DocNo: 001.E.1.1

**Grape**

**Software Architecture**

**Document**

**Version 1.0**

**By**:

Group Undefined

2015-04

**Group Member**:

Hunter Lin

Birdy

Listen

Morning

Syachi

**Document Language**:

English

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 2015.4.12 | 1.0 | Initialization of the report | Hunter Lin |
| Final Date | 2.0 |  |  |

Contents

[**1.** **Introduction** 5](#_Toc417418435)

[1.1. Purpose 5](#_Toc417418436)

[1.2. Scope 5](#_Toc417418437)

[1.3. Reference 5](#_Toc417418438)

[**2.** **Architectural Representation** 5](#_Toc417418439)

[**3.** **Architectural Goals and Constraints** 5](#_Toc417418440)

[**4.** **Use-Case View** 5](#_Toc417418441)

[4.1 Overview 5](#_Toc417418442)

[4.2 Architecturally Significant use cases 5](#_Toc417418443)

[**5.** **Logical View** 5](#_Toc417418444)

[5.1. Overview 5](#_Toc417418445)

[5.2. Front-end Interaction Mechanisms 6](#_Toc417418446)

[5.2.1. Front Controller 6](#_Toc417418447)

[5.2.2. Command Delegator 6](#_Toc417418448)

[5.2.3. Service Locator 6](#_Toc417418449)

[5.2.4. Security Handler 6](#_Toc417418450)

[5.3. Data Operation Mechanisms 6](#_Toc417418451)

[5.3.1. Persistency 6](#_Toc417418452)

[5.3.2. Session Facade 6](#_Toc417418453)

[5.4. Architecturally Significant Use Case Realization 6](#_Toc417418454)

[5.5. Architecturally Significant Model Elements 6](#_Toc417418455)

[5.6. Architecturally Significant Classes 7](#_Toc417418456)

[**6.** **Process View** 7](#_Toc417418457)

[**7.** **Deployment View** 7](#_Toc417418458)

[**8.** **Implementation View** 7](#_Toc417418459)

[**9.** **Size and Performance** 7](#_Toc417418460)

[**10.** **System Size** 7](#_Toc417418461)

**蓝色部分：Syachi**

**黄色部分：Hunter Lin**

**红色部分：Listen**

**紫色部分：Morning**

**绿色部分：Birdy**

1. **Introduction**

## 1.1. Purpose

This document provides a comprehensive architectural overview of Grape, using a number of different architectural views to depict different aspects of the system. It intends to capture and convey the significant architectural decisions, which have been made on the system.

## 1.2. Scope

This document should be an overview of the whole architecture and the way it should be modeled. Decisions made in this document affect how the system is modeled.

## 1.3. Reference

1. **Architectural Representation**
2. **Architectural Goals and Constraints**
3. **Use-Case View**

## 4.1 Overview

## 4.2 Architecturally Significant use cases

1. **Logical View**

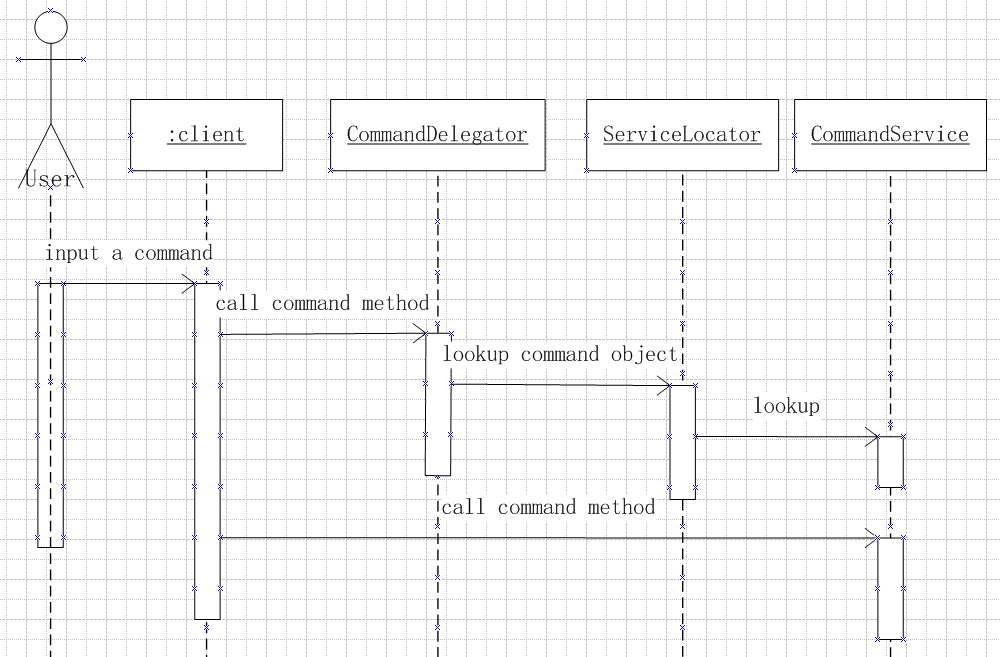
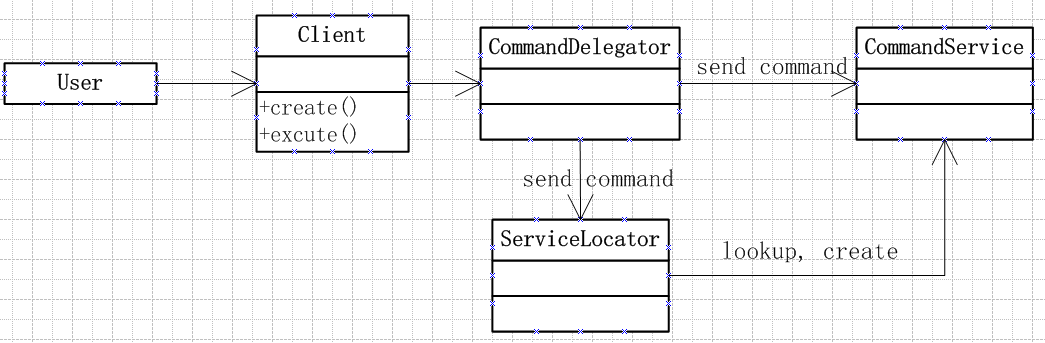
## 5.1. Overview

## 5.2. Front-end Interaction Mechanisms

### 5.2.1. Front Controller

### 5.2.2. Command Delegator

Class diagram and sequence diagram:



**How it works:**

A client (in most cases it is a command) that requires access to a business service component creates an instance of a session delegate called CommandDelegator.

**Key Point:**

Usually the command delegate is used with a session facade, typically there is an one-to-one relationship between the two.

A CommandDelegator uses a component called the ‘lookup service’ (ServiceLocator). The ‘lookup service’ (ServiceLocator) is responsible for hiding the underlying implementation details of the business service lookup code. (The ServiceLocator mechanism will be depicted later.)

**Advantages:**

1. Use a CommandDelegator to reduce coupling between presentation-tier clients and business services. The CommandDelegator hides the underlying implementation details of the command service, such as lookup and access details of the architecture.
2. If necessary, the delegate may cache results and references to remote command services. Caching can significantly improve performance.

### 5.2.3. Service Locator

### 5.2.4. Security Handler

## 5.3. Data Operation Mechanisms

### 5.3.1. Persistency

### 5.3.2. Session Facade

## 5.4. Architecturally Significant Use Case Realization

## 5.5. Architecturally Significant Model Elements

## 5.6. Architecturally Significant Classes

1. **Process View**

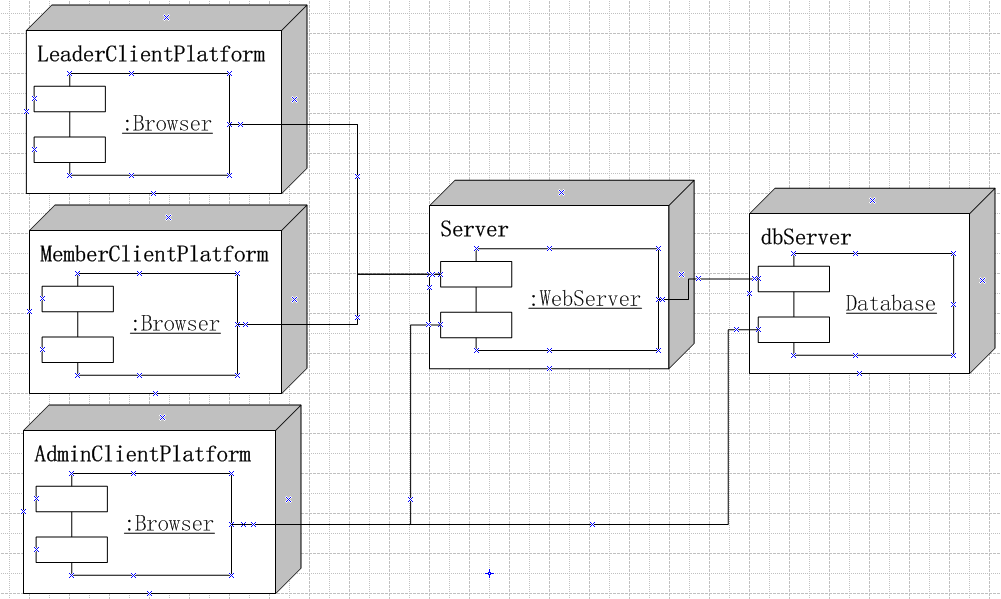
我觉得这部分没必要有，整合的时候可以删了

The process view of a system shows the assignment of active classes (classes that must run in their independent threads of control) to the operating systems processes and threads. In the case of our platform, the operating system resources are “hidden” underneath the container services. In other words, the infrastructure manages the operating system resources. In particular, the containers are placed in operating system processes and the containers in turn manage thread-pools and assign the threads to active objects.

Since the threads are managed by container automatically, we didn’t give a process view in our model.

1. **Deployment View**

The deployment view of a system shows the physical links between different nodes when the system works. Grape basically runs on a web server, with a dbServer providing access to data of users. Users can access to grape with a browser, while an admin can have access directly to the database.



1. **Implementation View**
2. **Size and Performance**
3. **System Size**