

南京大學

本 科 毕 业 论 文（设计）

院 系_____软件学院_____

题 目_____实训平台之估算模块的设计与实现_____

学生姓名_____谢明娟_____学 号_____071251170_____

年 级_____四_____专 业_____软件工程_____

指导教师_____荣国平_____职 称_____讲师_____

论文提交日期_____2011.6_____

南京大学本科生毕业设计中文摘要

毕业论文题目：实训平台之估算模块的设计与实现

软件学院 院系 软件工程 专业 07 级本科生姓名：谢明娟

指导教师（姓名、职称）：荣国平 讲师

摘要：

团队软件过程（TSP）是一种团队级别的过程框架，用于指导团队成员合理规划和管理项目任务，使用预订的资源成本开发出高质量的软件产品。现实项目团队在使用 TSP 进行团队管理时，使用支持工具能够帮助其达到更好的效果，这就是实训平台设计的初衷。在对项目任务做详细计划之前，须先引入 PSP 中的估算方法对任务做早期规划，制定时间与规模估算，作为后续计划的基础。针对这一需求，在实训平台中设计并实现了估算模块。

该模块使用了现下流行的轻量级框架组合 Struts、Spring、Hibernate 的系统结构。前端采用 Struts 处理页面请求，中间层采用 Spring 管理业务逻辑并隔离上下层，底层应用 Hibernate 负责与数据库交互并实现对象持久化。SSH 的整合使用使得模块被分层，各层次之间相互独立，整个模块结构清晰，提高了代码的可重用性，加快了开发速度。

同时，该模块还使用了 Ext JS 技术开发用户界面，通过从 jsp 文件中抽离出 js 代码，并在 js 中复用 Ext 组件，提高了 UI 代码的结构性与可重用性。由于 Ext JS 本身就是 Ajax 工具包，优化了 Ajax 的使用，使得异步 web 请求变得更加轻松。

本文中就估算模块进行了全面的分析总结，描述其背景，需求与最终运行结果，并从技术角度再现该模块的架构设计、详细设计与具体实现，为下一步的改进打下基础。

关键词：TSP，实训平台，估算，SSH，Ext JS

南京大学本科生毕业设计英文摘要

THESIS : The Design and Implementation of the Estimation Module in Training Platform

DEPARTMENT : Software Institute

SPECIALIZATION: Software Engineering

UNDERGRADUATE: Xie Mingjuan

MENTOR: Rong Guoping

ABSTRACT :

Team Software Process (TSP) is a team-level process framework to guide the team members of rational planning and management of projects tasks, and to develop high-quality software products with limited resources. When users apply TSP into real team management, using support tools will be helpful to achieve better results, and this is our goal to develop training platform. Before making a detailed plan of project tasks, users should work out an early plan as the basis of follow-up ones using the estimation method in PSP, which includes time and size estimation. In response to this demand, the team design and implement the estimation module in training platform.

The module uses the popular combination of Struts, Spring and Hibernate to construct its structure, all of them are lightweight frameworks. Struts is used in front end to process page requests; In middle layer Spring will help managing business logic and isolating the upper and lower layers; And Hibernate is responsible for communicating with database and achieving object persistence in the bottom layer. The module is layered with the help of SSH and the layers are independent with each other. As a result, the structure of the module is clearer, the code reusability is better, and the developing speed is faster.

Meanwhile, the module uses the technology called Ext JS to develop user interface. By pulling the js code out of jsp files and reusing Ext components in the js code, the team achieve better structured and reusable program. As Ext JS is itself a kind of optimized Ajax toolkits, it makes the asynchronous web requests to be easier.

In this paper, it makes a comprehensive analysis and summary of the estimate

module. It includes the description of background, requirements and final operating results, as well as a reproduction of designing and implementation process from a technical point of view. It will lay a foundation for further improvements.

KEY WORDS: TSP, training platform, estimation, SSH, Ext JS

目 录

南京大学本科生毕业设计中文摘要	I
南京大学本科生毕业设计英文摘要	II
目 录	IV
第一章 绪论	1
1.1 项目背景	1
1.2 国内外 TSP 支持工具现状	2
1.3 论文组织结构	3
第二章 估算模块技术与理论概述	4
2.1 SSH 框架技术	4
2.2 Ext JS 技术	5
2.3 理论基础	7
第三章 项目概述	10
3.1 项目总体划分图	10
3.2 项目模块描述	10
第四章 估算模块分析与设计	12
4.1 需求分析	12
4.2 分层架构	13
4.2.1 表现层	14
4.2.2 业务层	15
4.2.3 持久层	15
4.2.4 领域模型	16
4.3 详细设计	16
4.3.1 管理基础信息子模块	16
4.3.2 管理估算数据子模块	21
4.3.3 管理历史数据子模块	24
4.3.4 选择估算方法估算子模块	27
第五章 估算模块实现	28
5.1 管理基础信息子模块	28
5.2 管理估算数据子模块	30
5.3 管理历史数据子模块	31
5.4 选择估算方法估算子模块	33
5.5 估算模块运行效果	35

第六章 总结与展望	39
参考文献	40
致谢	41

第一章 绪论

1.1 项目背景

团队软件过程 (Team software process, TSP) 是一个具有良好定义的框架[1], 也是一种用于组织和管理团队的最佳实践。它提供了一种已被证明的, 用于帮助计划、评估、管理和控制项目开发的方法[2]。它能够帮助团队制定并且遵守项目策略, 流程以及计划。与 TSP 面向团队不同, 个人软件过程 (Personal software process, PSP) 则是一种面向个人的过程框架。它为软件开发人员提供了自我改进的方法与规范[3], 用于帮助估算、制定个人计划、跟踪工作进度并最终生成一个高质量的产品。无论是在概念上, 还是在具体的操作上, PSP 中的这些方法与规范都是 TSP 团队管理的基础[4]。它们的提出弥补了 CMM 在小规模软件开发组织上和细节实现上的不足, 三者共同组成了 CMM/CMMI-TSP-PSP 的过程框架体系, 涵盖了组织级、团队级与个人级这 3 个级别。

一款完整的支持工具在概念实践时扮演着十分重要的角色。在实际应用 PSP 与 TSP 进行项目管理时, 项目组也同样需要使用辅助工具来帮助团队管理过程, 收集并分析过程数据。实训平台就是这样一款基于 Web 的 TSP 支持工具, 它在功能上涵盖了 TSP 中所有的基本管理元素: 阶段管理, 数据记录与分析, 角色管理, 计划管理, 历史数据管理等。在实际使用中, 实训平台能够共享团队实时数据, 能够指导团队成员按照过程规范进行数据记录与项目开发, 有效支持个人与团队活动, 实践 PSP 与 TSP 管理方法。

在 TSP 中, 强调了历史数据的重要性, 其中一项重要的应用就是利用已有的经验为新项目做早期规划, 包括规模估算与资源估算, 以更早的反应开发成本, 同时, 一份准确的项目估算也是制定合理的项目计划的基础, 能够帮助降低项目失败的风险。在 PSP 中, 采用了代理 (Proxy) 的方式来进行规划, 即 PROBE (PROxy Based Estimation) 估算方法, 这一方法通过建立估算代理与实际度量数据之间的关联关系来达到目的, 在实训平台中, 设计并实现了估算模块, 用以支持 PROBE 估算方法。

1.2 国内外 TSP 支持工具现状

在现实项目中，工具支持对与 PSP 与 TSP 的应用有十分重要的作用。一些组织或个人都对支持工具所应该具备的能力做出过总结，也开发出一些支持工具的版本，这里将选取其中的 2 种进行简要介绍。

● TSPi 支持工具

TSPi 是 SEI 对 TSP 扩展出的学校版本，用来帮助指导毕业生和高等级的未毕业的学生应用 TSP 管理项目[5]。其支持工具涵盖了 TSP 的所有阶段和管理元素，但是它的管理形式比较单一，仅使用一系列的表格组织数据。团队成员之间不能实时共享数据，不利于信息传递。

估算模块被安排在 SUMS 工作表中，用户需要在表格中填写部件信息，用户友好程度不高。

● The Software Process Dashboard [6]

这是一款开源的 PSP/TSP 支持工具，旨在推动 PSP 与 TSP 地广泛使用。它支持个人级，团队级，组织级过程开发，同时支持 PSP 训练，并且提供更直观的图表分析结果。但是，该工具在实时共享数据方面依旧有欠缺，另外，它也缺乏角色权限管理。

估算模块被安排在 PROBE Wizard 中，用户根据提示按步完成估算。同时它提供了估算工具，脱离具体模块，用户可以自行输入估算数据，更加自由。但是界面效果依旧不理想。

除了以上介绍的这两种 TSP 支持工具外，还有其它类型的工具，此处不再详述。从上述工具描述中，可以发现一些缺陷，例如不能实时共享数据，界面不友好，缺乏角色权限管理等；实训平台将致力于消除这些缺陷，更加有效地支持 PSP/TSP 项目。

1.3 论文组织结构

本文共分为 6 章，每章内容如下：

第 1 章：绪论，介绍所属项目实训平台与估算模块的基本信息，叙述国内外其他 TSP 支持工具的现状，总结论文的结构。

第 2 章：估算模块技术与理论概述，就该模块所涉及到的关键技术做简要的介绍，包括 SSH 框架和 Ext JS 技术，同时介绍该模块的理论基础。

第 3 章：项目概述，对实训平台进行模块划分，并简要分析每个模块的功能。

第 4 章：估算模块分析与设计，对模块进行需求分析并详细设计，描述设计方案，包括架构详细设计，以及各个子模块的设计，关注特殊的设计手段。

第 5 章：估算模块实现，介绍估算模块的各子模块的具体实现，展示核心代码逻辑并用系统截图的方式介绍该模块的具体使用方式。

第 6 章：总结与展望，对本文做全面总结，并提出该模块中的不足与以后改进的方向。

第二章 估算模块技术与理论概述

2.1 SSH 框架技术

SSH 框架由 Struts、Spring 和 Hibernate 这三个框架组合而成，是 Java Web 开发中最流行的轻量级框架组合。它们之间以 Spring 为核心，Struts 和 Hibernate 都被装载到 Spring 中。利用框架各自不同的关注点，帮助实现 web 项目的分层结构。围绕 MVC 的设计模式，各个框架各司其职，共同搭建起系统的整体框架。

● Struts

本项目使用的 Struts 是 2.x 版本，它和 1.x 版本一样都是一个 MVC 框架，但是它与版本 1 之间并没有继承关系，而是起源自 WebWork 框架。这里将介绍 2.x 版本。

Strut2 中放弃使用 request, response 这些 Servlet API[7]（但是如果需要，还是可以获取到它们的对象），Action 类中包含属性的 setter 方法和 getter 方法，在该类被调用之前，会从 request 中获取参数值，填充到属性中。而后调用 Action 中的 execute 主方法，最后返回字符串作为处理结果。这里 Action 很像普通的 java 类（POJO, bean），更加便于 Spring 进行管理。

Struts2 支持 @ 注解实现零配置，零配置并不是真的没有配置文件，只是没有传统的 struts.xml 文件[7]。在 Action 中使用 @ 注解定义一些资源，例如使用 @Result 定义结果集，当存在多个结果页面时，使用结果集可以让返回信息变得一目了然。

● Hibernate

Hibernate 主要应用于数据的持久化，是一种 ORM（Object-Relative Database-Mapping，对象关系数据库映射）框架，通过配置文件 (*.hbm.xml) 或者注解的形式完成对象到数据库的映射，对 JDBC 进行上层封装，并且根据程序配置自动生成并执行 SQL 语句，省去使用 JDBC 接口、编写 SQL 语句的麻烦。同时，它也是跨数据库平台的一种框架，只需要修改配置文件就可以适用于不同的数据库，目前，它已经支持大部分常用数据库了，例如 Oracle, MySQL, MS SQL Server 等。

它使用 HQL（Hibernate Query Language）进行查询，它的语法与 SQL 类似，并在 SQL 之上添加了面向对象的思想，能够直接对实体类及其属性进行查询。

在 Hibernate 中，Session 和 Transaction 是两个非常重要的概念。Session 代

表用户的一次操作，有对应的工厂类 `SessionFactory`，用于维护不同的 `Session`；`Transaction` 代表用户的一次事务，包含对数据的各种类型操作。

● Spring

在 SSH 这个组合框架中，Spring 是构建者，也是 Struts 与 Hibernate 之间的桥梁。Spring 最初由 Rod Johnson 在 *Expert One-on-One:J2EE Design and Development* 一文中描绘，其目的是为了简化企业级的应用开发[8]。作为一个被广泛使用的容器框架，它配置 Bean 并管理其生命周期，能够通过配置整合其他框架，并且提供很多基础服务（例如事务管理）。Spring 中包含有两个非常重要的概念：AOP（面向切面编程）和 DI（依赖注入）。

AOP：Spring 为 AOP 提供了强有力的支持，从 2.0 版本开始，又新添加了建立切面的方法，增加了 `@AspectJ` 注解驱动和纯 POJO 切面[8]。

DI：另一个相关的概念是 IoC（控制反转），它是指改变主动获取信息的方式，是一种理念与原则，在这一原则的基础上，提出了 DI 这一具体的设计模式，即通过合理的方式将所需信息注入到部件中，注入方式包含 3 种，分别是接口注入、setter 注入和构造器注入，其中 setter 注入最为常见。DI 作为一种松耦合技术，成为 Spring 与另外两种框架整合时的一个重要的结合点。

AOP 与 DI 一样，都是实现解耦的重要方式，这也进一步提高了 Spring 在实际项目中的应用价值。

2.2 Ext JS 技术

Ajax 是 JavaScript、XML/JSON、CSS 的合称，是由 Jesse James Garrett 在 2005 年提出的概念，发展至今，它的应用已日益增多。通过 Ajax 的使用，改进“纯网页”模式，为用户带去更好的操作体验。为了方便开发人员轻松创建基于 Ajax 的应用，一系列的 Ajax 工具包被开发出来，它们提供了多种基于 JavaScript 的 UI 控件，同时通过高级抽象解决 Ajax 的浏览器不兼容问题[9]。

Ext JS 就是一种 Ajax 工具包，也是一个整合的程序界面开发平台[10]。它是由 Jack Slocum 于 2006 年开始酝酿，在 YUI（Yahoo! User Interface）Library 基础之上开发的一个 JavaScript 类库[9]。在更多的力量被注入到 Ext JS 的开发与改进中后，它的功能日益完善，成为众多企业级 Ajax 框架中非常流行的一种。

现在的 Ext 已经脱离了对 YUI Library 的依赖，通过使用“适配器”模式兼容其他 JavaScript 类库，例如 Prototype.js、jQuery，实现从这些类库到自有的底

层库的转换，转换图如图 2.1 所示。

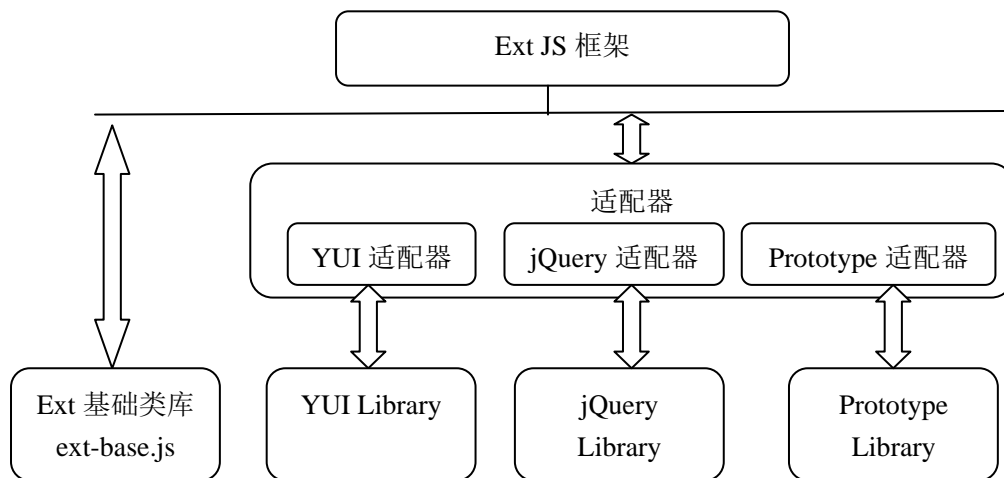


图 2.1 Ext 与底层库的转换[10]

在本项目中，使用的是 Ext 基础类库。

在使用 Ext 进行界面开发时，需要使用到几个核心组件。

(1) **Panel**：对应的类是 `Ext.Panel`，继承自 `Ext.Container`，将 UI 划分成了几个部分，支持添加子布局，是一种非常常用的组件。面板的一种特殊继承就是窗口 `Window`，在该项目中弹出窗口就是应用了窗口组件。

(2) **Ext 布局**：负责管理组件在容器中的组合方式，位置，大小尺寸等信息。`Ext` 为开发人员提供了丰富的布局方式，这个模块中会使用到垂直式列表布局（`ColumnLayout`）、表单专用布局（`FormLayout`）与填充式布局（`FitLayout`），通过在嵌套的容器中使用不同的布局，实现布局的组合使用，可以构建出更加复杂的界面。

(3) **Form 组件**：其下又包含多个组件，例如 `FormPanel` 组件、`ComboBox` 组件、`FieldSet` 组件等。

`FormPanel` 组件继承自 `Panel` 组件，通过参数的设置设定内部组件的显示方式，本模块中自定义的 `XPanel` 大部分都是继承自 `FormPanel` 组件；

`ComboBox` 组件非常灵活，可以在 js 文件中定义数组作为数据源，也可以使用远程数据作为数据源。

这里又要涉及到另外一个概念，即数据存储（`Store`），它是 `Ext` 用来进行数据交换的中间组件[11]，在 `ComboBox` 里，以及在之后描述的 `Grid` 组件中都使用 `Store` 用来进行数据交换，搜索，转换等操作。在定义 `Store` 时，还需要设定 `proxy`

以及 reader，分别用于指定提取数据的数据源以及数据读取器。

FieldSet 组件继承自 Panel，可以容纳其他的组件共同组成一个整体。通过将 collapsible 属性设置为 true，自动渲染一个展开/闭合轮换按钮在其头部，使得面板可以收缩。

(4) Grid 组件：表格组件，在需要显示大量数据的时候，它能够帮助清晰的，有条理的展示出来。Ext 除了为用户省去了设计表格界面效果的工作，还增加了更多的表格操作，例如排序，隐藏一行，单元格编辑等。Grid 最大的特点还是实现了数据模型与显示的分离，数据由相应的 Store 完成，Grid 只需要负责数据的界面显示与对表格操作的响应，这一设计在本项目中被广泛地使用。

组件也有生命周期，一个标准的组件在被运行时包含了以下 3 个过程[10]，这对于理解事件处理顺序有重要的意义。

- 初始化/创建：通过执行构造器组，获取配置项；登记事件（常用的有 show, beforeRender, render, beforeDestroy, destroy 事件等）；如果有设置，还需要初始化状态感知与插件；最后还有插件的渲染。
- 渲染过程（rendering）：这个阶段中按照组件的定义渲染对应的 UI，是最关键的步骤。具体流程涉及到事件 beforeRender 和事件 render 的触发，在后者触发之前，自定义的类与样式会生效。
- 销毁过程：涉及到事件 beforeDestroy 和 destroy 的触发，以及方法 beforeDestroy 和 onDestroy 的调用。

Ext 中异步请求无处不在，这个项目中主要使用了 Ext.Ajax.request 方法实现。该方法包含几个常用的组成部分，其中 url 表示请求的路径（可以具体到 Action 中的某个方法）；success 部分定义服务器返回成功的处理结果以后的动作；failure 部分定义服务器返回失败的处理结果以后的动作（可以添加一些错误处理）；params 部分描述请求所需要的参数，参数之间用逗号分隔；scope 表示回调函数的作用域[12]。

2.3 理论基础

PSP 中定义了 PROBE 估算流程的四个步骤[3]，具体内容见图 2.2，估算模块支持这个流程的实现。

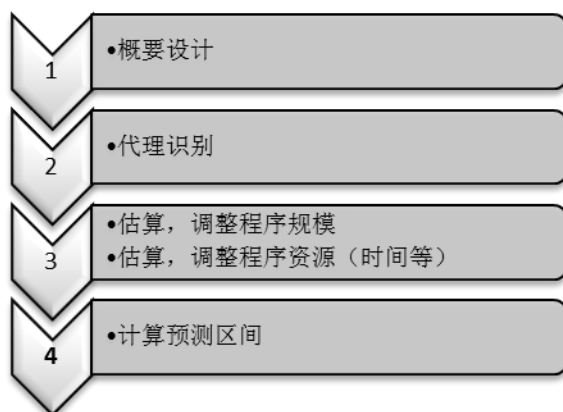


图 2.2 PROBE 估算流程[3]

- 概要设计----是由用户自己完成，也是后续估算的基础。用户将工作进行划分，直到一定粒度时，用户可以很方便的估算或者和已有的部件相关联。项目中，允许用户以部件为单位进行估算。
- 代理识别----寻找一个合适的代理便于估算准确。例如，在面向对象中，往往选择类作为代理。方法，例程，数据库表格等也都可以当作代理。选定以后，对概要设计后的部件集合按照代理进行规模估算，迭代相加以后，就获得了代理规模。项目中，会帮助计算代理规模，进行后续估算。
- 估算与调整----估算代理包含了太多的主观因素，因此在所难免的会有偏差。调整就是修正这种偏差。PSP 中采用线性回归的方法来修正估算。修正规模的公式为：

$$\text{Plan Size} = \beta_{0\text{size}} + \beta_{1\text{size}} * (E)$$

其中 $\beta_{0\text{size}}$ 和 $\beta_{1\text{size}}$ 是根据历史数据中的代理规模估算值和程序实际规模计算出的系数。以 $n \geq 3$ 组历史数据为例，

$$\beta_{1\text{size}} = \frac{(\sum_{i=1}^n x_i y_i) - (n x_{\text{avg}} y_{\text{avg}})}{(\sum_{i=1}^n x_i^2) - (n x_{\text{avg}}^2)}$$

式 1

$$\beta_{0\text{size}} = y_{\text{avg}} - \beta_{1\text{size}} x_{\text{avg}}$$

式 2

其中 x_{avg} 和 y_{avg} 表示平均值。

同理，对于资源计算公式为：

$$\text{Plan Time} = \beta_{0\text{time}} + \beta_{1\text{time}} * (E)$$

计算 $\beta_{1\text{time}}$ 和 $\beta_{0\text{time}}$ 的历史数据就是代理规模的估算值和程序的实际资源值，计算公式也与式 1,2 逻辑相同。

以上计算都由项目的业务逻辑与工具类共同完成。

- 计算预测区间----对调整后的估算结果进行再处理，计算置信区间，从而得出估算范围。

$$\text{浮动范围 Range} = t(p, df) \sigma \sqrt{1 + \frac{1}{n} + \frac{(x_k - x_{avg})^2}{\sum_{i=1}^n (x_i - x_{avg})^2}}$$

其中， n 表示数据点数目， σ 是标准差， $t(p, df)$ 表示自由度为 df （取 $n-2$ ），概率为 p （一般取 70%）的 t 分布。

$$\sigma = \sqrt{\frac{1}{n-2} \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2}$$

估算上限 $UPI = \text{Plan Size}(\text{Time}) + \text{Range}$, 下限 $LPI = \text{Plan Size}(\text{Time}) - \text{Range}$ 。

以上计算由项目完成，而后反馈用户结果。

● 相对大小矩阵

估算代理规模时，方法是一个比较常用的代理（也是计算 LOC 时默认的代理），Watts S. Humphrey 根据他的实验数据整理出相对大小矩阵，分别表示不同种类的方法的多种大小（VS-非常小，S-小，M-中等，L-大，VL-非常大）。默认情况下，项目使用表 2.1 对应的相对大小矩阵：

表 2.1 相对大小矩阵[3]

类型	VS	S	M	L	VL
Calculation	2.34	5.13	11.25	24.66	54.04
Data	2.60	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.51	21.62	28.93
Logic	7.55	10.98	15.98	23.25	33.83
Set-up	3.88	5.04	6.56	8.53	11.09
Text	3.75	8.00	17.07	36.41	77.66

当然，项目也应该允许用户使用其他类型，并且可以根据自己的情况调整这个矩阵。

● PROBE 方法选择

PSP 提供了 4 种 PROBE 方法(A,B,C 和 D)。选择方法时，理论要求使用统计学概念，即相关性 r 与显著性 s 。A 方法和 B 方法都需要 r 和 s 达到一定要求，而 C 和 D 方法不需要。同时，对历史数据数量也有要求。

根据历史数据的质量与数量，应该有选择的使用 PROBE 方法。

项目中，需要提供历史数据以及其质量分析，供用户进行方法选择。同时，对已选择的方法进行有效性验证，保证其正确性。

注：以上一节都是基于参考文献[3]。

第三章 项目概述

3.1 项目总体划分图

将实训平台按照功能划分，各个模块之间共同协作，共同支持团队活动管理，具体划分如图 3.1 所示。

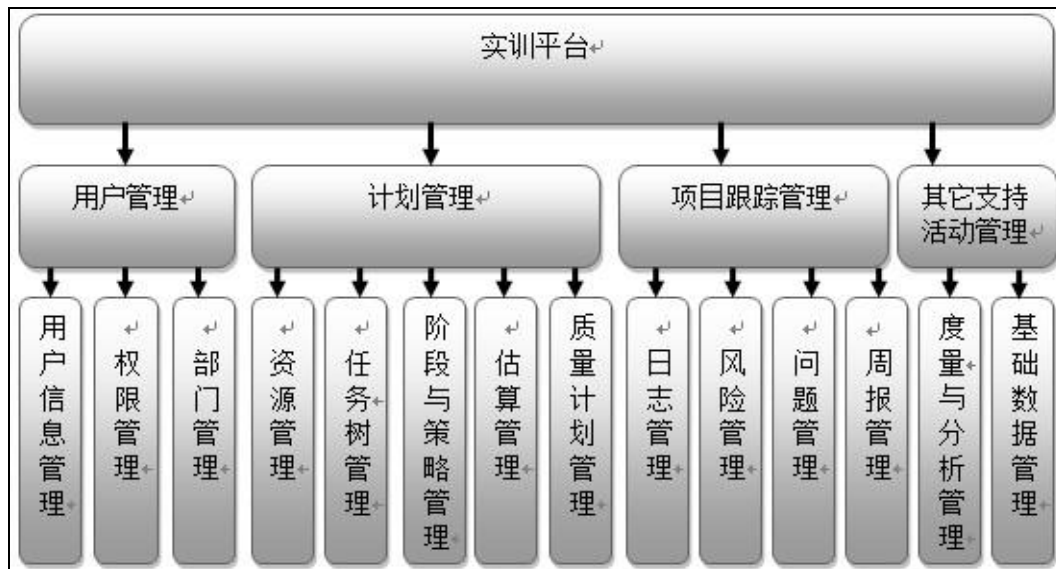


图 3.1 项目总体划分图

3.2 项目模块描述

● 用户管理模块

- (1) 用户信息管理：主要用于管理用户身份信息，包括用户名，工号，姓名，角色等基础信息，登记的用户可以被指定为负责人，共同参与项目管理。
- (2) 权限管理：用户会被指定角色，不同的角色对系统有不同的管理权限，以帮助他们关注于自己的职责，同时也可以保证信息安全。
- (3) 部门管理：部门是用户的身份信息之一，系统将其抽离出来，单独由部门管理模块管理。

● 计划管理

- (1) 资源管理：根据计划的任务列表与项目成员的时间资源列表，系统自动安排日程，并最终生成日程计划。

- (2) 任务树管理：以树的形式展示项目的开发与管理阶段，每个节点代表一项任务，任务之间可以是顺序关系也可以是父子关系。在节点，即任务的基础之上展开详细计划的制定、日志追踪、估算等活动。
- (3) 阶段与策略管理：系统预定义一些常用的阶段，例如计划阶段、需求阶段等。每个阶段会有阶段属性，用于限制该阶段最终成果的形式、记录内容、动作等。策略描述了阶段进行的方式方法，预定义了 PSP0/1/2 这三种策略，用户可以自行修改策略内容或添加其他策略。
- (4) 估算管理：使用 PROBE 估算方法，产生规模与时间的估算结果，作为资源安排的一项重要依据。本文将详细介绍这个模块。
- (5) 质量计划管理：质量计划是计划中的一项重要内容，帮助用户制定质量目标，用于后续跟踪管理。

● 项目跟踪管理

- (1) 日志管理：日志包含时间日志与障碍日志，分别用于记录任务的进行时间与开发中的缺陷。
- (2) 风险管理：帮助记录风险信息，用于识别潜在问题，帮助规避消极影响。信息包括严重性，可能性，项目等内容，是一种可以重用的历史数据。
- (3) 问题管理：用于记录项目活动所遇到的问题，包括严重性，类型，标题等信息，与项目相关，不作为公共历史数据。
- (4) 周报管理：用于跟踪项目进度，以周为单位，记录计划与实际数据，通过数据对比，观察实际进展与计划是否一致，如果有偏离，参照数据采取相应措施进行纠偏。

● 其它支持活动管理

- (1) 度量与分析管理：使用系统中收集到的过程数据进行分析，包括 PROBE 方法中的参数计算，以及 yield, PQI, COQ 等质量指标,帮助量化过程。
- (2) 基础数据管理：对系统中需要使用到的含多选项的内容进行管理，帮助用户自定义选项信息。

第四章 估算模块分析与设计

4.1 需求分析

上一节描述了模块的理论基础，下面介绍模块的 4 项具体子功能，如图 4.1 所示：

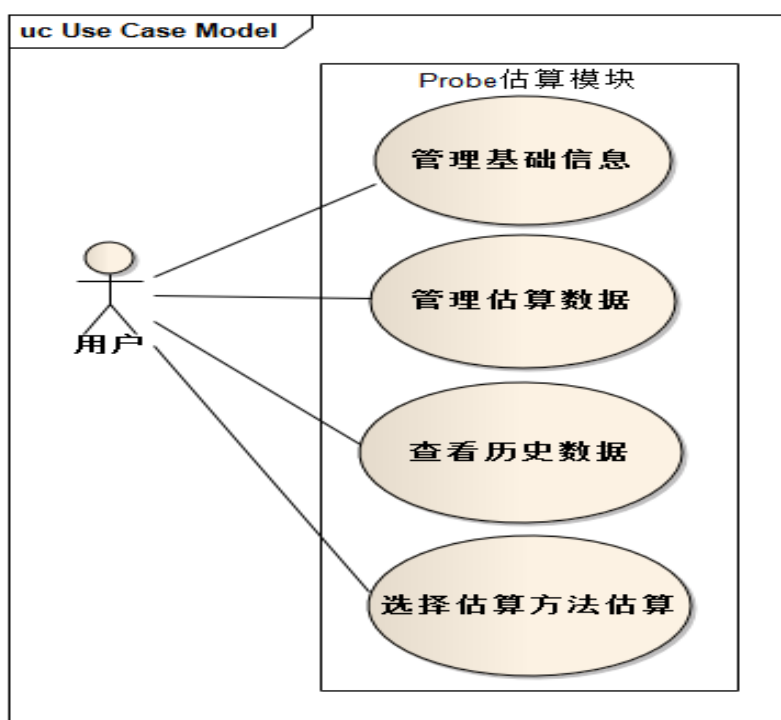


图 4.1 用例图

- 管理基础信息----基础信息包括计算单位（最常见的就是代码行和页数），编写语言，以及描述信息，提交以后就不能修改；
- 管理和查看估算数据----按照代理，填写条目的基本信息，包括条目名称，数量，类型，相对大小，计划规模和实际规模；同时，用户也可以根据自己的情况修改相对大小矩阵。
- 查看历史数据----历史数据在估算中发挥着至关重要的作用，它们是线性回归参数的来源。由于在 A,B 和 C 方法都需要使用到历史数据，所以可以按照方法的不同组织历史数据的显示。
- 选择估算方法估算----为了方便用户选择合适的方法，在按照方法组织历史数据的同时，也应该提供方法对应的参数值，包括规模估算和资源估算所需要的 r^2, β_0, β_1 的值。同时，可以让用户选择规模和资源估算的方法；为了辅助资源估算，当用户选择 D 方法时，也需要用户提供生产率的值，最后可以反馈给用户估算的结果。

4.2 分层架构

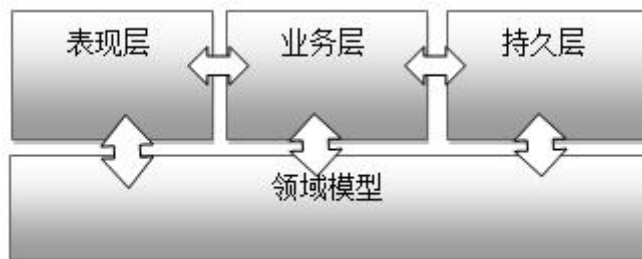


图 4.2 架构设计图[13]

模块被拆分成多个层次，概念图如图 4.2 所示，细节图如图 4.3 所示，每个层次都有其对应的任务。

- 表现层与页面相关，包括了页面请求与响应；
- 业务层则主要用于处理业务逻辑，管理事务，作为表现层与持久层的中间媒介，通过依赖注入的方式实现与表现层和持久层的通信，防止它们因互相通信而造成依赖；
- 持久层将关注点限定在了数据库中的信息，包括生成信息相关对象，以及对信息的操作；
- 领域模型层与上述 3 层都会存在交互，该层包含一系列的 java 类，这些类在与其它层交互时，会生成一系列的数据对象，同时也包含了一些工具类，被服务类或 action 调用。

它们以松耦合的方式组合在一起，降低了代码的复杂程度，便于后期的维护。

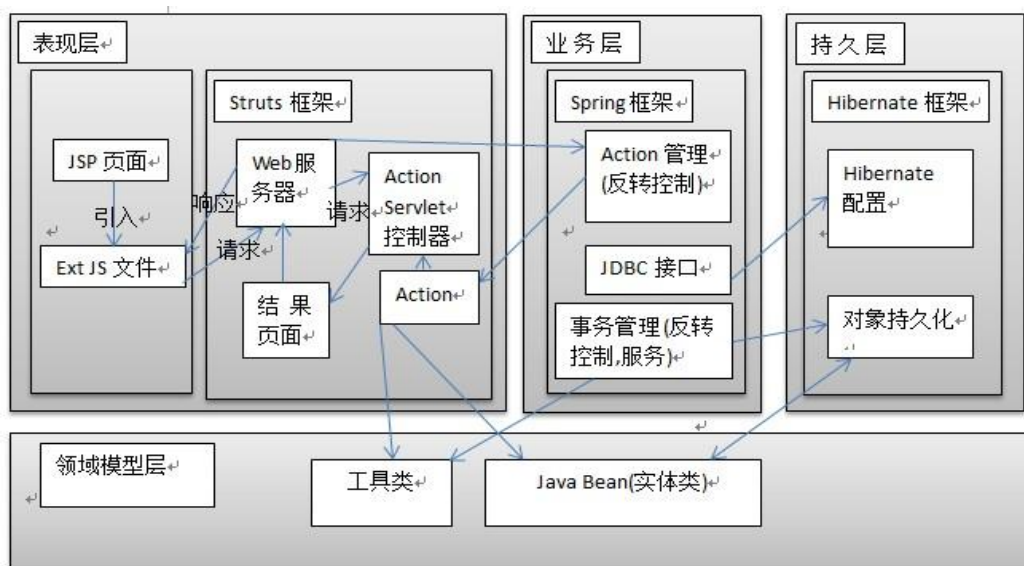


图 4.3 架构细节图

4.2.1 表现层

在该层中，使用的页面技术包括 jsp, Ext JS, css, 应用的框架是 struts2。

使用 struts2 集成 Spring 使用，使得在 struts2 在创建 Action 对象时，会尝试从 Spring 容器中查找，因此 Action 只需要在 Spring 相关的配置文件中以 Bean 的方式配置，并通过依赖注入的形式注入其它依赖的 Bean(Service)，这里将配置文件命名为 ApplicationContext-struts.xml，具体流程如图 4.4 所示：

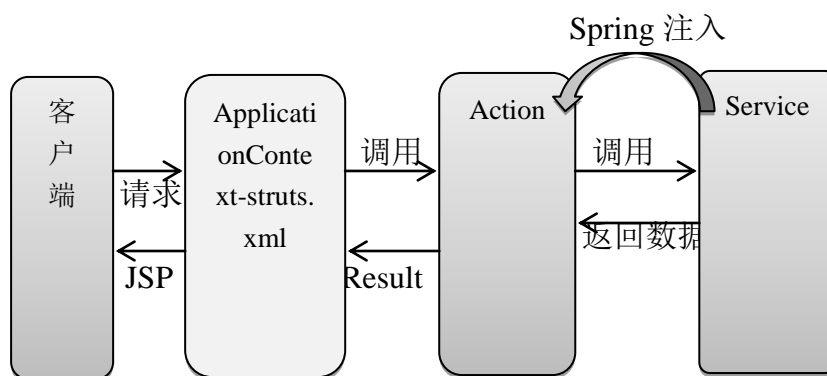


图 4.4 表现层流程

再深入客户端界面部分进行分析。

● 界面总体设计

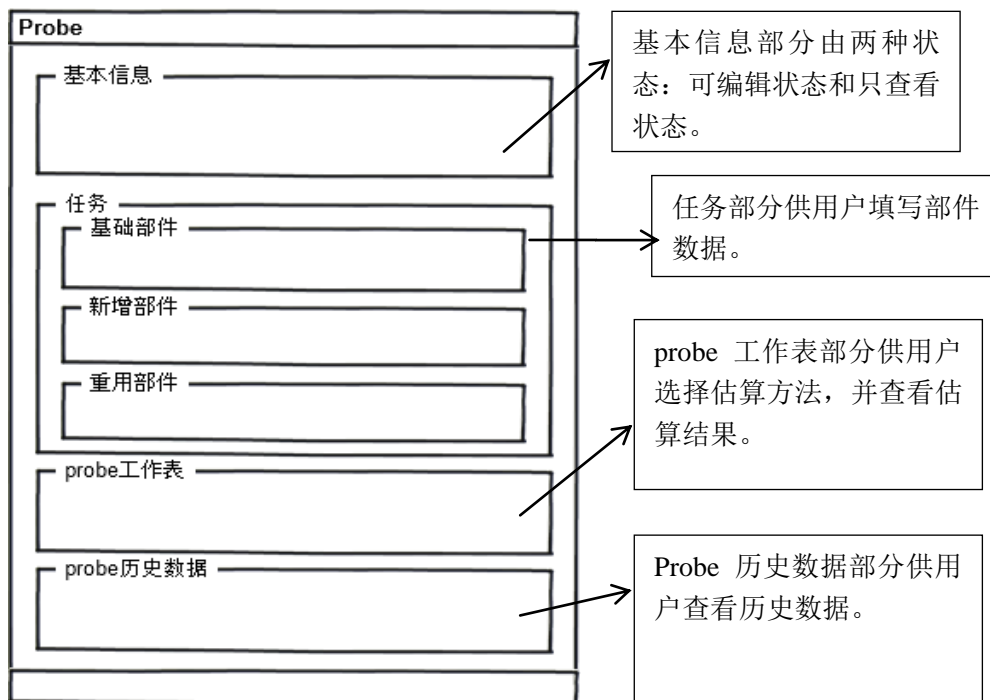


图 4.5 界面模拟图

由图 4.5 可以看出，界面具体分为 4 个块，而由于基础信息部分有两种状态，因此可以由 5 个块（页面中对应了 5 个 div）组成。关于每个块的细节模块将在 js 文件中给出，而页面风格将在 css 文件中定义。

这样做的结果就是：页面将变得非常简洁，因为它基本不包含有 js 代码，也不用考虑页面风格，易于维护。

4.2.2 业务层

该层主要应用了 Spring 框架来隔离表现层与持久层，拒绝因它们直接通信而造成的依赖关系。Spring 容器对 Action, Service, Dao 都使用 bean 的管理方式，使得业务逻辑变得很清晰，也使配置文件成为了关键的一环。逻辑如图 4.6 所示：

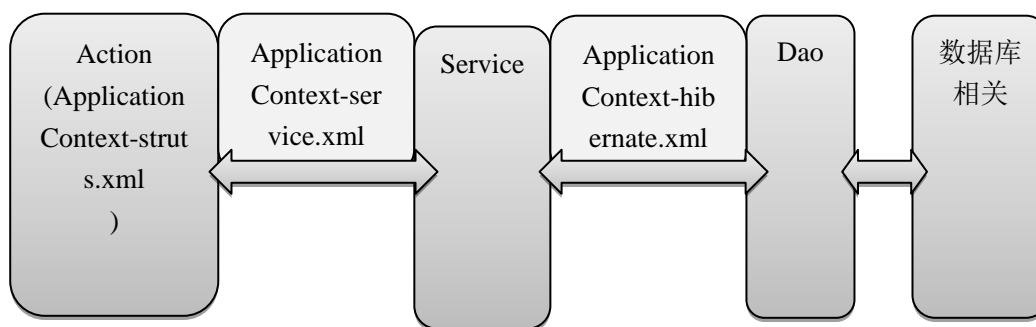


图 4.6 业务层逻辑

ApplicationContext-service.xml 中定义 Service bean，Dao 作为其 property 存在；ApplicationContext-hibernate.xml 中定义 Dao 的 bean，以及其他 hibernate 配置；通过在 Service 中使用 setter 的注入方式，注入关联 Dao，实现各种 Dao 的组装使用，形成 Dao 的上一层封装，从而达到上下两层分隔的效果。

4.2.3 持久层

该层主要使用了 Hibernate 框架，它是这四个层次中唯一一个与数据库系统直接发生联系的，是对数据库操作进行的上层封装。这里使用到的 hibernate 版本是 3.x。

Hibernate 连接数据库的配置信息也交由 Spring 管理，相关的配置信息所

文件命名为 Application-hibernate.xml;

需要编写的是实体类 X 对应的映射文件 X.hbm.xml，并且在 Application-hibernate.xml 中添加对此映射文件的配置。

由于数据库的连接信息都被写入了配置文件，修改相对简单，故数据库系统的选择就变得很灵活，系统在这一方面具有了很好的可移植性。

4.2.4 领域模型

领域模型主要包含一些实体类的 bean（例如），以及工具类（数学方法实现）。领域模型和上三层都有直接关联，但是这些联系都不复杂；关联如图 4.7 所示：

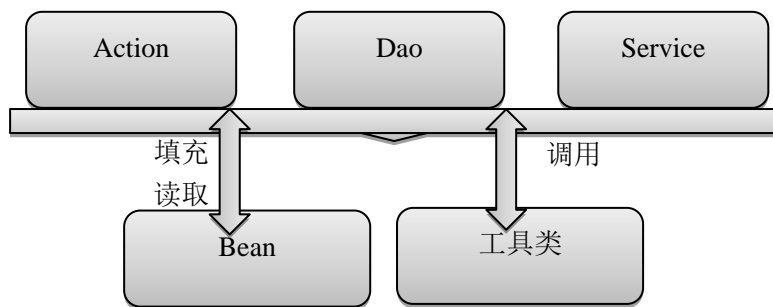


图 4.7 领域模型与其他层关系图

同时，这些实体类（Bean）都实现 java.io.Serializable 接口，使其可以被序列化与反序列化。

由于数据都被封装在实体类内部，所以当这些实体对象要修改属性的时候，只需要在 bean 中改变，无关其他业务逻辑，利于维护。

4.3 详细设计

4.3.1 管理基础信息子模块

- 界面设计要点 1-1：由于在确认提交以后就不能修改，所以可以分为编辑基础信息和只查看基础信息两部分。体现到 jsp 页面中分别由

grid-probe-categoryInfo 层和 grid-probe-basicInfo 层表示。其中 grid-probe-categoryInfo 层最主要的动作是提交数据，而 grid-probe-basicInfo 层的主要动作则是加载数据。

- 界面设计要点 1-2: 使用 Probe.XPanel 定义界面元素，使用 Probe.XStore 从下层获取数据。

如图 4.8 所示,在提交数据的时候,需要先从 taskManager, languageManager, sizeUnitManager, tagManager 中根据 id 找出对应的 Task, Language, SizeUnit, Tag 对象（可以为空），如果发生错误，则返回一个失败响应；如果成功，那么向当前 Task 对象里填充其余 3 个对象的信息，同时创建当前 task 的 TaskProbe 对象（probe 估算要用），最后将填充好的 Task 对象和 TaskProbe 对象通过对应的 service 保存到底层去。这里的 Task, Language, SizeUnit, Tag, TaskProbe 等都是领域模型层的 bean。

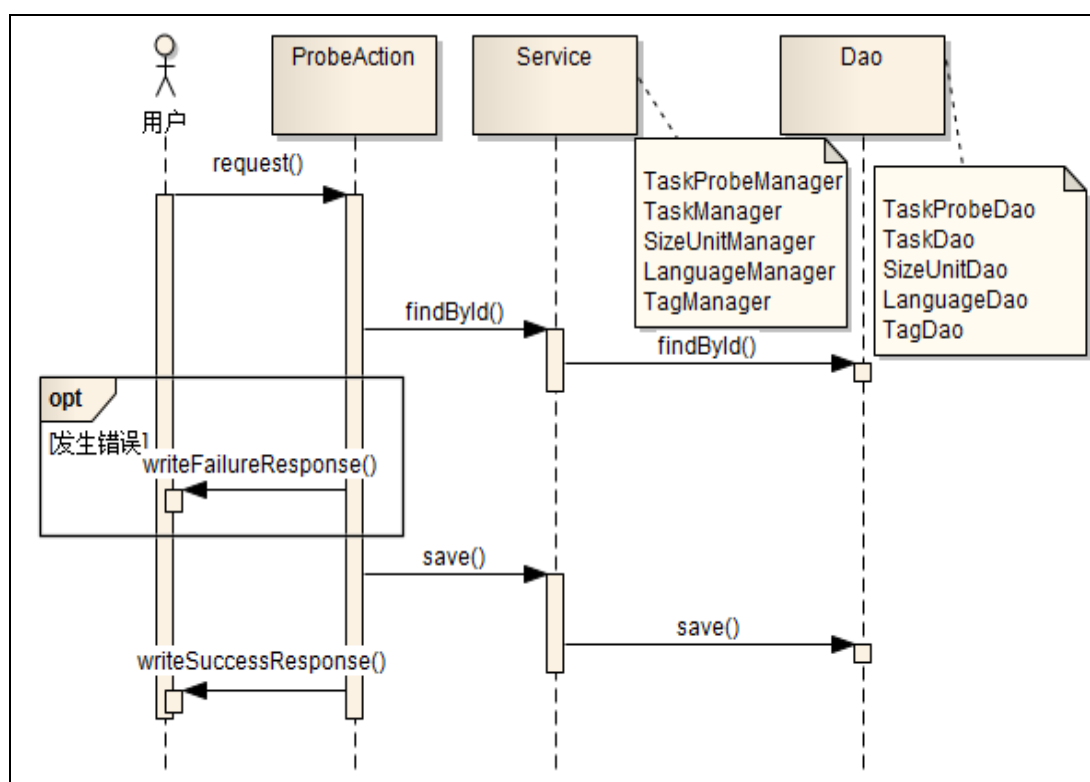


图 4.8 提交数据流程

如图 4.9，在加载基础信息数据的时候，先要通过 service 找到当前的 task，

如果 task 为空，那么返回失败响应；如果不为空，那么找到对应的数据集合，并返回正确响应。

- 设计要点 1-1：模块需要返回 Task 对象的基础信息，包含 language, sizeUnit 和 tag，由于包含多个对象，因此使用一个集合类 Category 对其进行封装，实现信息隐藏。

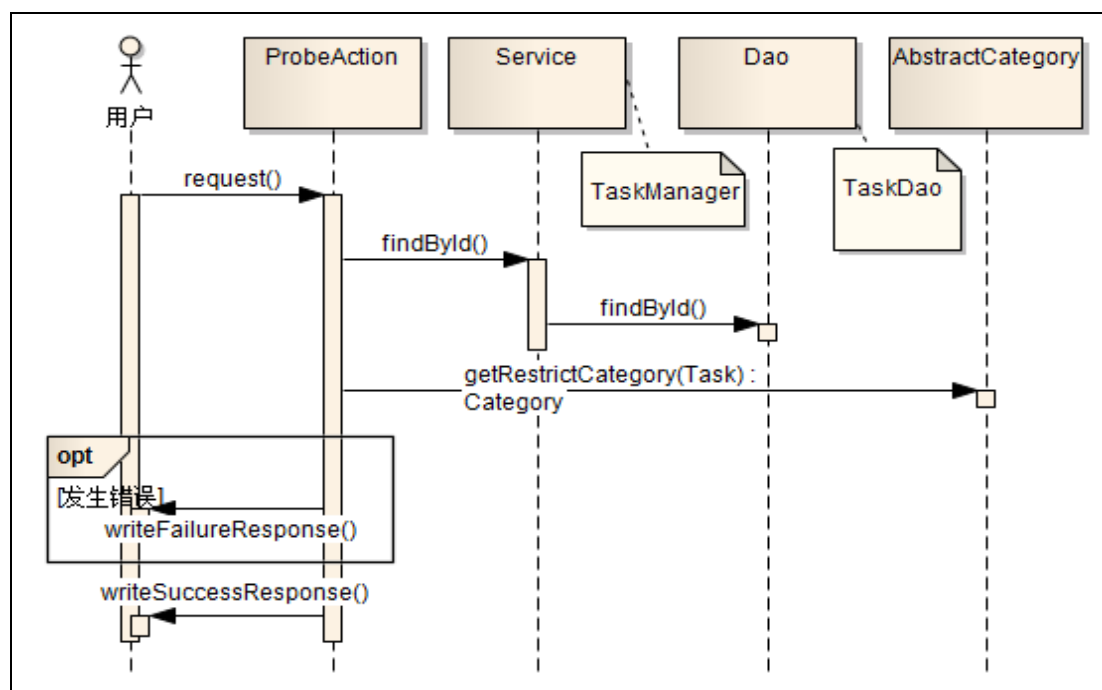


图 4.9 加载数据流程

图 4.11 展示的是 Dao 类之间的关系；

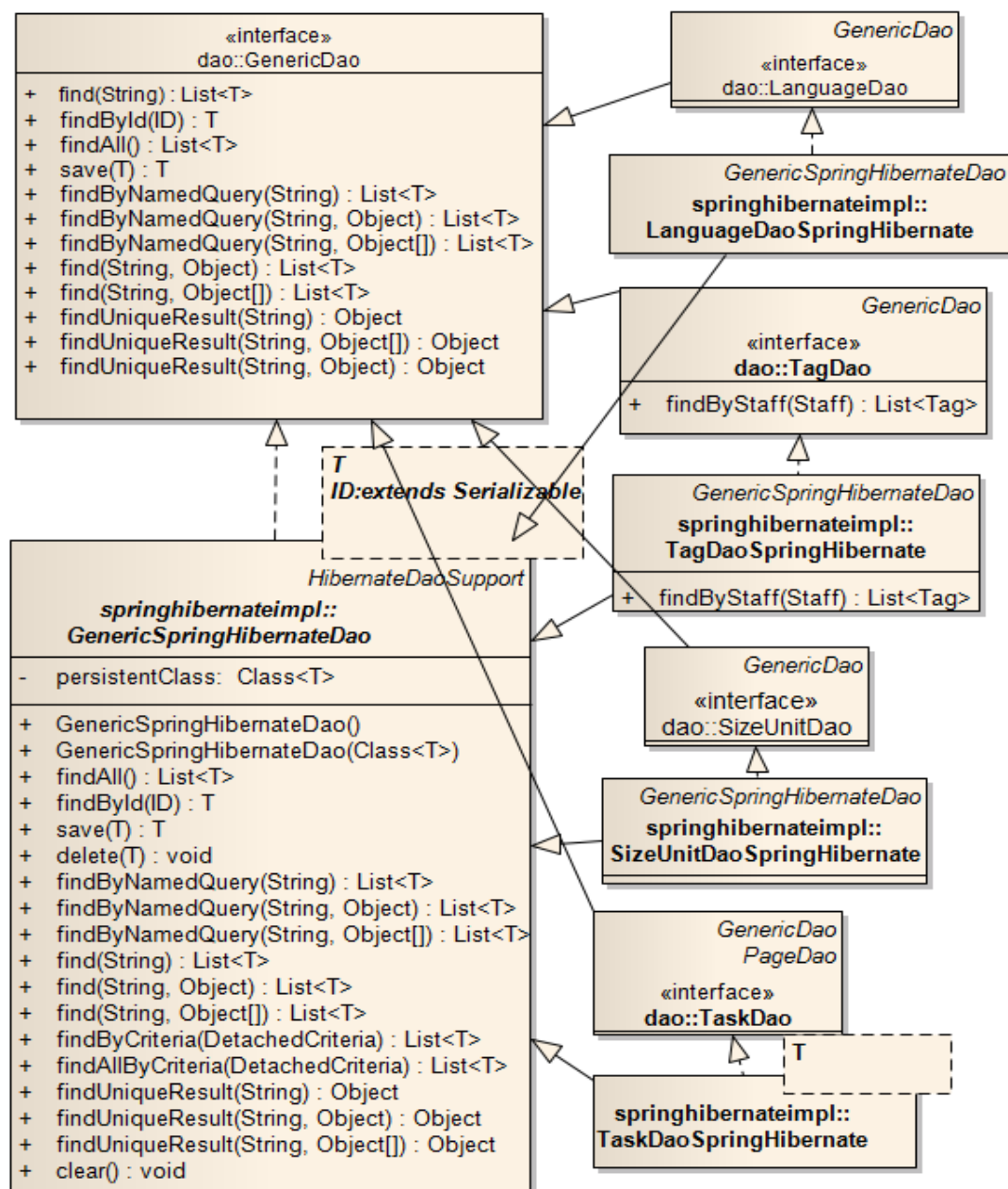


图 4.11 Dao 类图

- 设计要点 1-2: 模块分别将服务类和 Dao 中的共同部分（比如 save 方法）抽取出来，形成 BaseManager 接口和 BaseManagerImpl 类、GenericDao 接口和 GenericSpringHibernateDao 类，为了屏蔽参数类型差异，使用了模板,模板类中抽取出了共同的实现部分，并且在定义具体服务类和 Dao 时才指定 T 的具体类型，子类中则给出特殊方法的具体实现，这种设计的优点在于[14]：提高了模板类的可复用程度；共同实现只存在于模板类中，方便维护；这种实

现方式提供了一个框架，其他类型的 Dao 与 Manager 能够很方便地插入到其中，新的 Dao 和 Manager 只需要实现其特殊方法。

- 设计要点 1-3: 由于有 spring 的强大支持，service 类与 Dao 之间的耦合度变得非常小。除了在定义 service 类时指定了 TDao 的类型是对应 Dao 的父类以外，就只需要在配置文件（applicationContext-service.xml）中加以说明，由 Spring 完成依赖注入。

4.3.2 管理估算数据子模块

- 界面设计要点 2-1: 这部分界面也可以分成 3 个部分: 分别体现基础部分，增加部分以及复用部分。在每个部分的定义中，模块定义不同种类的对象实现不同功能。Probe.XPanel 用来定义布局，设置局部风格；Probe.XGrid 用来展示表格式数据排布，而且是只负责数据显示和操作控制；Probe.XStore 负责数据处理；[9]3 个部分分别有各自的 XGrid 以及 XStore，共同组成 PartsPanel。这样的使用方式使得显示模型和数据模型分离开来，使得代码简洁易懂，易于复用与维护。在之后的子功能的界面设计中也遵循这一设计原则，不再赘述。

这 3 个部分中增加部分最复杂，用户可以使用已定义部件类别（根据相对大小矩阵自动确定部件大小），也可以使用除定义之外的其他类别（自己确定部件大小），也可以改变相对大小矩阵。当用户完成数据编辑后，可以提交确认。

如图 4.12 所示，在加载相对大小矩阵数据的时候，先从 TaskManager 中找到当前的 task，从 task 中获取其 Category 和 Staff 对象，以此为参数，从 PartTypeStandardManager 中找到 PartTypeStandard 集合，如果没有，那么加载默认数据，并且返回这个集合。

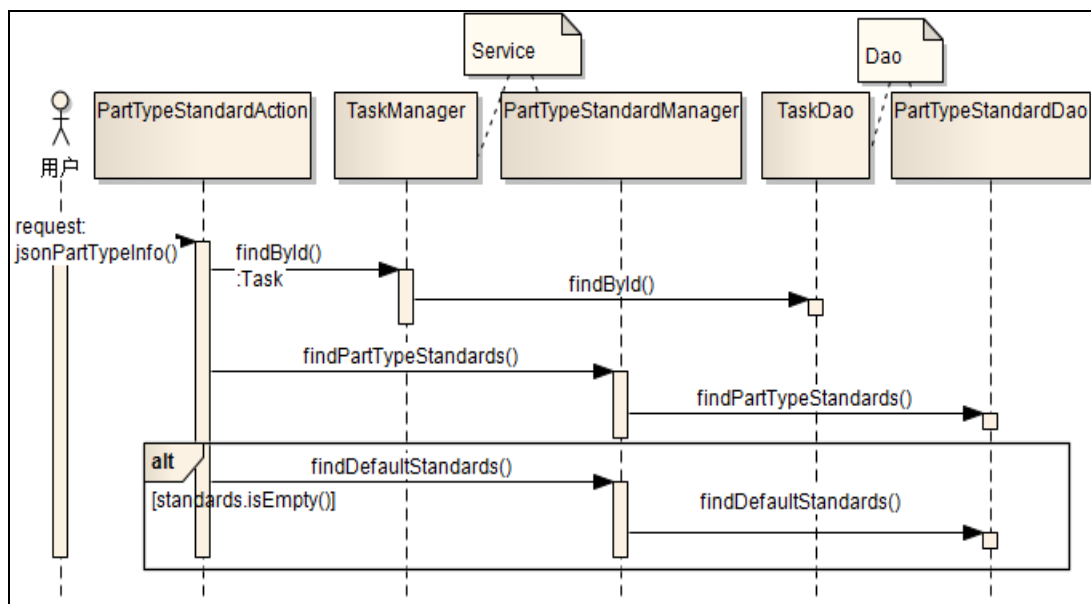


图 4.12 加载相对大小矩阵流程

如图 4.13 所示，在确认相对大小矩阵时，先要找到当前的 task，并且确认 task、以及 task 的 category 集合、staff 属性是否存在。确认之后针对每一条记录，通过服务类 partTypeStandardManager 查找记录，如果不存在，还需要创建一条新记录，最后调用服务类的保存方法保存到底层。

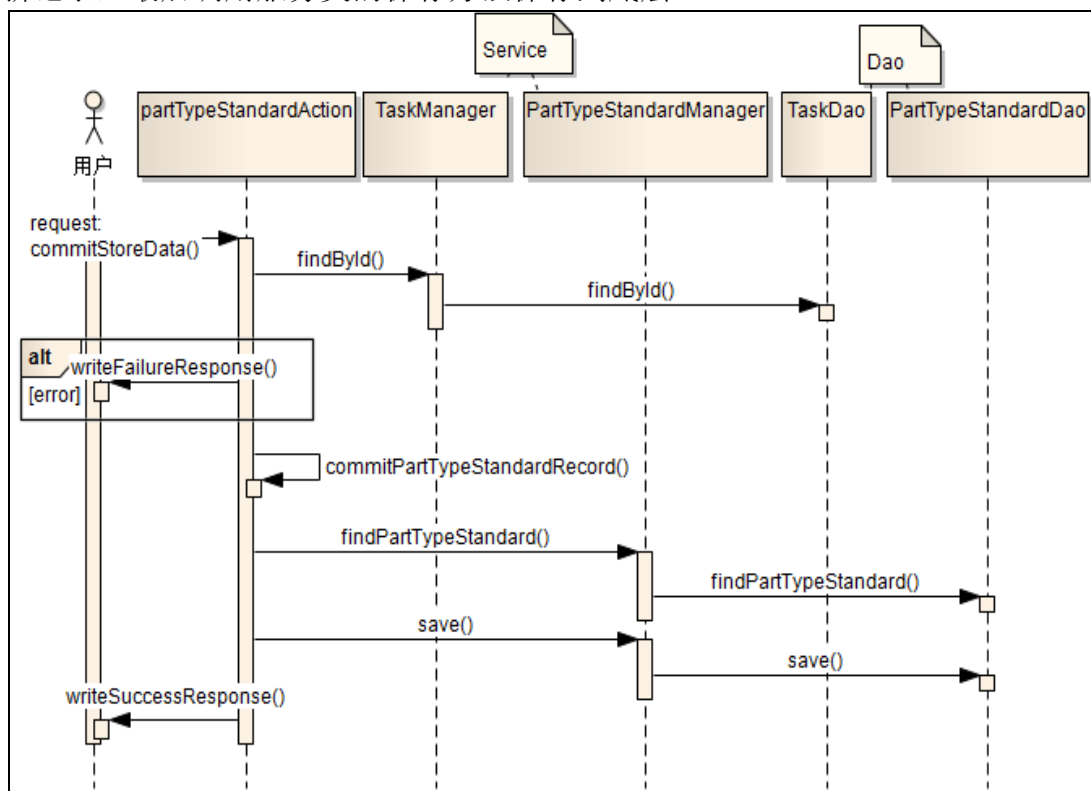


图 4.13 提交相对大小矩阵流程

如图 4.14 所示，在用户确认提交估算数据时，先找到当前 task，再获取 task 的 TaskProbe 对象，如果该对象不存在，还需要创建一个对象并且保存，用来保存整理后的各部件数据。再调用针对 base，add 和 reuse 部件的服务类，保存部件内容，最后移除被用户删除的条目即可（在 js 文件中定义属性名称‘deleted’，在 Action 中根据属性提取参数中的数据）。

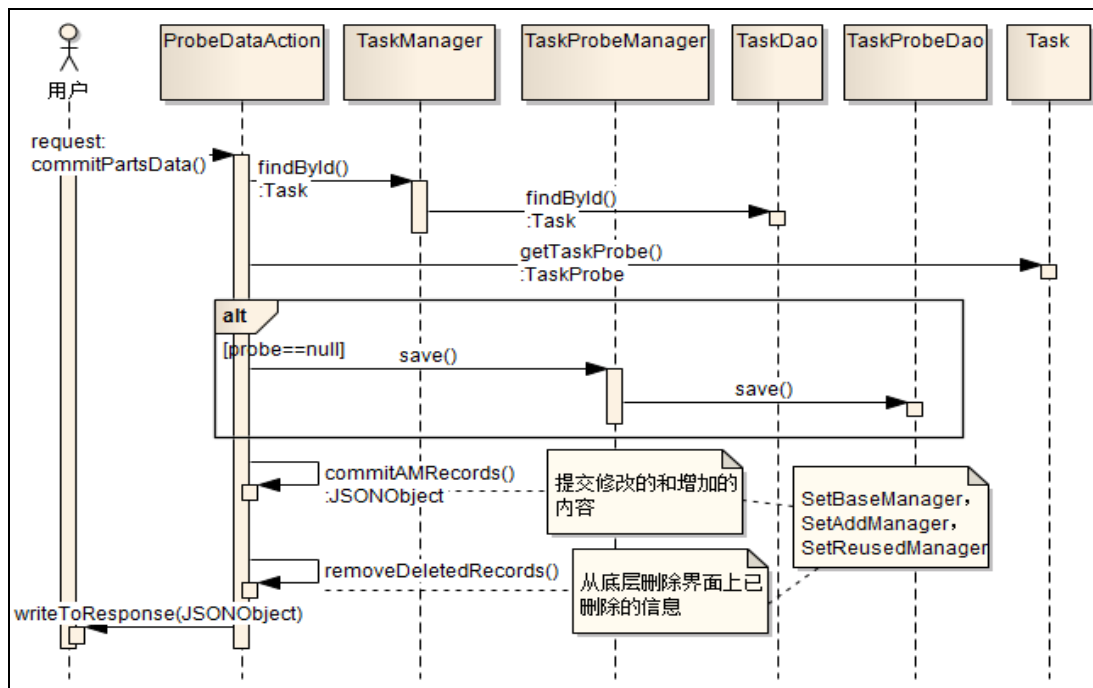


图 4.14 提交确认估算数据流程

- 设计要点 2-1: 这里都围绕着相对大小矩阵展开，该模块为这个矩阵的每一行记录设定了一个数据结构，命名为 **PartTypeStandard**。这个类包含的属性如图 4.15 所示，包含 task 的基础信息（用于区分不同种类的 task），度量对象类型（即 name，例如方法类型-Logic 等），度量对象大小（很大 vl，大 l，中等 m，小 s，很小 vs）。

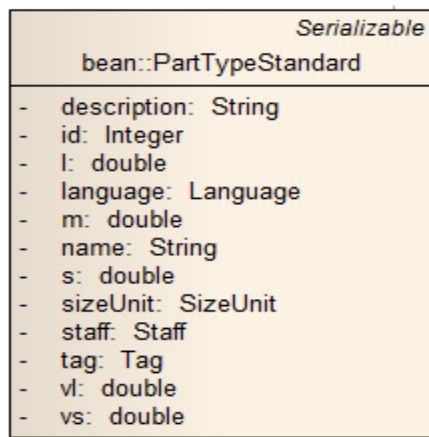


图 4.15 PartTypeStandard 属性图

4.3.3 管理历史数据子模块

由于在 PROBE A,B 和 C 方法中需要使用到历史数据，而所以该模块将历史数据按照方法的不同需求组织在一起，并且显示每种方法对应的参数计算结果。

- 界面设计要点 3-1：由于估算数据部分占用空间太大，而且历史数据并不总是需要查看，所以该部分数据就采用下拉/隐藏的方式显示，节省页面空间，具体使用 FieldSet 组件。

加载历史数据的具体业务流程如图 4.16。

首先，找到当前 task，并且确认 staff，category，并根据这些限制通过服务 probeHisDataManager 找到历史数据的结果；

- 设计要点 3-1：由于历史数据主要包含一个 list 以及长度等属性，因此模块抽取出来一个可以复用的数据结构，命名为 PaginationSupport，作为返回类型，类图见图 4.17；

然后，再从这个结构中抽取出需要的数据（通过 getProbeHisDataRecords 方法），即对每个 task 分解到功能模块级，抽取出功能模块的估算信息并返回一个结果集。

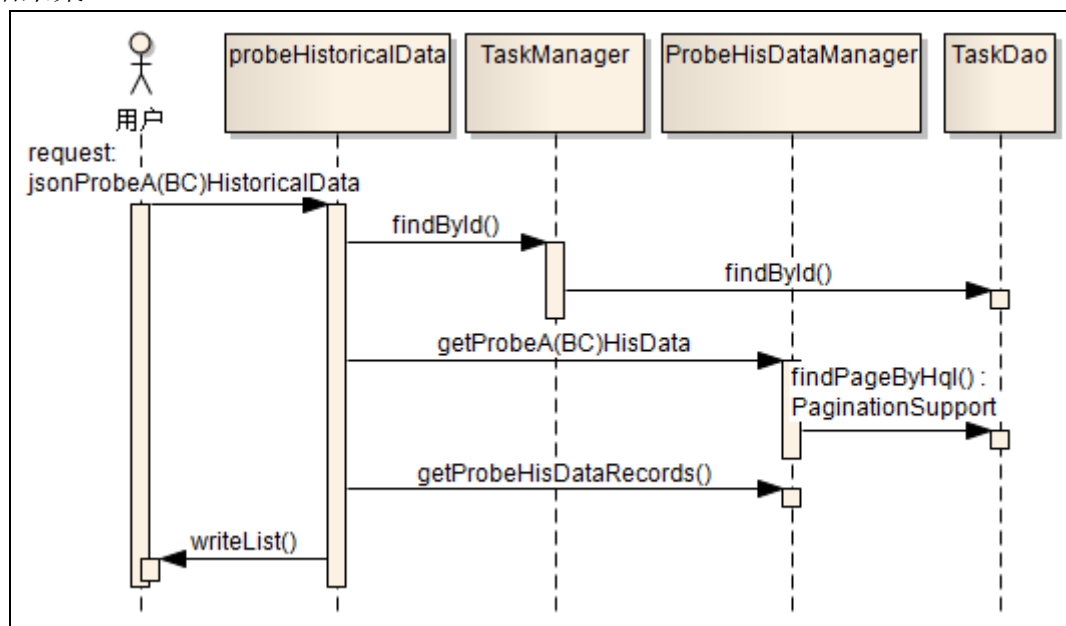


图 4.16 显示历史数据流程

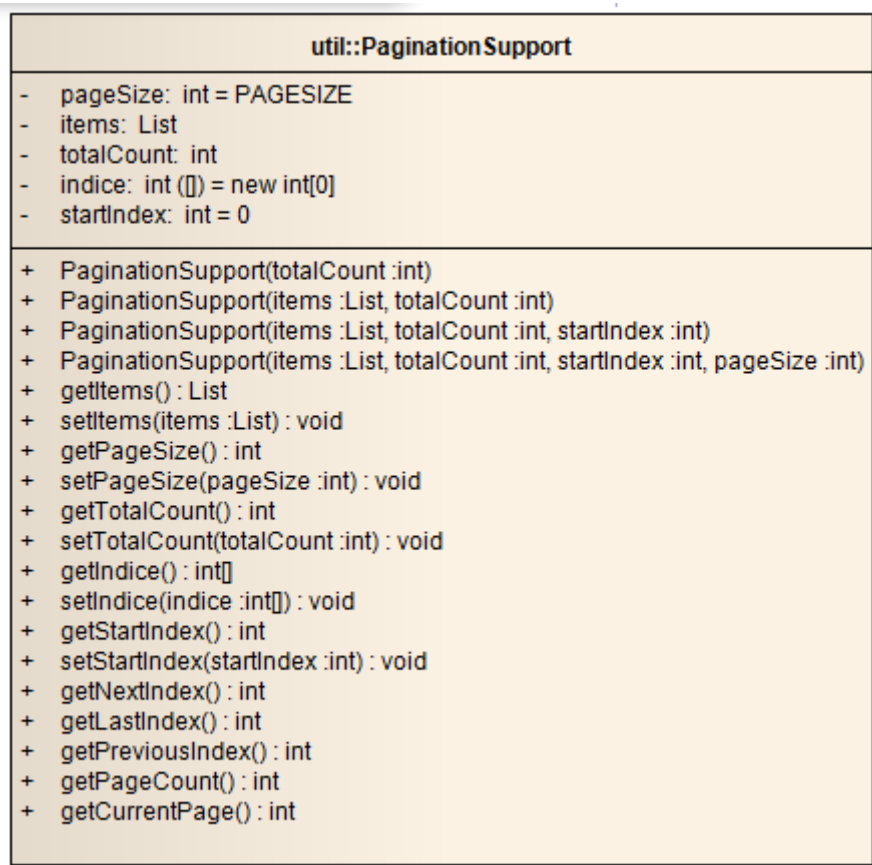


图 4.17 PaginationSupport 类图

如图 4.18，显示参数值时，找到当前 task 后，通过服务 probeHisDataManager 获取参数集合；

- 设计要点 3-2：这里最重要的是参数的计算，由于不同的 probe 方法有不同的计算方法，模块将计算的服务类抽象出来，分别是计算规模的 ProbeSizeCalculator 和计算时间的 ProbeTimeCalculator，具体不同的计算方法则是分别继承自这两个类；同时将代码的共同部分抽离出来，组织成方法 getProbeParameters，这些设计都会使代码更加简洁易懂，结构清晰，易于维护，具体类图见图 4.19。

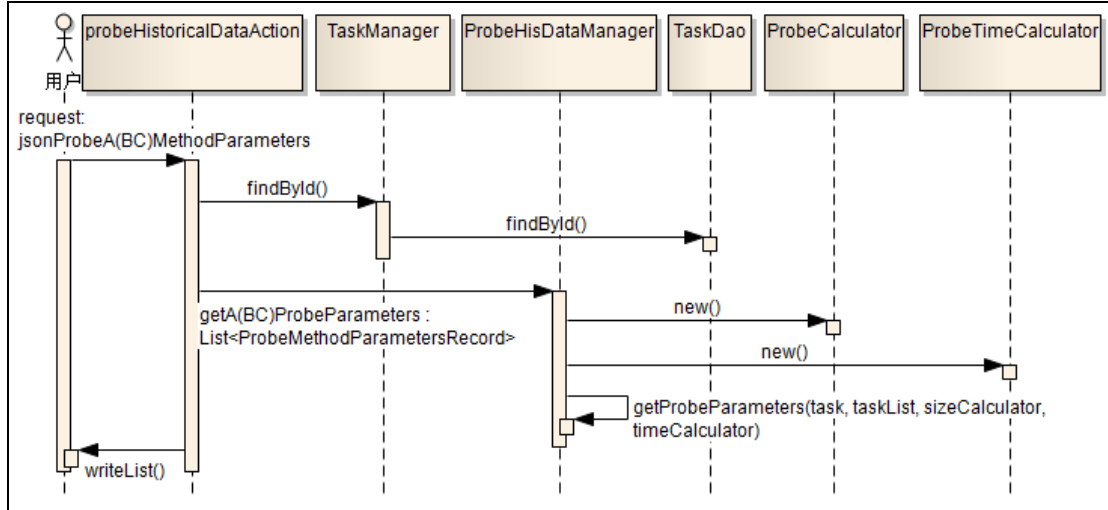


图 4.18 显示参数值流程

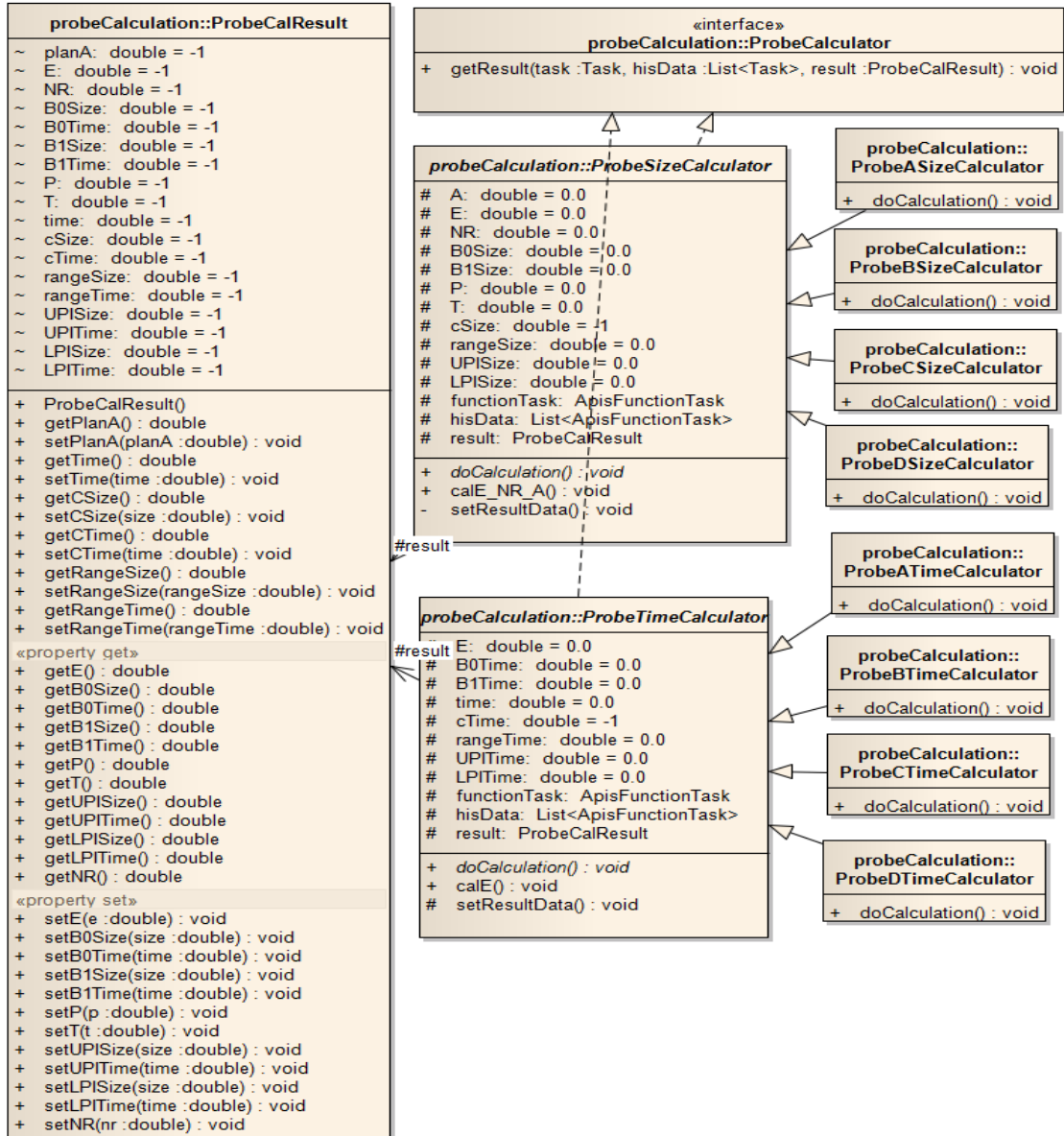


图 4.19 参数计算相关类图

4.3.4 选择估算方法估算子模块

在这里，用户需要选择使用的规模估算和时间估算的方法，而后提交数据进行计算，最后系统会反馈估算结果。

如图 4.20 所示，用户确认提交估算方法以后，模块首先确认估算方法的准确性，而后保存估算方法；根据所选择的方法以及历史数据，计算出估算结果。

- 设计要点 4-1：对于计算类，使用了简单工厂的设计方式，通过使用 ProbeCalculatorFactory 类的 getSize(Time)Calculator 返回计算类的对象。

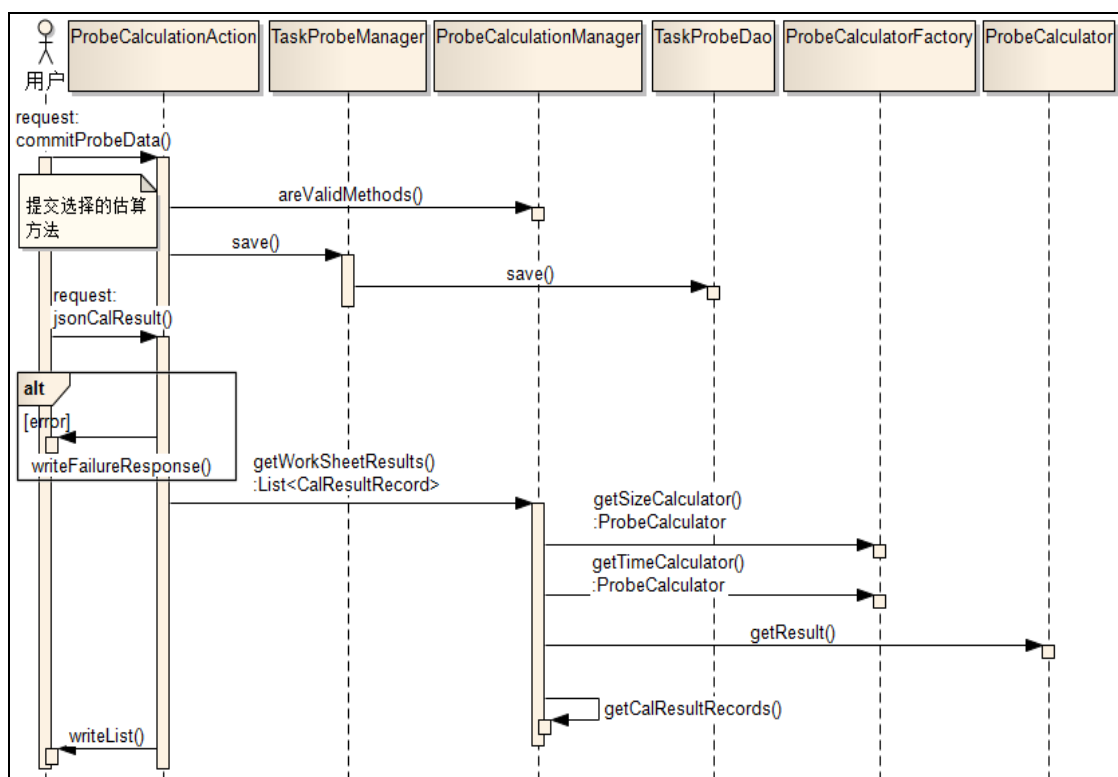


图 4.20 估算流程

第五章 估算模块实现

5.1 管理基础信息子模块

界面上，具体使用 grid-probe-categoryInfo 层和 grid-probe-basicInfo 层中的哪一个，是在 probe.js 中定义的，先尝试加载基础信息，如果可以成功加载，那么就在 grid-probe-basicInfo 层中显示 BasicInfoPanel（render 方法），如果不成功，一种情况是在 grid-probe-categoryInfo 层中显示 CategoryInfoPanel，另外一种情况就是没有权限编辑，并显示错误信息，具体实现见图 5.1：

```
Ext.Ajax.request({
    url : 'probe.do?method:loadProbeBasicInfo',//先尝试加载基础信息
    success: function(response,options) {
//省略了反馈结果responseArray与用于区分界面是否可编辑的变量editable；如果基础信息是可以加载的，那么说明已经提交过数据，不可以更改。

        if(responseArray.success=='true'){

            Probe.createBasicInfoPanel(responseArray.data);

            Probe.createMainPanel(editable);//create the main panel

            Probe.loadMainData();//加载其他数据

        }else {

//没有加载到数据，要么是还没有基础信息，要么是没有权限编辑

            if(editable){

//如果基础信息部分可以编辑，那么就创建CategoryInfoPanel

                Probe.createCategoryInfoPanel();

                Probe.createMainPanel(editable);

            }else{

//没有权限编辑

                Ext.Msg.alert('Probe','You can not do probe for the owner of this module');}
```

图 5.1 层选择代码

这里，接受并处理用户请求的是 ProbeAction.java，其中最主要的方法是 commitCategoryData 方法以及 loadProbeBasicInfo 方法，分别用于提交基础信息以及加载基础信息。这些都是通过调用 service 完成业务逻辑的（各种 Manager），并且使用 bean(Task 等)填充数据并保存数据，具体实现见图 5.2 与图 5.3：

```
//使用Service-taskManager,languageManager,sizeUnitManager,tagManager获取当前task
Task task = this.taskManager.findById(taskId);

Language language = this.languageManager.findById(this.languageId);

SizeUnit unit = this.sizeUnitManager.findById(this.sizeUnitId);

Tag tag = (this.tagId == null) ? null : this.tagManager.findById(this.tagId);
//此处省略用unit,language,tag,taskProbe填充task的过程
//调用service的保存方法
this.taskManager.save(task);

this.taskProbeManager.save(probe);
//返回正确结果响应
this.writeSuccessResponse();
```

图 5.2 commitCategoryData 核心代码

```
//使用Service-taskManager获取当前task
Task task = this.taskManager.findById(taskId);
//当task不存在时的错误处理
if (task == null) {
    log.warn("Can not get task with id:" + taskId);
    this.writeFailureResponse(PSPErrorCodeEnum.TASK_NULL);
} else {
    Category category = AbstractCategory.getRestrictCategory(task);
    this.writeSuccessResponse(task, category, editable);//反馈结果
}
```

图 5.3 loadProbeBasicInfo 核心代码

在具体的 Service 和 Dao 的实现中，使用了模板，为了指定 T 的类型，一方面，程序在 Manager 类和 Dao 类定义的时候指定，具体实现如图 5.4:

```
//服务类接口定义
public interface TaskManager extends BaseManager<TaskDao, Task>

//Dao接口定义
public interface TaskDao extends GenericDao<Task, Integer>,PageDao<Task, Integer>
```

图 5.4 定义 Manager 和 Dao

另一方面，在配置文件 applicationContext-Service.xml 与 applicationContext-Struts.xml 中加以定义，将 Action，Service，Dao 联系起来，让 Spring 容器管理注入，减少了依赖，尤其是 Service 与 Dao 之间的关联，体现了业务层与持久层之间的联系。具体配置信息实例见图 5.5：

```
<bean id="taskManager" class="apis.server.service.impl.TaskManagerSpring">
    <property name="dao" ref="taskDao" />
    <property name="taskMemberDao" ref="taskMemberDao" />
</bean>
```

图 5.5 配置文件

在所有将要使用到的 Action,Manager,Dao 里，模块都是这样定义与配置的，之后将不再赘述。

5.2 管理估算数据子模块

这一部分界面的设计是按照 panel-grid-store 的层次关系展开的。

整个估算数据部分使用 Probe.PartsPanel 组件定义，主要分成 3 个部分，由 3 个 Grid 组件表示，分别是 Probe.BasePartGrid，Probe.AddedPartGrid 和 Probe.ReusedPartGrid，每个 Grid 组件中都定义对应数据存储，分别是 Probe.BasePartStore，Probe. AddedPartStore，Probe. ReusedPartStore，在 store 中使用 http 数据代理，从传入的 URL 中请求数据，并通过 JsonReader 对象读取结果数据，数据被显示到 Grid 中。

实现窗口显示相对大小矩阵时，在 panel 的基础上又增加了一层 window 组件，即 Probe.PartTypeStandardWindow。

Js 中通过 URL 指定对 Action 中具体方法的调用。

相对大小矩阵操作对应的 Action 是 PartTypeStandardAction，最主要的方法是用于加载数据的 jsonPartTypeInfo 方法和用户确认相对大小矩阵的 commitStoreData 方法。

估算数据的提交操作对应的 Action 是 ProbeDataAction，调用 commitPartsData 方法，该方法主要完成 2 项工作：确认增加和修改的记录，从底层删除用户在界面上删除的记录。

难点在于在 Action 中获取页面上增加、修改与删除的记录。首先在提交请求

前，获取估算部分的更新数据 data，将 data.basePart (=data1) 定义为基础部分的更新数据，这些更新数据又包含 3 部分 data1.added,data1.modified,data1.deleted, 分别表示基础部分增加，修改和删除的数据；data 被作为参数提交以后，在 Action 中通过 getObject("basePart") 获取基础部分更新数据，再通过 getArrayAttribute(...,"added/modified/deleted",...)方法获取更详细的增加，修改或删除的数据，实现了记录的获取。

页面与 Action 之间的数据传输以 JSONObject 的数据格式传输。图 5.6 是从页面到 Action 的参数传递，图 5.7 是从 Action 到页面的数据传输，并在页面中加以解码，这种数据传输格式被广泛应用于当前项目中。

```
params: { //编码
    probeData:Ext.util.JSON.encode(data), taskId:this.taskId
}
```

图 5.6 编码

```
//Action中传递数据到js文件中
JSONObject jsonObj = new JSONObject();
jsonObj.put("success", "false");
jsonObj.put("message", e.getValue());
jsonObj.put("editable", editable);
this.writeToResponse(jsonObj);
//js文件中解码
var responseArray = Ext.util.JSON.decode(response.responseText);
```

图 5.7 传递数据和解码

5.3 管理历史数据子模块

为了实现隐藏/拉伸，界面上使用了 FieldSet 组件。将 panel-gird-store 的层次结构变成 panel-fieldSet-grid-store 结构，在 Probe.HistoricalDataPanel 中定义 Probe.ExpandLoadFieldSet（继承自 Ext.form.FieldSet）的实例，这些实例以不同 gird 作为参数，历史数据对应的 grid 是 Probe.HistoricalDataGrid，参数对应的 grid 是 Probe.MethodParametersGrid；A 方法与 BC 方法分别对应不同的 grid，通过传入不同的参数实现。在 Probe.ExpandLoadFieldSet 的定义中，最重要的定义是

expandHandler，用于在页面展开之前加载数据。

历史数据相关的业务逻辑包含在 ProbeHistoricalDataAction.java 中，最主要的方法是用于加载历史数据的 jsonProbeA(BC)HistoricalData 方法以及加载参数的 jsonProbeA(BC)MethodParameters 方法。方法的逻辑已经在设计部分给出，不再详述。

加载参数时，首先要调用服务计算参数值。不同的 Probe 估算方法对应不同的计算参数的方法，映射到代码中是 ProbeA(BC)Size(Time)Calculator 类的 doCalculation 方法的实现不同。

Probe AB 方法中使用线性回归计算规模与时间的估算范围，图 5.8 中以 B 方法的规模估算为例，给出代码实现：

```
//获取代理规模估算数据列表，以及实际程序规模数据列表
    LinkedList<Double> planSizeList = ProbeCalculationUtil.getPlanAM(this.hisData);
    LinkedList<Double> actualSizeList = ProbeCalculationUtil.getActualAM(this.hisData);
//计算相关性r的平方
    cSize = ProbeCalculationUtil.getCorrCoSquare(planSizeList,actualSizeList);
//计算 $\beta_{0size}$ 以及 $\beta_{1size}$ 
    B0Size = ProbeCalculationUtil.getB0(planSizeList, actualSizeList);
    B1Size = ProbeCalculationUtil.getB1(planSizeList, actualSizeList);
//获得估算规模大小
    P = ProbeCalculationUtil.getP(B0Size, B1Size, this.E);
//计算浮动范围
    rangeSize = ProbeCalculationUtil.getRange(this.E, planSizeList,actualSizeList);
//计算规模估算上限和下限
    UPISize = P + rangeSize;
    LPISize = P - rangeSize;
```

图 5.8 B 方法规模估算

这里 ProbeCalculationUntil 类是领域模型层的工具类，用于计算理论基础中所涉及到的数学公式。

Probe C 方法使用比例法计算规模与时间的估算，图 5.9 中以 C 方法的规模估算为例，给出代码实现：

```
//省略获取代理规模计划数据toDatePlannedSize与实际程序规模数据toDateActualSize
//如果计划值为0，实际值不为0，那么 $\beta_{1size}$ 设为0，如果都为0，报错
if (MathUtil.isZERO(toDatePlannedSize)) {
    if (!MathUtil.isZERO(toDateActualSize))
        B1Size = 1.0;
    else
        throw new ProbeNotEnoughDataException("not enough historical data for size
method C");
} else { //计划值和实际值都不为0，相除计算比例，即回归参数
    B1Size = MathUtil.divide(toDateActualSize, toDatePlannedSize);
}
P = ProbeCalculationUtil.getP(0, B1Size, this.E); //计算最终估算值
```

图 5.9 C 方法规模估算

Probe D 方法使用猜测， β_0 设为 0， β_1 设为 1，使用 getP 方法即可。

5.4 选择估算方法估算子模块

界面继续以 panel-grid-store 的层次结构展开，在 js 文件中定义了 Probe.WorksheetPanel，Probe.WorksheetGrid 以及 Probe.ProbeCalResultStore。

处理估算方法的业务逻辑包含在 ProbeCalculationAction.java 中；其中 commitCalResult 方法用来确认保存 probe 估算数据，jsonCalResult 方法用来获取计算结果。

提交方法进行估算时，模块需要验证 probe 方法的正确性：

一方面，在提交估算数据之前，在 js 页面中会加以验证，图 5.10 中给出了代码实现。

```
//调用下拉框的验证方法
var valid_input1 = this.sizeCombo.validate();
var valid_input2 = this.timeCombo.validate();
//当时间估算方法选择D方法时，需要输入工作效率，这也需要验证
if(this.timeCombo.getValue == 'D' && !this.productivity.getValue)
    valid_input2 = false;
//对下拉框内容进行验证，如果失败，返回
var valid = (valid_input1 && valid_input2);
if(!valid) return ;
```

图 5.10 js 页面验证方法

另一方面，在服务 `probeCalculationManager` 里提供了方法 `areValidMethods`，用于验证选择的 `probe` 方法是否有效，这个方法在 `Action` 中调用。在该方法中，先调用 `TaskProbe.isValidProbeMethodString` 确认方法名是不是 A,B,C,D，而后计算结果，看计算过程是否会产生异常，从而判断 `probe` 方法是否有效。

工厂 `ProbeCalculatorFactory` 类包含 2 个静态方法：`getSizeCalculator` 方法和 `getTimeCalculator` 方法，根据参数返回不同 `Probe` 方法的计算类实例。以 `getSizeCalculator` 为例，给出代码实现，如图 5.11 所示。

```
public static ProbeCalculator getSizeCalculator(String probeSize) {
    char probe = StringUtil.getFirstLowerChar(probeSize);
    ProbeCalculator cal = null;
    //根据参数返回对应实例
    switch (probe) {
        case 'a': cal = new ProbeASizeCalculator();
                break;
        case 'b': cal = new ProbeBSizeCalculator();
                break;
        case 'c': cal = new ProbeCSizeCalculator();
                break;
        case 'd': cal = new ProbeDSizeCalculator();
                break; }
    return cal; }
```

图 5.11 工厂方法

5.5 估算模块运行效果

- 在初次打开 PROBE 界面时，要求用户填写基础信息，界面效果如图 5.12 所示；规模单元默认选项是 Page（页）和 LOC（代码行），语言默认选项是 C++,C#和 Java，这两项是必须要填写的，而标签由自己填写，是可选的。

图 5.12 填写基础信息界面

- 填写基本信息并点击提交按钮以后，出现主界面，界面效果如图 5.13 所示（隐藏了主要面板的内容信息）；

图 5.13 PROBE 估算主界面

- “基本信息”展开后显示成功提交过的估算基本信息，界面效果如图 5.14 所示；

基本信息

任务:

5

规模单元:

Page

语言:

Java

标签:

图 5.14 “基本信息”查看界面

- “任务”展开后显示部件信息，包含 3 个组成部分：基础、新增和复用，用户可以进行增删改操作，界面效果如图 5.15 所示；

任务

部件-基础

新增基础部件

移除基础部件

连接至SVN

Name	Plan Base	Plan Del	Plan Mod	Plan Add	Act Base	Act Mod	Act Del	Act Add
a	12	0	1	1	11	1	1	1
(1 Item)								

部件-新增

新增添加部件

移除新增部件

连接至SVN

调整相对大小

Name	Part Type	Plan Items	Rel Sz	Plan Size	*	Act Items	Act Size	*
New Part	Calculation	1	M	11.25	<input checked="" type="checkbox"/>	1	10	<input checked="" type="checkbox"/>
(1 Item)		1		11.25		1	10	

部件-重用

新增重用部件

移除重用部件

Name	Plan Size	Act Size
c	11	11
(1 Item)		11

提交

图 5.15 任务部分界面

- 如果基本信息中的规模单元选择的是 LOC，那么本模块将提供相对大小

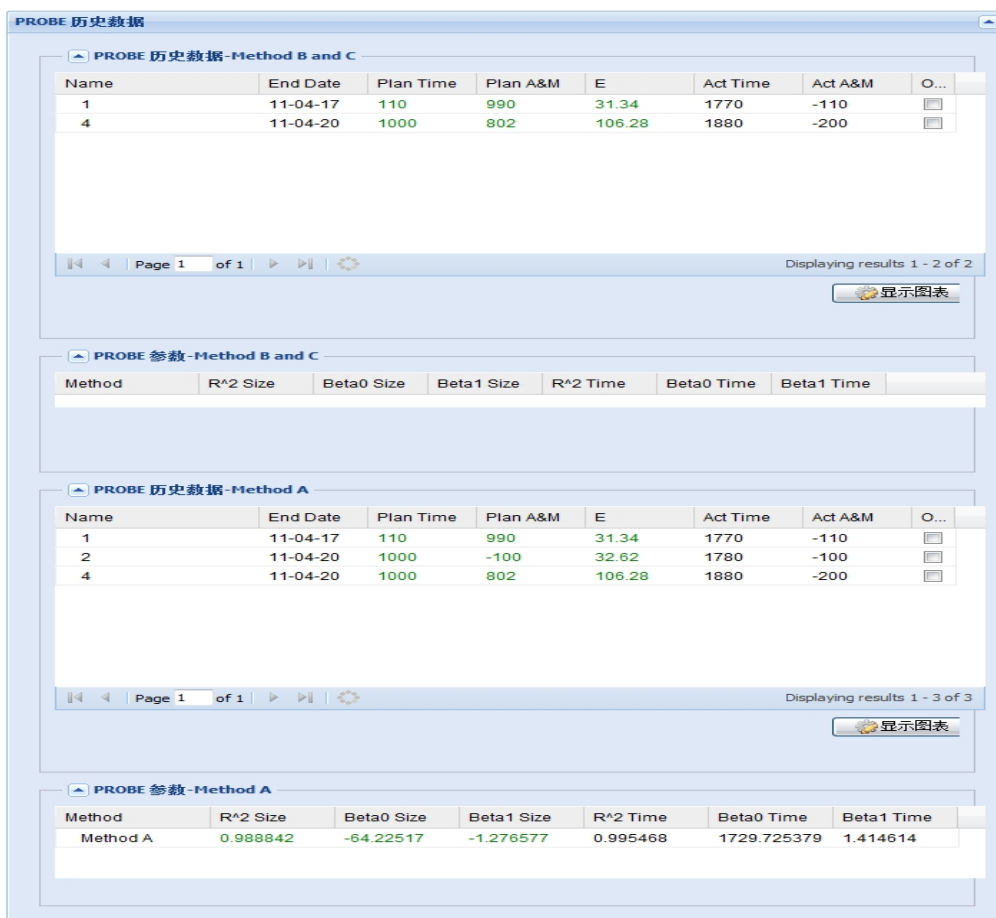
矩阵来自动生成计划部件规模大小。点击新增部分的“调整相对大小”按钮，出现相对大小矩阵面板（默认值是 2.3 节中表 2.1 里的数据），用户可以对数据进行调整，面板效果如图 5.16 所示。



Name	VS	S	M	L	VL
Calculation	2.34	5.13	11.25	24.66	54.04
Data	2.6	4.79	8.84	16.31	30.09
I/O	9.01	12.06	16.15	21.62	28.93
Logic	7.55	10.98	15.98	23.25	33.81
Set-up	3.88	5.04	6.56	8.53	11.09
Text	3.75	8	17.07	36.41	77.66
Other	0	0	0	0	0

图 5.16 相对大小矩阵

- “PROBE 历史数据”展开以后，显示历史数据与参数值。这些数据以方法进行组织，运行效果如图 5.17 所示；



PROBE 历史数据-Method B and C							
Name	End Date	Plan Time	Plan A&M	E	Act Time	Act A&M	O...
1	11-04-17	110	990	31.34	1770	-110	
4	11-04-20	1000	802	106.28	1880	-200	

PROBE 参数-Method B and C						
Method	R^2 Size	Beta0 Size	Beta1 Size	R^2 Time	Beta0 Time	Beta1 Time
Method A	0.988842	-64.22517	-1.276577	0.995468	1729.725379	1.414614

PROBE 历史数据-Method A							
Name	End Date	Plan Time	Plan A&M	E	Act Time	Act A&M	O...
1	11-04-17	110	990	31.34	1770	-110	
2	11-04-20	1000	-100	32.62	1780	-100	
4	11-04-20	1000	802	106.28	1880	-200	

图 5.17 历史数据

- 当用户填写完部件信息并提交以后，下一步就是进行估算。展开“PROBE 工作表”，出现估算面板，如图 5.18 所示。用户选择规模估算和时间估算的 PROBE 方法，如果时间估算选择了 D 方法，还需要输入生产效率，然后点击“计算结果”按钮，如果选择的方法有效，系统会自动进行计算，反馈预测区间等计算结果。

PROBE 工作表

Probe 规模: D

Probe 时间: D

生产率(每小时): 10

Name	Cal Function	Size	Time
Added Size(A):	$A = BA + PA$	11.25	
Estimated A&M(E):	$E = BA + PA + M$	11.25	
Correlation(R^2):			
Regression Parameter(B0):	Size and Time	0	0
Regression Parameter(B1):	Size and Time	1	6
Projected A&M(P):	$P = B0 + B1 * E$	11.25	
Estimated Total Size(T):	$T = P + B - D - M + R$	11.25	
Estimated Total New Reusable(NR):	sum of * Items	11.25	
Estimated Total Development Time:	$Time = B0 + B1 * E$		67.5
Prediction Range:	Range	0	0
Upper Prediction Interval:	$UPI = P + Range$	0	0
Lower Prediction Interval:	$LPI = P - Range$	0	0
Prediction Interval Percent:		0.7	0.7

计算结果 应用到子任务

图 5.18 工作表

第六章 总结与展望

为了更有效的支持 TSP 在现实项目团队中的应用，尤其是利用时间和规模估算制定早期规划，本文从设计和实现角度描述了实训平台中估算模块，便于团队成员在项目早期即制定一份合理的规划，为后续详细计划打下基础。本文主要研究成果如下：

- 论述了使用工具支持 TSP 进行团队管理的必要性，尤其是估算对于制定项目计划的重要性，分析并指出了现有的一些 TSP 支持工具的局限性，突出了 TSP 支持工具，即实训平台以及其估算模块设计与实现的现实意义。
- 描述了实训平台估算模块的需求，给定了设计与实现的目标。
- 阐述了实训平台估算模块的总体分层架构，通过 SSH 架构的集成使用，使得整个模块层次分明，每个框架负责不同层次，层次之间相对独立，功能明确，耦合程度降低，提高了程序的可维护性与可重用性。
- 给出了每个子功能的详细设计与实现，尤其突出设计原则与设计方法的使用。

在下一步的工作中，项目组需要对系统做进一步的完善与改进：

- 在估算时，选择合适的 PROBE 方法尤为重要，实训平台系统对于 PROBE 方法的有效性验证还存在一定的缺陷，包括设计不合理（需要执行两次相同的参数计算）、条件缺乏验证（相关性和显著性验证）、极端数据不能自动剔除等问题，这些都是以后对估算模块的改进工作中的重点。
- 对于整个实训平台系统，已经进行过一系列的改进，包括剔除了一些冗余的功能，将缺陷日志改成表格下载方式等，但还需要做进一步的改进：详细标明缺陷阶段信息、改进数据的展示方式、优化日程安排算法、导出项目数据并加以分析等，以增加实训平台的现实应用价值。

综上所述，通过本次毕业设计，我对实训平台这一 TSP 支持工具有了更加深入的理解，为以后做进一步的项目改进打下了坚实的基础；同时，在技术方面，对 SSH 架构设计方案与 Ext JS 界面设计技术的应用有了更直观的感受，对于一些优秀的设计模式的理解也逐步加深，于我而言，这是一笔宝贵的经验财富。

参考文献

- [1] Watts S. Humphrey. Introduction to the Team Software Process . 第一版. 北京. 清华大学出版社. 2002. 4 页
- [2] SEI - TSP Overview. <http://www.sei.cmu.edu/library/upload/TSPoverview.pdf>
- [3] Watts S. Humphrey. PSPSM: A Self-Improvement Process for Software Engineers. 第一版.北京.人民邮电出版社.2006. 3 页、314 页、70 页-108 页
- [4] James McHale, Timothy A. Chick, Eugene Miluk . Implementation Guidance for the Accelerated Improvement Method (AIM) . SPECIAL REPORT : CMU-SEI-2010-SR-032 . 22 页
- [5] Introductory Team Software Process(TSPi).
<http://www.sei.cmu.edu/tsp/tools/tspi/>
- [6] The Software Process Dashboard Initiative. <http://www.processdash.com/>
- [7]刘京华 . Java Web 整合开发王者归来 . 第一版 . 北京 . 清华大学出版社 . 2010.1 . 456 页-462 页, 532 页-533 页
- [8]Craig Walls Ryan Breidenbach . 毕庆红 王军等. Spring in Action. 第二版. 北京. 人民邮电出版社. 2008.10. 3-4 页,79 页
- [9]张鑫、黄灯桥、杨彦强. JavaScript 凌厉开发----Ext 详解与实践. 第一版. 北京. 清华大学出版社.2009.3. 5 页-7 页, 104 页
- [10] 张鑫、黄灯桥、杨彦强. JavaScript 凌厉开发----Ext JS3 详解与实践. 第一版. 北京. 清华大学出版社.2010.4. 3 页-10 页、81 页-87 页
- [11] ext store. <http://blog.csdn.net/luckstar1999/archive/2009/06/24/4295985.aspx>
- [12] Ext.Ajax.request – fuwenbint 的专栏.
<http://blog.csdn.net/fuwenbint/archive/2010/11/13/6006495.aspx>
- [13] Wiring Your Web Application with Open Source Java.
<http://onjava.com/pub/a/onJava/2004/04/07/wiringWebapps.html>
- [14] Eric Freeman, Elisabeth Freeman, Kathy Sierra, Bert Bates. Head First Design Patterns. 第一版. America. O'Reilly Media , Inc. 2004. 288 页

致谢

时光荏苒，四年的本科生涯与这篇论文一样，即将在这一刻画上句点，付出与收获也要在此刻做出总结。无论快乐还是忧伤，努力还是迷茫，都被深深地镌刻在我的青春岁月里，并将在我的生命中留下永恒的记忆，对此我充满感激。

首先，我要感谢荣国平老师与邵栋老师。他们在论文的写作过程中给与了我非常耐心的指导，从论文的选题、研究思路到内容结构，他们都认真负责地进行了一一解析，指导并鼓励我们去探究重要的技术与理论，得以使我最后顺利完成此次毕业设计。

我要感谢实训平台项目的全体开发人员，他们通过辛勤的努力在短期内完成了实训平台项目，给予了我们观摩与学习项目的机会。

同时，我要感谢项目改进小组的其它两位同学：庄晨熠和李静怡，他们在论文的思路和具体内容的确定上给予了我很大的帮助，让我受益匪浅。

感谢我的舍友，大学四年给予了我很多帮助与鼓励，在各方面给予了我宝贵的建议。

感谢我的家人，正是在他们的默默鼓励和支持下，我才得以顺利地完成大学学业。

最后，衷心感谢所有曾经关心与帮助过我的老师，同学和朋友，谢谢！