第十二次作业——软件老化

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:	软件老化

目录

1 软件老化可信度量理论

1.1 可信计算框架

为了评估软件系统随时间老化而降低其可信度，我们采用了一种称为可信计算框架的方法。该框架从研究软件老化相关的 bug 出发，通过提取失信证据、设计度量元，并根据这些度量元计算各个类别的可信度，最终汇总成一个综合的软件可信度量模型。

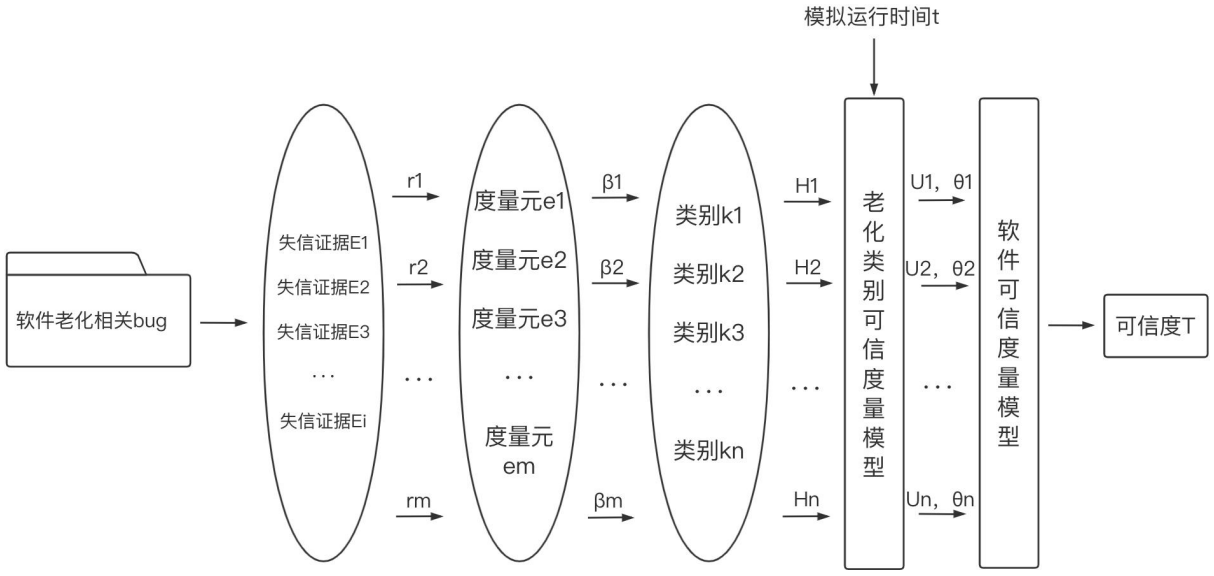

图 1: 可信计算框架

具体步骤如下：

1. **提取失信证据**

   首先，从公共漏洞库中搜集与软件老化有关的 bug 报告。例如，在以下四个开源软件系统中收集了大量错误报告：

   - Linux 内核（Linux 2.6）：`https://bugzilla.kernel.org`
   - MySQL 数据库管理系统（MySQL 5.1）：`http://bugs.mysql.com`
   - Apache HTTPD 服务器（Apache HTTPD 2）：`https://issues.apache.org/bugzilla`
   - Apache AXIS 框架: `https://issues.apache.org/jira/secure/IssueNavigator.jspa`

2. **设计度量元**

   基于上述失信证据，定义一系列度量元来量化不同类型的软件老化问题。每个度量元反映了特定的老化特征，比如内存泄漏、数值误差累积等。例如，针对浮点数寄存器误差和系统功能异常两个度量元（ARU 类别），分别有 1 个实例（爱国者导弹事件）和 3 个实例（Linux3134, Linux5870, Linux7959）。

3. **分类并计算各分类的可信度**

   将所有度量元按照导致软件老化的原因分成五个主要类别：MEM（内存管理）、STO（存储空间）、LOG（逻辑资源）、NUM（误差累积）及 ARU（未知原因）。然后为每种类别计算其熵值 $H_i(t)$，即该类别在时刻 t 的信息不确定性，以及对应的可信度 $U_i(t)$。

4. **度量元权重计算 β_i**

对于每一个度量元，我们需要确定它的相对重要性 β_i。这可以通过比较同一类别内各个度量元的重要性来完成。例如，在 ARU 类别中，“浮点数寄存器误差”对应了一个实例，“系统功能异常”则对应了三个实例。因此，它们之间的权重分别为：

$$\beta_1 = \frac{(1/3)}{(10/3)} = 0.10, \quad \beta_2 = \frac{(3)}{(10/3)} = 0.90$$

5. **输入模拟运行时间，得到预期时间点的软件可信度**

   使用公式预测给定时间点 t 时软件的可信度 $T(t)$。这里以某一具体类别为例，说明如何利用熵 $H_i(t)$ 计算 $K_i(t)$，进而获得整个系统的可信度 $T(t)$。

6. **汇总综合的软件可信度**

   最后，将所有类别的可信度加权求和，得到综合的软件可信度 $T(t)$，其中 θ_i 代表各类别的权重。公式如下：

$$T(t) = \sum_{i=1}^{n} \theta_i \times U_i(t)$$

其中，$U_i(t)$ 是第 i 个类别在时刻 t 的可信度，θ_i 是该类别的权重。

通过以上步骤，我们可以建立一个完整的软件可信度计算框架，用以评估长期运行后的软件系统的可靠性和稳定性。

1.2 度量元权重计算

度量元权重计算是软件老化可信度量模型中的第一个重要环节，其目的是确定各个度量元在评估软件老化可信度时的相对重要程度。详细分析如下：

1. **计算原理：**

   权重 β_i 代表了各度量元与同类型度量元相比的相对重要程度，数值越大说明该度量元对分类可信度的影响也就越大。通过每个度量元在统计数据中出现次数之比反应各度量元重要性的不同，该度量元对应的错误报告越多，说明该度量元对应的软件老化问题出现的越频繁，那这个度量元就越重要，对分类可信度的影响也就越大。

2. **具体计算方法（以类别 ARU 为例）：**

   (a) 类别 ARU 共有 2 个度量元：$A1$ 浮点数寄存器误差（1），$A2$ 系统功能异常（3）。括号内数字表示该度量元包含的实例数目。

   (b) 构造矩阵如下：

	$A1(1)$浮点数寄存器误差	$A2(3)$系统功能异常	行总计	(权重i)
$A1(1)$		$\frac{1}{3}$	$\frac{1}{3}$	0.10
$A2(3)$	$\frac{3}{1}$		3	0.90
列总计	3	$\frac{1}{3}$	$\frac{10}{3}$	

(c) 计算权重 β_1 和 β_2：$\beta_1 = \frac{\frac{1}{3}}{\frac{10}{3}} = 0.10$，$\beta_2 = \frac{\frac{3}{10}}{\frac{3}{3}} = 0.90$

3. **通用公式推导：**

(a) 设类别 k 中有 m 个度量元，分别为 $e_1, e_2, \cdots, e_m$，其对应的实例数目分别为 $n_1, n_2, \cdots, n_m$。

(b) 对于度量元 e_i，其在矩阵中的行总计为 $r_i = \sum_{j=1}^{m} \frac{n_i}{n_j}$（当 $i = j$ 时，$\frac{n_i}{n_j} = 1$）。

(c) 该类别所有行总计之和为 $R = \sum_{i=1}^{m} r_i$。

(d) 则度量元 e_i 的权重 $\beta_i = \frac{r_i}{R}$。

在实际计算中，对于不同的软件老化原因分类（如 MEM、STO、LOG、NUM 等），均按照上述方法分别计算其包含的度量元权重。例如，在类别 LOG 中，有 7 个度量元，通过类似的计算过程得出每个度量元的权重，如嵌套字功能异常 $(\beta_1) = 0.2340$、适配器选择错误 $(\beta_2) = 0.1064$ 等。这种计算方式能够合理地反映各个度量元在软件老化评估过程中的重要性差异，为后续的软件可信度计算提供重要依据。

1.3 熵值计算

度量元熵值计算是软件老化可信度量模型中的第二个重要环节，用于衡量软件运行状态的混乱程度。

1. **度量元熵值的定义与计算原理：**

基于软件老化的失信证据度量元的熵是在单一该度量元影响下，表示软件运行状态混乱程度的量。其考虑了预期内运行状态和风险状态，因为风险状态越多软件运行偏离预期的可能性越大，系统越混乱，熵就会增大。熵值计算公式为

$$S = \log_{10} r(t)$$

其中 $r(t)$ 为度量元对应的最大风险值。由于度量元可能有两个风险值，这里取最大风险值进行计算。例如在一些实际案例中，像 Linux 的某些缺陷导致的度量元，其风险值可根据具体的软件占用资源情况、对系统性能影响时间等因素确定，进而通过该公式计算出熵值。

2. **类别熵值的计算方法：**

软件老化原因分类的熵是在由同一种软件老化原因导致的基于软件老化的失信证据度量元影响下，衡量软件运行状态混乱程度的量，且是软件运行时间的函数。其计算公式为

$$H(t) = \sum_{i=1}^{n} \beta_i \log_{10} r_i(t)$$

其中 β_i 为第 i 个度量元的权重（采用 Brassard 的计算方式），$r_i(t)$ 为第 i 个度量元的最大风险值。以 MEM 类别为例，如果有多个与内存管理相关的度量元，如内存需求过大、内存泄漏等，先确定每个度量元的权重 β_i 和最大风险值 $r_i(t)$，然后按照公式计算该类别在时刻 t 的熵值。单一类型的熵值越高，代表该类型度量元可能引发软件老化的风险越大，软件可信度随时间降低的速度也就越快。

1.4 类别可信度计算

类别可信度计算是软件老化可信度量模型中的第三个重要环节，用于评估不同软件老化原因类别对软件可信度的影响。其具体计算过程为：

1. 设计得到老化类别可信度量模型，各类别可信度与时间 t 的关系函数为

$$K_i(t) = e^{-H_i(t)}$$

其中 $K_i(t)$ 为第 i 个类别在时刻 t 的可信度，$H_i(t)$ 为第 i 个软件老化类别在时刻 t 的熵，t 为软件运行时间。

2. 为了优化可信度的值域，将各类别 i 可信度对软件可信度分类可信度的值域 $[0,1]$ 进行调整，采用十分制，值域为 $[1,10]$。因此 i 类别可信度的十分制表示为

$$U_i(t) = \begin{cases} 10e^{-H_i(t)}, 0.1 \le e^{-H_i(t)} \le 1 \\ 1, 0 \le e^{-H_i(t)} < 0.1 \end{cases}$$

1.5 软件可信度计算

软件可信度计算是软件老化可信度量模型中的最后一个重要环节，其模型为：

$$T(t) = \begin{cases} \prod_{i=1}^{n} U_i(t)^{\theta_i}, & 1 \le U_i(t) \le 10 \\ \sum_{i=1}^{n} \theta_i = 1, & 0 \le \theta_i \le 1 \end{cases}$$

在此公式中，$\theta_i (i = 1, \cdots, n)$ 代表各类别的权重，这些权重反映了不同类别在软件可信度评估体系中的相对重要程度。通过将各类别经过前面一系列计算所得到的可信度 $U_i(t)$，按照其对应的权重 θ_i 进行幂运算后连乘（当 $1 \le U_i(t) \le 10$ 时），或者在满足权重和为 1 且 $0 \le \theta_i \le 1$ 的条件下进行相关运算，最终实现将各分类可信度汇总整合，从而得出软件在时刻 t 的综合可信度 $T(t)$。

2 编程计算软件可信属性

2.1 题目

编程或使用课程提供的 Python 代码，实现下列功能：

- **输入**：类别，以及类别中每个度量元的含例子个数。
- **输出**：在时刻 `t = 0` 和 `t = 10` 情况下，四个类别的熵和可信度，以及软件可信度.

已知的可信属性如下图所示：

MEM	M1内存需求过大	M2资源占用	M3权限管理错误	M4程序异常繁殖	M5空指针相关问题	M6内存越界	M7缓冲区溢出	M8删除表未释放内存	M9引用未清空	M10循环过度	M11异常终止	M12回滚机制问题	M13僵尸进程过多	M14高并发缓存问题	M15内存泄漏
权重	0.0506	0.1845	0.0238	0.0238	0.0774	0.0774	0.0774	0.0238	0.0238	0.0506	0.0506	0.0238	0.0238	0.1042	0.1845
t=0	**4**	4,1	1	1,4	1	1	4	1,1	1	1	4,1	1,1	1,4	1	4
t=10	**4**	4,7	7	4,4	9	4	7	4,7	7	4	7,7	7,4	7，4	9	7
STO	S1磁盘空间泄露		S2索引节点爆满		S3数据插入问题		S4写入失败								
权重	0.25		0.25		0.25		0.25								
t=0	**1,4**		1,1		4		4								
t=10	**4,7**		7,9		7		7								
LOG	L1 嵌套字功能异常		L2未刷新缓存		L3死锁		L4无效更新		L5临时文件未及时清理		L6适配器选择错误		L7嵌套字用尽		
权重	0.2340		0.1064		0.1064		0.2340		0.1064		0.1064		0.1064		
t=0	**1**		1		4		1		1		1		7		
t=10	**7**		10		7		4		7		10		9		
NUM	N1数值溢出		N2舍入误差累计												
权重	0.2		0.8												
t=0	4		1												
t=10	**7**		4												
ARU	A1浮点数寄存器误差		A2系统功能异常												
权重	0.10		0.90												
t=0	1		1,4												
t=10	7		7,7												

图 2: 已知可信属性

2.2 代码

根据软件老化可信度量模型，编写 Python 代码如下：

```python
import math

n = int(input("类别数："))

theta = []
theta_str = input("各个类别的权重：").split("␣")
for t in theta_str:
theta.append(float(t))

m = []
m_str = input("各个类别的度量元数量：").split("␣")
for num in m_str:
m.append(int(num))

R = [] # 存储权重和最大风险
BETA = []

for i in range(n):
beta = []
beta_str = input("第{0}个类别-各个度量元的权重：".format(i + 1)).split("␣")
for b in beta_str:
```

```
22      beta.append(float(b))
23      BETA.append(beta)
24
25      r = []
26      r_str = input("第{0}个类别-各个度量元的该时刻最大风险值：".format(i + 1)).
            split("␣")
27      for value in r_str:
28      r.append(float(value))
29      R.append(r)
30
31      Hs = [] # 熵
32      Us = [] # 类别可信值
33
34      for i in range(n):
35      H = 0
36      for j in range(m[i]):
37      H += BETA[i][j] * math.log10(R[i][j])
38      U = max(10 * math.exp(-H), 1)
39      Hs.append(H)
40      Us.append(U)
41
42      print("各类别的熵：{0}".format(Hs))
43      print("各类别的可信值：{0}".format(Us))
44
45      T = 1
46      for i in range(n):
47      T *= math.pow(Us[i], theta[i])
48      print("可信值：{0}".format(T))
```

2.3 结果

当 `t = 0` 时，代码运行结果如下图所示：

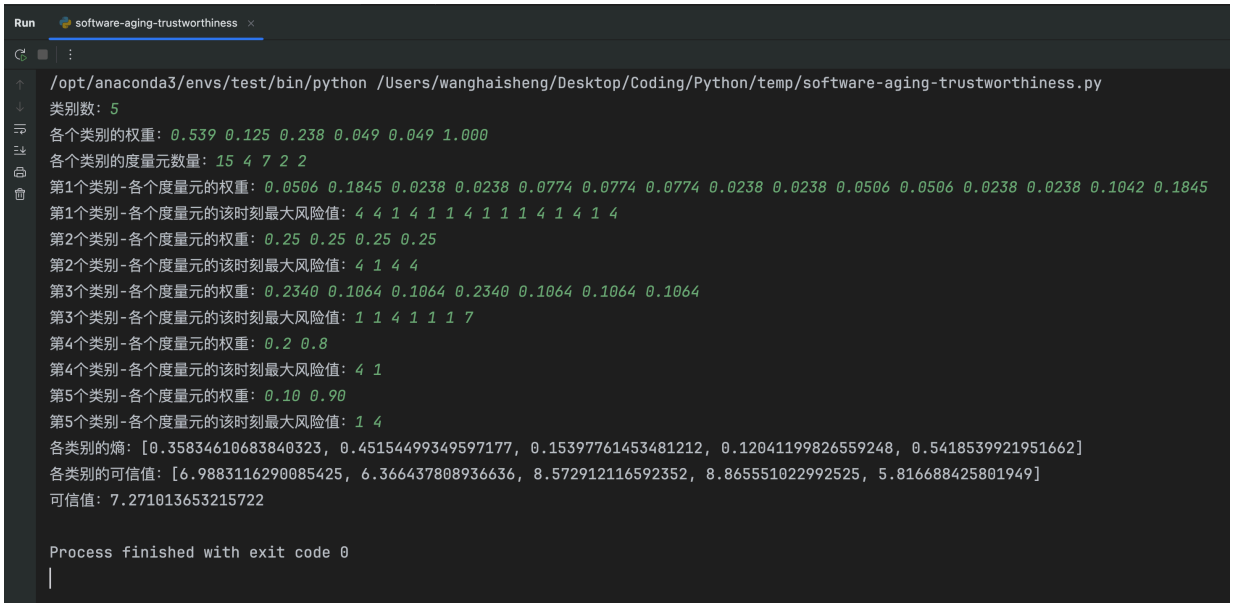

图 3: `t = 0` 运行结果

当 `t = 10` 时，代码运行结果如下图所示：

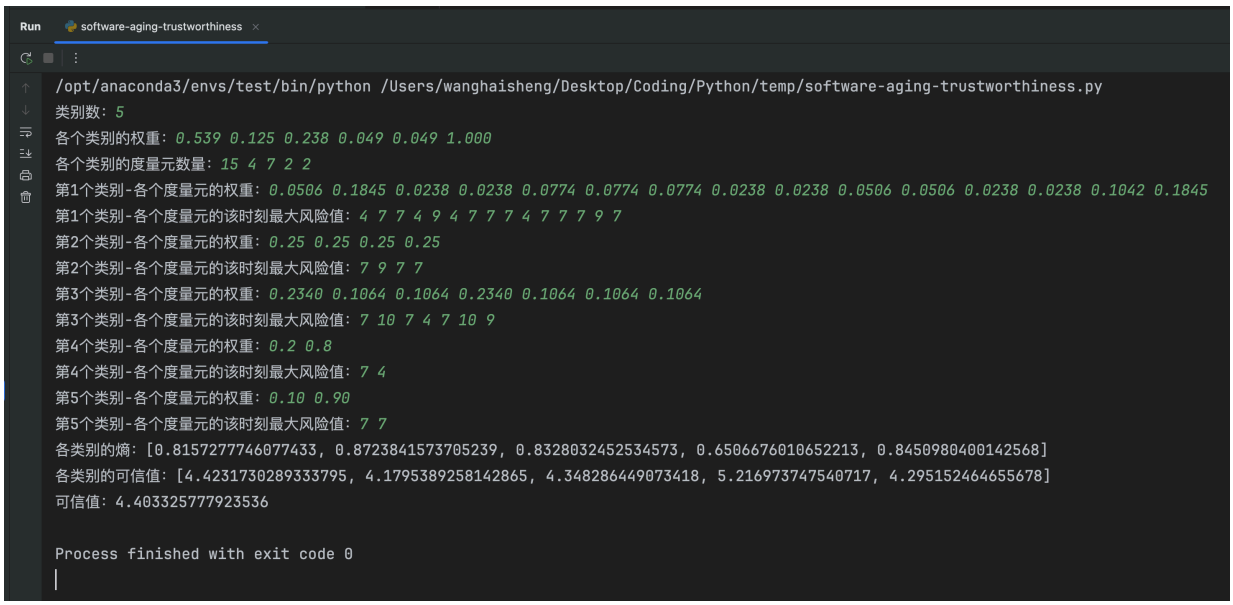

图 4: `t = 10` 运行结果

由此可得，四个类别的熵和可信度如下表所示：

	时间	MEM	STO	LOG	NUM	ARU
熵	t=0	0.358	0.452	0.154	0.120	0.542
	t=10	0.816	0.872	0.833	0.651	0.845
可信度	t=0	6.988	6.366	8.573	8.866	5.817
	t=10	4.423	4.180	4.348	5.217	4.295

可信度如下表所示：

时间	软件可信度
t=0	7.271
t=10	4.403