacă nu este importantă, poți folosi metoda `OrderBy` pentru a sorta listele înainte de a le compara.