第七章面向组件的软件可信性度量模型

陈仪香

2024年11月18日

Outline

1. 7.1 指标体系
2. 7.2 基于组件的软件可信性度量模型

第七章面向组件的软件可信性度量模型

- 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。
- 本章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；
- 其次构建基于属性的组件可信度量模型；
- 然后构建基于组件的软件可信性度量模型；
- 最后通过示例展现软件可信性度量的过程。

7.1 组件可信量化评估指标体系

可信属性选取

结合组件特性选取了可靠性、安全性、可维护性、实时性、功能性五个属性，并在此基础上增加了团队和管理两个属性。

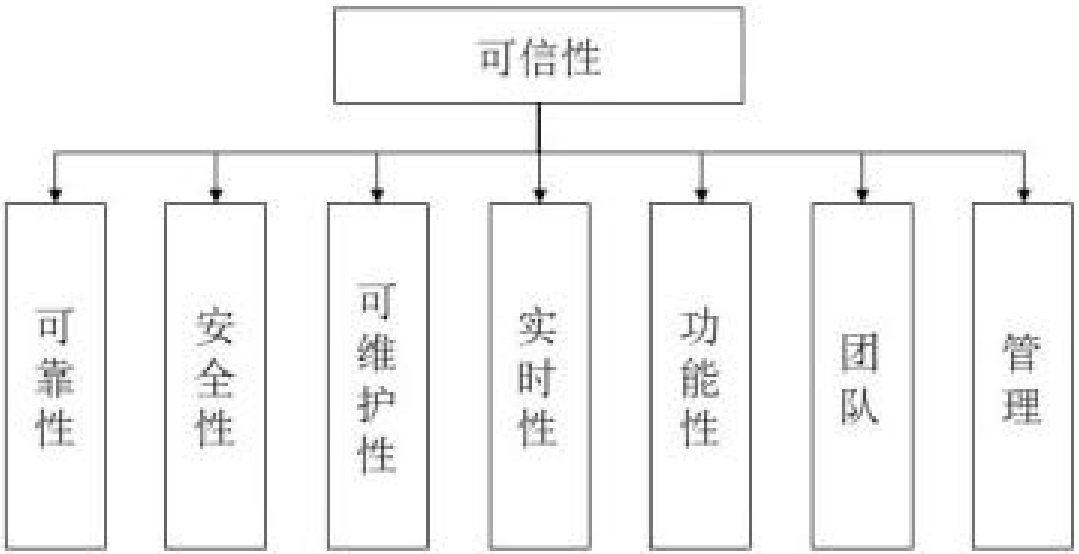
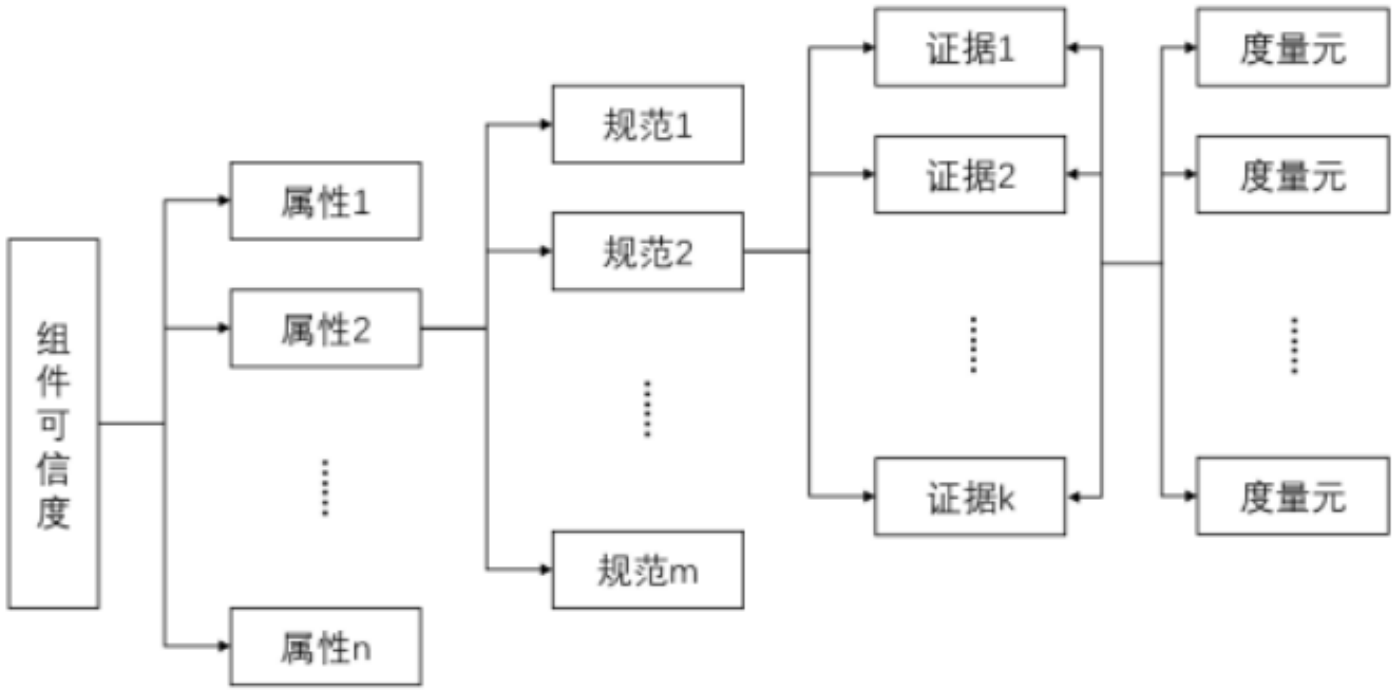


Figure: 组件可信属性

评估体系

将可信性分解为七个可信属性。可信属性又向下进一步分解为可信规范，然后根据组件开发过程数据，将规范分解为可信证据，再由证据对应到可以度量的度量元。为了对组件进行全面的可信性度量，从组件特征和开发过程两个方面建立了涵盖7 个属性，16 个规范，56 个证据和64 个度量元的四层可信评估指标体系。



组件功能性

- 组件是能完成独立功能的程序模块，因此组件可信性需要体现功能性。
- 功能性是指组件提供满足明确和隐含要求功能的能力，
- 功能性向下可分解为文档规范定义功能、建模仿真和评审这三条规范。
- 文档规范定义功能可以从文档的角度反映组件功能的完整程度，若组件需求文档记录的功能不完整准确，可能会对后续开发产生影响。
- 建模的目的是为了准确表达组件的功能需求，自然语言会产生歧义。仿真的目的也是为了说明组件功能需求避免产生歧义。建模仿真能够反映功能需求准确程度。
- 评审是从专家对功能进行评审的角度去反映组件功能的适合准确程度。

组件功能性

- 文档规范定义功能指记录组件功能文档的规范程度，从需求规格说明文档中是否详细记录对组件功能需求，是否详细说明了组件的软件和硬件环境，是否对业务逻辑接口等方面进行详细描述以及文档编写是否符合公司规范这四个方面去分解证据。
- 建模仿真指对组件进行建模并仿真，从是否进行建模，是否进行仿真，比较模型和实际组件实现的功能差别和组件中的概念是否基于物理假设这四个方面去分解证据。
- 评审指专家对组件功能进行评审，可以从是否有专家评审，是否是独立评审，专家是否精通该领域和评审是否是独立评审这四个方面去分解证据。
- 功能性属性分解如表7.1

功能性属性度量元

属性	规范	证据	度量元
功能性	文档规范定义	1、需求规格说明文档清晰、详细地记录了组件的功能需求。 2、需求规格说明文档详细说明了组件的软件环境和硬件环境。 3、需求规格说明文档对组件业务逻辑包括输入、输出、接口以及通信协议进行详细说明。 4、需求规格说明文档遵循公司规定的文档编写标准	A、满足证据1 、2 、3 、4 B、满足证据1 、2 、3 C、满足证据1 、2 D、满足证据1
	建模仿真	1、对需求进行建模并符合建模规则。 2、对功能进行仿真并且能实现其需求。 3、比较测试对象和仿真模型在相同输入下的反应，以检测模型的行为和具体实体之间的差别。 4、证明概念所基于的物理假设在多种真实的运行条件下。	A、满足证据1 、2 、3 、4 B、满足证据1 、2 、3 C、满足证据1 、2 D、满足证据1
	评审	1、精通该领域的专家进行第三方评审。 2、公司内部专家进行独立评审。 3、公司内部专家进行评审。 4、无专家评审。	A、满足证据1 B、满足证据2 C、满足证据3 D、满足证据4

可靠性属性

- 可靠性指组件维持规定功能的能力，组件可靠性可向下分解为故障去除率、容错能力和测试这三条规范。
- 故障去除率是软件开发实践中一个有用的指标，它帮助开发人员评估调试的有效性并估计额外的工作量。
- 容错能力从组件在发生异常情况下依旧能提供正常服务的角度说明组件的可靠性。
- 测试是从组件的测试结果和多种测试方式的角度去反映组件的可靠程度。

可靠性属性

- 故障去除率指组件中解决的故障与总故障的比例，从这个角度分解了一条证据，由于度量元的值要保持在1 到10 之间，度量元的值就是组件中解决的故障与总故障的比例乘以10，如果算出的度量元值小于1，则默认度量元的可信值为1。
- 容错能力指组件发生故障并能保持工作的能力，从组件发生故障到恢复的时间、组件核心模块是否备份和组件模块发生故障时是否有冗余模块进行替代这三个方面去分解证据。
- 测试是指测试组件是否完成规定功能，从组件测试结果是否符合功能和性能需求，是否对组件进行故障注入测试，是否测试组件资源使用情况并进行评估和是否针对组件所运行的环境进行测试，并分析测试环境与目标环境之间的差异这四个方面去分解证据。
- 可靠性属性分解如表7.2所示。

可靠性属性度量元

Table: 可靠性属性度量元7.2

属性	规范	证据	度量元
可靠性	故障去除率	组件中解决的故障与总故障的比例。	$r = \frac{f}{F}$， 其中F 为故障总数量， f 为已解决的故障数量， 如果$r \leq 0.1$，则$R = 1$， 否则$R = r \times 10$
	容错能力	1、记录组件发生故障到恢复的时间。 2、组件核心模块进行备份设计。 3、功能模块发生故障有冗余模块进行替代。	A、满足证据1 、2 、3 B、满足证据1 、2 C、满足证据1 D、无要求
	测试	1、针对组件所运行的环境进行测试， 并分析测试环境和目标环境之间的差异。 2、测试组件资源使用情况并进行评估。 3、对组件进行故障注入测试。 4、组件测试结果符合功能和性能需求。	A、满足证据1 B、满足证据2 C、满足证据3 D、满足证据4

安全属性

- 安全性是指组件出现故障或意外失效后，组件尽量避免灾难性后果的能力，可向下分解为内部安全和外部安全这两条规范。
- 内部安全是从组件内部出现故障采取安全措施的角度去反映组件的安全性。
- 外部安全是从与该组件相关的外部设备发生故障采取安全措施的角度去反映组件的安全性。

安全属性证据

- 组件的内部安全指组件自身的安全保障措施，从是否考虑组件所有的正常工作情况与异常工作情况，是否设置故障容错时间间隔，是否设置故障缓解措施和功能是否通过安全性检测这四个方面去分解证据。
- 组件外部安全指与组件交互的外部设备安全保障措施，从是否随机检测与组件相关的硬件故障措施，是否有发现外部故障后的告警措施和是否有与该组件交互的外部设备的故障控制措施这三个方面去分解证据。
- 安全性属性分解如表7.3所示。

安全属性度量元

Table: 安全属性度量元7.3

属性	规范	证据	度量元
安全性	内部安全	1、考虑组件所有的正常工作情况以及异常工作情况。 2、设置故障容错时间间隔。 3、设置故障缓解措施。 4、功能通过安全性检测	A、满足所有证据 B、满足其中三条证据 C、满足其中两条证据 D、满足其中一条证据
	外部安全	1、有随机检测与组件相关的硬件故障措施。 2、有发现外部故障后的告警措施。 3、有与该组件交互的外部设备的故障控制措施。	A、满足证据1 、2 、3 B、满足证据1 、2 C、满足证据1 D、无要求

可维护属性

- 可维护性是可信属性之一。可维护性指组件可被修改的能力，可向下分解为程序编写规范和可修改性这两条规范。
- 程序编写规范代表了程序的编写风格需要统一，一个组件可能由不止一位开发人员进行开发，统一的编程风格将提高代码的可阅读性，便于后期维护。
- 可修改性从组件可被修改的角度去反映组件的可维护性。

可维护性证据

- 程序编写规范指统一程序编写规范，便于后期维护，从文件的结构、工程和模块名称是否正确，子模块及其输入输出的名称是否正确，代码中每个函数是否都有注释和表明函数功能与代码格式是否完全符合公司编码规范这四个角度去分解证据。
- 可修改性指组件被修改的能力，从是否明确修改前后文档的双向追踪关系，是否明确了组件已实现或已修改部分能够被确认的能力的要求，和是否明确了组件可被诊断自身的缺陷或失效原因或标识其待修改部分的能力的要求，这三个角度去分解证据。
- 可维护性属性分解如表7.4所示。

可维护性度量元

Table: 可维护属性度量元7.4

属性	规范	证据	度量元
可维护性	程序编写规范	1、文件的结构、工程和模块名称是正确的。 2、子模块及其输入输出的名称是正确的。 3、代码中每个函数都有注释，表明函数功能。 4、代码格式完全符合公司编码规范。	A、满足证据1 、2 、3 、4 B、满足证据1 、2 、3 C、满足证据1 、2 D、满足证据1
	可修改性	1、明确修改前后文档的双向追踪关系。 2、明确了组件已实现或已修改部分能够被确认的能力的要求。 3、明确了组件可被诊断自身的缺陷或失效原因或标识其待修改部分的能力的要求。	A、满足所有证据 B、满足其中两条证据 C、满足其中一条证据 D、无要求

实时属性

- 实时性作为可信属性之一。实时性指组件在指定时间内完成操作或提交输出的能力，实时性可向下分解为实时处理和实时稳定性这两个规范。
- 实时处理指组件按照规定时间进行响应的能力，从是否明确各任务响应时间并设定范围值以使响应时间在范围值之内，是否明确数据处理延迟最大时间，若数据处理延迟有相应的解决措施，和是否明确系统出故障时处理时间并设定最大处理时间，这三个方面去分解证据。
- 实时稳定性指组件在实时处理时避免过大抖动的能力，从是否明确了处理时间最大抖动定义，处理时间最大抖动设计是否具体，处理时间最大抖动是否实现了，时间抖动测试是否完整，这四个方面去分解证据。
- 实时性属性度量元如表7.5所示。

实时属性度量元

Table: 安全属性度量元7.3

属性	规范	证据	度量元
实时性	实时处理	1、明确各任务响应时间并设定范围值，使响应时间在范围值之内。 2、明确数据处理延迟最大时间，数据处理延迟有相应的解决措施。 3、明确系统出故障时处理时间并设定最大处理时间	A、满足证据1 、2 、3 B、满足证据1 、2 C、满足证据1 D、无要求
	实时稳定	1、明确了处理时间最大抖动定义， 2、具体设计了处理时间的最大抖动时间， 3、处理时间最大抖动已实现， 4、时间抖动测试完整，	A、全部满足四条证据 B、满足其中三条证据 C、满足其中两条证据 D、满足其中一条证据

团队属性

- 团队是指参与组件开发的团队人员工作能力，可以向下分解为管理人员和研发人员这两条规范。
- 管理人员是从项目中管理者的角度去反映团队的能力，管理人员规范是指项目中管理人员的工作能力。从项目经理与产品经理的工作年限和工作资历这两个方面向下分解了四条证据。
- 研发人员是从研发者的角度去反映团队的能力。研发人员规范是指项目中研发人员的工作能力。从开发人员和测试人员的工作年限和工作资历这两个方面向下分解了四条证据。

团队属性规范

- 管理人员规范是指项目中管理人员的工作能力。从项目经理与产品经理的工作年限和工作资历这两个方面向下分解了四条证据。
- 研发人员规范是指项目中研发人员的工作能力。从开发人员和测试人员的工作年限和工作资历这两个方面向下分解了四条证据。
- 团队属性分解以及度量元见表7-6.

团队属性度量元

Table: 团队属性度量元7.6

属性	规范	证据	度量元
团队	管理人员	1、项目经理：有5 年以上工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务；产品经理：有5 年以上工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务。 2、项目经理：有5 年以上工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务；产品经理：有3～5 年工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务。 3、项目经理：有3～5 年工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务；产品经理：有3～5 年工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务。 4、项目经理：有3～5 年工作经验多次参与研发项目，并且作为骨干或管理人员带领团队完成研发任务；产品经理：有1～3年工作经验，多次参与研发项目，能够独立完成研发任务。	A、满足证据1 B、满足证据2 C、满足证据3 D、满足证据4

团队属性度量元

Table: 团队属性度量元7.6续

属性	规范	证据	度量元
团队	研发人员	1、开发人员：有3 年以上工作经验多次参与研发项目，能够独立完成研发任务； 测试人员：有1～3 年工作经验多次参与研发项目，能够独立完成研发任务。 2、开发人员：有1～3年工作经验多次参与研发项目，能够独立完成研发任务； 测试人员：有1～3 年工作经验，仅作为执行人员参加过项目研发。 3、开发人员：有1～3年工作经验多次参与研发项目，能够独立完成研发任务； 测试人员：有0～1 年工作经验，仅作为执行人员参加过项目研发。 4、开发人员：有0～1年工作经验，仅作为执行人员参加过项目研发； 测试人员：有0～1年工作经验，仅作为执行人员参加过项目研发。	A、满足证据1 B、满足证据2 C、满足证据3 D、满足证据4

管理属性

- 项目管理是在一定的约束条件下，以高效率地完成项目为目标，按照项目内在的逻辑规律进行有效的计划、组织、协调、控制的系统管理活动。
- 因此将管理作为可信属性之一。管理是指组件开发过程中，对组件设计工作进行全面管理的能力，可以向下分解为文档管理、资源管理和风险管理这三条规范。
- 文档管理就是对开发过程中的所有文档进行管理。
- 资源管理就是对开发过程中的人力资源、时间资源、计算机资源进行管理。
- 风险管理就是在风险来临时，能够应对风险的能力。

管理属性证据

- 文档管理指组件开发过程中文档管理能力，可以从是否有详细的需求说明文档且文档中需求被完整记录并易于理解，是否有详细的设计文档，文档中详细记录数据、接口等设计细节，是否有详细的源代码文档，并且有适当注释，使源码易于理解，以及是否有详细的测试文档，记录测试过程中的所有数据。
- 资源管理指组件开发过程中对时间、人力、资源、费用的管理能力，可以从是否有针对项目的开发计划，规定每个阶段的具体开始时间和结束时间，是否保证在正确时间有正确的资源，为每个人员分配任务，是否为每个工作人员合理安排工作量及计算机资源和是否对项目进行跟踪与监控，实时反映项目的进度、费用这四个方面去分解证据。
- 风险管理指组件开发过程中对风险的管理能力，可以从是否记录所有潜在风险及每个风险所具有的特点，是否对风险进行评估，并量化风险可能产生的影响和是否制定风险应对措施这三个方面分解证据。
- 管理属性分解如表7.7所示。

管理属性分解表与度量元

属性	规范	证据	度量元
管理	文档管理	1、有详细的需求说明文档，文档中需求被完整记录并易于理解。 2、有详细的设计文档，文档中详细记录数据、接口等设计细节。 3、有详细的源代码文档，并且有适当注释，使源码易于理解。 4、有详细的测试文档，记录测试过程中的所有数据	A、满足所有证据 B、满足其中三条证据 C、满足其中两条证据 D、满足其中一条证据
	资源管理	1、有针对项目的开发计划，规定每个阶段的具体开始时间和结束时间。 2、保证在正确时间有正确的资源，为每个人员分配任务。 3、为每个工作人员合理安排工作量及计算机资源。 4、对项目进行跟踪与监控，实时反映项目的进度费用。	A、满足所有证据 B、满足其中三条证据 C、满足其中两条证据 D、满足其中一条证据
	风险管理	1、记录所有潜在风险及每个风险所具有的特点。 2、对风险进行评估，并量化风险可能产生的影响。 3、制定风险应对措施。	A、满足证据1 、2 、3 B、满足证据1 、2 C、满足证据1 D、无要求

度量元种类

度量元设定原则一共有4 种。

- 第一种：证据间是递进关系。例如，A 满足证据1、2、3、4，B 满足证据1、2 、3，评分高的度量元包含评分低的度量元的证据。
- 第二种：证据间没有递进关系，只看满足证据的个数。例如，A 满足所有证据，B 满足其中3 条证据，度量元等级越高，包含证据越多。
- 第三种：证据间也是递进关系，不过证据中包含的信息很完整。例如，A 满足证据1 ，B 满足证据2，在定义证据时，证据1 就已经涵盖了证据2 的情况。
- 第四种：度量元可以直接通过公式计算出来。例如故障去除率的度量元。$r = \frac{f}{F}$,其中F 为故障总数量,f 为已解决的故障数量, 如果$r \leq 0.1$，则$R = 1$，否则$R = r \times 10$。

度量元取值

Claim

- *A* 等级最高，对应到相应的数值是*10*；其次是*B* 等级，对应到相应的数值是*7*；再然后是*C*，对应到相应的数值是*4*；最后是*D* 等级，对应到相应的数值是*1*。
- $r = \frac{f}{F}$, 其中F 为故障总数量, f 为已解决的故障数量, 如果$r \leq 0.1$，则$index = 1$，否则$index = r \times 10$

例子：电池充放电速度计算组件–BCD_SCC

- 选出该组件的五个版本, 分别为V1 、V2 、V3 、V4 、V5。
- 组件结合本文提出的可信指标评估体系，请相关专家根据可信证据在实际组件开发过程数据的映射进行打分，得出的分值即规范的可信值。
- BCD_SCC 五个版本中各可信属性对应的规范可信值如表7.9所示。

Table: 功能性的子属性准确性度量元

属性	规范	V1	V2	V3	V4	V5
功能性	文档规范定义功能	10	10	10	10	10
	建模仿真	10	10	10	10	10
	评审	7	7	7	7	10
可靠性	故障去除率	8	7	9	7	8
	容错能力	7	10	10	10	10
	测试	10	7	7	10	10
安全性	内部安全	10	10	7	10	10
	外部安全	7	10	10	10	10
	程序编写规范	10	10	7	7	7
	可修改性	7	7	10	10	10
实时性	实时处理	7	7	10	10	10
	实时稳定	7	10	9	8	10
团队	管理人员	10	10	10	10	10
	研发人员	7	7	7	7	7
管理	文档管理	10	10	10	10	10
	资源管理	7	10	7	7	7
	风险管理	7	7	10	10	10

属性量化

Claim

- 从属性可信分解表可以看到，度量元实际上是量化了可信属性的规范。
- 从度量元的值就是对应的规范量化值。
- 一个属性一般分解成*2-4*个规范，因而从规范量化值可以得到属性的量化值。
- 如何从规范量化值计算出属性的量化值？

属性量化

设一个属性*attr*有m个规范$normal_1, \cdots, normal_m$，则属性*attr*的可信值通过下面计算公式求得：

Claim

$$T_{attr} = norm_1^{\beta_1} \times norm_2^{\beta_2} \times \cdots \times norm_m^{\beta_m}$$

其中$norm_i$是第i个规范，β_i是第i个规范的权重，$\sum_{i=1}^{m} \beta_i = 1$。

组件的可信性度量

设一个组件A有有n个属性$attr_1, \cdots, attr_n$，则组件A的的可信值通过下面计算公式求得：

Claim

$$T_A = attr_1^{\alpha_1} \times attr_2^{\alpha_2} \times \cdots \times attr_n^{\beta_n}$$

其中$attr_i$是第i个组件属性，α_i是第i个属性的权重，$\sum_{i=1}^{m} \alpha_i = 1$。

权重计算

使用层次分析法原理计算出属性权重和规范权重作为组件度量的权重。请专家按照1 到9 比例标度对属性与属性之间或规范与规范的重要程度进行判断，构建正互反判断矩阵，利用层次分析法计算出各属性或规范的权重值。表7.8为正互反判断矩阵的标度含义。**9-point Saaty's scale:**

Scale	Meaning
1	y_i and y_j are equally(同等) important
3	y_i is weakly more（稍微）important than y_j
5	y_i is strongly more （明显）important than y_j
7	y_i is demonstrably more （强烈）important than y_j
9	y_i is absolutely more （绝对）important than y_j
2,4,6,8	compromising between slightly differing judgement.（相邻判断的中间值）

若$a_{ij} = 3$则$a_{ji} = \frac{1}{3}$。

属性正互反判断矩阵及权重

Table: 属性正互反判断矩阵及权重

属性	功能性	可靠性	安全性	可维护性	实时性	团队	管理	属性权重
功能性	1	1/3	1/2	2	3	2	2	0.139
可靠性	3	1	1/2	3	3	3	2	0.237
安全性	2	2	1	2	2	3	3	0.257
可维护性	1/2	1/3	1/2	1	2	2	1	0.108
实时性	1/3	1/3	1/2	1/2	1	2	1	0.083
团队	1/2	1/3	1/3	1/2	1/2	1	1/2	0.063
管理	1/2	1/2	1/2	1	1	2	1	0.115

功能性规范正互反判断矩阵及权重

Table: 功能性规范正互反判断矩阵及权重

规范	文档规范定义功能	建模仿真	评审	规范权重
文档规范定义功能	1	1/2	1/2	0.198
建模仿真	2	1	1/2	0.312
评审	2	2	1	0.490

可靠性规范正互反判断矩阵及权重

Table: 可靠性正互反判断矩阵及权重

规范	故障去除率	容错能力	测试	规范权重
故障去除率	1	2	1	0.400
容错能力	1/2	1	1/2	0.200
测试	1	2	1	0.490

安全性规范正互反判断矩阵及权重

Table: 安全规范正互反判断矩阵及权重

规范	内部安全	外部安全	规范权重
内部安全	1	1	0.500
外部安全	1	1	0.500

可维护性规范正互反判断矩阵及权重

Table: 可维护规范正互反判断矩阵及权重

规范	程序编写规范	可修改性	规范权重
程序编写规范	1	3	0.750
可修改性	1/2	1	0.250

实时性规范正互反判断矩阵及权重

Table: 实时性规范正互反判断矩阵及权重

规范	实时处理	时间稳定	规范权重
实时处理	1	2	0.198
时间稳定	1/2	1	0.312

管理规范正互反判断矩阵及权重

Table: 管理规范正互反判断矩阵及权重

规范	文档管理	资源管理	风险管理	规范权重
文档管理	1	2	1/2	0.298
资源管理	1/2	1	1/3	0.164
风险管理	2	3	1	0.538

团队规范正互反判断矩阵及权重

Table: 团队规范正互反判断矩阵及权重

规范	管理人员	研发人员	规范权重
管理人员	1	1/2	0.333
研发人员	2	1	0.667

功能性规范正互反判断矩阵及权重

- 根据规范的可信值，结合可信属性计算模型可以计算出每个属性的可信值；
- 再根据属性的可信值，结合可信性计算模型最终计算出组件每个版本的可信值。
- 五个版本的组件可信值、属性可信值及可信分级如表7.18所示。

五个版本的组件可信性度量值

Table: 五个版本的组件可信性度量值

版本/属性	功能性	可靠性	安全性	可维护性	实时性	团队	管理	组件值
V1	8.39	8.52	8.37	9.15	7	7.82	7.78	8.26
V2	8.39	7.52	10	9.15	7	7.82	8.25	8.45
V3	8.39	8.67	8.37	7.65	7	7.82	9.43	8.22
V4	8.39	8.67	10	9.15	10	8.88	9.43	9.21
V5	10	9.15	10	9.15	10	8.88	9.43	9.43

基于组件的软件可信性度量模型

- 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。
- 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。
- 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。
- 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。

基于组件的软件可信性度量模型

- 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$\beta_1, \cdots, \beta_m$。
- 设关键组件和非关键组件的权重分别为α和β，要求$\alpha > 1/2 > \beta > 0$。
- 组件间没有替换性。

Claim (基于组件的软件可信性度量模型)

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

基于组件的软件可信性分级模型

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

例子：游戏服务器

- 选取GitHub 上的一个游戏服务器结构为例，其包含9 个组件，分别是网关组件、游戏大厅组件、配置组件、核心服务器组件、注册组件、后台管理组件、工具组件、消息组件、游戏AI 组件，
- 根据组件交互及互相调用关系并结合拥有三年以上开发经验的编程人员意见选出4个关键组件：网关组件、游戏大厅组件、配置组件和核心服务器组件，5个非关键组件：注册组件、后台管理组件、工具组件、消息组件、游戏AI 组件。
- 本节示例由拥有五年以上开发经验的相关领域专家判断软件关键部分和软件非关键部分、关键组件之间、非关键组件之间的重要程度，构建正互反判断矩阵，并利用层次分析法计算出所需权重。
- 表7.21为软件关键部分和软件非关键部分正互反判断矩阵，表7.22为关键组件正互反判断矩阵，表7.23为非关键组件正互反判断矩阵。

例子：游戏服务器–各类权重

Table: 关键组件和非关键组件的正互反判断矩阵及权重

	关键组件	非关键组件	权重
关键组件	1	3	0.750
非关键组件	1/3	1	0.250

例子：游戏服务器–各类权重

Table: 关键组件正互反判断矩阵及权重

组件名	网关组件	游戏大厅组件	配置组件	核心服务器组件	权重
网关组件	1	2	1/2	1/3	0.167
游戏大厅组件	1/2	1	1/2	1/3	0.121
配置组件	2	2	1	1/2	0.262
核心服务器组件	3	3	2	1	0.450

Table: 关键组件正互反判断矩阵及权重

	网关组件	游戏大厅组件	配置组件	核心服务器组件	TD
LLSM	0.167	0.118	0.262	0.453	3.21
CSM	0.167	0.121	0.262	0.450	3.11
Matlab	0.169	0.119	0.261	0.452	3.17

Claim

选取*CSM*计算的权重。

例子：游戏服务器–各类权重

Table: 非关键组件的正互反判断矩阵及权重

组件名	注册组件	后台管理组件	工具组件	消息组件	游戏AI组件	权重
注册组件	1	1/2	1	1/3	1/2	0.113
后台管理组件	2	1	2	1/2	1	0.213
工具组件	1	1/2	1	1/2	1	0.140
消息组件	3	2	2	1	2	0.349
游戏AI组件	2	1	1	1/2	2	0.185

Table: 非关键组件权重计算

组件名	注册组件	后台管理组件	工具组件	消息组件	游戏AI组件	TD
LLSM	0.113	0.213	0.140	0.349	0.185	3.836
CSM	0.112	0.214	0.140	0.348	0.186	3.8409
Matlab	0.112	0.213	0.142	0.348	0.185	6.243

Claim

选取*LLSM*计算的权重。

例子：游戏服务器–各组件可信值

Table: 组件可信值

组件名称	网关	游戏大厅	配置	核心服务器	注册	后台管理	工具	消息	游戏AI
可惜值	9.536	7.531	9.167	8.423	7.556	7.858	8.979	7.708	9.265

例子：游戏服务器可信度量与可信级别

Claim (游戏服务器可信度量)

$$
\begin{aligned}
T &= 0.75 \times (9.536^{0.167} \times 7.531^{0.121} \times 9.167^{0.252} \times 8.423^{0.450}) \\
&+ 0.25 \times (7.556^{0.113} \times 7.858^{0.213} \times 8.979^{0.140} \times 7.708^{0.349} \\
&\quad \times 9.265^{0.185}) \\
&\approx 0.75 \times 8.484 + 0.25 \times 8.162 \\
&\approx 6.363 + 2.041 \\
&= 8.404
\end{aligned}
$$

Claim (游戏服务器可信等级)

*III*级（可信值在$[7.0, 8.5)$区间，并且低于7.0的关键组件少于等于3个）

基于组件的软件可信性度量模型

Claim

把这款游戏服务器软件提升到*IV*甚至*V*级，提升哪些组件的可信性？

作业

Claim

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下面三页。
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。

作业

Table: 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	

作业

Table: 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	

作业

Table: 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	