**SA思考题10**

**1、请描述Conway's law(康威定律)**

**答：**

**康威定律：**设计系统的组织其产生的设计等价于组织间的沟通结构。

- Melvin Conway(1967)

**反向理解：**如果系统架构不支持，你无法建立一个高效的组织。

**通俗解释：**产品必然是其（人员）组织沟通结构的缩影。

Mike Amundsen 归纳了如下**四个核心观点:**

**1）第一定律：组织沟通方式会通过系统设计表达出来。**

对于复杂的，需要协作完成的系统开发，沟通是必须要持续提升的问题。

每个团队由5-10人组成（沟通成本 = n(n-1)/2 - 《人月神话》），在团队内部进行频繁的、细粒度的沟通。对于团队外部，定义好接口，契约，只进行粗粒度的沟通。这样可以降低沟通成本，同时也符合高内聚，低耦合原则（代码和人员管理有些时候确实是相通的）。

**2）第二定律：时间再多一件事情也不可能做的完美，但总有时间做完一件事情。**

这就是我们在用kanban管理迭代时几乎都有一列是BAU（Business As Usual ），其中会包括一些日常修复的Bug Story。敏捷开发中将迭代引入，做到持续交付，快速验证，迅速反馈，持续改进。

**3）第三定律：线型系统和线型组织架构间有潜在的异质同态特性。**

通俗的理解是，你想要架构成为什么样，就将团队分成怎样的结构。比如前后端分离的团队，架构就是基于前后端分离。在基于微服务设计的团队里，一个很好的理念是自管理。团队内部对于自己所负责的模块高度负责，进行端对端的开发以及运维。

**4）第四定律：大的系统组织总是比小系统更倾向于分解。**

合久必分，分久必合，团队以及架构都是在不断优化的。一个团队随着人员的增加，沟通以及管理成本一定会增加。

**2、什么是透明装饰模式，什么是半透明装饰模式？请举例说明。**

**答：**

**1）透明装饰模式**

**在透明装饰模式中，要求客户端完全针对抽象编程**，装饰模式的透明性要求客户端程序不应该将对象声明为具体构件类型或具体装饰类型，而应该全部声明为抽象构件类型。对于客户端而言，具体构件对象和具体装饰对象没有任何区别。也就是应该使用如下代码：

|  |
| --- |
| **Component c, c1; //使用抽象构件类型定义对象**  c = new ConcreteComponent()；  c1 = new ConcreteDecorator (c)； |

而不应该使用如下代码：

|  |
| --- |
| **ConcreteComponent c; //使用具体构件类型定义对象**  c = new ConcreteComponent()； |

或

|  |
| --- |
| **ConcreteDecorator c1; //使用具体装饰类型定义对象**  c1 = new ConcreteDecorator(c)； |

**例：**在一个图形界面构件库的设计方案中使用透明装饰模式，在客户端中存在如下代码片段：

|  |
| --- |
| **……**  **Component component,componentSB,componentBB; //全部使用抽象构件定义**  **component = new Window();**  **componentSB = new ScrollBarDecorator(component);**  **componentBB = new BlackBorderDecorator(componentSB);**  **componentBB.display();**  **……** |

使用抽象构件类型Component定义全部具体构件对象和具体装饰对象，客户端可以一致地使用这些对象，因此符合透明装饰模式的要求。

透明装饰模式可以让客户端透明地使用装饰之前的对象和装饰之后的对象，无须关心它们的区别，此外，还可以对一个已装饰过的对象进行多次装饰，得到更为复杂、功能更为强大的对象。在实现透明装饰模式时，要求**具体装饰类的operation()方法覆盖抽象装饰类的operation()方法**，除了**调用原有对象的operation()外还需要调用新增的addedBehavior()方法来增加新行为**。

**2）半透明装饰模式**

透明装饰模式的设计难度较大，而且有时我们需要单独调用新增的业务方法。为了能够调用到新增方法，我们不得不**用具体装饰类型来定义装饰之后的对象**，而具体构件类型还是可以使用抽象构件类型来定义，这种装饰模式即为半透明装饰模式，也就是说，对于客户端而言，具体构件类型无须关心，是透明的；但是具体装饰类型必须指定，这是不透明的。

**例：**在一个文件对象功能增加实例中，为了能够调用到在Approver中新增方法approve()，客户端代码片段如下所示：

|  |
| --- |
| **……**  **Document doc; //使用抽象构件类型定义**  doc = new PurchaseRequest();  **Approver newDoc; //使用具体装饰类型定义**  **newDoc = new Approver(doc);**  **……** |

半透明装饰模式可以给系统带来更多的灵活性，设计相对简单，使用起来也非常方便；但是其最大的缺点在于**不能实现对同一个对象的多次装饰**，而且客户端需要有区别地对待装饰之前的对象和装饰之后的对象。在实现半透明的装饰模式时，我们只需在具体装饰类中增加一个独立的addedBehavior()方法来封装相应的业务处理，由于客户端使用具体装饰类型来定义装饰后的对象，因此可以单独调用addedBehavior()方法来扩展系统功能。