Institut für Echtzeitsysteme
und Softwaretechnik
Prof. Dr. Derk Rembold

Hochschule
Albstadt-Sigmaringen
Albstadt-Sigmaringen University

2022

Praktikum Betriebssicherheit

Aufgabenblatt 2

Fehlerbaumanalyse

Anforderungen:

- Die Aufgabe wird in Python programmiert.
- Die Aufgabe wird von jedem Teilnehmer einzeln erstellt!
- Der Teilnehmer kommt rechtzeitig zur Abnahme auf den Dozenten zu. Die Abnahme erfolgt für jeden Teilnehmer einzeln. Die Kenntnis des Quellcodes wird erwartet.
- Quellcode in Python wird auf Ilias hochgeladen. Die Lokation wird im Praktikum bekanntgegeben. Das File hat folgendes Format:
    - <Name>_<Vorname>_<Matrikelnummer>_Aufgabe_1_Programmcode.py

- **Es gibt eine Frist für die Abnahme und Hochladen der Files. Diese wird im Praktikum bekanntgegeben.**
- **Alle Abgaben werden elektronisch auf Plagiat geprüft.**

Einleitung

Der Fehlerbaum ist ein Werkzeug zur logischen Verknüpfung von Komponenten und Teilsystemen. Unteranderem will man die Verfügbarkeit und die Nicht-Verfügbarkeit ermitteln, in Abhängigkeit der Ausfallraten und der Reparaturraten. Der Fehlerbaum besteht aus einer Menge von Verknüpfungselementen. Bei dieser Aufgabe soll der Standardeingang, die Nicht-Verknüpfung, die Und-Verknüpfung und die Oder-Verknüpfung behandelt werden.

Aufgabe

Es soll mit Hilfe von den Verknüpfungselementen Standardeingang, Nicht-Verknüpfung, Und-Verknüpfung und Oder-Verknüpfung ein beliebiger Fehlerbaum aufgebaut werden können. Dafür sollen die vier Elemente als Klassen definiert werden. Die Klassen sollen Listen (nodes, siehe Bild 2) als Attribute enthalten. Die Instanzen der Klassen sollen über die Listen weiteren Verknüpfungselementen aufnehmen können.

Der folgende Fehlerbaum (Bild 1) soll mit einem Programm modelliert werden:

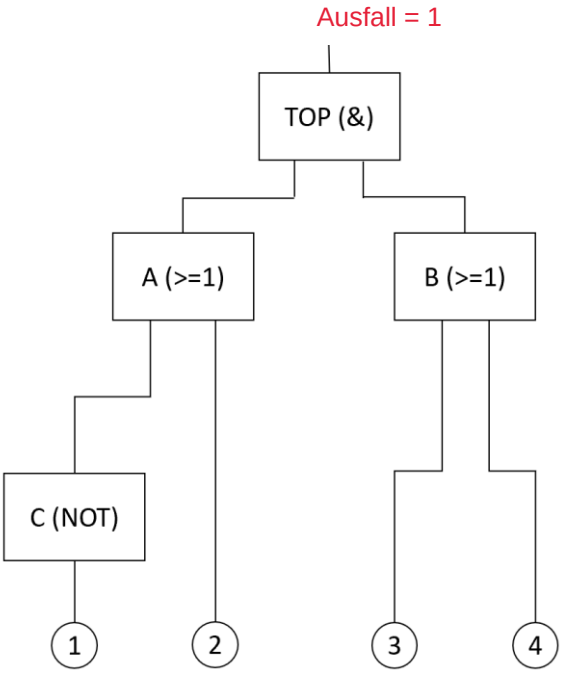

Bild 1: Fehlerbaum

TOP ist dabei ein Und-Verknüpfungselemente, A und B sind Oder-Verknüpfungselemente und C ist ein Nicht-Verknüpfungselement. Standardeingänge sind als 1, 2, 3, 4.

Die Verknüpfungselemente sollen als Klassen definiert werden mit folgender Definition, siehe Bild 2, Bild 3, Bild 4, Bild 5:

```
class ANDNODE:
    def __init__(self,name):
        self.name = name
        self.nodes=[]
    def add(self,node):
        #...
        return
    def availability(self):
        #...
        return self.avail
```

Bild 2: Und-Verknüpfungselement

```
class ORNODE:
    def __init__(self,name):
        self.name = name
        self.nodes=[]
    def add(self,node):
        #...
        return
    def availability(self):
        #...
        return self.avail
```

Bild 3: Oder-Verknüpfungselement

eine klasse für NOT und eine für EVENT (siehe nächstes Slide)

```
class NOTNODE:
    def __init__(self,name):
        self.name = name
        self.nodes=[]
    def add(self,node):
        #...
        return
    def availability(self):
        #...
        return self.avail
```

Bild 4: Nicht-Verknüpfungselement

```
class EVENT:
    def __init__(self,name, la, mu):
        self.name = name
        #...
    def add(self,node):
        #...
        return
    def availability(self):
        #...
        return self.avail
```

hier kommt verfügbarkeit basierend auf den lambdas und mus

Bild 5: Standardeingang

a.) Bauen Sie aus den instanziierten Verknüpfungselementen den Fehlerbaum aus Bild 1 auf. Verwenden Sie dabei die Methoden add, um darunterliegende Objekte an die Verknüpfungselemente anzuhängen. Beispiel:

TOP = ANDNODE(‚TOP‘)
A = ORNODE(‚A‘)
E1 = EVENT(‚1‘, 1/1000, 1/4)
name lambda mu(reparaturrate)
TOP.add(A)
TOP.add(E1)

Kontrollfrage: Wie würde der Fehlerbaum nach diesen Befehlsfolgen aussehen?

b.) Es soll aus dem Fehlerbaum die Verfügbarkeit und die Nicht-Verfügbarkeit ermittelt werden. Dafür sollen die Methoden availability implementiert werden. Die Methode availability berechnet rekursiv die Verfügbarkeit aus den Verfügbarkeiten der darunterliegenden Knoten. Beispiel:

avail = TOP.availability()

siehe Screenshots

c.) Programmieren Sie den Graphen mit Graphviz/Matplotlib/OpenCV und drucken Sie diesen, sodass ihr Graph dem Graphen aus Bild 1 gleicht.

d.) Verwenden Sie das Python Modul coverage, um nachzuweisen, dass alle Methoden durchlaufen worden sind.

=> Überdeckungsgeschichte C0 etc.

e.) Verändern Sie bei der Abnahme den Fehlerbaum und berechnen Sie in Anwesenheit des Dozenten die neue Verfügbarkeit. Der Fehlerbaum wird vom Dozenten vorgegeben. Der vom Programmcode erzeugte Graph soll dabei automatisch angepasst werden. Zeigen Sie mit Hilfe der Python coverage-Moduls, dass alle Methoden durchlaufen worden sind.