UNIVERSITA' DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

tesi di laurea

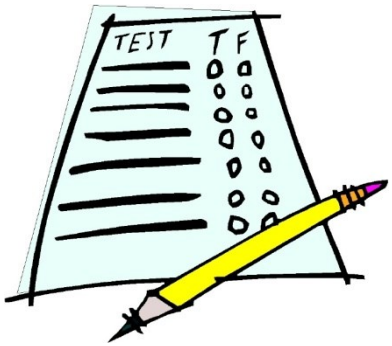
Testing di concorrenza di una applicazione web : un caso di studio

Anno Accademico 2011/12

relatore
Prof. Porfirio Tramontana

candidato
Domenico Siervo
Matr. M63/180

OBIETTIVI DELLA TESI



- Trovare casi di test per applicazioni web

- Trovare casi di test concorrenti nelle applicazioni web

- Automatizzare l'individuazione dei casi di test concorrenti

- Sviluppare un supporto all'esecuzione e ripetizione dei casi di test concorrenti

- Verifica dei risultati ottenuti

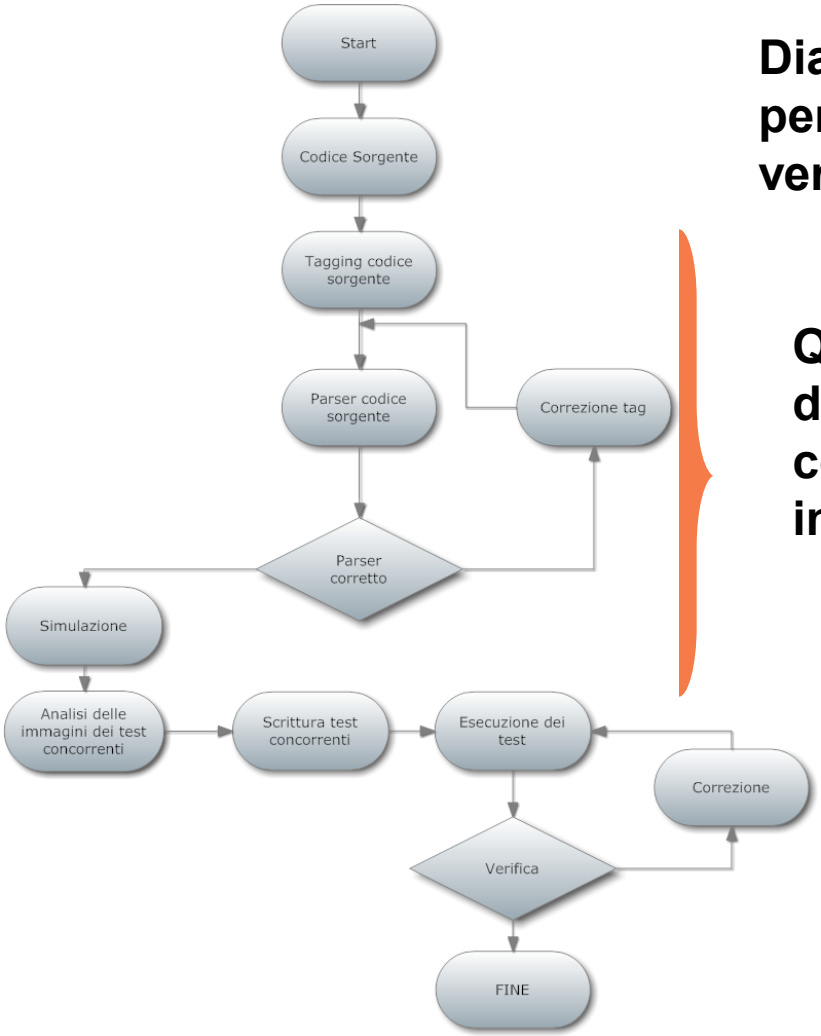
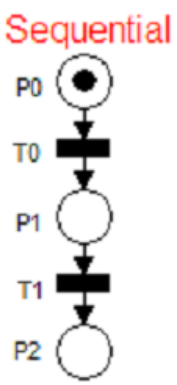


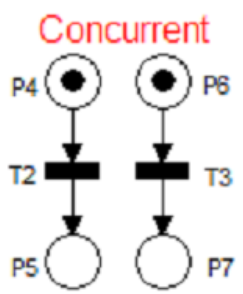
Diagramma di flusso del processo per l'individuazione , simulazione e verifica dei casi di test

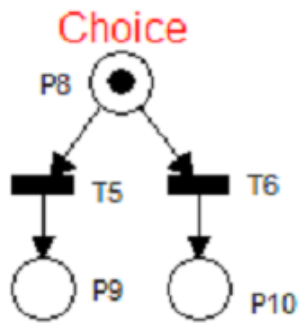
Questa parte del diagramma viene gestita dalla classe ParserNet che trasforma il codice sorgente in rete di Petri e lo simula indicando i casi di test

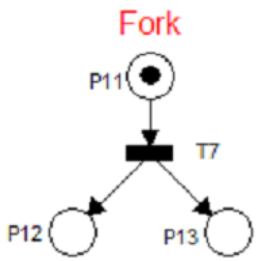
La parte di testing vera e propria viene fatta dalla libreria Unitest con l'aggiunta di Unithread per i test concorrenti

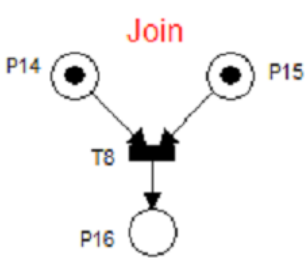
Reti di Petri

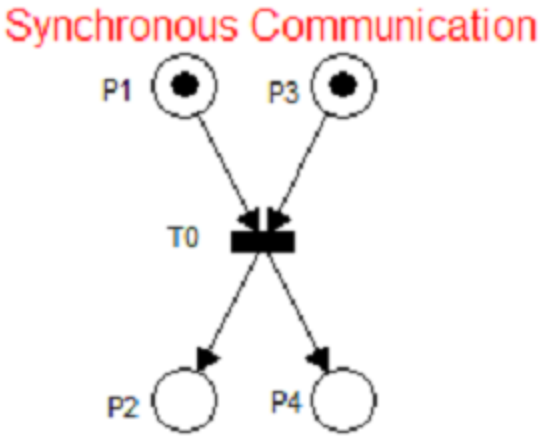


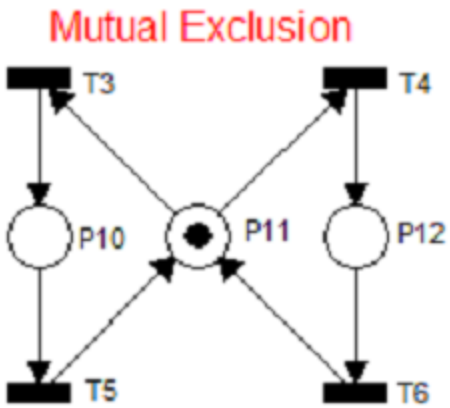


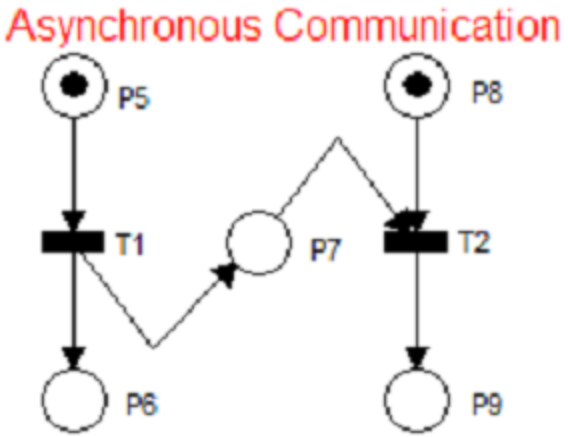




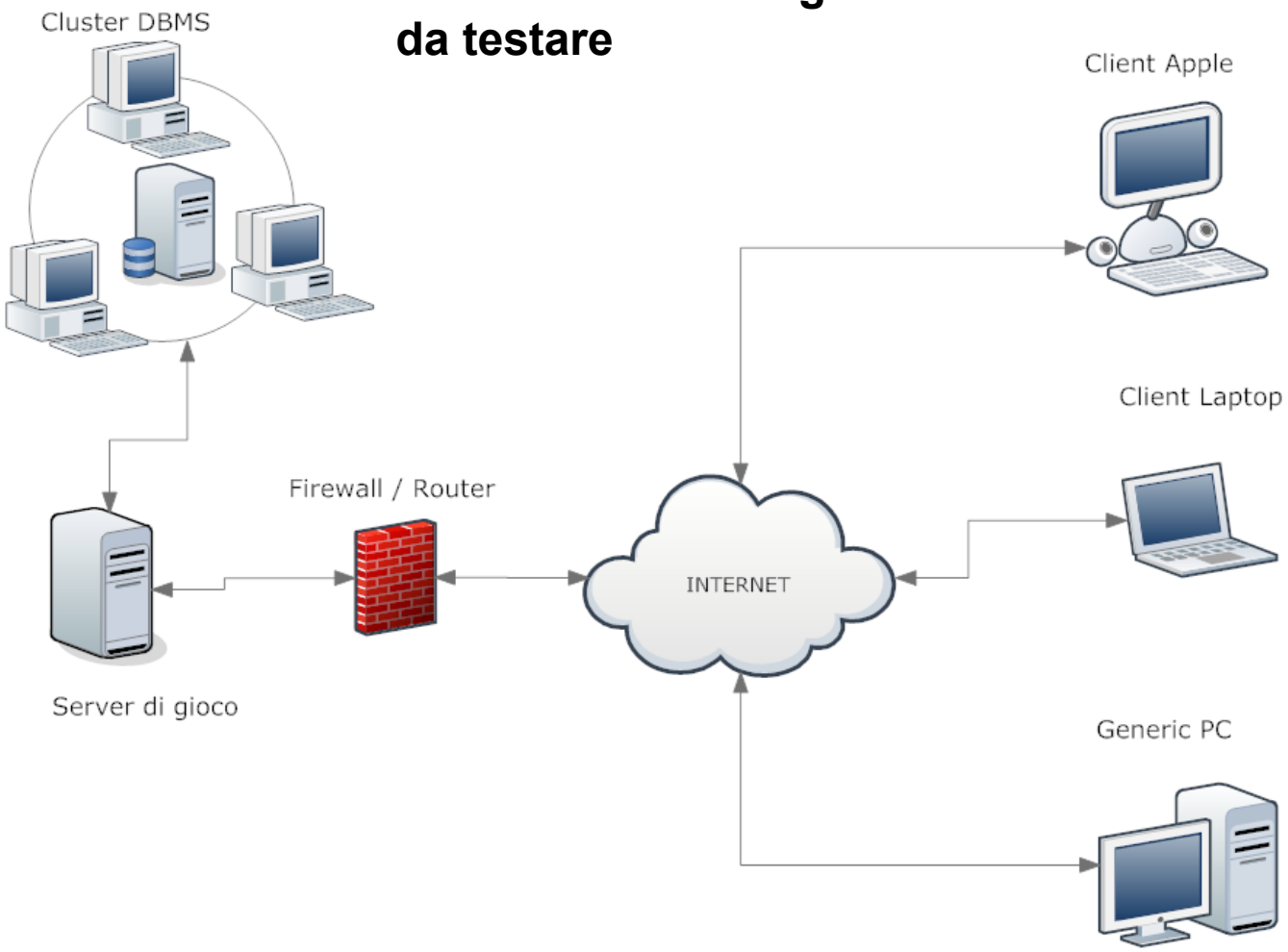




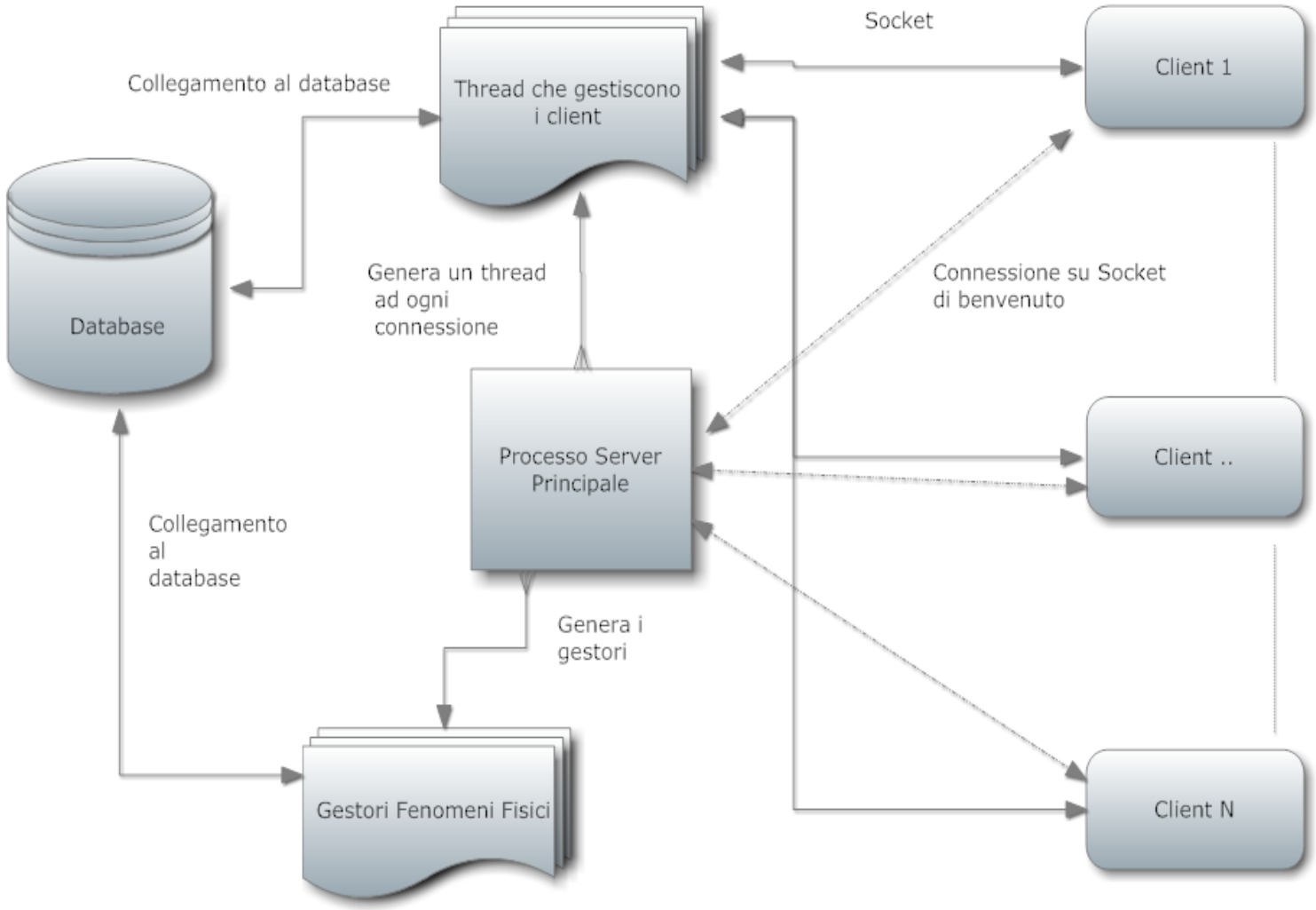




Schema fisico del gioco da testare



Testing di concorrenza di una applicazione web : un caso di studio

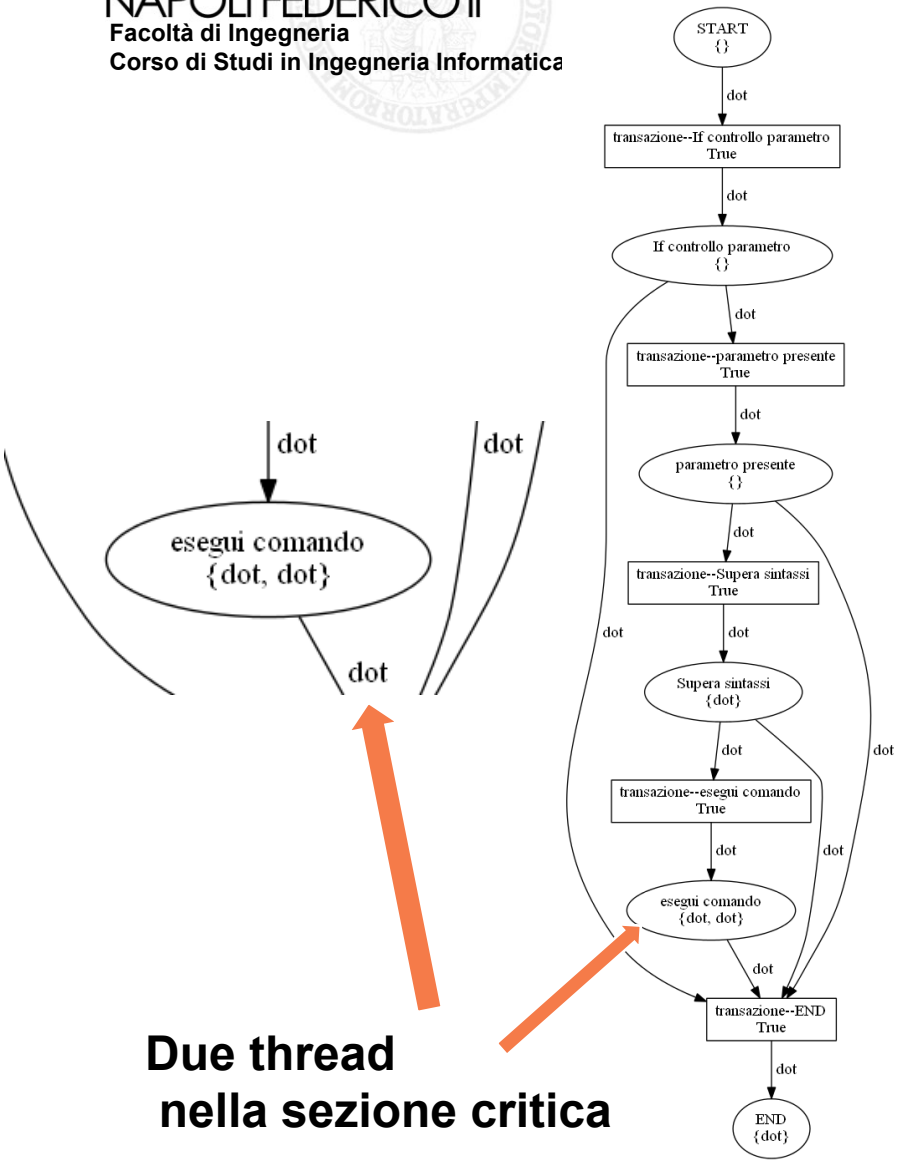


```
def prendi(cursore,Sock,ListaUt,UD,tupla):##==START [If controllo parametro]
    '''Comando usato per prendere gli oggetti e i soldi da terra o da contenitori sintassi prendi N*.ogg [N*.cogg]'''
    if tupla==None or tupla=="" :##==NODE If controllo parametro[END,parametro presente]
        tupla=('''Prova ad inserire un'oggetto!<br>''')
        pak2=ProtoServ.Imp_Tupla(tupla,'CMD')
        Astrazioni.Send_1(Sock,pak2)
    else:##==NODE parametro presente[END,Supera sintassi]
        a=string.split(tupla, " ")
        if len(a)==3:
            dove=a[2]
        else:
            dove="tutti"
        p=Astrazioni.Sintassi_2par(cursore,Sock,ListaUt,UD,tupla)
        if p!=None:##==NODE Supera sintassi[END,esegui comando]
            io=p[0] - 1
            ogg=p[1]
            punto=p[2]
            ic=p[3] - 1
            cogg=p[4]
        #esecuzione del comando
            Eprendi(cursore,Sock,ListaUt,UD, io, punto, ogg, ic, cogg, dove)##==CRITIC esegui comando[END]
    ##==END--
```

Codice da testare opportunamente taggato

```
1477 Errore di sintassi : Manca il tag START
1478 ------------------ NUOVA RIGA------------
1479
1480 ------------SIMULAZIONE----------------
1481 Stati della rete di petri:
1482 []
1483
```

```
4 ------------SIMULAZIONE----------------
5 Stati della rete di petri:
6 [Place('Supera sintassi', MultiSet([]), tAll), Place('END', MultiSet([]), tAll), Place(
  'esegui comando', MultiSet([]), tAll), Place('If controllo parametro', MultiSet([]),
  tAll), Place('START', MultiSet([dot]), tAll), Place('parametro presente', MultiSet([]),
  tAll)]
7 Attenzione testare Prendi-state4-place esegui comando-token 0.png
8 Attenzione testare Prendi-state4-place esegui comando-token 1.png
9 FINE SIMULAZIONE
```



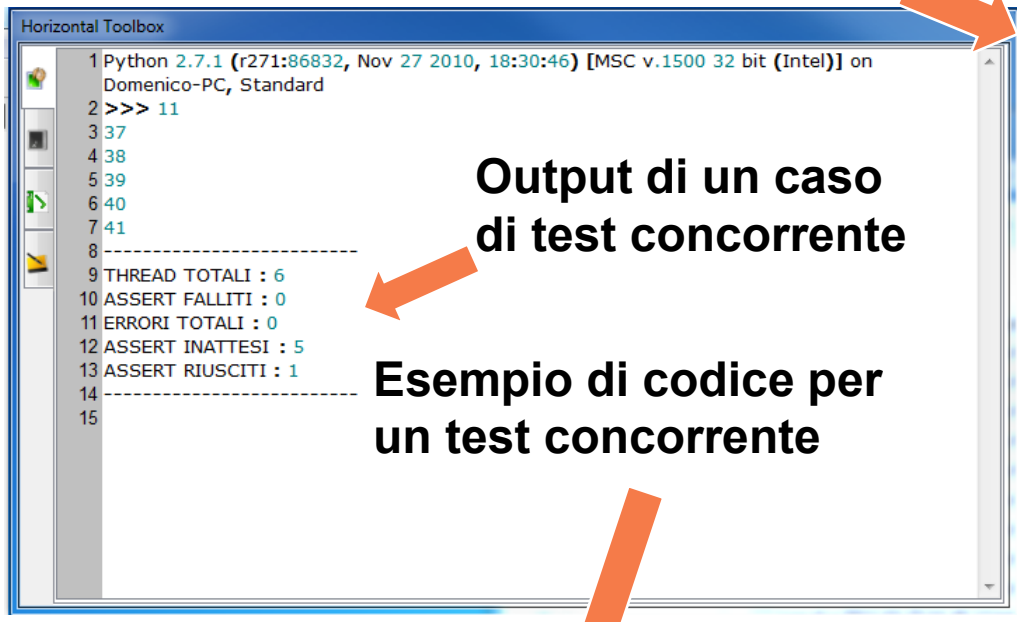
Questo è il grafo ricavato dal codice sorgente mostrato nella slide precedente dopo la simulazione in cui viene indicata la seguente immagine da analizzare in quanto sono presenti due thread che accedono alla stessa sezione critica contemporaneamente Indicati nel grafo con la dicitura "dot".

Inoltre si potrebbero ricavare altri casi di test dal grafo visualizzando tutti i percorsi che vengono effettuati dai "dot" durante la simulazione.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
Facoltà di Ingegneria
Corso di Studi in Ingegneria Informatica

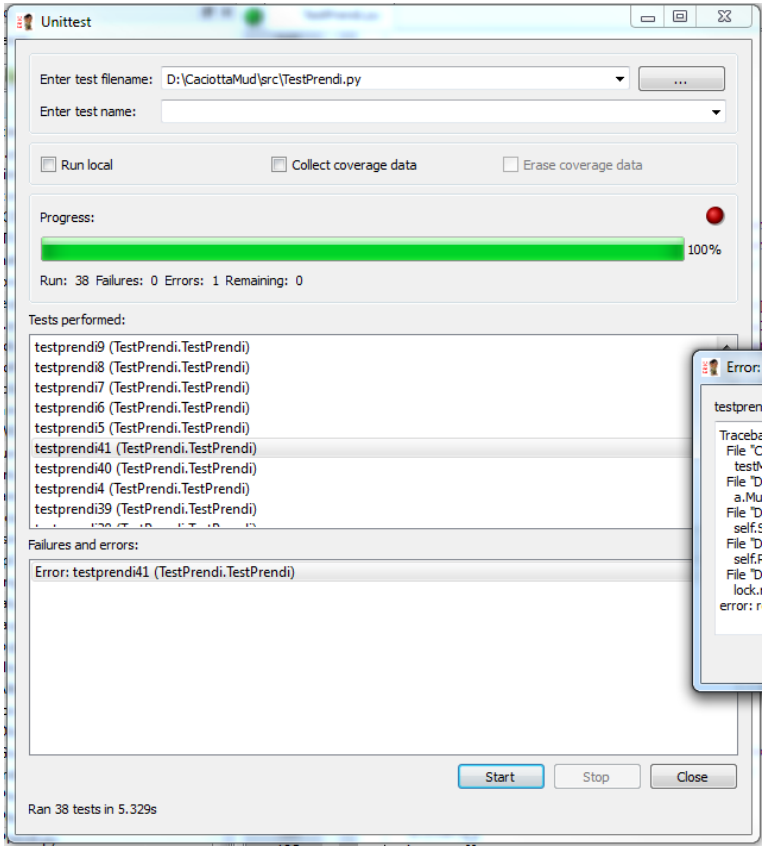
Testing di concorrenza di una applicazione web : un caso di studio

Interfaccia Unittest



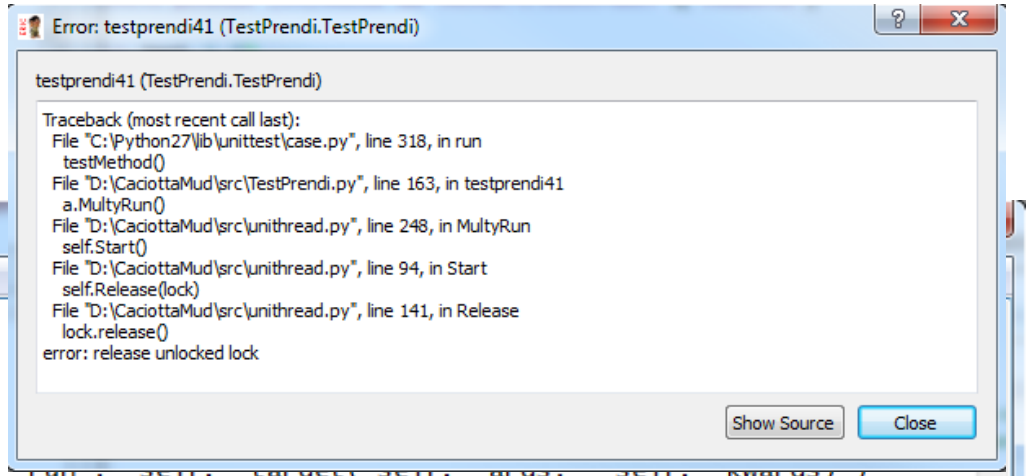
Output di un caso di test concorrente

Esempio di codice per un test concorrente




```
152     def testprendi41(self):
153         print "41"
154         t1=[self.Socka, self.Lista, 'TddAlfa', '*tor']
155         t2=[self.Sockb, self.Lista, 'TddBeta', '*tor']
156         t3=[self.Sockb, self.Lista, 'TddBeta', 'tor']
157         listain=[t1, t2, t3, t1, t2, t2]
158         test={'<font color="white">Prendi [10]torce da terra.</font><br>':['TddAlfa'],
159               '<font color="white">Non vedi nulla del genere qui !</font><br>':['TddBeta','TddAlfa'],
160               '<font color="white">TddAlfaAggettivo prende [10]torce da terra.</font><br>':['TddBeta']
161               }
162         a=exeprendi(listain,0,  'test 41', '\\Log\\', test, 1, 0)
163         a.MultyRun()
```

Output di un caso di test fallito



Stack error salvato nei log

Testing

-Una volta rilevati e scritta la classe di test possiamo ripetere nuovamente i nostri casi di test concorrenti tutte le volte che vogliamo uno dopo l' altro fino alla completa soddisfazione di tutti i casi di test , in maniera congrua alla filosofia Unittest

Unit Tests

Conclusioni

-Possiamo visualizzare il grafo del nostro software

-Possiamo ricavare casi di test concorrenti in maniera semiautomatica

-Possiamo ripetere quante volte vogliamo i nostri test concorrenti grazie alla libreria Unittest e Unitthread

-La pecca di tale tecnica consiste nel tagging del codice sorgente che oltre ad essere dispendioso in termini di tempo potrebbe portare alla creazione di errori fasulli.