第九讲作业——基于组件的软件可信性度量模型

课程名称:	软件质量分析	指导教师:	陈仪香
姓　　名:	王海生	学　　号:	10235101559
年　　级:	2023 级	主　　题:	游戏服务器系统的可信度量

目录

1 题目要求

1. 编程实现基于组件的软件可信性度量模型。

2. 游戏服务器系统包含 13 个组件，分别是网关组件（$cp1$）、游戏大厅组件（$cp2$）、公共组件（$cp3$）、配置组件（$cp4$）、核心服务器组件（$cp5$）、总捕鱼游戏组件（$cp6$）、用户捕鱼游戏组件（$cp7$）、注册组件（$cp8$）、游戏 AI 组件（$cp9$）、后台管理组件（$cp10$）、工具组件（$cp11$）、消息组件（$cp12$）、$Maven$ 插件组件（$cp13$）。其中 $cp1 - cp5$ 为关键组件，$cp6 - cp12$ 为非关键组件。他们之间的正互反判断矩阵以及各组件的可信值见讲授 ppt 最后三页。

3. 计算在关键组件组与非关键组件组的权重分别取为 $(\alpha, \beta) = (0.7, 0.3), (0.6, 0.4), (0.55, 0.45)$ 的情况下，游戏服务器系统的可信度量值以及可信等级。

关键组件正互反判断矩阵及权重：

组件名	CP1	CP2	CP3	CP4	CP5	属性权重
CP1	1	2	1/2	2	1/4	
CP2	1/2	1	2	3	1/2	
CP3	2	1/2	1	1	1/2	
CP4	1/2	1/3	1	1	1/2	
CP5	4	2	2	2	1	

非关键组件正互反判断矩阵及权重：

组件名	CP6	CP7	CP8	CP9	CP10	CP11	CP12	CP13	属性权重
CP6	1	3	2	1/2	2	1	3	3	
CP7	1/3	1	2	1	2	2	2	2	
CP8	1/2	1/2	1	1/2	1	1/2	3	3	
CP9	2	1	2	1	3	1/2	3	3	
CP10	1/2	1/2	1	1/3	1	1	2	2	
CP11	1	1/2	2	2	1	1	2	2	
CP12	1/3	1/2	1	1/3	1/2	1/2	1	2	
CP13	1/3	1/2	2	1/3	1/2	1/2	1/2	1	

组件可信值：

组件名	CP1	CP2	CP3	CP4	CP5	CP6	CP7
可信值	8.430	8.530	6.042	9.094	8.289	6.192	8.020
组件名	CP8	CP9	CP10	CP11	CP12	CP13	
可信值	7.984	8.713	9.211	7.777	7.897	8.075	

2 相关知识点

2.1 总体思路

1. 在基于组件的软件中，软件系统性能是其组件性能的反映。为了计算软件的可信性，先要计算出组件的可信性。
2. 在本节课的 PPT 中，首先建立组件可信量化评估指标体系，将可信性分解成若干个可信属性，再将可信属性向下分解为可信证据，依据可信证据的不同组合构建度量元；
3. 其次构建基于属性的组件可信度量模型；
4. 然后构建基于组件的软件可信性度量模型。

2.2 基于组件的软件可信性度量模型

1. 软件系统中有若干组件，组件可以分为两大类，即关键组件和非关键组件。
2. 关键组件是指一个软件必须具备的组件，该组件的可信性对系统整体可信性影响较大。
3. 非关键组件是指软件自身除了关键组件外可能还需要的其他组件。
4. 关键组件和非关键组件都能影响软件可信性。但关键组件更影响软件可信度量。
5. 设软件系统 S 有 n 个关键组件 $FC_1, \cdots, FC_n$ 和 m 个非关键组件 $NFC_1, \cdots, NFC_m$，关键组件的权重分别为 $\alpha_1, \cdots, \alpha_n$，非关键组件的权重为 $\beta_1, \cdots, \beta_m$。
6. 设关键组件和非关键组件的权重分别为 α 和 β，要求 $\alpha > 1/2 > \beta > 0$。
7. 组件间没有替换性。

基于组件的软件可信性度量模型的计算如下：

$$T_s = \alpha(FC_1^{\alpha_1} \times \cdots \times FC_n^{\alpha_n}) + \beta(NFC_1^{\beta_1} \times \cdots \times NFC_m^{\beta_m})$$

可信等级可以用下面表格中的方法确定：

软件可信度量值要求	可信属性要求	可信等级
$9.5 \leq T$	1. 低于 9.5 分的关键组件个数不超过 $n - [n \times 2/3]$ 2. 没有低于 8.5 分的组件	V
$8.5 \leq T < 9.5$ 或者 $T > 9.5$ 且不能评为 V 级别	1. 低于 8.5 分的关键组件个数不超过 $n - [n \times 2/3]$ 2. 没有低于 7.0 分的组件	IV
$7.0 \leq T < 8.5$ 或者 $T > 8.5$ 且不能评为 IV 级别及以上者	1. 低于 7.0 分的关键组件个数不超过 $n - [n \times 2/3]$ 2. 没有低于 4.5 分的可信属性	III
$4.5 \leq T < 7.0$ 或者 $T > 7.0$ 且不能评为 III 级别及以上者	1. 低于 4.5 分的关键组件个数不超过 $n - [n \times 2/3]$	II
$T < 4.5$ 或者 $T > 4.5$ 且不能评为 II 级别及以上者	无要求	I

2.3 拓展：层次分析法

2.3.1 基本原理

层次分析法（Analytic Hierarchy Process, AHP）是由美国运筹学家托马斯 • 萨蒂（Thomas L. Saaty）在 20 世纪 70 年代提出的一种用于多准则决策的数学工具。它通过将复杂的问题结构化，分解为一个层次化的模型，并利用两两比较的方式评估各元素之间的相对重要性，从而计算出各个属性或因素的权重。

在 AHP 中，正互反判断矩阵是表示成对比较结果的矩阵，具有以下特性：

- 它是一个方阵，假设我们有 n 个属性，则矩阵大小为 $n \times n$。
- 矩阵中的元素 a_{ij} 表示第 i 个属性相对于第 j 个属性的重要性比率，通常使用 Saaty 的 1-9 标度来量化。
- 对角线上的元素总是 1，因为每个属性相对于自身的比较结果应该是相等的。
- 它满足互反性，即如果 $a_{ij} = k$，则 $a_{ji} = 1/k$。

2.3.2 属性权重的计算

为了从正互反判断矩阵计算属性的权重，可以遵循以下步骤：

步骤 1：构建判断矩阵。根据专家意见或相关数据，建立所有属性之间的成对比较矩阵。

步骤 2：计算几何平均值。对于每一行，计算其所有元素的几何平均值。对于第 i 行，权重 w_i 可以通过下面的公式计算：$w_i = (a_{i1} \times a_{i2} \times \cdots \times a_{in})^{\frac{1}{n}}$

步骤 3：归一化处理。由于几何平均值可能不会自然地形成一个总和为 1 的权重向量，因此需要进行归一化处理。归一化后的权重 W_i 可以通过下面的公式计算：$W_i = \frac{w_i}{\sum_{k=1}^{n} w_k}$

步骤 4：一致性检验。计算一致性比率（Consistency Ratio, CR），以检验判断矩阵的一致性。如果 CR 小于 0.1，一般认为判断矩阵具有满意的一致性；否则，可能需要重新评估和调整判断矩阵。一致性指标（Consistency Index, CI）和随机一致性指标（Random Consistency Index, RI）用于计算 CR：$CR = \frac{CI}{RI}$ 其中 CI 由最大特征根 λ_{max} 得出，而 RI 依赖于矩阵的维度并且是预定义的。

步骤 5：确定权重。如果一致性检验通过，那么归一化后的向量即为最终的权重向量，代表了每个属性的相对重要性。

2.3.3 属性间相对重要性的量化

Saaty 的 1-9 标度提供了一种量化属性间相对重要性的方法，具体如下：

- 1 表示两个属性同等重要；
- 3 表示一个属性稍微重要于另一个；
- 5 表示一个属性明显重要于另一个；
- 7 表示一个属性强烈重要于另一个；
- 9 表示一个属性极端重要于另一个；
- 2, 4, 6, 8 是上述相邻判断的中间值；
- 倒数用于表示相反的重要性关系。

3 题目解析

3.1 计算关键组件和非关键组件的属性权重

代码：

```
import numpy as np

# 计算属性权重的函数，传入正互反判断矩阵
def calculate_attribute_weights(matrix):
        n = matrix.shape[0]

        # 计算几何平均值
        geometric_means = np.power(np.prod(matrix, axis=1), 1 / n)

        # 归一化处理
        weights = geometric_means / np.sum(geometric_means)
        return weights
```

```
13
14      # 1) 计算关键组件和非关键组件的属性权重
15      key_weights = calculate_attribute_weights(key_components_matrix)
16      non_key_weights = calculate_attribute_weights(non_key_components_matrix)
17
18      print("关键组件属性权重:", key_weights)
19      print("非关键组件属性权重:", non_key_weights)
```

计算结果：

```
1       关键组件属性权重: [0.16020622 0.19957385 0.16020622 0.11195645 0.36805725]
2       非关键组件属性权重: [0.18373096 0.1500157  0.10384807 0.18373096 0.09727315
            0.14471694 0.07130095 0.06538326]
```

将属性权重填入表格中：

关键组件正互反判断矩阵及权重：

组件名	CP1	CP2	CP3	CP4	CP5	属性权重
CP1	1	2	1/2	2	1/4	0.16020622
CP2	1/2	1	2	3	1/2	0.19957385
CP3	2	1/2	1	1	1/2	0.16020622
CP4	1/2	1/3	1	1	1/2	0.11195645
CP5	4	2	2	2	1	0.36805725

非关键组件正互反判断矩阵及权重：

组件名	CP6	CP7	CP8	CP9	CP10	CP11	CP12	CP13	属性权重
CP6	1	3	2	1/2	2	1	3	3	0.18373096
CP7	1/3	1	2	1	2	2	2	2	0.1500157
CP8	1/2	1/2	1	1/2	1	1/2	3	3	0.10384807
CP9	2	1	2	1	3	1/2	3	3	0.18373096
CP10	1/2	1/2	1	1/3	1	1	2	2	0.09727315
CP11	1	1/2	2	2	1	1	2	2	0.14471694
CP12	1/3	1/2	1	1/3	1/2	1/2	1	2	0.07130095
CP13	1/3	1/2	2	1/3	1/2	1/2	1/2	1	0.06538326

3.2 计算游戏服务器系统的可信度量值 T_s

代码：

```
# 2) 计算游戏服务器系统的可信度量值T_s
key_component_product = np.prod(np.power(component_trust_values[:5], key_weights))
non_key_component_product = np.prod(np.power(component_trust_values[5:], non_key_weights))

T_s = alpha * key_component_product + beta * non_key_component_product

print(f"当(α, β)=({alpha}, {beta})时, 可信度量值T_s:", T_s)
```

计算结果：

```
当(α, β)=(0.7, 0.3)时, 可信度量值T_s: 7.968354549775479
当(α, β)=(0.6, 0.4)时, 可信度量值T_s: 7.948107650109263
当(α, β)=(0.6, 0.4)时, 可信度量值T_s: 7.948107650109263
```

3.3 确定可信等级

代码：

```
# 根据可信度量值确定可信等级的函数
def determine_trust_level(T_s):
        if 9.5 <= T_s:
                return "V"
        elif 8.5 <= T_s < 9.5:
                return "IV"
        elif 7.0 <= T_s < 8.5:
                return "III"
        elif 4.5 <= T_s < 7.0:
                return "II"
        else:
                return "I"

level = determine_trust_level(T_s)
print(f"可信等级: {level}")
```

计算结果：

```
当(α, β)=(0.7, 0.3)时, 可信度量值T_s: 7.968354549775479
可信等级: III
当(α, β)=(0.6, 0.4)时, 可信度量值T_s: 7.948107650109263
```

```
4       可信等级: III
5       当(α, β)=(0.55, 0.45)时, 可信度量值T_s: 7.937984200276155
6       可信等级: III
```

4 结果总结

代码运行结果如下图所示：

```
C:\Users\hswan\.conda\envs\temp\python.exe C:\Users\hswan\Desktop\Coding\Temp\homework.py
关键组件属性权重: [0.16020622 0.19957385 0.16020622 0.11195645 0.36805725]
非关键组件属性权重: [0.18373096 0.1500157  0.10384807 0.18373096 0.09727315 0.14471694
 0.07130095 0.06538326]
当(α, β)=(0.7, 0.3)时，可信度量值T_s: 7.968354549775479
可信等级: III
当(α, β)=(0.6, 0.4)时，可信度量值T_s: 7.948107650109263
可信等级: III
当(α, β)=(0.55, 0.45)时，可信度量值T_s: 7.937984200276155
可信等级: III

Process finished with exit code 0
```

图 1: 运行结果

5 完整代码

```
1       import numpy as np
2
3
4       # 计算属性权重的函数, 传入正互反判断矩阵
5       def calculate_attribute_weights(matrix):
6               n = matrix.shape[0]
7
8               # 计算几何平均值
9               geometric_means = np.power(np.prod(matrix, axis=1), 1 / n)
10
11              # 归一化处理
12              weights = geometric_means / np.sum(geometric_means)
13              return weights
14
15
16      # 根据可信度量值确定可信等级的函数
```

```
17      def determine_trust_level(T_s):
18              if 9.5 <= T_s:
19                      return "V"
20              elif 8.5 <= T_s < 9.5:
21                      return "IV"
22              elif 7.0 <= T_s < 8.5:
23                      return "III"
24              elif 4.5 <= T_s < 7.0:
25                      return "II"
26              else:
27                      return "I"
28
29
30      # 关键组件正互反判断矩阵
31      key_components_matrix = np.array([
32              [1, 2, 1 / 2, 2, 1 / 4],
33              [1 / 2, 1, 2, 3, 1 / 2],
34              [2, 1 / 2, 1, 1, 1 / 2],
35              [1 / 2, 1 / 3, 1, 1, 1 / 2],
36              [4, 2, 2, 2, 1]
37      ])
38
39      # 非关键组件正互反判断矩阵
40      non_key_components_matrix = np.array([
41              [1, 3, 2, 1 / 2, 2, 1, 3, 3],
42              [1 / 3, 1, 2, 1, 2, 2, 2, 2],
43              [1 / 2, 1 / 2, 1, 1 / 2, 1, 1 / 2, 3, 3],
44              [2, 1, 2, 1, 3, 1 / 2, 3, 3],
45              [1 / 2, 1 / 2, 1, 1 / 3, 1, 1, 2, 2],
46              [1, 1 / 2, 2, 2, 1, 1, 2, 2],
47              [1 / 3, 1 / 2, 1, 1 / 3, 1 / 2, 1 / 2, 1, 2],
48              [1 / 3, 1 / 2, 2, 1 / 3, 1 / 2, 1 / 2, 1 / 2, 1]
49      ])
50
51      # 组件可信值
52      component_trust_values = np.array([8.430, 8.530, 6.042, 9.094, 8.289, 6.192,
            8.020,
53      7.984, 8.713, 9.211, 7.777, 7.897, 8.075])
54
55      # 1) 计算关键组件和非关键组件的属性权重
56      key_weights = calculate_attribute_weights(key_components_matrix)
57      non_key_weights = calculate_attribute_weights(non_key_components_matrix)
58
59      print("关键组件属性权重:", key_weights)
```

```
60      print("非关键组件属性权重:", non_key_weights)
61
62      # 关键组件和非关键组件组的权重取值情况
63      alpha_beta_list = [(0.7, 0.3), (0.6, 0.4), (0.55, 0.45)]
64      for alpha, beta in alpha_beta_list:
65
66      # 2) 计算游戏服务器系统的可信度量值T_s
67      key_component_product = np.prod(np.power(component_trust_values[:5],
            key_weights))
68      non_key_component_product = np.prod(np.power(component_trust_values[5:],
            non_key_weights))
69
70      T_s = alpha * key_component_product + beta * non_key_component_product
71
72      print(f"当(α,␣β)=({alpha},␣{beta})时, 可信度量值T_s:", T_s)
73
74      # 3) 确定可信等级
75      level = determine_trust_level(T_s)
76      print(f"可信等级:␣{level}")
```