华东师范大学软件学院课程作业

课程名称：软件质量分析	年级：2023 级本科	姓名：张梓卫
作业主题：韧性与优雅降级	学号：10235101526	作业日期：2024/12/10
指导老师：陈仪香	组号：	

目录

一 第一题：软件韧性

1 软件韧性的定义

根据第八章课件中的第一页内容 8.1 所示：
软件韧性是指软件在遭受攻击或出现故障时，在不中断服务的情况下尽快恢复到正常工作状态的能力。

2 软件韧性的特点

其包含了三个可信属性和十个子属性，分别为：

- 可生存性
  - 可用性
  - 安全性
  - 容错性
- 可恢复性
  - 恢复性能
  - 恢复成本
  - 恢复时间
- 适应性
  - 可拓展性
  - 可重配置
  - 可学习性
  - 自治性

其中，可信属性的定义如下：
而三个可信属性” 旗下” 的十个子属性的定义表格如下：

二 第二题：性能的分级度量

1 对性能进行分级度量

实际工作时的性能，以初始性能为基准，不同系统或组件有不同的指标量，例如 CPU，内存等的性能体现为工作频率等，屏幕的性能体现为分辨率、触摸反馈准确率等。

属性	定义
可生存性	系统在受到攻击、出现故障或事故的情况下，及时完成其任务的能力。
可恢复性	系统及时恢复其服务的能力。
适应性	系统调整自身或其资源以面对改变的形势或环境来维持正常工作的能力。

表 1: 韧性属性定义表

属性	子属性	定义
可生存性	可用性	在规定的时间段内系统能够按照要求运行的能力。
可生存性	安全性	系统保护信息和数据的能力。
可生存性	容错性	在存在包括故障、错误或攻击等威胁的情况下，系统的功能状态能够提供适当服务的程度。
可恢复性	恢复性能	系统恢复其服务的能力。
可恢复性	恢复成本	系统在恢复过程中耗费的非时间成本。
可恢复性	恢复时间	系统在恢复过程中涉及到的一系列时间、延时等。
适应性	可拓展性	系统可以添加新的功能。
适应性	可重配置	系统重新调整资源与进程关系的能力。
适应性	可学习性	系统从动态环境中学习、做出适应性决策来提高性能的能力。
适应性	自治性	自主控制系统在不确定环境下长期运行良好，并在没有外部干预的情况下自动恢复复杂故障的能力。

表 2: 软件韧性属性表

2 以可生存性的定义加以说明

回到上一张 PPT，可生存性的容错性度量元定义为：在存在包括故障、错误或攻击等威胁的情况下，系统的功能状态能够提供适当服务的程度。

根据该定义，可通过以下方法进行性能分级度量：

- 初始性能基准：假设系统在无威胁条件下，能够正常提供完整服务功能，作为初始性能基准。
- 威胁条件下的性能分级：
  1. **完全容错级别**：系统在威胁环境下仍能保持初始性能，提供所有服务功能。
  2. **部分容错级别**：系统受到部分威胁后，性能有所下降，但仍能提供主要服务功能。
  3. **基本容错级别**：系统受到严重威胁，仅能提供关键服务或最低限度的功能。
  4. **无容错级别**：系统完全无法抵抗威胁，服务功能中断。

通过对容错性能的分级度量，可以清晰地评估系统在各种威胁条件下的韧性表现，并为优化和改进提供依据。

在示例中，其韧性证据为组件损失表现：组件受到攻击前的表现 B_{origin} 与受到攻击后降低到的最低表现 $B_{disrupt}$ 之差

三 第三题：网关组件的变化及维护

1 更新后的性能变化

在客户大量增加后，性能发生的变化有很多：

- 可用性（1）：版本更新前为 A，现为 C，此时网关组件无法无法向部分用户提供与客户端进行网络互联功能，无法帮部分用户选择网络质量最佳的服务器
- 可用性（2）：版本更新前为 A，现为 B，此时网关组件在中国以外地区仍能提供达到质量要求的服务。
- 容错性：版本更新前为 A，现为 B，此时受到影响的用户占总用户的 30 %，损失表现约为 25 %.
- 恢复性能：版本更新前为 A，现为 D，组件在规定时间内恢复了损失表现的 30 % 以下
- SURV（可生存性）权重值：更新前为 9.660，版本更新后为 7.996
- REC（可恢复性）权重值：更新前为 10，版本更新后为 6.006
- ADAPT（可适应性）权重值：更新前为 5.250，版本更新后不变，仍为 5.250
- CP 可信度量值：更新前为 8.321，版本更新后为 6.353

2 运行商维护解决

游戏开发人员及时增加了服务器以解决了该问题。同时调整了更新策略：让一部分正在游玩的玩家继续游玩，未游玩的玩家才需进行更新，以免导致服务器突发压力过大。

3 解决效果

- 可用性（1）：版本更新后为 C，修复后为 A，此时组件当前关键功能和非关键功能全部可用
- 可用性（2）：版本更新后为 B，修复后为 A，此时网关组件在各地区均能提供达到质量要求的服务
- 容错性：版本更新后为 B，修复后为 A，此时组件损失表现不超过受到攻击前表现的 40 %
- 恢复性能：版本更新前为 D，修复后为 A，组件在规定时间内恢复了损失表现的 90 % 以上
- 可拓展性：修复前为 D，修复后为 B，组件添加功能时不影响正在运行的功能，但用户下次使用时需更新
- SURV（可生存性）权重值：修复前为 7.996，修复后为 9.660
- REC（可恢复性）权重值：修复前为 6.006，修复后为 10
- ADAPT（可适应性）权重值：修复前为 5.250，修复后为 6.383
- CP 可信度量值：修复前为 6.353，修复后为 8.768

通过这些变化可以看出：

- 修复后，组件的关键性能指标得到了显著改善，尤其是可用性、容错性和恢复性能达到了理想状态，充分保障了系统的正常运行和服务质量。
- 可生存性（SURV）、可恢复性（REC）和可适应性（ADAPT）权重值显著提高，表明系统在面对威胁和故障时具备更强的韧性，能够快速响应和恢复。
- 组件在恢复后的扩展性有所提升（从 D 提升到 B），尽管仍需优化，但已经能够在一定程度上支持功能拓展需求。
- CP 可信度量值从修复前的较低水平（6.353）提升到修复后的高水平（8.768），表明系统整体的可靠性和稳定性显著增强。
- 总体而言，修复工作不仅解决了当前问题，还提升了系统在未来运行中的综合表现能力，为用户提供了更加高效、稳定的服务。

四 第四题：软件优雅降级的目的

软件优雅降级的目的在于确保系统在遭受攻击或发生内部故障后，能够通过牺牲部分性能或功能来保证软件系统的持续运行，而非直接停止或崩溃，从而提升系统的容错能力和可靠性。这种机制可以实现以下目标：

优雅降级模型的系统具备的两个能力：故障定位能力、故障隔离能力，能够使得系统在发生故障后，仍能够正常提供服务，从而保证系统的可信性。

- 保障系统的服务能力：即使部分组件失效，系统仍能提供核心功能，确保用户体验的基本需求。
- 降低维护成本：避免因全面系统故障导致的高昂修复成本，通过有限的降级措施及时隔离故障。
- 提高系统的稳定性：优雅降级通过故障隔离机制，将故障影响局限于局部区域，确保系统整体的运行可靠性。
- 增强系统韧性：通过快速定位和隔离故障，优雅降级为系统提供了一种灵活应对复杂问题的手段。

五 第五题：优雅降级模型架构

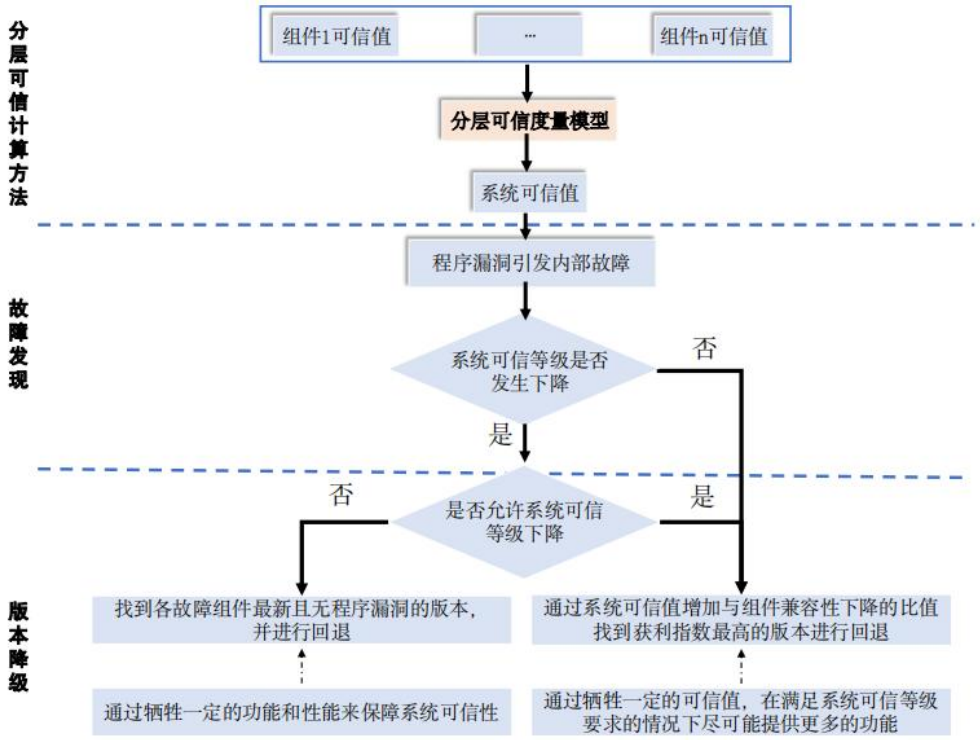

图 1: 优雅降级模型架构

优雅降级模型的架构包含以下几个关键部分：

分层结构：系统按照功能模块和组件进行分层，如权限管理、数据更新、数据修改、注册登录、购票下单等功能模块，每层包含若干组件。每个组件的可信度与其在系统中的调用频率、重要性权重和依赖关系相关。

故障发现机制：通过实时监控系统和组件的可信度，识别系统可信值的下降情况，定位发生故障的组件。

故障隔离机制：通过版本回退或功能降级，将故障组件隔离，以确保其他组件和系统整体的正常运行。使用算法（如二分搜索算法）快速找到故障组件的最优回退版本，同时尽量降低版本回退对系统兼容性的影响。

版本选择和回退成本控制：使用分支定界法或类似的优化算法，根据组件的重要性权重和回退成本，选择最优的组件版本回退方案，以实现系统可信等级的恢复或保持。

六 第六题：如何选择组件版本

可通过基于分层可信度量模型所开发的软件系统分层可信度量工具对各组件可信度的实时监控，可根据组件可信度的下降情况及下降原因完成故障组件的定位。

为了保证能在较短时间内找到故障组件的最优回退版本，可以采用二分搜索算法的思想，通过不断缩小搜索区间直到找寻到各故障组件的最优回退版本。

但如果可信等级已经发生了下降，此时需通过基于分支定界法的故障组件选取算法来搜寻最优的故障组件选择方案。

核心的公式如下：

(1) 自重要性 $S(cp)$

定义：衡量单个组件的自身重要性，基于其在系统中被调用的频率。公式如下:

$$S(cp) = \frac{\text{frq}_i}{\sum_{j=1}^{n} \text{frq}_j} \times 10$$

其中:

- frq_i: 组件节点 cp_i 的调用频率。
- n: 系统中的组件总数。

(2) 出度重要性 IMout(cp)

定义：根据组件的依赖关系，计算其对其他组件的影响。公式如下:

$$\text{IMout(cp)} = \max\{w_{i,j}\} \times \text{outdegree}$$

其中:

- $w_{i,j}$: 从 cp_i 到 cp_j 的依赖关系的权重。
- outdegree: 组件的出度（依赖关系数量）。

(3) 综合重要性 $IM(cp)$

定义：组件综合重要性指标，结合出度和自重要性。公式如下:

$$IM(cp) = p \times \text{IMout(cp)} + (1-p) \times S(cp)$$

其中:

- p: 权重系数，衡量出度和自重要性的重要性比例。

(4) 回退成本 $Cost(i)$

公式定义：综合考虑组件版本回退的影响，计算其成本：

$$\text{Cost}(i) = \alpha_i \times \sum_{r_{i,j} \in R_i} \left(\alpha_j \times \frac{\Delta X_i}{X_i}(10 - |X_i - X_j|) + \alpha_j \times \frac{\Delta Y_i}{Y_i}(10 - \beta|Y_i - Y_j|) \right)$$

其中：

- α_i: 组件 cp_i 的权重。
- R_i: 组件 cp_i 的依赖集合。
- $\Delta X_i = |X_i^* - X_i|$: 主版本号的差值。

- $\Delta Y_i = |Y_i^* - Y_i|$: 次版本号的差值。
- X_i^*, Y_i^*: 当前版本的主版本号和次版本号。
- β: 次版本号权重参数，用于调整次版本号差异的影响。