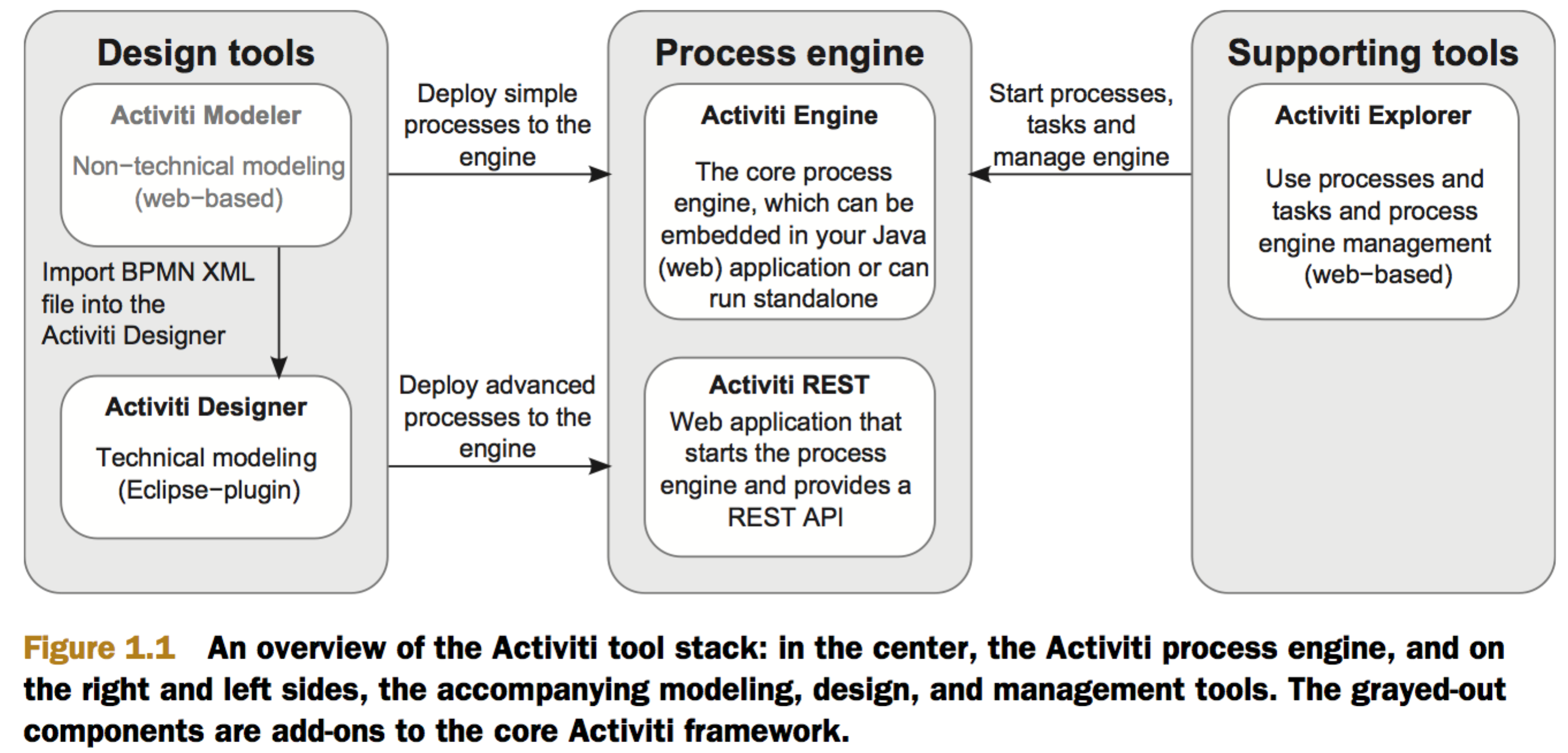
Activiti

# 基础知识

activiti是轻量级的工作流和业务流程管理（BPM）平台，核心是BPMN2流程Java引擎。



Activiti Engine：核心组件。

Activiti Modeler：web上的BPMN2.0建模工具；

Activiti Designer：Eclipse 插件，在Eclipse中定义和执行流程。

Activiti Exploer：部署和控制流程，查看数据库表，管理用户。

Activiti REST：web应用，提供Activiti Engine 的REST接口。



## Getting Started

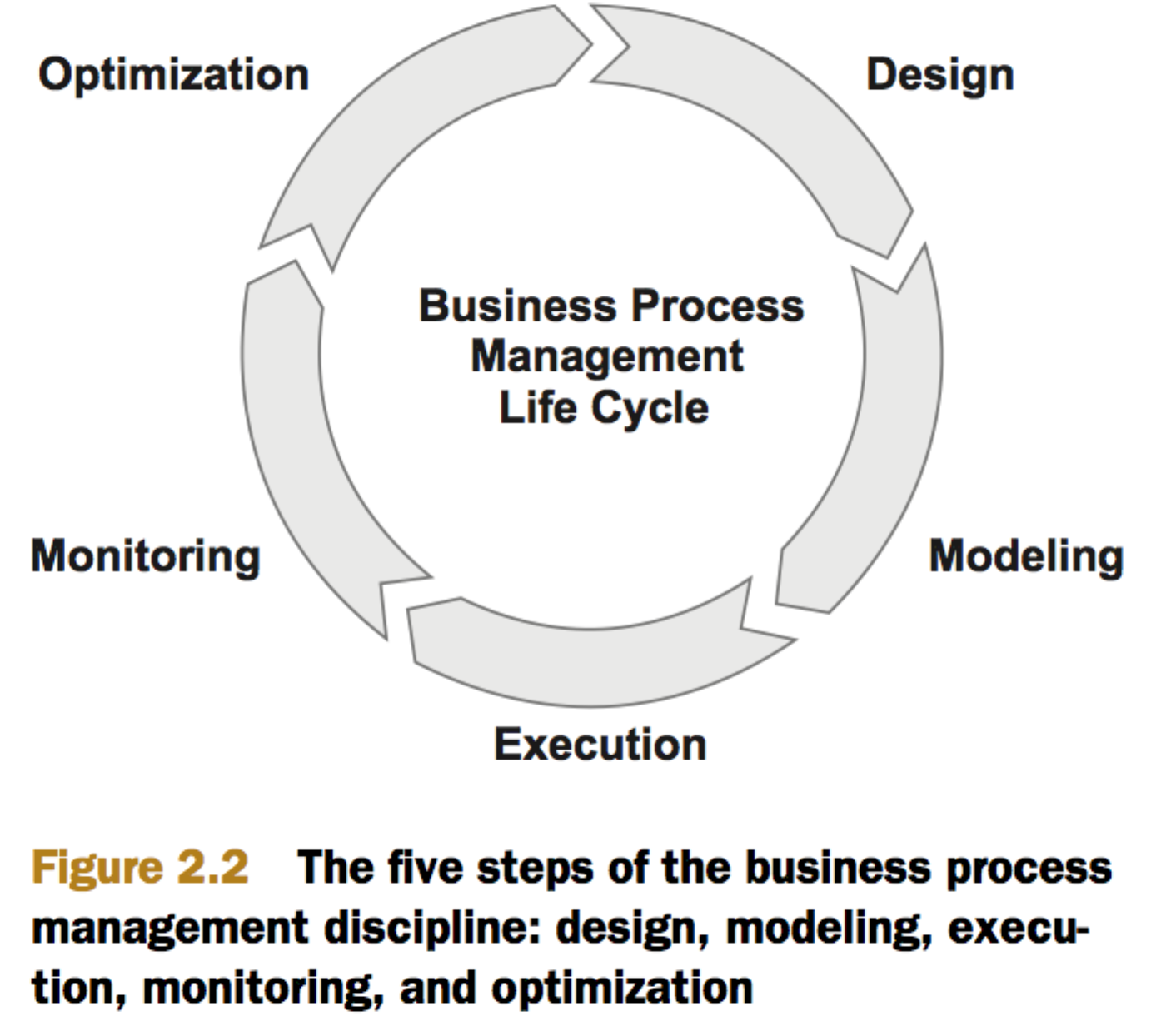
将activiti-explorer.war拷贝到Tomcat的webapps中，启动Tomcat，访问 http://localhost:8080/activiti-explorer。explorer运行于内嵌的H2数据库中。

用户名密码：

|  |  |  |
| --- | --- | --- |
| 账号 | 密码 | 角色 |
| kermit | kermit | admin |
| gonzo | gonzo | manager |
| fozzie | fozzie | user |

|  |  |  |  |
| --- | --- | --- | --- |
| Webapp名称 | URL | 描述 |  |
| Activiti Explorer | http://localhost:8080/  activiti-explorer | 流程引擎的用户控制台。使用它来启动新流程，分配任务， 查看并认领任务，等等。这个工具也可以用来管理Activiti引擎。 |  |

## BPMN2.0设计



Design：设计业务流程，确定最高等级的活动，讨论可能的变化。

Modeling：建模，规范化业务流程。

Execution：利用BPMS（business process management system）如activiti执行。

Monitoring：检查流程是否满足绩效指标。

Optimization：优化流程。

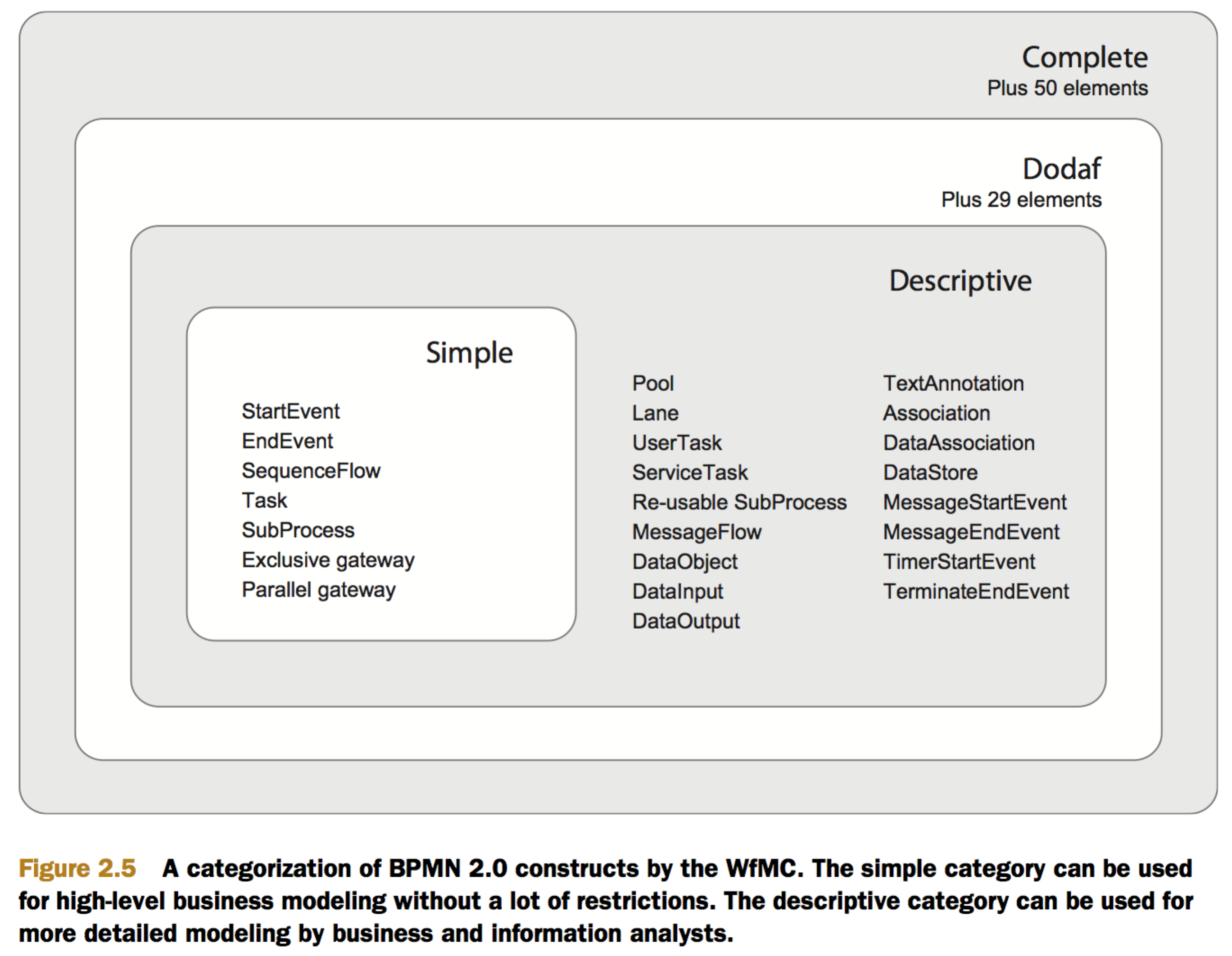
Silver在BPMN Method Style中将BPMN2.0的组件分为三个等级：

Level1用于最高等级建模。

Level2用于细节建模，包括事件和异常处理。

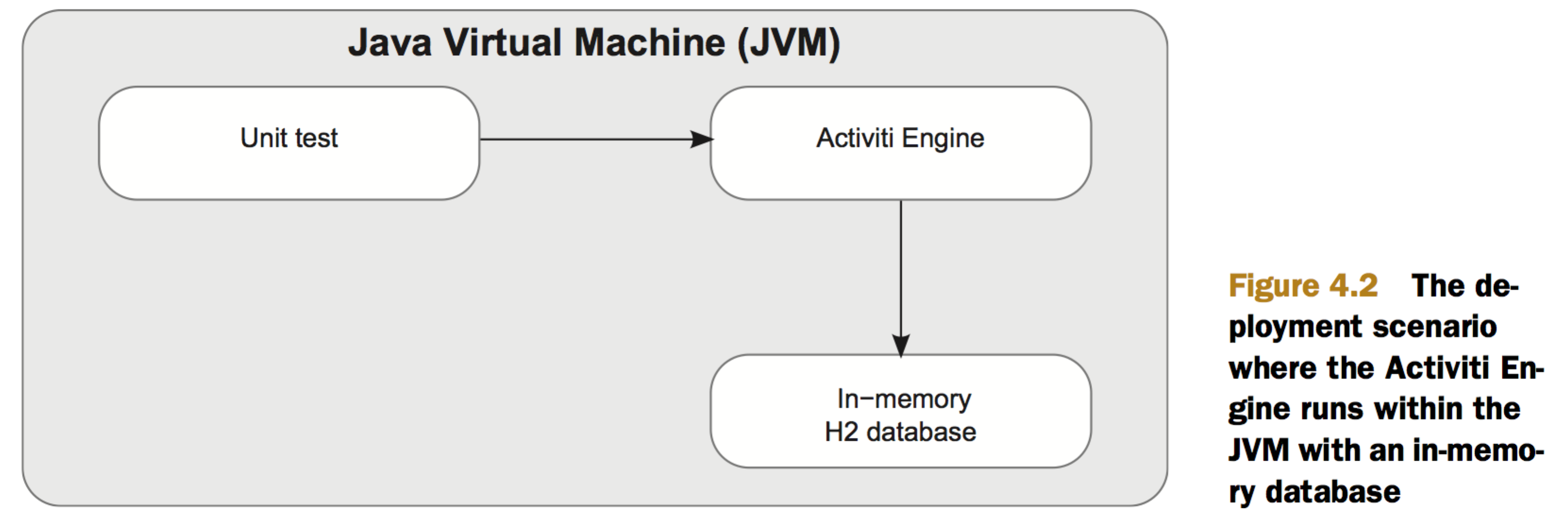
Level3是执行建模，部署到流程引擎中。

WfMC的分级：Descriptive对应Level1，DODAF对应Level2，Complete对应Level3。区别是将Simple作为最高等级的建模抽象。

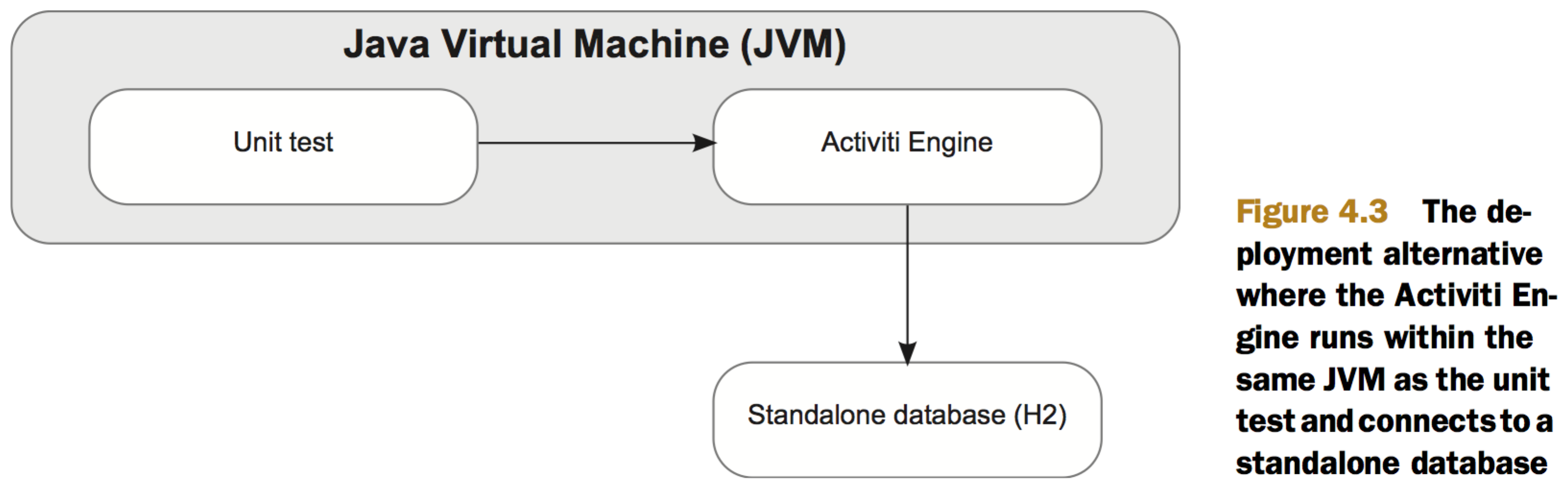


## Activit engine运行环境

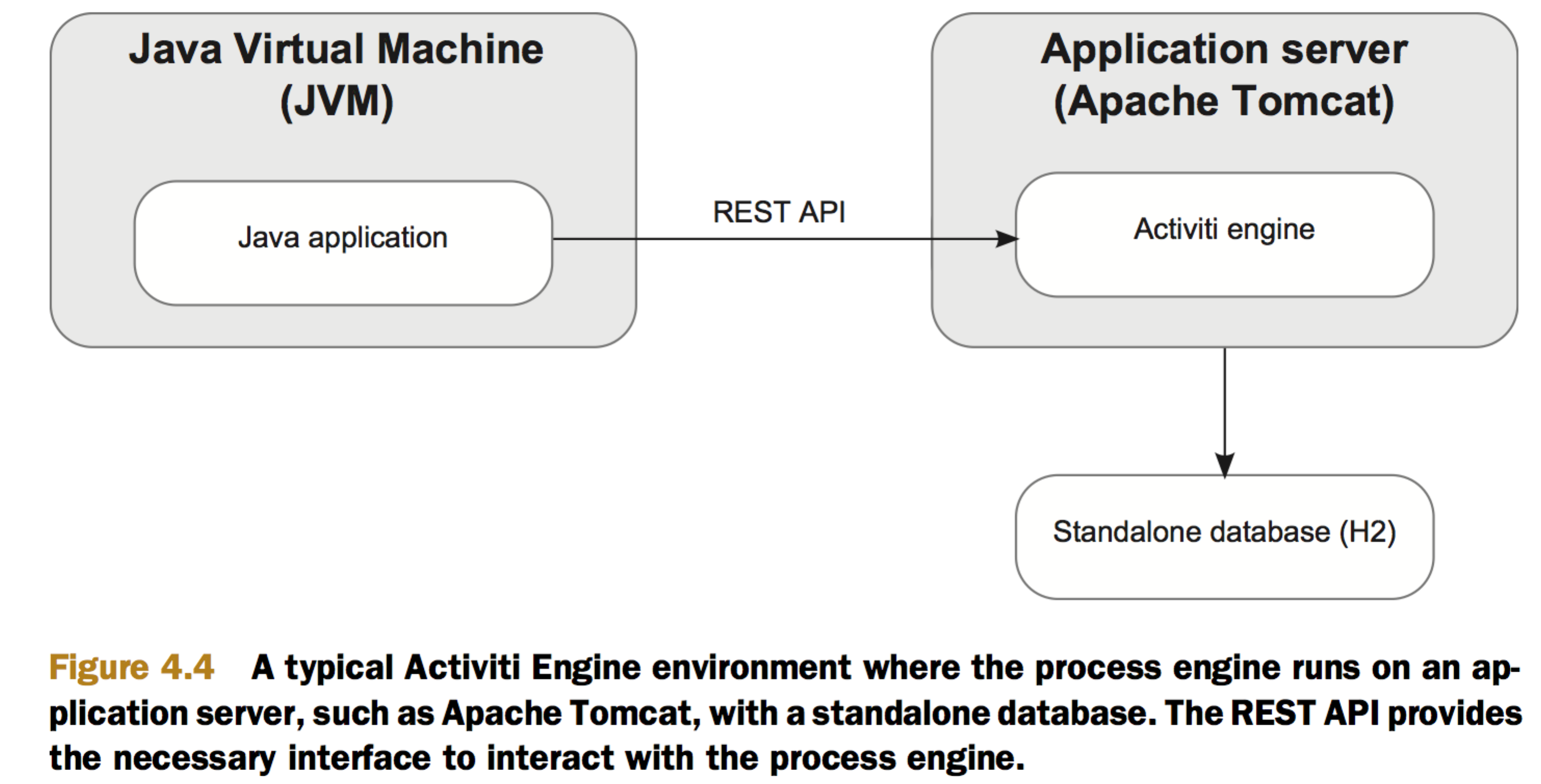
1. 内存h2数据库



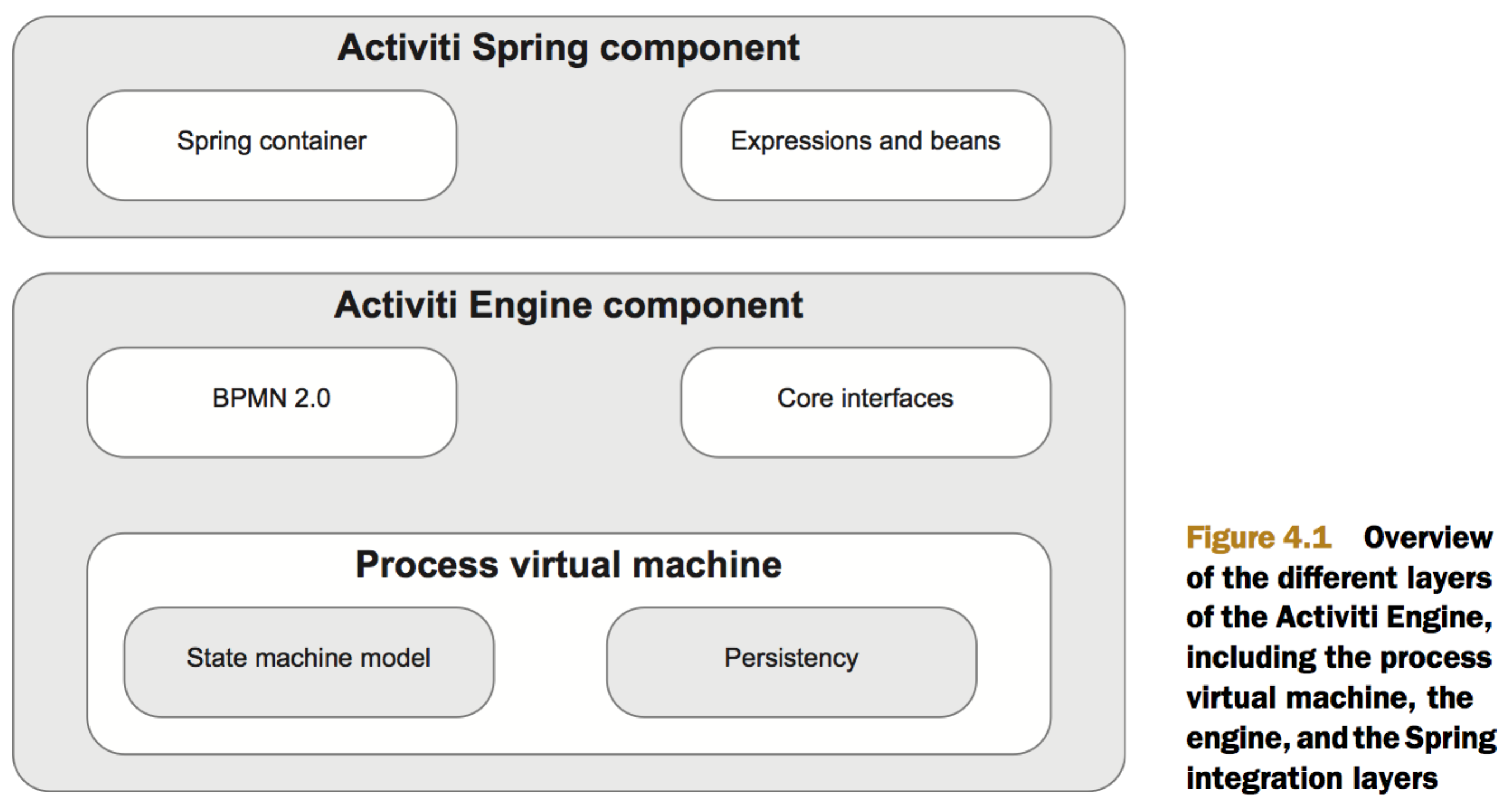
1. 独立数据库



1. 在tomcat中的独立数据库



## Activiti Engine架构



## Service Task实现方式

### 实现JavaDelegate接口

类需实现JavaDelegate的接口

public void execute(DelegateExecution execution)

配置：

<serviceTask id="servicetask1" name="Validate order" activiti:class="com.meituan.ValidateService"></serviceTask>

### 不实现JavaDelegate接口

<serviceTask id="serviceTask1" name="Validate order"

activiti:expression="#{bookOrder.validate(isbn)}" activiti:resultVariableName="validatetime"/>

bookOrder不用实现JavaDelegate接口，但须实现Serializable接口。

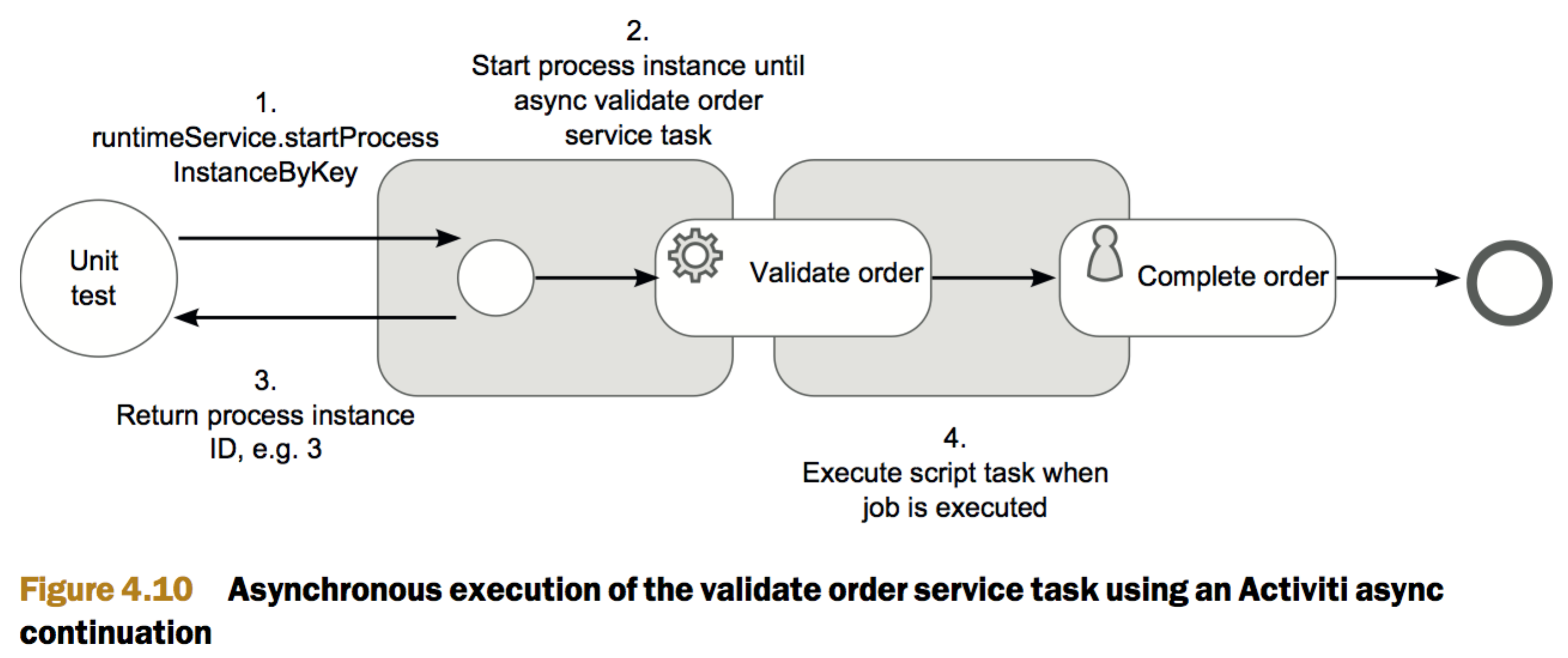
其返回值付给validatetime

activiti:expression="#{bookOrder.isbn}"

将调用bookOrder的setIsbn方法

### 异步执行

<serviceTask id="servicetask1" name="Validate order" **activiti:async="true"** activiti:class="com.meituan.ValidateService"></serviceTask>



### 注入参数

类中设置org.activiti.engine.impl.el.Expression

Expression isbn;

isbn.getValue(execution)

<serviceTask id="servicetask1" name="Validate order" activiti:class="com.meituan.ValidateServiceWithFields">

<extensionElements>

<activiti:field name="validatetext" stringValue="Validation done at"/>

<activiti:field name="isbn">

<activiti:expression>

${isbn > 999999 ?isbn:1000000+isbn}

</activiti:expression>

</activiti:field>

</extensionElements>

</serviceTask>

### Script Task

scriptFormat：指明脚本语言（Groovy默认），符合JSR-223标准。

<scriptTask id=”” scriptFormat=”groovy”>

<script>

out:println “xxx”

sum = 0;

for(i in inputArray){

sum += i;

}

def bookVar = "BPMN 2.0 with Activiti"

execution.setVariable("bookName", bookVar);

</script>

</scriptTask>

脚本中直接读流程变量inputArray，sum会自动加入流程变量，如果避免该规则，则设置sum为局部变量，def sum = 0 ，或通过execution显示设值

# 关键对象

## ActivitiRule

Activiti JUnit4的单元测试功能封装在ActivitiRule中，ActivitiRule继承自TestWatchman。创建ActivitiRule时，会搜索classpath下的activiti.cfg.xml文件以构造Activiti Engine。

ActivitiRule的主要用途是在单元测试前部署流程定义，通过@Deployment(resources=” , ”)指定流程定义文件。

**@Deployment会在测试完后清空流程实例，用户任务等。**

## Deployment

流程部署对象。可包含各种资源，如流程定义，业务规则，流程定义图像等。

## ProcessDefinitions

流程定义，定义流程中各个节点的结构和行为，部署成功后自动创建流程定义。对应一个.bpmn文件中Process部分。

ProcessDefinitionID举例：simplebookorder:1:3为流程定义ID：版本号：所在数据库ID

## ProcessInstances

流程实例，流程定义的执行过程，对每个流程定义来说，同一时间会有很多实例在执行。

## Task

任务，流程在流转过程中的一个原子活动，流程中的工作不能被拆分到更细。在Activiti 中的Task仅指有角色参与的任务，即定义中的UserTask。

## Execution

执行计划，流程实例和流程执行中的所有节点都是Execution，如UserTask、ServiceTask 等。

# 关键接口

## ProcessEngineConfiguration

清除数据库记录

CommandExecutor commandExecutor = processEngineConfiguration.getCommandExecutorTxRequired();

commandExecutor.execute(new Command<Object>() {

public Object execute(CommandContext commandContext) {

DbSqlSession session = commandContext.getSession(DbSqlSession.class);

session.dbSchemaDrop();

session.dbSchemaCreate();

return null;

}

});

ProcessEngineConfiguration.createXXX()返回ProcessEngineConfiguration对象。

createXXX分为两类：一类传入activiti.cfg.xml文件路径；一类采用默认内置的H2数据库。

## ProcessEngine

流程引擎接口，提供流程管理和运作的所有接口。

两种创建ProcessEngine方式：

1. **ProcessEngines获取默认ProcessEngine**

ProcessEngine processEngine = ProcessEngines.getDefaultProcessEngine();

搜索classpath下得activiti.cfg.xml文件

1. **ProcessEngineConfiguration. buildProcessEngine()**

使用：

获取各种Service的接口

## RepositoryService

提供部署（Deployments）和流程定义（Process Definitions）管理。管理静态资源。

获取：

RepositoryService repositoryService = processEngine.getRepositoryService();

使用：

1. 查询部署

**DeploymentQuery** deploymentQuery= repositoryService.createDeploymentQuery();

1. 查询流程定义

**ProcessDefinitionQuery** processDefinitionQuery = repositoryService.createProcessDefinitionQuery();

1. 部署和删除流程定义

DeploymentBuilder deploymentBuilder = repositoryService.createDeployment();

**Deployment** deployment = deploymentBuilder.addClasspathResource(String resource). deploy();

String deploy\_id = deployment.getId();

repositoryService.deleteDeployment(deploymentID, true);

如果为true，则流程定义和其关联的流程实例一起删除

如果为false，则只删除流程定义，不删除流程实例，如果有关联的流程实例，则抛异常

## RuntimeService

RuntimeService runtimeService = processEngine.getRuntimeService();

运行时服务接口，提供流程启动服务，运行中流程查询，运行变量设置和获取。管理动态资源。

1. 启动流程实例

ProcessInstance processInstance = runtimeService.startProcessInstanceByKey(

"bookorder", variableMap);

由于采用同步执行，startProcessInstanceByKey会立刻创建并执行流程实例，在遇到wait状态时返回流程实例，UserTask是让其进入Wait状态的情况之一，Script Task不是。

startProcessInstanceById(String processDefinitionID)，由RepositoryService部署时返回processDefinitionID

区别是同一流程名称PD\_KEY可能有多个流程定义，startProcessInstanceByKey总是以最新的流程定义创建流程实例，startProcessInstanceById是以某个流程定义创建流程实例。

1. 查询流程实例

List<ProcessInstance> instanceList = runtimeService .createProcessInstanceQuery().processDefinitionKey("bookorder") .list();

返回bookorder的所有流程实例，如果需要指定某个实例，用processDefinitionId

1. 查询流程实例变量

runtimeService.getVariable(processInstance.getId(), "validatetime");

## TaskService

用户任务接口（UserTask），提供运行时查询、认领、完成、删除任务及变量设置用户管理等服务。

获取：

TaskService taskService = processEngine.getTaskService();

使用

1. 查询任务

**TaskQuery** tq = [TaskService](eclipse-javadoc:%E2%98%82=mtoa/src%5C/main%5C/java%3Ccom.meituan.oa.pf.bpm.service.impl%7BBpmBizServiceImpl.java%E2%98%83BpmBizServiceImpl%5EtaskService%E2%98%82TaskService).createTaskQuery();

tq.taskCandidateUser();//任务所有者

tq.processDefinitionKey();//

tq.taskId(String id); //任务ID

tq.taskCandidateUser(String userID); //候选人ID

tq.taskAssignee(String userID); //负责人ID

tq.taskCreatedAfter(Date date); //任务创建时间在这之后

tq.taskCreatedBefore(Date date); //任务创建时间在这之前

tq.taskDescriptionLike(String desc); //任务描述正则式

//按照任务创建时间升序创建查询

TaskQuery tqu = tq.orderByTaskCreateTime().asc();

//返回查询到的Task

List<Task> results = tqu.list();

1. 认领和完成任务

Task task = taskService.newTask();

taskService.saveTask(task);

taskService.addCandidateUser(taskID, userID);

taskService.claim(taskID, userID);

taskService.complete(taskID);

任务优先级为[0,100]，默认为50

## ManagementService

流程引擎管理接口。

ManagementService managementService = processEngine.getManagementService();

1、查询数据库的表和表的元数据；

2、查询和管理异步操作的功能，如定时器，延迟暂停激活等。

## IdentityService

用户和组管理接口。

获取：

IdentityService identityService = processEngine.getIdentityService();

使用：

1. 用户操作

User newUser = identityService.newUser(String userID);

identityService.saveUser(newUser);

UserQuery userQuery = identityService.createUserQuery();

1. 组操作

Group newGroup = identityService.newGroup(String groupId);

identityService.saveGroup(newGroup);

GroupQuery groupQuery = identityService.createGroupQuery()

1. 其他操作

identityService.createMembership(String userID, String groupID);

将用户加入组

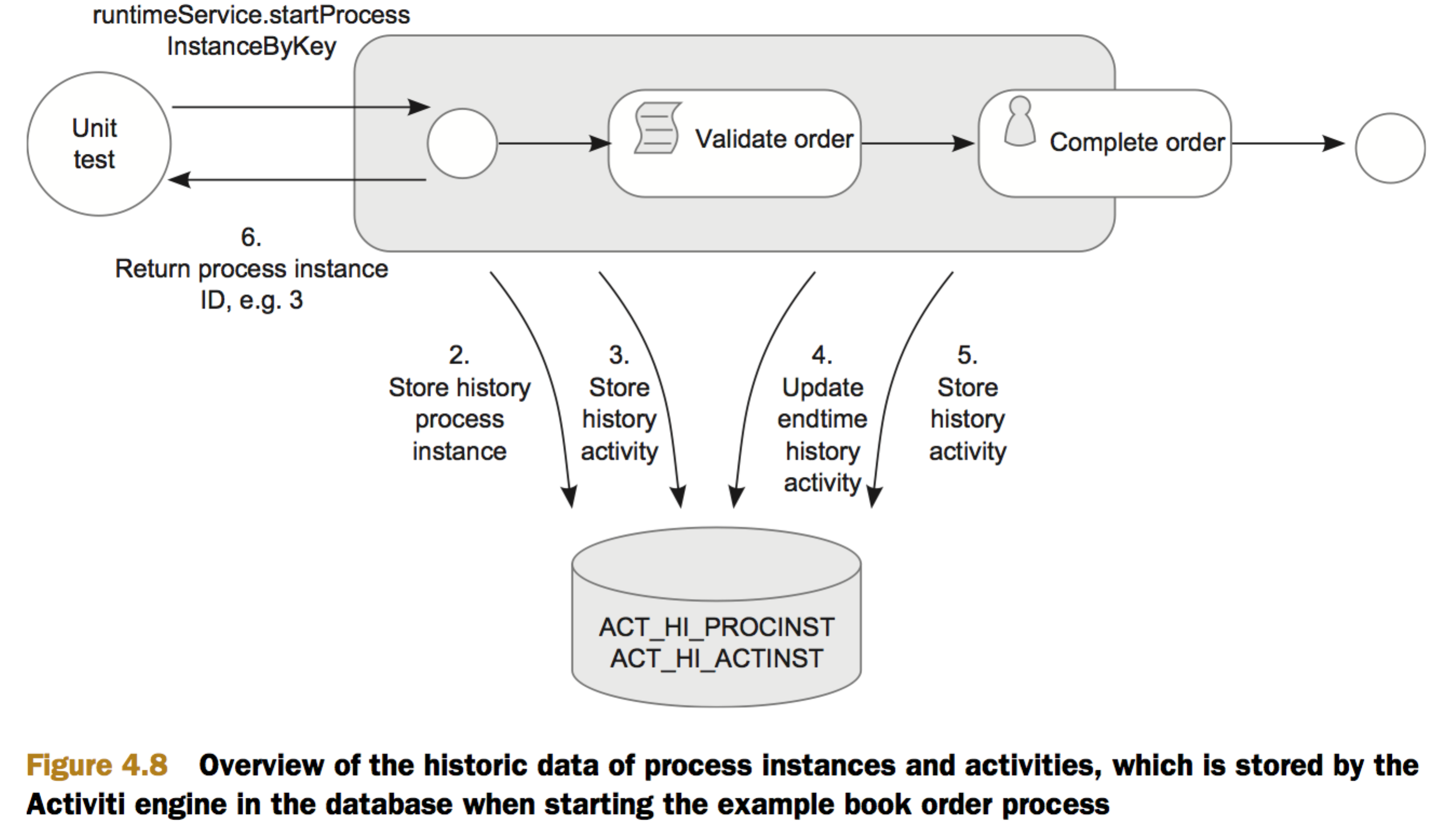
identityService.setAuthenticatedUserId("John Doe");

设置任务发起人，也可在bpmn文件中配置userID=starter的为活动发起人

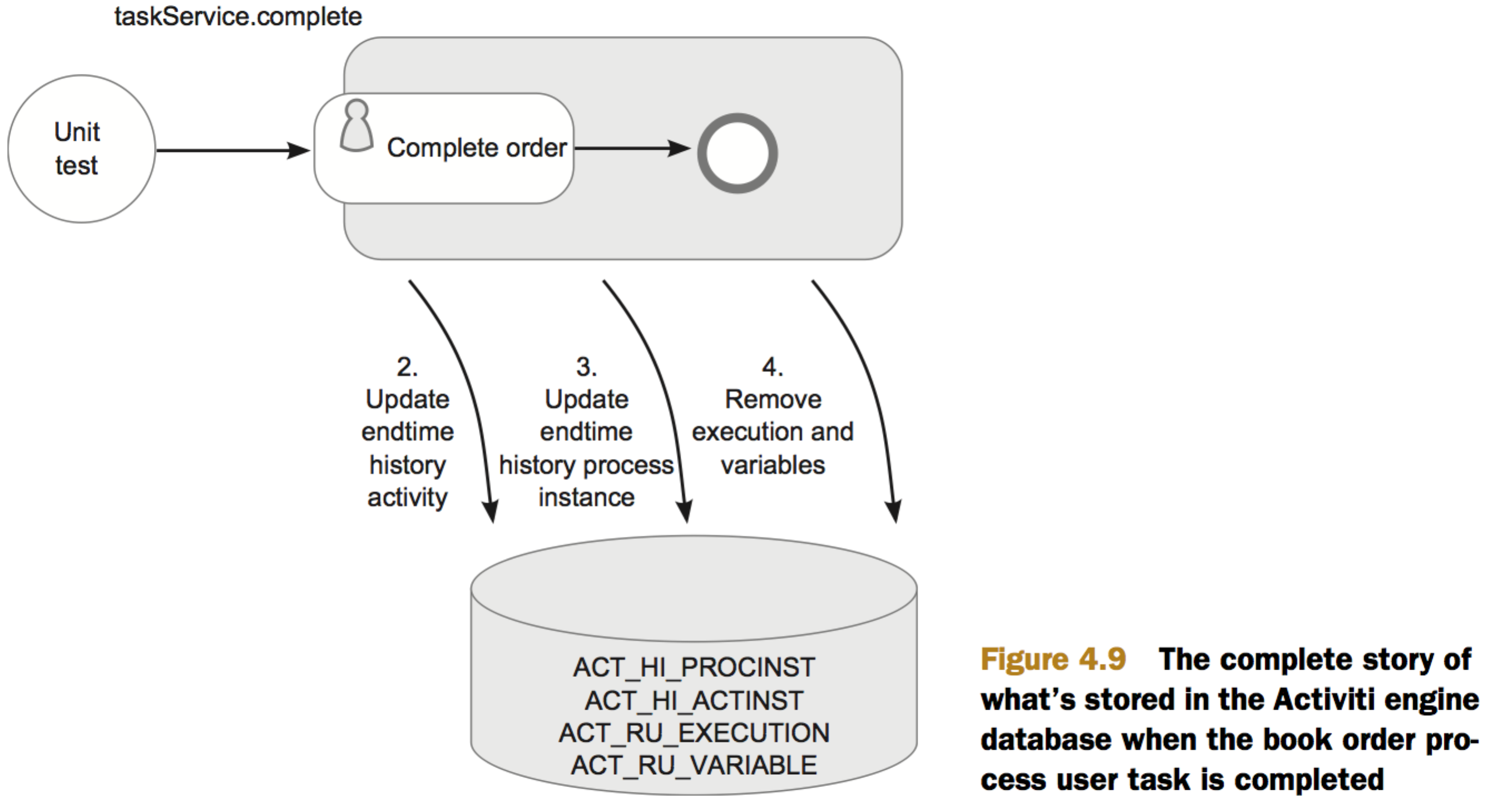
<startEvent id="startEvent" activiti:initiator="starter" />

## HistoryService

流程处理查询接口，包括执行中流程查询和历史流程查询。



流程启动时保存流程信息，活动启动时保存活动信息，更新活动结束时间，



活动结束时更新活动结束时间，流程结束时更新流程结束时间，最后删除运行时的信息和变量。当full时，流程变量将保存到ACT\_HI\_DETAIL中。

通过activiti.cfg.xml中设置Activiti Engine的属性，设置输出日志等级：

<bean id="processEngineConfiguration" class="org.activiti.spring.SpringProcessEngineConfiguration">

<property name="deploymentResources" value="classpath\*:/\*.bpmn" />

<property name="history" value="full" />

</bean>

None：不输出

Activity：归档流程和活动信息

Audit：默认，归档流程、活动、表属性

Full：在Audit基础上，归档流程变量和用户任务的表属性。

获取：

HistoryService historyService = processEngine.getHistoryService();

使用：

1. 获取历史流程

**HistoricProcessInstanceQuery** historicProcessInstanceQuery=historyService

.createHistoricProcessInstanceQuery()

**HistoricProcessInstance** historicProcessInstance =

historicProcessInstanceQuery .processInstanceId(processInstanceID).singleResult();

1. 获取历史活动

**HistoricActivityInstanceQuery** historicActivityInstanceQuery = historyService.createHistoricActivityInstanceQuery()

**HistoricActivityInstance**

1. 获取历史细节（在full等级生效）

**HistoricDetailQuery** historicDetailQuery = historyService.createHistoricDetailQuery()

## FormService

可选，提供了启动表单和任务表单两个概念。 启动表单会在流程实例启动之前展示给用户， 任务表单会在用户完成任务时展示。

获取：

FormService formService = processEngine.getFormService();

配置：

<activiti:formProperty id="name" name="Name" required="true" type="string" />

默认type为string，还可为long，boolean，enum，date。

使用：

List<FormProperty> formList = formService

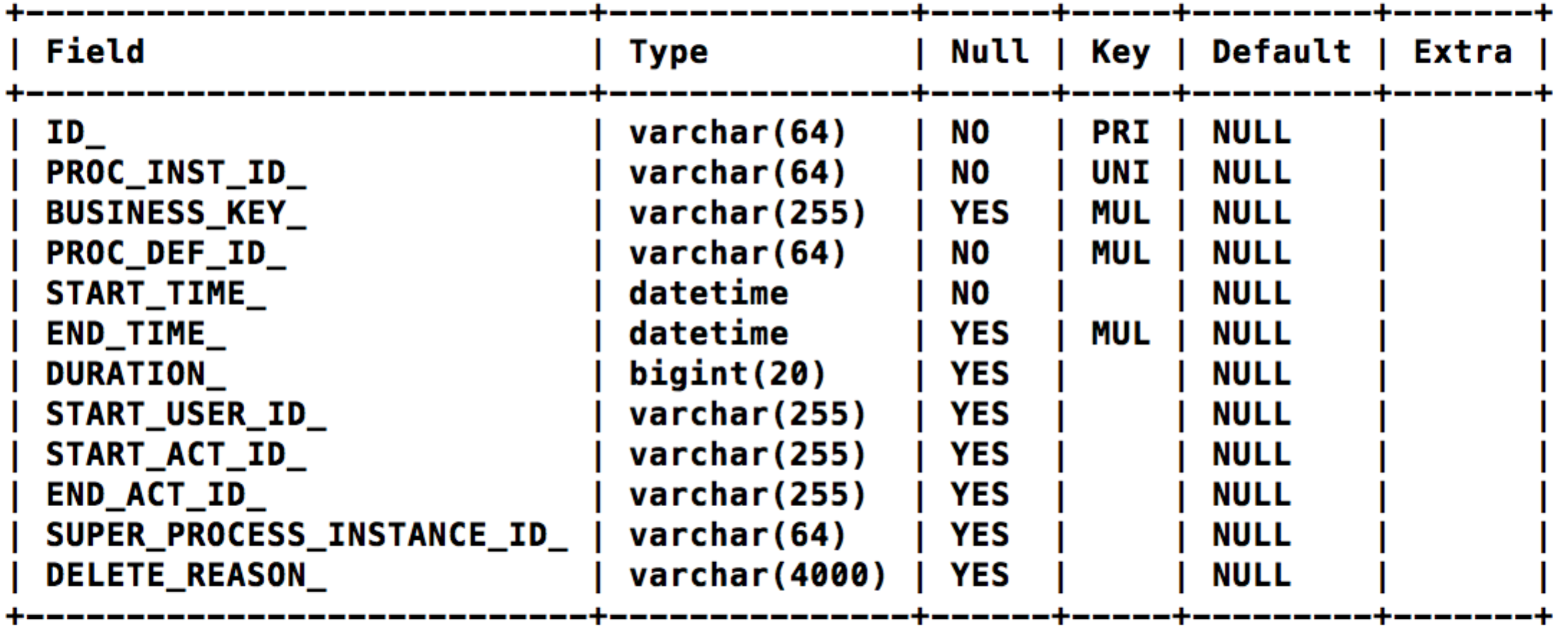
.getStartFormData(definition.getId()).getFormProperties();

formService.submitStartFormData(definition.getId(), formProperties);

# 表结构

## ACT\_HI\_PROCINST

历史流程实例



## ACT\_HI\_VARINST

历史活动

