分布式温控系统动态结构设计

姓名：吴铭钞 奚工理 许磊 慕雨诚 戴亚敏

班级：2017211318

组号：F组

2020年5月20日

0. 文档说明

动态结构针对用例中的每个用例进行设计，描述了分层结构中的各个软件对象如何实现用例场景。本文档以分布式温控系统的领域模型、操作契约等为基础，设计了良好的系统分层结构并进行职责分配，分为客户、管理员、前台、经理、调度五部分，以 UML 的交互图 Sequence diagram 来描述软件对象的协作过程。。

1. 客户

1. 系统事件：ChangeTargetTemp(RoomId,TargetTemp)

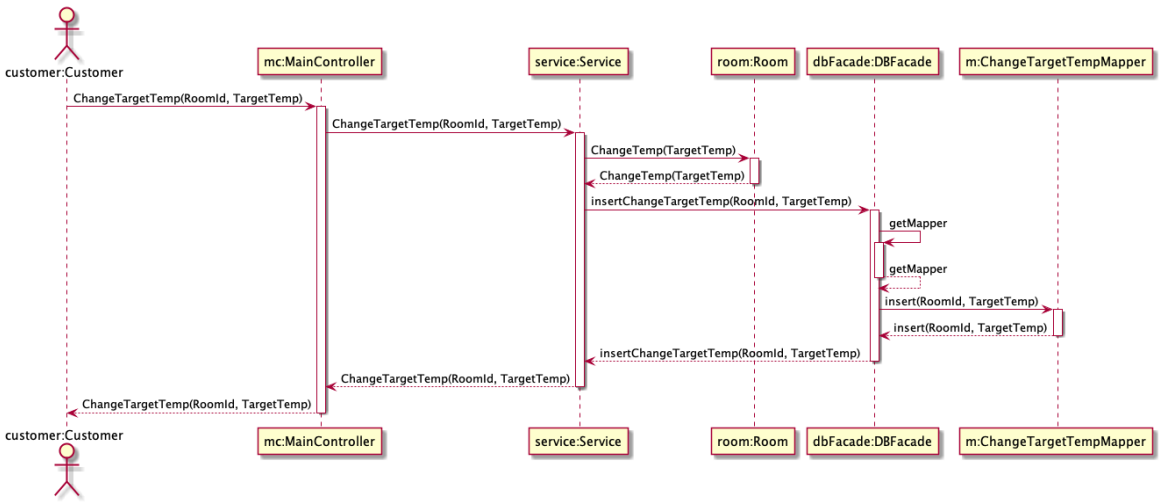
操作契约：

- 后置条件：
  a) 调度对象与房间建立关联（调度对象首先接受调温请求）
  b) 调度对象与服务对象建立关联（调度对象将请求转发给服务对象）

设计用例实现过程：

a) MainrController 作为控制器对象接受系统事件 ChangeTargetTemp
b) 控制器将请求转发给服务对象 Service
c) 执行服务对象 Service 的 ChangeTargetTemp 方法
d) 执行房间对象 Room 的 ChangeTargetTemp 方法
e) Service 对象通过持久化层持久化记录

Sequence diagram 如下：



2. 系统事件：ChangeFanSpeed (RoomId,FanSpeed)

操作契约：

- 后置条件：

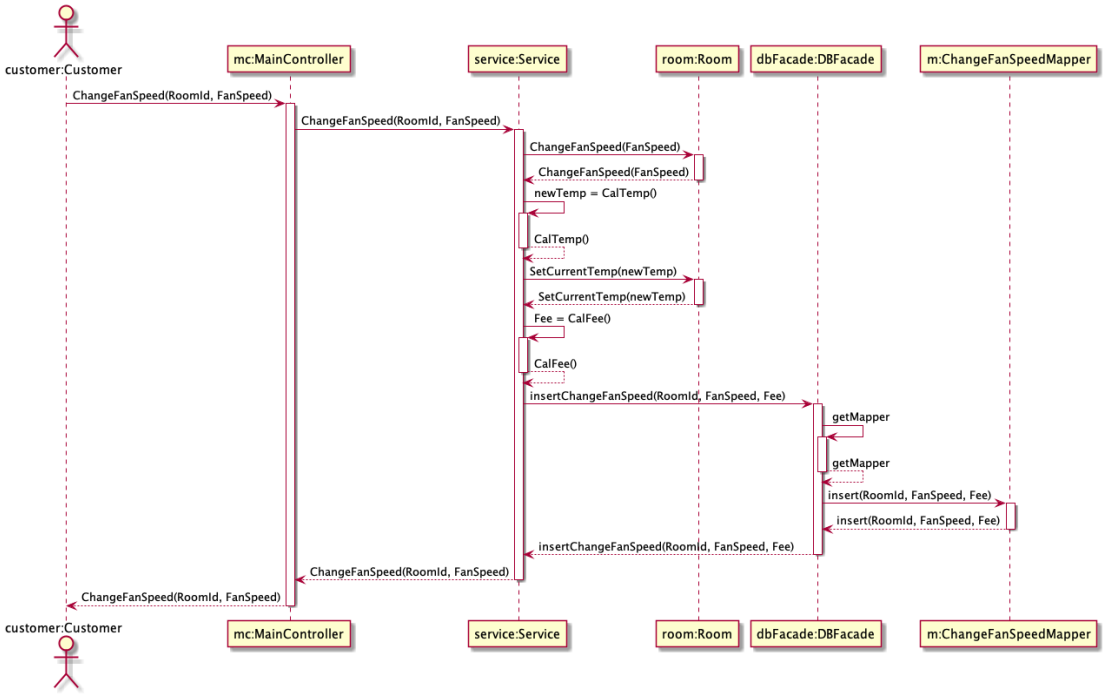
a) 调度对象与房间建立关联（调度对象首先接受调温请求）
b) 调度对象与服务对象建立关联（调度对象将请求转发给服务对象）

设计用例实现过程：

a) MainController 作为控制器对象接受系统事件 ChangeFanSpeed
b) 控制器将请求转发给服务对象 Service

c) 执行服务对象 Service 的 ChangeFanSpeed 方法
d) 执行房间对象 Room 的 ChangeFanSpeed 方法
e) 执行服务对象 Service 的 CalTemp 方法
f) 执行房间对象 Room 的 SetCurrentTemp 方法
g) 执行服务对象 Service 的 CalFee 方法
h) Service 对象通过持久化层持久化记录

Sequence diagram 如下：



2. 管理员

1. setdefault 设置默认空调状态

系统事件	修改默认空调设置
交叉引用	管理员设置空调默认设置
前置条件	管理员已经登陆系统且空调主机已经打开
后置条件	中央空调默认设置风速、温度、运行模式被修改 从控机属性风速、温度、运行模式被修改 主机 m 与从机 f 进行类关联。
事件名称	功能
Setdefault(temp,speed,mode)	系统管理员向控制器 adminController 发送设置申请。
Setdefault_m(temp,speed,mode)	控制器向主机 main 的实例 m 发送修改申请，修改 m 的默认风速、温度和模式
Setdefault_f(temp,speed,mode)	主机 m 循环地与从控机 follower 的实例 f

	进行类关联，修改 f 的默认风速、温度和模式

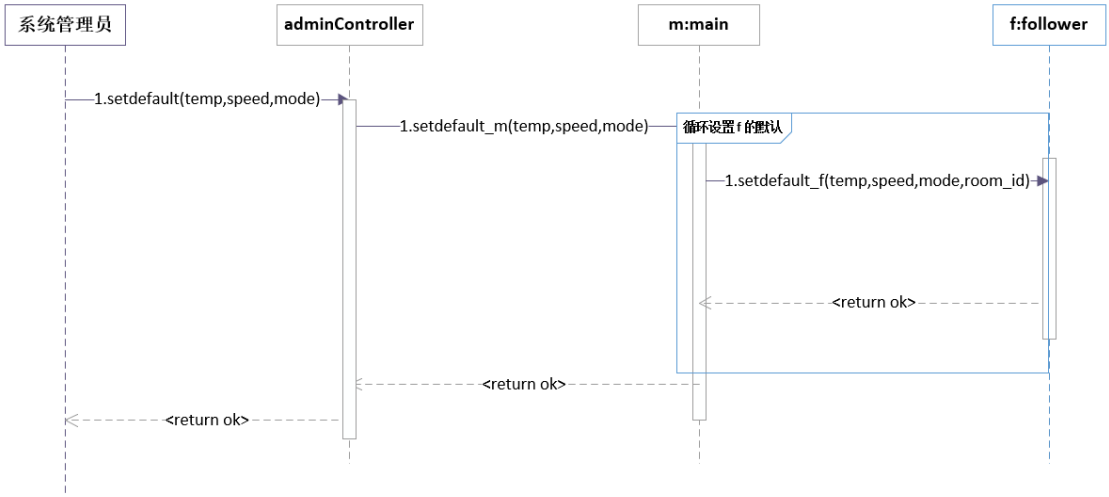
设计用例实现过程：

a) AdminController 作为控制器接受系统事件 setdefault

b) AdminController 将消息传给主机类的实例 m，m 的风速、温度、运行模式属性被修改

c) 主机类 m 与从机类的实例 f 进行关联，f 的风速、温度、运行模式被修改。

Sequence diagram 如下：



2．监控空调用例(checkinfo)

要求确定该监控指令所对应的控制器对象以及负责获取所有房间状态信息的监控对象

系统事件	查询各从控机状态
交叉引用	监测从控机工作状态
前置条件	中央空调已成功打开并正常工作，管理员已经登录系统
后置条件	1. 一个（概念类）空调信息表创建 2. 空调信息表与从机进行关联 3. 主机与从机进行关联 4. 空调信息表被赋值 5. 空调信息表的信息一分钟返回一次。
事件名称	功能
Checkinfo(room_id)	系统管理员向控制器 adminController 发送监控申请,并将需要监控的房间号 room_id 作为参数传入。
Info_request(room_id)	控制器向主机 main 的实例 m 发送监控申请，并将需要监控的房间号 room_id 传入
M_info_request(room_id)	主机 m 与对应 room_id 号的从控机 follower 的实例 f 进行类关联，每隔一分钟查看一次空调状态(speed,temp,mode)

Additem()	向空调信息表中添加信息

设计用例实现过程：

1、 adminController 作为控制器接受系统事件 checkinfo

2、 adminController 将信息发送给主机类的实例 m

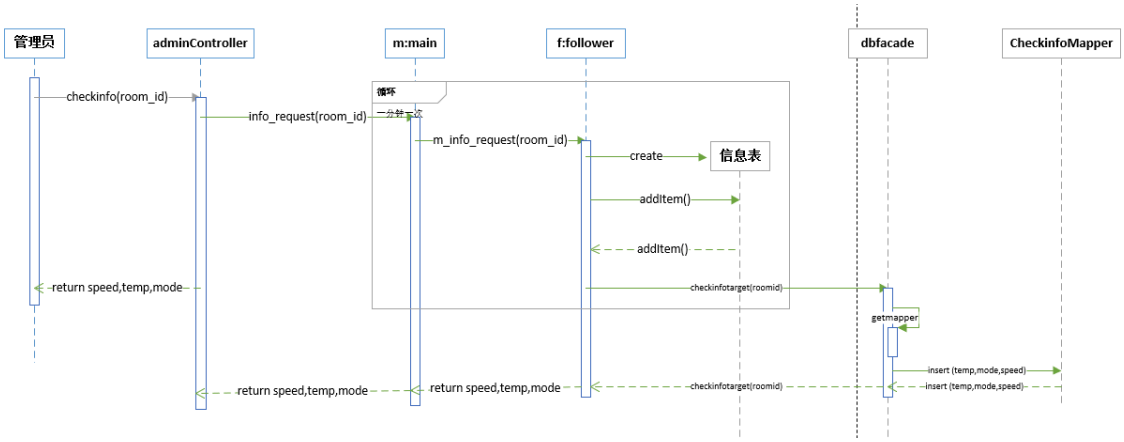
3、 m 与从机类的实例 f 通过 room_id 进行关联。

4、 f 创建信息表，能够记录每次的信息记录

5、 信息表的信息返回给请求者。

6、 通过持久化层持久化记录。

Sequence diagram 如下：



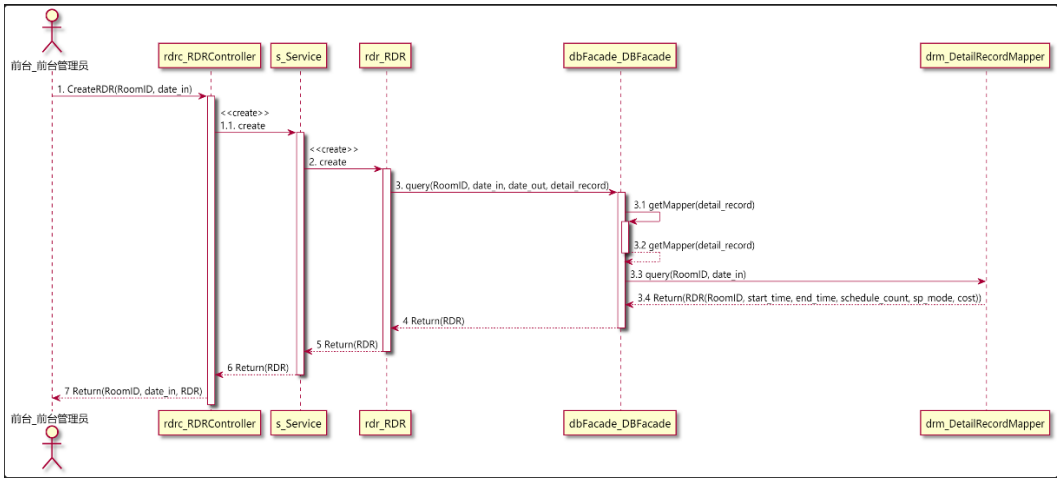
3. 前台

1. CreateRDR

前置条件：前台管理员对象被创建，顾客对象发出获取详单请求。

后置条件：

- 服务对象 s 被创建
- 详单对象 RDR 被创建
- 详单对象 RDR 的各个属性被赋值



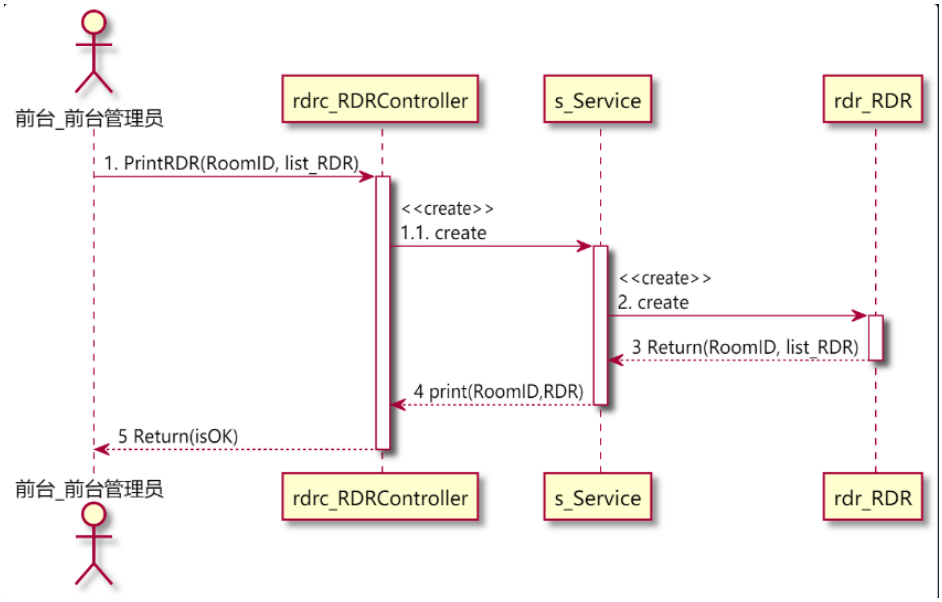
设计用例实现过程：

a) rdrc 作为控制器接收系统事件 CreateRDR
b) 根据操作契约的第一条创建服务对象 s，s 实例由 rdrc 创建，具有缺省的关联关系
c) 初始化 s，即该实例能够记录多个 RDR 详单记录，s 负责创建一个 RDR 集合
d) rdr 对象被创建，像持久化层发出请求，查询获取对应的持久化层详单记录信息
e) 通过持久化层信息，修改 rdr 对象的属性

2. PrintRDR

前置条件：详单对象 RDR 被创建
后置条件：详单对象 RDR 被反应到 UI 界面
设计用例实现过程：

a) rdrc 作为控制器接收系统事件 PrintRDR
b) 根据操作契约的第一条创建服务对象 s，s 实例由 rdrc 创建，具有缺省的关联关系
c) 初始化 s，即该实例能够记录多个 RDR 详单记录，s 负责创建一个 RDR 集合
d) rdr 对象返回数据给服务对象 s，由 s 负责处理系统事件



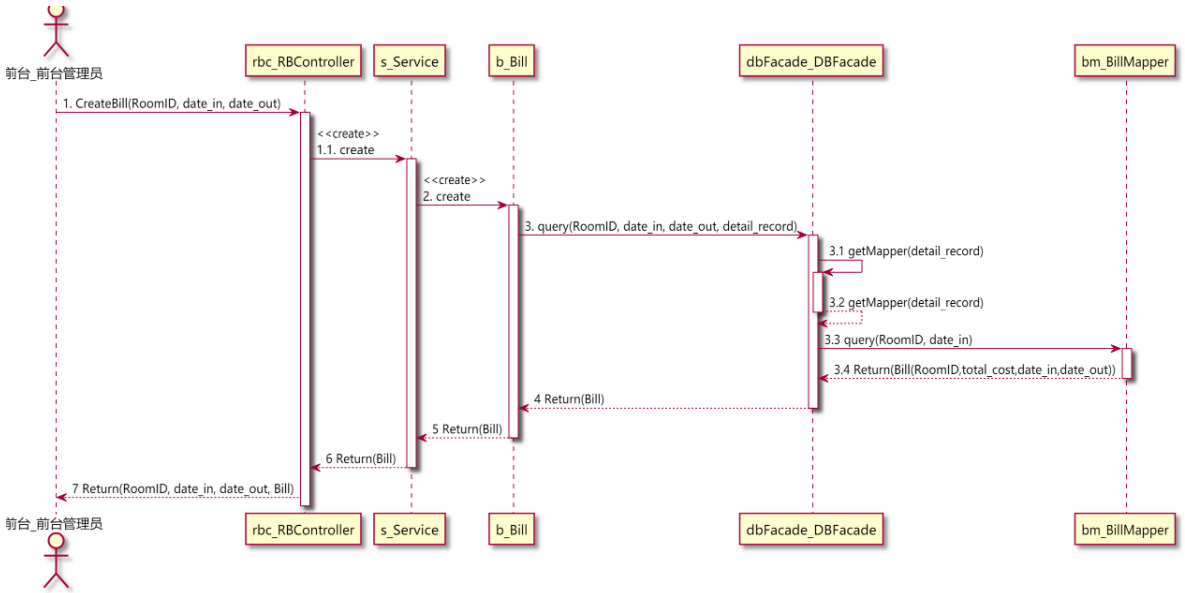
3. CreateBill

前置条件：前台管理员对象被创建，顾客对象发出获取账单请求。
后置条件：

- 服务对象 s 被创建
- 账单对象 RDR 被创建
- 账单对象 RDR 的各个属性被赋值

设计用例实现过程：

a) rbc 作为控制器接收系统事件 CreateBill
b) 根据操作契约的第一条创建服务对象 s，s 实例由 rbc 创建，具有缺省的关联关系
c) 初始化 s，即该实例能够记录 Bill 账单记录，s 负责创建一个 Bill 对象
d) Bill 对象被创建，像持久化层发出请求，查询获取对应的持久化层账单记录信息
e) 通过持久化层信息，修改 Bill 对象的属性



4. PrintBill

前置条件：详单对象 RDR 被创建

后置条件：详单对象 RDR 被反应到 UI 界面

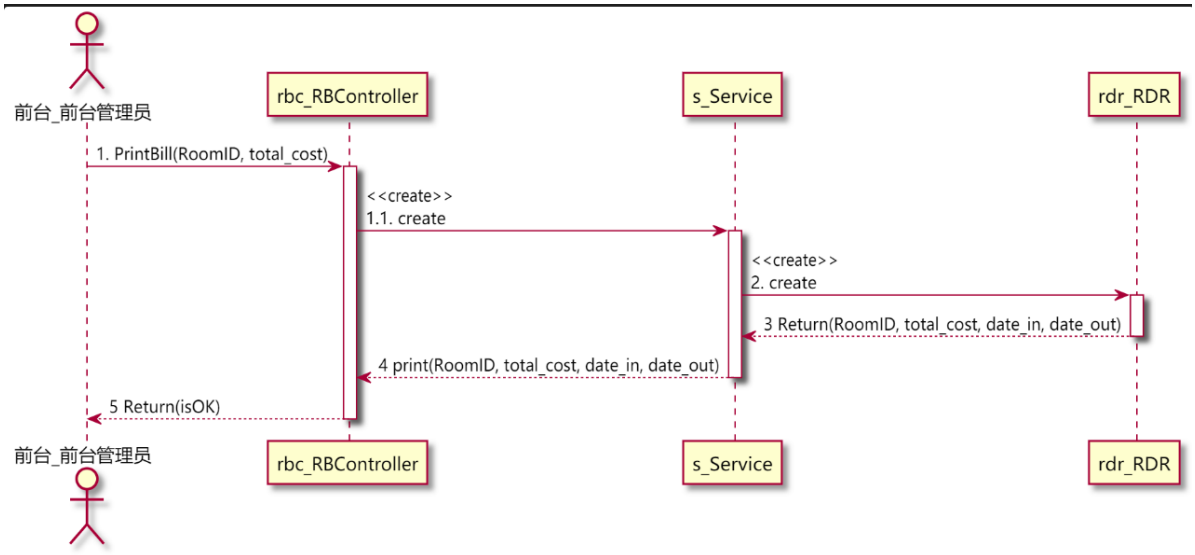
设计用例实现过程：

a) rbc 作为控制器接收系统事件 PrintBill

b) 根据操作契约的第一条创建服务对象 s，s 实例由 rbc 创建，具有缺省的关联关系

c) 初始化 s，即该实例能够记录 Bill 账单记录，s 负责创建一个 Bill 对象

d) Bill 对象返回数据给服务对象 s，由 s 负责处理系统事件



4.经理

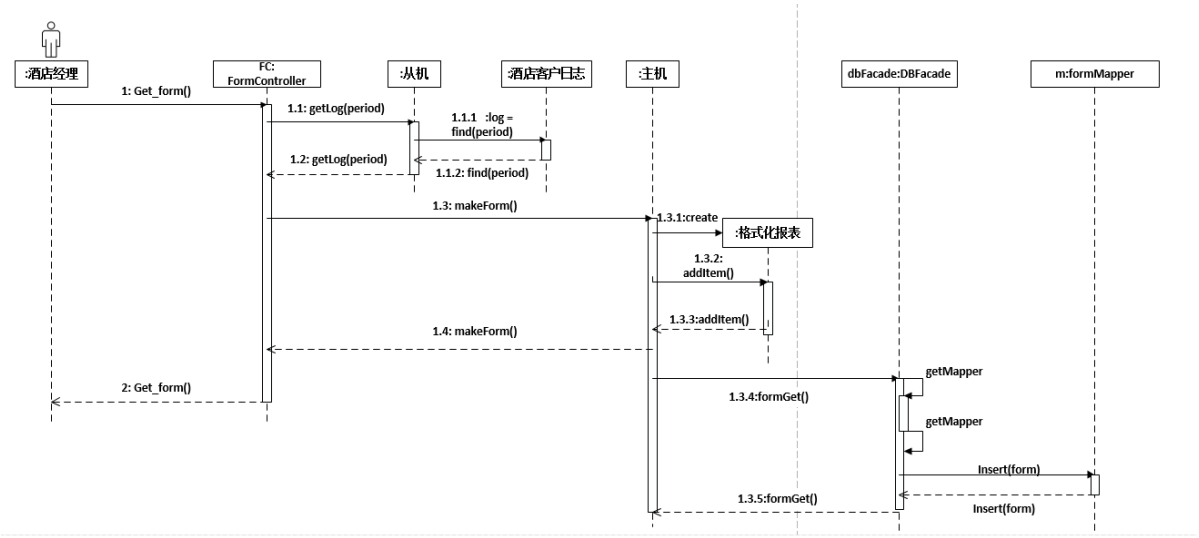
1.Get_form(perid)

分析过程：

问题 1：控制器的选择
用例控制器，选择 FormController 作为用例控制器
问题 2：创建格式化报表
格式化报表由主机来创建（创建者模式）
问题 3：酒店客户日志获取
酒店客户日志由从机获取（信息专家模式）

操作	Get_form(period)　请求获取一段时间内的报表
交叉引用	生成格式化报表
前置条件	经理账号登陆
后置条件	1.一个（概念类）格式化报表创建； 2. 格式化报表与主机建立关联 3. 格式化报表被赋值 4. 格式化报表在酒店客户操作的基础上，与酒店客户日志建立关联

根据以上分析，构建动态模型如下：



5. 调度

ChangeFanSpeed(RoomId, FanSpeed)

前置条件：
管理员在运行空调系统时，生成一个 MainController 实例并进行初始化，MainController 创建并维护服务队列和等待队列。

设计用例实现过程：

1. 选择 MainController 作为调度对象（即控制器对象）
2. MainrController 作为控制器对象接受系统事件 ChangeFanSpeed
3. 执行控制器的 GetServiceNumAndRunningNum 方法，判断当前服务对象数是否达到上限。

3.1 若未达到上限

3.1.1 控制器对象将请求转发给 ServiceList，再由 ServiceList 将请求转发给服务对象 Service。

3.1.2 执行房间对象 Room 的 ChangeFanSpeedTemp 方法

3.1.3 Service 对象通过持久化层持久化记录

3.2 若达到上限，则需要进行调度

3.2.1 控制器对象调用 IsAtServiceList 方法，判断当前服务请求的 Room 是否正在被服务。若是，控制器对象将请求转发给 ServiceList，再由 ServiceList 将请求转发给服务对象 Service，释放其所占用的资源，并从 WaitingList 中选取一个请求分配资源。

3.2.2 调用控制器对象的 GetLowestSpeed 方法，得到当前服务队列的最低风速 LowestSpeed，由此判断当前服务请求能否获得资源。

3.2.2.1 FanSpeed > LowestSpeed

a) 调用 ServiceList 的 GetNumLowestSpeed 方法，得到等于当前最低风速的正在被服务的请求的个数。

若为 1，则由 ServiceList 将请求转发给服务对象 Service，释放其所占用的资源。

若不为 1，则调用 ServiceList 的 ReleaseByServiceTime 方法，将请求转发给服务对象 Service，释放资源。

b) 控制器对象将请求转发给 ServiceList，再由 ServiceList 将请求转发给服务对象 Service。

执行房间对象 Room 的 ChangeFanSpeedTemp 方法

Service 对象通过持久化层持久化记录

3.2.2.1 FanSpeed == LowestSpeed

a) 调用 MainController 的 AddWaiting 方法，把当前服务请求加入 WaitingList。

b) 在等待时间内监听，若有服务对象的资源得到释放，则从从 WaitingList 中选取一个请求分配资源。

c) 在等待时间为 0 时，（若不考虑有新的更高级别的请求到达的情况），此时该服务一定能够分配资源。

首先调用 ServiceList 的 ReleaseByServiceTime 方法，将请求转发给服务对象 Service，释放资源。

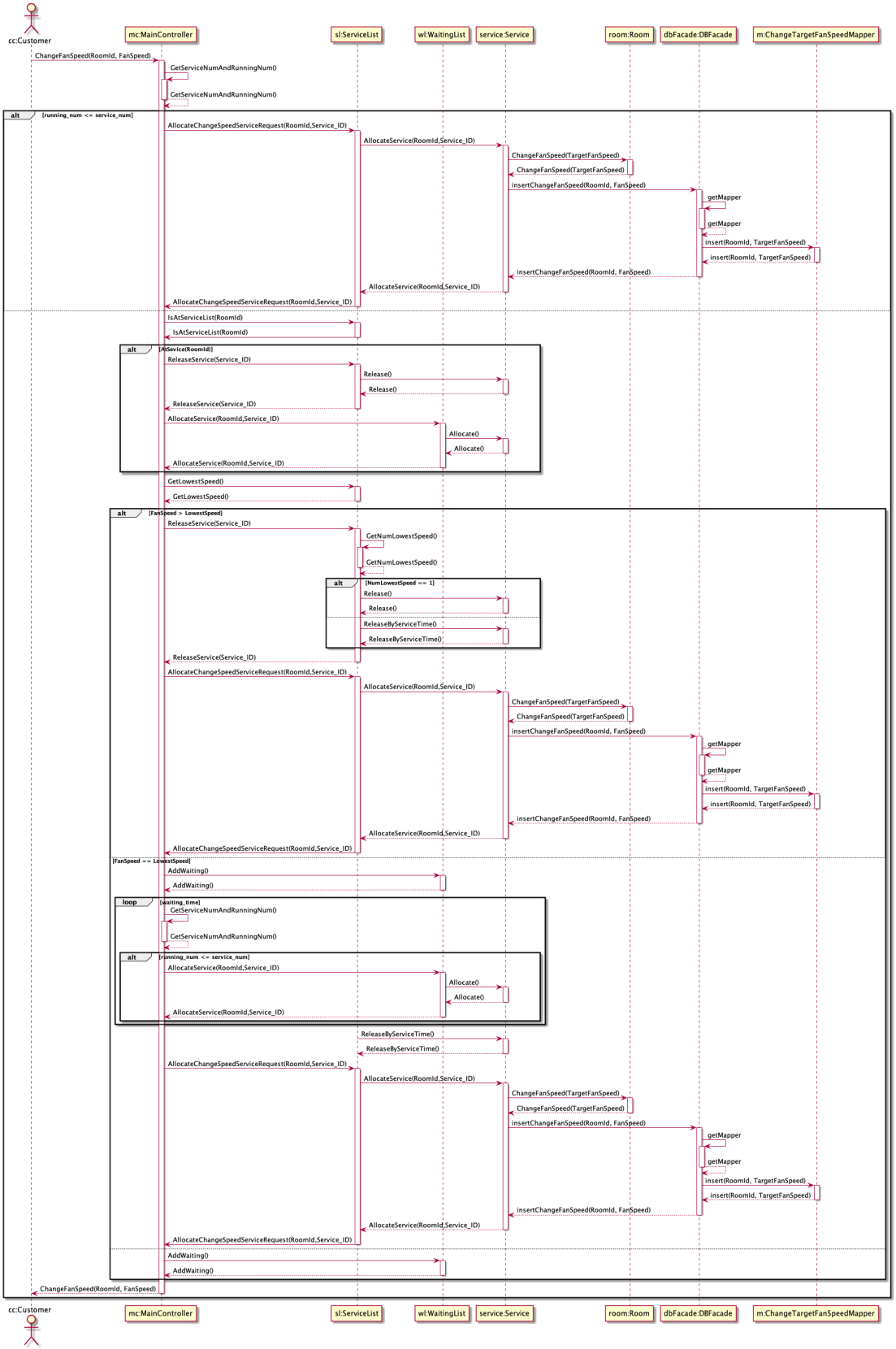
d) 控制器对象将请求转发给 ServiceList，再由 ServiceList 将请求转发给服务对象 Service。

e) 执行房间对象 Room 的 ChangeFanSpeedTemp 方法

f) Service 对象通过持久化层持久化记录

3.2.2.1 FanSpeed > LowestSpeed

调用 MainController 的 AddWaiting 方法，把当前服务请求加入 WaitingList。


cc:Customer
mc:MainController
sl:ServiceList
wl:WaitingList
service:Service
room:Room
dbFacade:DBFacade
m:ChangeTargetFanSpeedMapper
ChangeFanSpeed(RoomId, FanSpeed)
GetServiceNumAndRunningNum()
GetServiceNumAndRunningNum()
alt [running_num <= service_num]
AllocateChangeSpeedServiceRequest(RoomId,Service_ID)
AllocateService(RoomId,Service_ID)
ChangeFanSpeed(TargetFanSpeed)
ChangeFanSpeed(TargetFanSpeed)
insertChangeFanSpeed(RoomId, FanSpeed)
getMapper
getMapper
insert(RoomId, TargetFanSpeed)
insert(RoomId, TargetFanSpeed)
insertChangeFanSpeed(RoomId, FanSpeed)
AllocateService(RoomId,Service_ID)
AllocateChangeSpeedServiceRequest(RoomId,Service_ID)
IsAtServiceList(RoomId)
IsAtServiceList(RoomId)
alt [AtSevice(RoomId)]
ReleaseService(Service_ID)
Release()
Release()
ReleaseService(Service_ID)
AllocateService(RoomId,Service_ID)
Allocate()
Allocate()
AllocateService(RoomId,Service_ID)
GetLowestSpeed()
GetLowestSpeed()
alt [FanSpeed > LowestSpeed]
ReleaseService(Service_ID)
GetNumLowestSpeed()
GetNumLowestSpeed()
alt [NumLowestSpeed == 1]
Release()
Release()
ReleaseByServiceTime()
ReleaseByServiceTime()
ReleaseService(Service_ID)
AllocateChangeSpeedServiceRequest(RoomId,Service_ID)
AllocateService(RoomId,Service_ID)
ChangeFanSpeed(TargetFanSpeed)
ChangeFanSpeed(TargetFanSpeed)
insertChangeFanSpeed(RoomId, FanSpeed)
getMapper
getMapper
insert(RoomId, TargetFanSpeed)
insert(RoomId, TargetFanSpeed)
insertChangeFanSpeed(RoomId, FanSpeed)
AllocateService(RoomId,Service_ID)
AllocateChangeSpeedServiceRequest(RoomId,Service_ID)
[FanSpeed == LowestSpeed]
AddWaiting()
AddWaiting()
loop [waiting_time]
GetServiceNumAndRunningNum()
GetServiceNumAndRunningNum()
alt [running_num <= service_num]
AllocateService(RoomId,Service_ID)
Allocate()
Allocate()
AllocateService(RoomId,Service_ID)
ReleaseByServiceTime()
ReleaseByServiceTime()
AllocateChangeSpeedServiceRequest(RoomId,Service_ID)
AllocateService(RoomId,Service_ID)
ChangeFanSpeed(TargetFanSpeed)
ChangeFanSpeed(TargetFanSpeed)
insertChangeFanSpeed(RoomId, FanSpeed)
getMapper
getMapper
insert(RoomId, TargetFanSpeed)
insert(RoomId, TargetFanSpeed)
insertChangeFanSpeed(RoomId, FanSpeed)
AllocateService(RoomId,Service_ID)
AllocateChangeSpeedServiceRequest(RoomId,Service_ID)
AddWaiting()
AddWaiting()
ChangeFanSpeed(RoomId, FanSpeed)