软件体系结构设计调研及SAD（二）

> 姓名： 张烜
>
> 班级： 2班
>
> 学号： 201705130112

实验目的:

1. 尝试不同的体系结构设计
2. 继续补充和修改自己项目的SAD。
3. 记录项目及小组的工作进度。

实验内容:

0. 学习、检索课本5.17及以下推荐的参考书或网上检索新的有关软件体系结构的资料。

我选择的书籍是《聊架构：洞见架构之道》 王概凯，大概是因为觉得需要对于架构有一定的了解，在写代码到一定程度之后，就需要对架构或者说整体有一定的了解了，那这时候一本讲架构的书就显得尤为重要。

但是打开书籍发现这本书并没有作者独特的见解，而更像是漫谈，整本书像是一本散文集，每个篇章都是一个故事，整体缺少一个主线。说白了，看完觉得空空。也许是我自己才学疏浅，没明白作者想要传达给读者怎样一种思想，或者要解答怎样的困惑。

粗读本书，有几个感悟记录一下。

1.架构师必然是有权利推动架构演变的人。架构最终都要落地，没有组织权力，就谈不上架构了。

2.软件开发要拆分业务功能的生命周期。

3.软件开发者必须熟知业务。软件是业务功能在计算机上的体现，必要情况下需要软件开发者充当业务的角色

4.单元测试要写，但是不可有模拟的情况，这样的单元测试是没有意义的，debug就可以完成了。单元测试要脱离环境部署。简单来说main函数就可以跑起来。

5.设计模式是成熟的写代码简洁的方式，任何成熟的框架中都有设计模式的体现。

1. KWIC： 管道和过滤器

1.1 介绍

在这种情况下，有四个过滤器:输入、移位、按字母顺序排列和输出。每个过滤器处理数据并将其发送给下一个过滤器。控件是分布式的:每个过滤器可以在它拥有要计算的数据时运行。过滤器之间的数据共享被严格限制在通过管道传输的范围内。。

这个解决方案有几个很好的特性。首先，它保持了处理的直观性。其次，它支持重用，因为每个过滤器可以独立工作(提供上游过滤器以其期望的形式生成数据)。通过在处理序列的适当位置插入过滤器，可以很容易地将新函数添加到系统中。第三，它支持简化修改，因为过滤器在逻辑上独立于其他过滤器。

过滤器分为四个部分：

- 输入
- 循环移位
- 字母顺序器（这个必须等待所有数据才能产生输出）
- 输出

1.2 实现

1.2.1 *input*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

__author__ = 'mejty'


class Input:
    def __init__(self, filename):
        self.filename = filename

    def lines(self):
        with open(self.filename) as file:
            return [line.strip() for line in file]
```

1.2.2 *alphabetizer*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class Alphabetizer:
    def sorted_lines(self, shifted_lines):
        yield from sorted(shifted_lines)
```

1.2.3 *circular_shift*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from collections import deque


class CircularShift:
    def shifted_lines(self, lines):
        for line in lines:
            words = line.split(" ")
            dec = deque(words)
            for i in range(0, len(dec)):
                dec.rotate()
                yield " ".join(dec)
```

1.2.4 *output*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


class Output:
    def __init__(self, filename):
        self.filename = filename

    def save_to_file(self, sorted_lines):
        with open(self.filename, "w") as file:
            for line in sorted_lines:
                file.write("{line}\n".format(line=line))
            file.close()
```

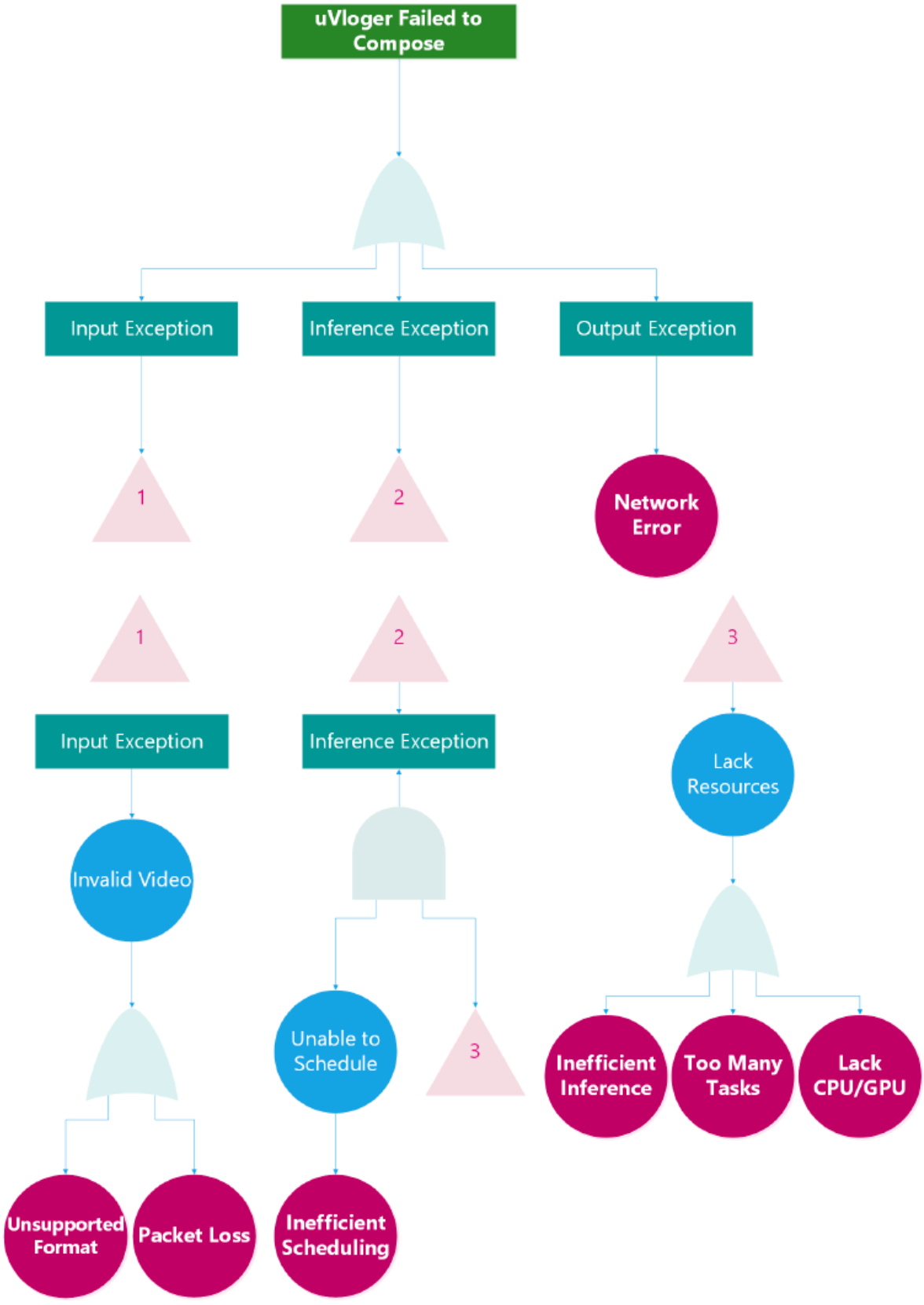
1.2.5 *kwic*

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from input import Input
from circular_shift import CircularShift
from alphabetizer import Alphabetizer
from output import Output

input = Input("../input.txt")
circular_shift = CircularShift()
alphabetizer = Alphabetizer()
output = Output("../output.txt")

output.save_to_file(alphabetizer.sorted_lines(circular_shift.shifted_lines(input
.lines())))
```

2. 第五章课后习题14，故障树转割集树练习。


uVloger Failed to Compose
Input Exception
Inference Exception
Output Exception
1
2
Network Error
1
2
3
Input Exception
Inference Exception
Lack Resources
Invalid Video
Unable to Schedule
3
Inefficient Inference
Too Many Tasks
Lack CPU/GPU
Unsupported Format
Packet Loss
Inefficient Scheduling

Unsupported Format	x_1
Packet Loss	x_2
Inefficient Scheduling	x_3
Inefficient Reference	x_4
Too Many Tasks	x_5
Lack GPU/CPU	x_6
Network Error	x_7
Input Exception	a_1
Inference Exception	a_2
Output Exception	a_3

根据表运算可以得到割集划分

$$
T = a_1 + a_2 + a_3
\\= (x_1+x_2) + (x_3(x_4+x_5+x_6)) + x_7
\\= x_1 + x_2 + x_3x_4+x_3x_5 + x_3x_6 + x_7
$$

$$
\begin{aligned}
T &= a_1 + a_2 + a_3 \\
&= (x_1 + x_2) + (x_3(x_4 + x_5 + x_6)) + x_7 \\
&= x_1 + x_2 + x_3x_4 + x_3x_5 + x_3x_6 + x_7
\end{aligned}
$$

3. 补充和完善体系结构设计文档SAD

新的SAD已经更新在小组文档