

北京交通大学

本科毕业设计（论文）

一款基于 GEF 的  
列车全仿真模型 MON 屏编辑软件的实现

**Realization of an editing software  
of monitor screen in the full simulation model of trains  
based on GEF**

学    院：\_\_\_\_软件学院\_\_\_\_

专    业：\_\_\_\_软件工程\_\_\_\_

学生姓名：\_\_\_\_XXX\_\_\_\_

学    号：\_\_\_\_XXXXXX\_\_\_\_

指导教师：\_\_\_\_XXX\_\_\_\_

北京交通大学

2016 年 5 月

# 学士学位版权使用授权书

本学士学位作者完全了解北京交通大学有关保留、使用学士学位的规定。特授权北京交通大学可以将学士学位的全部或部分内容编入有关数据库进行检索，提供阅览服务，并采用影印、缩印或扫描等复制手段保存、汇编以供查阅和借阅。

（保密的学位论文在解密后适用本授权说明）

学位论文作者签名：

指导教师签名：

签字日期：      年    月    日

签字日期：      年    月    日

## 中文摘要

**摘要：**经过第六次全面大提速之后，中国铁路交通走入了快速铁路时代，动车组高铁列车的搭建规模与运营里程已经达到了世界领先水平。但是，速度提升的背后存在着故障应急处理体系不完善的巨大安全隐患。这不仅关系到交通系统运输任务的高效完成，而且关系到全国人民的生命安全。因此，本系统通过创建动车组列车全仿真模型，为列车相关人员提供故障应急处理培训，以及为动车研究人员提供故障注入与分析平台。经过系统培训之后的列车相关人员，如司机、列车员，能够积累与实车行驶相同的故障处理经验，有效缩短延误时间，保障人民的生命财产安全。而动车的研究人员可以通过故障的注入与显示得到全面可靠的数据，方便进一步进行故障分析。

本论文阐述了动车组列车全仿真模型故障注入、分析与排除系统中的 MON 屏可视化图形编辑软件的设计与实现过程。该实验台将创建一个动车组列车模型的全仿真环境。研究人员通过向环境中注入以及排除故障，形成一系列典型的故障分析库，从而实现了对人员、模型和故障信息的有效管理。MON 屏可视化编辑软件与仿真机建立数据通信机制，实现了可视化图形方式编辑仿真列车模型与监控界面，将故障信息内容通过司控台传递并显示在编辑软件上，能够与整个仿真模型组成为一个整体。

在整个项目中，本人主要负责项目需求分析与建模、MON 屏编辑软件的详细设计与开发实现过程。根据客户需要建立编辑软件的功能需求，将软件开发的整个流程融入到本项目的软件开发过程中，设计并完成了工程管理、图片编辑、数据通信与脚本关联四个模块的开发任务。本项目采用了 Eclipse RCP 作为基本工作平台，GEF 图形编辑框架实现核心图形控件建模、关联、显示以及面板可视化编辑的功能，将 Navigator 导航资源管理器、PV Manager 数据通信模式与 Jython Script 脚本模式，采用插件方式作为编辑软件的辅助功能。本项目 Beta 版本的软件已经测试上线，用户对于软件的反馈非常良好。关于软件测试版本的改进意见与优化方案，本论文已在后半部分做出阐述。为了满足用户更加丰富的需求以及完善系统功能，软件的拓展与更新也会持续推进。

**关键词：**动车仿真模型；可视化图形编辑；GEF 框架；插件开发

## ABSTRACT

**ABSTRACT:** After the sixth fully speed up of railway in 2007, China entered into the age of high-speed railway. Construction scale and transportation mileage of CRH has reached the world advanced level. However, behind the accelerating of railway, it raises that huge security risk of imperfect emergency system in dealing with malfunction. It is not only related to the highly efficiency of railway transportation, but also the precious lives of people across the country. Therefore, according to realizing a fully stimulation model of trains, the system provides CRH related staff for the emergency training of handling troubles and platform of trouble injection and analysis for training researchers. After training, related staff, such as training driver and crew, are able to gaining experience consistent with real handling experience which can shorten fixing time efficiently and ensure people's property and lives safety. Training researchers using trouble injection and illustrating gains reliable data from all perspective which benefit to analyzing malfunction.

This paper introduces designing and realization of visual editing software of monitor screen in full simulation model of trains. The lab implements the all real stimulation environment. MON screen visual editing software builds connection with full stimulation model to transfer monitoring data. This communication mechanism implements getting malfunction from full stimulation model to display on edited page that is the same as the real monitor page. In this respect, editing software as a part of the full stimulation model is the important eye.

Around the whole project, i am responsible for requirement and modeling, realizing and designing system in detail. The study completes a series of software developing process. The visual graphics editing software which uses Eclipse RCP as the basic workbench, builds GEF frame to implement visual editing function including modeling, communicating, displaying. Auxiliary functions - navigator, PV connection, Jython script, are added into main program in the plugin way. Beta version has been on line for test and user's feedback is very good. About improvement and optimizing solutions, the paper will describe them in detail at the second half. In order to satisfy user's requirement and perfect system functions, the software will continue updating and maintaining all the time.

**KEYWORDS:** CRH full stimulation model; visual graphics editing;  
GEF frame; plugin development

## 目 录

中文摘要 .....	I
ABSTRACT .....	II
目 录 .....	III
1. 引言 .....	1
1.1 项目背景 .....	1
1.2 课题研究来源 .....	2
1.3 课题研究意义 .....	3
1.4 课题研究的主要内容 .....	3
1.5 个人主要工作 .....	4
1.6 论文组织结构 .....	4
2. MON 屏编辑软件需求分析 .....	5
2.1 总体功能需求 .....	5
2.2 系统功能性需求分析 .....	7
2.2.1 工程文件管理模块 .....	7
2.2.2 图形编辑模块 .....	12
2.2.3 数据通信模块 .....	16
2.2.4 脚本编辑关联模块 .....	17
2.3 系统非功能性需求分析 .....	19
2.3.1 性能 .....	19
2.3.2 易用性 .....	19
2.3.3 可移植性 .....	20
2.3.4 可维护性 .....	20
3. MON 屏编辑软件架构设计 .....	21
3.1 系统体系架构设计 .....	21
3.2 核心技术点介绍 .....	21
3.2.1 SWT/JFace 绘制基本窗口部件 .....	21
3.2.2 Eclipse RCP 提供基本工作台功能 .....	22
3.2.3 高内聚低耦合的 PLUG-IN 插件 .....	22
3.2.4 GEF 图形编辑框架实现核心功能 .....	23
3.2.5 MVC 模式详解 .....	24
3.3 开发环境 .....	24
4. MON 屏编辑软件功能模块设计与实现 .....	25
4.1 工程管理模块 .....	25
4.1.1 设计描述 .....	25
4.1.2 时序图 .....	25

4.1.3 核心类的结构设计.....	26
4.1.4 实现效果图.....	27
4.2 图形编辑模块.....	29
4.2.1 设计描述.....	29
4.2.2 时序图.....	29
4.2.3 核心类的结构设计.....	30
4.2.4 核心功能实现.....	34
4.2.5 实现效果图.....	38
4.3 数据通信模块.....	40
4.3.1 设计描述.....	40
4.3.2 时序图.....	40
4.3.3 核心类的结构设计.....	41
4.3.4 实现效果图.....	41
4.4 脚本关联模块.....	42
4.4.1 设计描述.....	42
4.4.2 时序图.....	42
4.4.3 核心类的结构设计.....	43
4.4.4 实现效果图.....	44
<b>5. 项目整体优化方案.....</b>	<b>45</b>
5.1 数据通信功能优化.....	45
5.1.1 通信变量分渠道绑定 PV 值.....	45
5.1.2 绑定变量的列表显示.....	45
5.2 脚本编写功能优化.....	46
5.3 基本图形控件功能优化.....	46
<b>6. 结论 .....</b>	<b>47</b>
<b>参考文献 .....</b>	<b>48</b>
<b>致 谢 .....</b>	<b>49</b>
<b>附 录 .....</b>	<b>50</b>

## 1. 引言

### 1.1 项目背景

20 世纪 90 年代至 21 世纪，我国轨道交通运输进入快速发展的新时期。随着国民经济的迅速发展、城市规模的不断扩大，国民对于公共交通运输有了更高更快的要求，铁路轨道交通的作用因而显得愈发突出。2008 年以来，中国的铁路建设投资量迅猛增加，铁路制造业也逐年随之高速增长。我国铁路车辆设备的投资由 2008 年的 176 亿增长到近年来的 1430 亿元，又一次创出历史新高。作为“十二五”的收官之年，也是贡献最大的一年，2015 年全年完成固定铁路资产投资 8238 亿元，其中基建投资 5925 亿元；投产新线 9531 公里，其中高铁 3306 公里；年内 61 个新项目顺利开工建设<sup>[1]</sup>。铁路建设投资力度加大、路网规模和质量显著提升的时期。众多新线开通运营后，高铁成网效应更加明显，铁路运输能力进一步提高，旅客出行更加便捷。现在，动车组列车速度高速增长，从北京到海南由原来十年前的平均速度只有 40 公里的“绿皮车”用几天几夜的时间，到现在日行千里的动车只用几个小时就到了蓝天碧水的中国最南端海南岛。下表展示了中国铁路总公司计划统计部提供的 2015 年 1 月到 12 月，国家铁路主要指标完成情况。

表 1-1 2015 年 1~12 月国家铁路主要指标完成情况

指标	计算单位	本年完成	上年完成	比上年 增 减	比上年 增减%
一、铁路运输					
1.旅客发送量	万人	249558	227180	22378	9.9
2.旅客周转量	亿人/公里	11905.30	11193.77	711.53	6.4
3.货运总发送量	万吨	271387	306942	-35555	-11.6
4.货运总周转量	亿吨/公里	21598.37	25103.42	-3505.05	-14.0
5.总换算周转量	亿吨/公里	33503.67	36297.19	-2793.51	-7.7
二、基本建设投资	亿元	5924.58	5512.61	411.97	7.5

可见，众多新线开通运营后，高铁成网效应更加明显，铁路运输能力进一步提高，旅客出行更加便捷。而主要带动铁路运输需求的高铁动车组列车，在运输速度的提高与运营里程的增加的过程中，同时出现了至关重要的交通运输安全问题。运营车辆的通信信号、控制系统等在运输过程中突发故障都需要健全的故障处理方案实施有效处理。2011年7月23日晚20点30分，由北京南站开往福州站的D301次动车，在运行至甬温线双屿路段时与前行的D3115次动车发生追尾事故。事故造成D301次列车第1至4节车厢脱线，D3115次列车第15、16节车厢脱线，40人死亡，约200人受伤<sup>[1]</sup>。

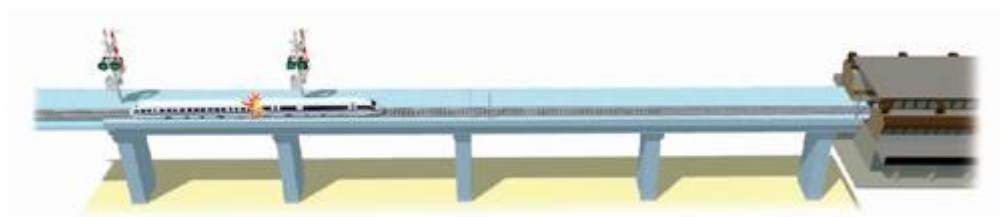


图 1-1 甬温线动车组列车事故模型示意图

仅根据这一项事例就能看出，列车故障没有完备的紧急处理体系而造成的生命财产损失的是非常巨大的。因此，建立健全动车组列车的故障分析与处理体系成为发展具有运能大、安全舒适、可持续能力的高速铁路的重中之重。

## 1.2 课题研究来源

本项目来源于本人在北京经纬恒润科技有限公司实习期间，参与开发的动车组列车全仿真模型故障注入、分析与排除系统。该系统创建了动车组列车实际运行的全仿真环境，实现了实验台模拟行驶过程中发生的全部故障。研究人员通过向环境中注入以及排除故障，形成一系列典型的故障分析库，从而实现对人员、模型和故障信息的有效管理。本项目作为系统的一个重要分支，主要承担起列车模型图形化编辑软件与故障显示的实现任务。项目要求能够实现一款不同于其他图形编辑软件的系统，根据专业背景创建专业的文件编辑显示类型、专业图形控件与图形操作方式。项目本着专项专用原则完成与整个模拟实验台的无缝衔接。



### 1.3 课题研究意义

根据国家《中长期铁路网规划》，2020 年中国铁路营业里程规划目标为 12 万公里以上，其中客运专线为 1.6 万公里，电气化率为 60%。铁路是国家重要的公共基础设施、国民经济大动脉和大众化的交通工具。安全是铁路运输的生命线，是运输生产永恒的主题<sup>[1]</sup>。铁路运输安全不仅影响着企业本身的生产效率和经济效益，也对社会和经济造成重大影响。铁路运输安全是运输生产系统运行秩序正常，运输设备完好无损，运输紧急处理体系健全的综合表现。铁路运输生产的根本任务就是安全运输高效发展。而铁路运输生产的作用、性质和特点，决定了铁路运输必须遵守铁路安全制度，必须拥有高效完善的故障应急处理能力。在国民经济发展进入新常态、铁路管理体制转轨的新形势下，要实现“十三五”时期铁路建设的的提量保质，重点抓好铁路安全体系建设工作意义重大。动车组列车运输相关人员使用全仿真模型下的故障应急处理培训，掌握良好的故障处理能力是对国家和人民生命财产安全最好的保障。本项目旨在通过实验台全仿真模型对列车相关人员进行模拟培训，使其积累与实车完全相同的故障处理经验。从而在实际行车过程中对于紧急故障的快速、准确处理，保障列车的安全运行，也保障人民群众的生命和财产安全。

### 1.4 课题研究的主要内容

列车全模型仿真系统是开发一款针对某动车组列车的故障分析与研究实验台。该实验台将创建一个列车运行的全模型仿真环境。研究人员通过向环境中注入以及排除故障，形成一系列典型的故障分析库，从而实现对人员、模型和故障信息的有效管理。这款实验台系统主要包括：司控台软件、故障注入与分析软件、故障考试软件、MON 屏编辑与显示软件以及运行列车全模型的仿真机。本项目主要研究的内容就是 MON 屏编辑与显示软件的设计与实现。软件能够实现用图形化方式编辑出列车监控屏幕的内容，并且仿真机能够通过数据通信中心将监控的故障信号传递给编辑软件并显示出来。

## 1.5 个人主要工作

在整个项目中，本人主要负责项目需求分析与建模、MON 屏编辑软件的详细设计与开发实现。需求分析与建模中，本人根据与客户的沟通内容与动车行业的学习理解用户需求，做出相应的需求建模与原型设计。与开发人员小组对需求产品给出确定的建议与解决方案，并与客户达成一致意见后进行产品的开发实现。在软件开发过程中，本人作为编辑软件主要的开发人员，完成了工程管理、图片编辑、数据通信与脚本关联四个模块的开发任务，严格依据需求文档做出符合列车模型的图形化控件、信息通讯介质、故障注入渠道与显示模式运行方式。最后，与整个系统的其他的开发人员完成动车组列车全仿真模型实验台的系统调试与运行任务。

## 1.6 论文组织结构

本论文以实际项目动车组列车全仿真模型的故障分析与研究实验台的一个重要分支——MON 屏编辑软件作为研究对象，依据软件开发的规范化流程：需求分析与建模、系统架构设计、功能模块详细设计与实现、提出优化方案的顺序来进行论述的。

第一章：引言，主要从项目的背景、课题研究来源与意义、课题研究的主要内容与个人负责的主要工作四大方面对论文进行简要介绍。

第二章：系统需求分析，从项目简要介绍、系统功能性需求、非功能性需求三个方面，将系统从现实要求变为软件需求进行论述。

第三章：系统总体架构，从系统总体架构设计、核心技术点介绍、开发平台与应用技术三个方面，对系统的框架结构进行论述。

第四章：系统模块设计与实现，从工程管理、图形编辑、数据通信、脚本关联四个模块分别对系统的设计方案与实现原理进行详细阐述。

第五章，系统整体的优化方案，对系统功能性、非功能性方面的优化进行了简要的方案设计阐述。

第六章，总结，对全文内容做出总结。

## 2. MON 屏编辑软件需求分析

### 2.1 总体功能需求

CRH(China Railway High-speed)中国铁路高速，即中国和谐号动车组列车的简称，以其明显高于普通列车的 250km/h 运营速度日渐成为国民出行的优质选择之一。[1]本项目要求整个实验台实现等比例制作动车组列车的仿真模型。由司控台作为核心“司令部”完成实验台的总指挥任务。司控台操作列车模型运行，并将列车运行时全部故障类型通过搭建的故障注入与分析软件传递给仿真机，由仿真机做出实物展现且发送故障数据给 MON 屏编辑软件。列车相关人员在故障培训时根据故障应急处理手册，通过仿真模型对故障做出处理。编辑软件能够同时接收故障数据，并在 MON 屏编辑软件的监控状态时演示出故障的发生情况，供列车故障专业技术研究员统计与分析。实验台同时要求配备一个故障考试软件。技术研究员将故障数据分析整理为一系列的试题，协同故障处理方式，提供给接受专业培训的相关人员进行能力测试。整个项目的动车组列车故障研究实验台由列车全仿真模型、故障注入与分析软件、MON 屏编辑软件与显示器、故障考试软件、司控台六个部分组成，如下图所示：



图 2-1 动车组列车全仿真模型故障注入与分析实验台示意图

MON 屏可视化图形编辑软件是动车组列车全仿真模型故障注入与分析系统的一个重要分支，连同 MON 屏界面的编辑显示器一起组成故障在计算机上编辑的显示功能。

本论文是这一款可视化的图形编辑软件的详细设计与实现过程的论述。项目要求编辑软件提供给用户通过创建图形控件、布局界面，实现绘制列车的仿真模型，以及司机、列车员、记录仪监控屏幕页面的功能。由于系统实验台要求进行故障注入与分析，要求软件建立数据通信中心与仿真机联通，实现监控数据传递故障注入与显示功能。软件拥有编辑模式与监控模式两种方式供用户选择，在监控模式下实现显示列车故障信号的功能，方便研究人员进行故障分析与研究。

MON 屏可视化图形编辑软件的功能主要分为七大部分：

- **工程文件管理功能：**要求实现各个种类工程文件的创建及编辑功能，提供资源导航视图右键菜单的快捷操作方式；
- **图形编辑功能：**要求提供给专业图形控件及布局方式，用户可以在编辑面板中编辑显示器界面；
- **脚本关联功能：**为监控状态下界面根据用户编辑的脚本程序执行显示故障内容服务；
- **数据通信功能：**完成连接司控台与编辑软件，传递仿真机故障的监控数据搭建专属的数据通道；
- **模板编辑供模块：**要求提供包含基本控件与逻辑实现的 OPI, Logic 等界面工程的专业模板；
- **视图窗口编辑：**提供给用户在软件使用中显示各种类操作视图如属性列表等的功能；
- **获取帮助：**主要为用户提供软件使用的帮助示例文档；

下图 2-2 是从用户视角阐述了整个编辑软件的功能模块划分。由于整个编辑软件完成的工作量巨大以及系统交付时间有限，本人参与并完成了其中的四大核心部分的开发工作：工程文件管理、图形编辑器、脚本编辑关联以及数据通信。

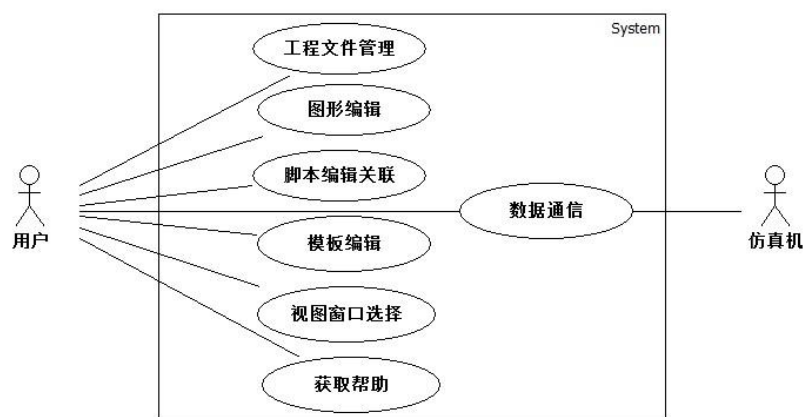


图 2-2 系统总体功能用例图

## 2.2 系统功能性需求分析

### 2.2.1 工程文件管理模块

#### （一）功能描述

该模块是对编辑软件中创建的工程文件进行管理，实现用户创建、删除、复制、粘贴、显示、保存、刷新和导入导出工程文件的功能。

#### （二）用例图

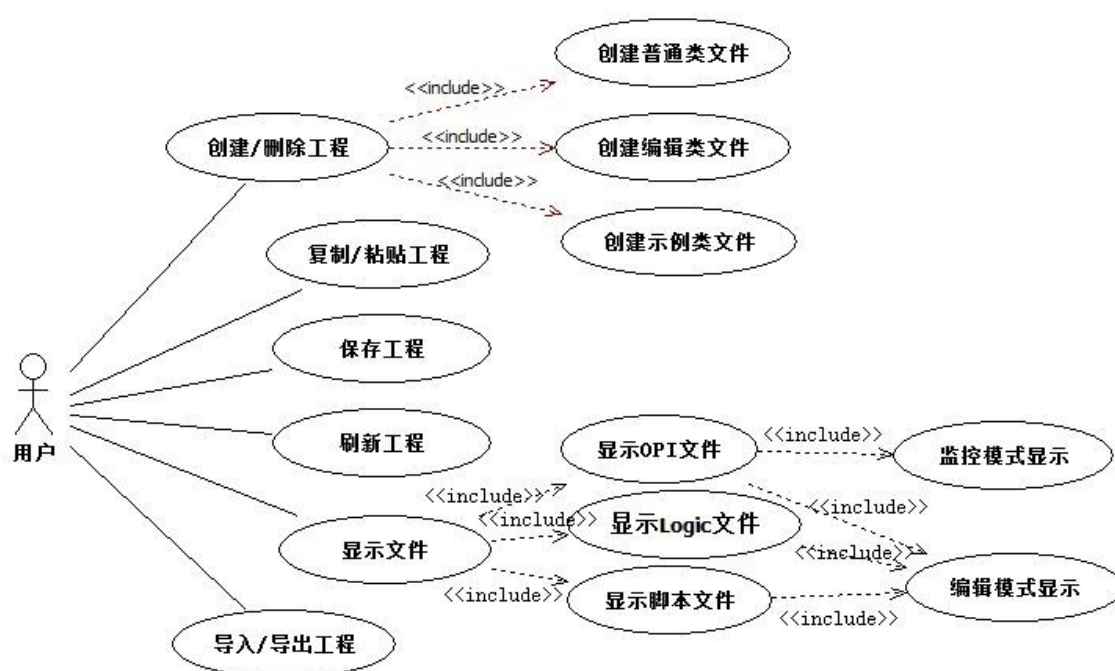


图 2-3 工程文件管理子系统用例图

#### （三）用例分析

1. 用户在编辑软件上实现创建、删除、复制、粘贴、保存、刷新工程文件；
2. 用户创建的工程文件包括有普通类文件、编辑类文件、示例类文件三种；
3. 用户显示 OPI 文件方式可以选择监控模式显示或编辑模式显示，Logic 文件与脚本文件只能以编辑模式显示；
4. 用户可以导入已存在的工程文件，也可以导出文件到指定位置；

## （四）模块详细需求描述

表 2-1 创建工程功能子模块需求描述


功能名称	创建工程
功能描述	用户在编辑软件中创建一个工程文件用于进行编辑工程
实现方式	<ol style="list-style-type: none"> <li>1. 用户在软件顶部菜单工具第一列文件（file）工具栏选择新建工程；或者在资源导航视图栏中空白处点击鼠标右键，弹出的工具栏第一行显示新建文件按钮（new）；</li> <li>2. 用户选择文件类型后以对话框形式展现创建步骤；</li> <li>3. 用户需选择创建文件的路径与自定义名称后才可成功创建；</li> <li>4. 用户新建的工程文件在用户选择的本地文件目录下被创建；</li> </ol>
图标设定	
子菜单项	工程文件包括：普通类文件（工程文件、文件夹、文本文件）、 图片编辑类文件（OPI 文件、Logic 文件、脚本文件）、 示例文件（BOY 示例、LOGIC 示例）；
备注	无

表 2-2 删除工程功能子模块需求描述


功能名称	删除工程
功能描述	用户在编辑软件的资源导航视图中删除已创建的文件
实现方式	<ol style="list-style-type: none"> <li>1. 用户在导航栏视图中选中某项工程文件，单击鼠标右键，在弹出的菜单中选中删除（delete）操作，即可删除选中的文件；</li> <li>2. 用户在导航栏视图中选中某项工程文件，顶部菜单工具第一列文件（file）工具栏选择删除（delete）选项，即可删除选中文件；</li> <li>3. 用户删除的文件即内容同时在创建的本地工程目录下删除；</li> <li>4. 用户删除时需弹出对话框再次询问是否确定删除；</li> </ol>
图标设定	
子菜单项	无
备注	快捷键操作：Ctrl+Delete

表 2-3 复制工程功能子模块需求描述


功能名称	复制工程
功能描述	用户在编辑软件中复制文件
实现方式	<ol style="list-style-type: none"> <li>1. 用户在导航栏视图中选中某项工程文件，单击鼠标右键，在弹出的菜单中选中复制（copy）操作，即可复制选中的文件；</li> <li>2. 用户在导航栏视图中选中某项工程文件，顶部菜单工具第一列文件（file）工具栏选择复制（copy）选项，即可复制选中文件；</li> <li>3. 用户可选择编辑面板中的文字或图形控件，在右键弹出的菜单中复制；</li> </ol>
图标设定	
子菜单项	无
备注	快捷键操作：Ctrl+C

表 2-4 粘贴工程功能子模块需求描述


功能名称	粘贴工程
功能描述	用户在编辑软件中粘贴文件
实现方式	<ol style="list-style-type: none"> <li>1. 用户在同一软件中的不同工程文件目录下可以粘贴已复制的文件，在不同的软件中复制的工程文件不能被粘贴在本软件的工程目录下；</li> <li>2. 用户在不同软件中粘贴的文本可以被粘贴在选择文本框或可编辑属性内容中；</li> <li>3. 用户粘贴的工程文件在选择的文件目录下被创建在本地文件夹中；</li> </ol>
图标设定	
子菜单项	无
备注	快捷键操作：Ctrl+V

表 2-5 保存工程功能子模块需求描述

功能名称	保存工程
功能描述	用户保存已创建编辑的文件及内容
实现方式	<ol style="list-style-type: none"> <li>1. 用户在导航栏视图中选中某项工程文件，单击鼠标右键，在弹出的菜单中选中保存（save）操作，即可保存选中的文件；</li> </ol>

表 2-5 保存工程功能子模块需求描述 &lt;序&gt;


实现方式	2. 用户在导航栏视图中选中某项工程文件，顶部菜单工具第一列文件（file）工具栏选择保存（save）选项，即可保存选中文件；
图标设定	
备注	快捷键操作：Ctrl+S

表 2-6 刷新工程功能子模块需求描述


功能名称	刷新工程
功能描述	用户在编辑软件的资源导航视图中刷新文件
实现方式	1. 用户在资源导航视图中空白处单击鼠标右键，在弹出的菜单中选择刷新（refresh），全部文件将被刷新； 2. 用户选中某项文件单击鼠标右键，在弹出的菜单中选择刷新（refresh），被选中文件将被刷新；
图标设定	
备注	快捷键操作：F5

表 2-7 显示工程功能子模块需求描述


功能名称	显示工程
功能描述	用户对已选中的文件选择显示方式显示内容
实现方式	1. OPI 类型文件有编辑模式与监控模式两种类型，编辑模式下用户可以在编辑面板中编辑图形各项属性，监控模式下文件根据关联的脚本内容与传递的 PV 值执行程序，用户不具有编辑的能力，只具有监控其变化的能力； 2. Logic 类型的文件只具有编辑模式，在其编辑面板中更改 PV 值关联的逻辑值会对关联了同一个 PV 值的监控模式下的 OPI 文件中的控件产生影响； 3. 脚本类型文件只具有编辑模式，编辑好的文件直接保存，在被关联的 OPI 文件的监控模式下运行时可以直接对其起作用；
图标设定	



表 2-7 显示工程功能子模块需求描述 &lt;序&gt;

子菜单项	显示方式包括：编辑模式显示（open with editor） 监控模式显示（open with runtime）
备注	快捷键操作：Ctrl+Shift+W

表 2-8 导入工程功能子模块需求描述



功能名称	导入工程
功能描述	用户可以将已存在的工程文件导入到
实现方式	1. 用户在资源导航视图空白处点击鼠标右键工具栏中选择导入（import）选项，在弹出的对话框中选择压缩文件与已存在文件； 2. 用户在“下一步”中选择导入文件的路径就可成功导入； 3. 用户在顶部菜单工具第一列文件（file）工具栏选择导入（import）选项，实现方式与第 1、2 步骤相同；
图标设定	
子菜单项	可导入的文件类型 包括： 压缩文件（Archive File）、已存在文件（Existing Project in works-pace）、 BOY 示例文件（BOY Example）、Logic 示例文件（Logic Example）；
备注	快捷键操作：Ctrl+I

表 2-9 导出工程功能子模块需求描述

功能名称	导出工程
功能描述	用户对已选中的工程文件导出到指定目录
实现方式	1. 用户在资源导航视图中选中工程文件，单击右键工具栏中选择导出（export）选项，在弹出的对话框中选择导出文件路径与工程文件包含的内容； 2. 用户在资源导航视图中空白处选择导出（export）操作，弹出的对话框显示全部资源工程文件，用户可以通过勾选框选择需要导出的文件以及文件路径； 3. 如果选择了某项工程，对话框的侧边栏显示工程包含的所有文件内容，用户可以自定义勾选导出内容；
图标设定	

## 2.2.2 图形编辑模块

### （一）功能描述

本模块实现了软件的核心功能——可视化图形编辑。在该功能模块中，用户首先创建一个 OPI 格式的文件，通过使用调色板中的图形控件在 OPI 文件的编辑面板中绘制司机、列车员监控屏幕页面即监控界面。图形控件分为基本图形、监控类图形、控制类图形和其他类图形，主要使用监控类与控制类图形控件。每一个控件都有对应的属性列表，用户可以通过编辑面板中鼠标拖拽或者选定控件在属性面板修改控件的属性。控件之间可以通过透视面板修改布局。

### （二）用例图

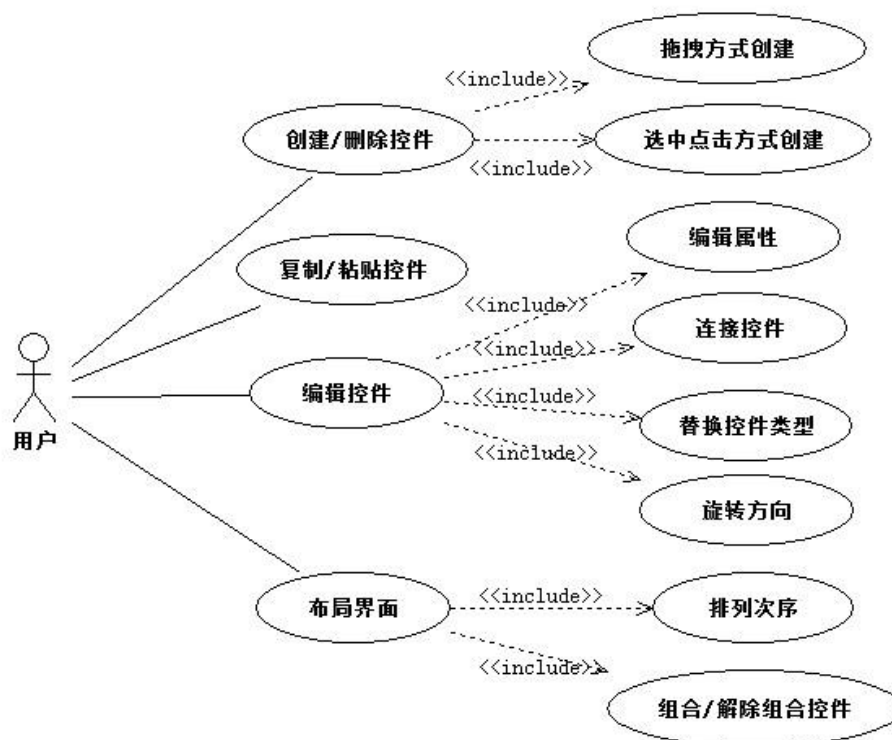


图 2-4 图形编辑子系统用例图

### （三）用例分析

1. 用户可以通过从调色板拖拽方式或者选中后单击鼠标的方式创建控件；
2. 用户可以复制和粘贴编辑面板中的图形控件；
3. 用户可以在编辑面板中用鼠标拖拽或属性列表中点击属性框编辑控件的属性，
4. 用户使用连接工具可以连接两个或多个控件；

5. 用户点击右键弹出菜单可以选择替换当前控件类型为其他类型，选择旋转方向和选择度数（顺时针 90 度、顺时针 180 度、逆时针 90 度、逆时针 180 度）；
6. 用户可以在编辑面板中右键菜单选择排列次序，选中多个控件选择组合控件或者解除组合，也可以通过视图面板布局控件的排列次序与组合控件；

#### （四）模块详细需求描述

表 2-10 创建控件功能子模块需求描述

功能名称	创建控件
功能描述	用户在编辑面板中创建图形控件
实现方式	<p>1. 在软件中央最大面积部分为编辑面板，编辑面板右侧放置可伸缩调色板视图，当用户在面板中操作时，调色板自动隐藏，当鼠标放置在右侧感应区时，调色板自动展开；</p> <p>2. 用户在调色板中选中控件，在编辑面板单击鼠标即可创建；或者在调色板中选中，通过鼠标拖拽方式创建控件；</p> <p>3. 创建的控件都以默认大小显示，创建后用户可以自行定义其属性；</p> <p>4. OPI 文件与 Logic 文件都具有 4 类控件，但两类文件只能共用普通类控件与其他类型控件，不能共用监控类与控制类控件；</p>
图标设定	无
分类说明	<p>图形控件类型 包括：</p> <p>普通类控件（Graphics）——直线、带箭头连接线、自定义曲线、矩形、椭圆形、自定义图形、标签、图片</p> <p>监控类控件（Monitor）——可更新文本框、LED 监控指示灯、进度条（OPI）</p> <p>电路板、传感器、示波器（Logic）</p> <p>控制类控件（Control）——按钮、多选按钮、可输入文本框、复选框（OPI）</p> <p>开关、或门、与门、非门（Logic）</p> <p>其他类控件（Others）——表格、组合容器、标签容器、树形容器、表格形容器</p>
备注	只有 OPI 文件与 Logic 文件工程显示调色板，其他类型不显示

表 2-11 删除控件功能子模块需求描述


功能名称	删除控件
功能描述	用户可以删除已创建的控件
实现方式	1. 用户在创建图形控件后可以选中控件，单击鼠标右键，在弹出的菜单中选择删除选项，控件就可以在编辑面板中删除；
图标设定	
分类说明	无
备注	快捷键操作：Ctrl+Delete

表 2-12 复制控件功能子模块需求描述


功能名称	复制控件
功能描述	用户可以复制已创建的控件
实现方式	1. 用户在编辑面板中选中图形控件，单击鼠标右键，在弹出的菜单中选择复制(copy)选项，控件就可以被复制；
图标设定	
分类说明	无
备注	快捷键操作：Ctrl+C

表 2-13 粘贴控件功能子模块需求描述


功能名称	粘贴控件
功能描述	用户可以粘贴被复制的图形控件
实现方式	1. 用户在编辑面板中单击鼠标右键，在弹出的菜单中选择粘贴(paste)选项； 2. 同一个工程同一类型的文件中允许粘贴控件，不同类型的文件之间不允许复制粘贴控件；
图标设定	
分类说明	OPI 文件与 Logic 文件中只能共用普通类控件与其他类控件，不能共用监控类控件与控制类控件；
备注	快捷键操作：Ctrl+V

表 2-14 编辑控件功能子模块需求描述

功能名称	编辑控件
功能描述	用户在编辑面板中编辑图形界面
实现方式	<ol style="list-style-type: none"> <li>1. 用户通过调色板在编辑面板中创建控件，控件的属性在面板右侧的属性列表视图中显示；</li> <li>2. 控件之间可以通过带箭头的连接线传递数据值，带箭头的连接线选中后，一次点击为数据源，第二次点击箭头指向数据靶控件，之间形成一条连接线；</li> <li>3. 控件可以在同一类型之间互相替换，用户选中控件点击鼠标右键，在弹出的菜单中选择替换控件类型，可替换的控件类型显示为正常颜色，不可替换的控件类型显示为灰色不可选，替换之后的控件显示当前控件类型的属性值；</li> <li>4. 用户点击右键弹出菜单可以选择替换当前控件类型为其他类型，选择旋转方向和选择度数；</li> </ol>
分类说明	<p>控件的属性包括：</p> <p>基本属性：名称、控件类型、PV 值</p> <p>行为属性：执行动作、是否可见、是否可被编辑、关联的脚本</p> <p>边框属性：边框颜色、边框类型、边框宽度</p> <p>显示属性：背景颜色、前景颜色、字体提示标</p> <p>位置属性：伸缩量、长度、宽度、X 坐标、Y 坐标</p> <p>旋转方向和度数 包括：顺时针 90 度、顺时针 180 度、逆时针 90 度、逆时针 180 度</p>
备注	无

表 2-15 布局控件功能子模块需求描述

功能名称	布局控件
功能描述	用户对多个控件可以布局操作
实现方式	<ol style="list-style-type: none"> <li>1. 在软件隐藏视图中有布局透视图，布局透视图显示控件的组合布局情况；</li> <li>2. 选中 2 个或以上的控件可以在右键菜单中选择组合控件，组合好的控件为一个整体，同时移动，变换同一大小，也可以选择解除组合；</li> <li>3. 在布局透视图选择控件可以组合/解除组合；</li> </ol>

表 2-15 布局控件功能子模块需求描述 <序>

实现方式	4. 在布局界面时使用其他类控件中的容器类控件可以按照容器的布局方式排列； 控件的排列次序根据创建先后顺序确定，先创建先排序原则；用户可以通过鼠标右键菜单选择排列次序；
分类说明	布局方式排列 包括：简单无序组合、标签组合、树形排列、表格排列 排列次序选择 包括：前置一位、置为最前、后置一位、置为最后
备注	无

2.2.3 数据通信模块

（一）功能描述

监控屏幕的显示内容需要与仿真机建立数据通讯机制。该模块就是在监控界面（OPI 文件）搭建时给监控控件绑定数据变量（PV Name）。在仿真机模型通过数据通信中心发送给服务器（Zookeeper）的监控数据发生变化时，Zookeeper 服务器会将数据内容通过管理平台（PV Manager）发送给前端界面，显示给用户具体的故障信息。

（二）用例图

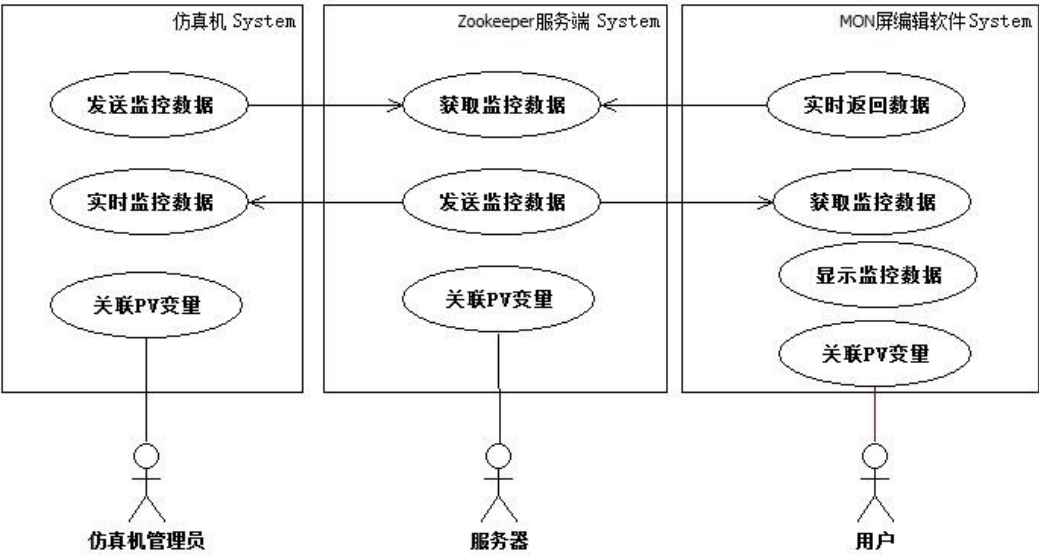


图 2-5 数据通信子系统用例图

（三）用例分析

- 1. 仿真机机器部件关联 PV 变量，并能够发送监控变量、实时获取监控数据；
- 2. Zookeeper 服务器接收仿真机监控数据，通过 PV 变量发送数据给编辑界面端；

3. 用户在编辑软件的属性列表里为图形控件关联对应的 PV 变量；
4. 编辑软件通过 PV 变量接收与返回实时数据，并将监控数据显示在屏幕上；

（四）模块详细需求描述

表 2-16 数据通信功能模块需求描述

功能名称	数据通信
功能描述	列车的运行状态通过 PV 变量显示在界面上
实现方式	1. 用户编辑界面时，需要显示监控数据的控件在属性列表中关联对应的 PV 变量； 2. 列车在运行状态时，司控台会接收列车的运行数据并发送给编辑软件，同样在监控状态的编辑软件，通过关联的 PV 变量获取监控数据并显示在界面上； 3. PV 变量编辑形式为：服务器地址：// 变量名（zoo://parameter_name）
备注	被关联的 PV 值需要在 Zookeeper 服务器中注册完成才可通信

2.2.4 脚本编辑关联模块

（一）功能描述

该模块实现了用户通过创建脚本文件控制 OPI 文件的显示。创建的脚本文件格式可以是 JavaScript 或者 Python 两种，两种格式只是书写语法不同，作用相同。脚本文件通过编辑面板的属性列表进行关联，监控数据发生变化关联控件的属性做出映射变化。

（二）用例图

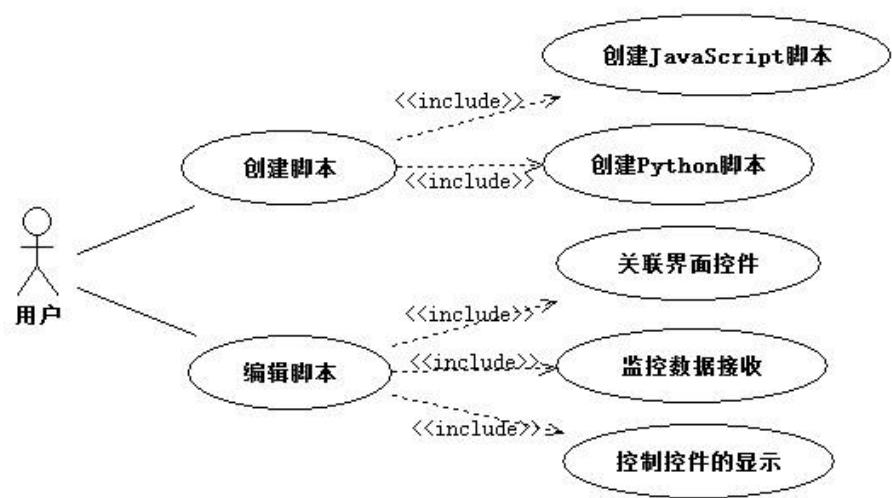


图 2-6 脚本编辑关联子系统用例图

### （三）用例分析

1. 用户可以创建并编辑两种类型的脚本文件：JavaScript 类型、Python 类型；
2. 用户在控件或页面的属性列表中设定关联的脚本文件；
3. 用户可以通过脚本接收监控数据，以及控制关联控件的属性显示；

### （四）模块详细需求描述

表 2-17 创建脚本功能子模块需求描述

功能名称	创建脚本
功能描述	用户可以创建脚本文件控制界面的显示
实现方式	1. 用户通过单击鼠标右键，在弹出的菜单中选择新建工程（new），新建工程类别中选择脚本文件； 2. 脚本文件创建成功后编辑面板显示可编辑的文本面板，用户就可以在面板中编辑脚本文件内容了；
分类说明	脚本文件类型 包括： JavaScript 脚本、Python 脚本
备注	无

表 2-18 编辑脚本功能子模块需求描述

功能名称	编辑脚本
功能描述	用户可在编辑面板编辑脚本程序
实现方式	1. 用户编辑脚本文件的语法与 js / python 语法一致； 2. 用户创建的脚本文件在编辑面板中编写执行程序，程序可以获取所关联界面中的控件并通过 PV 值设定用户需要的控件变化形式； 3. 脚本文件需要通过界面控件的脚本（script）属性关联，点击脚本属性，弹出的对话框中显示全部工程文件；用户可以选择工程中的脚本并选择触发脚本的 PV 值； 4. 被关联的脚本文件在 PV 值发生变化时接收数据并触发脚本文件的执行； 5. 监控状态执行的脚本文件为保存成功的脚本文件；
分类说明	无
备注	两类脚本编写时需要引入的类包可以自定义编写并添加在 plugin 文件包下引用



## 2.3 系统非功能性需求分析

### 2.3.1 性能

表 2-19 系统性能需求

系统非功能性指标	性能需求描述
响应速度	从仿真机模型发送监控数据到软件接收，一条消息响应时间不得大于 0.5 毫秒（Zookeeper 服务器对发送与接收数据时间不得大于 0.1 毫秒）；
吞吐量	单位时间内软件处理的监控数据量大于等于 200 条；
超时错误率	处理事务超过 5 秒的错误率要低于 5%；
资源利用率	CPU 使用率：软件进程与系统进程消耗的 CPU 时间百分比可接受上限不超过 85%； 内存利用率：至少有 10%可用内存，内存使用率可接受上限为 85%；

### 2.3.2 易用性

表 2-20 系统易用性需求

系统给功能性指标	易用性需求描述
易理解性	1. 考虑到使用者为铁路相关人员，使用手册可以使用专业术语，但对功能的解释要清楚易于理解； 2. 软件的图形控件设计需要尽量符合实体，易于用户判断和使用，不能过于抽象化；
易学习性	用户文档提供给用户学习软件要结构清晰、步骤完整、功能叙述详细，达到管理人员经过一周培训即可使用软件；
易操作性	1. 软件的设计需要为每一个控件添加 Tooltip 控件提示标签，每一个控件的使用都直接明了、没有歧义； 2. 软件设计的操作流程应符合铁路工作的通用操作流程； 3. 软件应合理安排编辑面板的视图，默认打开用户最常用的视图面板，软件首次打开需恢复到上一次关闭软件时的界面状态；

### 2.3.3 可移植性

表 2-21 系统可移植性需求

系统非功能性指标	可移植性需求描述
适应性	软件无需采用特殊的环境准备就可适应目标环境的配置；
易安装性	软件的安装自带 Java 环境包，不需要在每一台使用的机器上都安装使用环境，解压软件包即可以直接运行；
兼容性	软件需能够兼容 windows7/8、Linux 系统具备一般的兼容性；
易替换性	软件比较难被同类画图软件替换使用，具有较强的专业隔离性；

### 2.3.4 可维护性

表 2-22 系统可维护性需求

系统非功能性指标	可维护性需求描述
易分析性	软件需要易于诊断功能或逻辑缺陷，或者判断失效原因或待修改的部分，以做出相应的调整；
易改变性	软件易于根据测试错误或软件缺陷，或者环境进行改变，但不改变软件的基本框架和基础功能实现；
稳定性	软件结构清晰，功能明确，需具有较强的稳定性；
易测试性	根据模块的功能划分，软件各功能之间具有独立性，易于测试用例和自动化测试使用；

### 3 .MON 屏编辑软件架构设计

#### 3.1 系统体系架构设计

MON 屏编辑软件是一款搭建在 Eclipse RCP (Rich Client Platform) 富客户端开发平台上基于 GEF (Graphical Editing Framework) 图形编辑框架设计的可视化图形编辑桌面应用程序<sup>[8]</sup>。RCP 提供了可移动、可叠加的窗口组件以及菜单栏、工具栏、按钮、表格基本组件的支持。建立在 Eclipse 提供的强大的开放平台上, 该项目利用其底层支持方便地将开发的插件注册在平台上运行<sup>[3]</sup>。插件 Navigator 资源导航视图实现了 Eclipse 对软件的接口将应用程序中创建的工程文件进行管理的功能注册在 RCP 平台上得以使用。图形编辑插件基于 GEF 图形编辑框架, 创建编辑图形所需的编辑器面板和控制调色板。以 MVC 模式创建的 GEF 图形编辑框架根据需求设计四大类型的图形控件 Model 业务模型, 提供通知机制给 Control 控制层。Control 控制层主要由 EditPart 实现调用策略方法为 View 视图层发送 Command 操作消息命令完成模型-控制器-视图的合理配置。图形控件的图形绘制利用 Draw2D 技术完成。而完成 RCP 底层工作台绘制的是 SWT, 将 Draw2D 中的图形控件与 SWT 连接起来的是 LightweightSystem (轻量级控件) 注册模式。[5] 插件 PV Manager 与插件 Jython 分别以 Eclipse 支持的 PLUGIN 插件方式注册在 RCP 平台上完成数据通信与脚本编辑关联功能的使用。

下面将对其中核心的应用技术进行详细阐述。

#### 3.2 核心技术点介绍

##### 3.2.1 SWT/JFace 绘制基本窗口部件

SWT(Standard Widget Toolkit)是 Eclipse 窗口小部件工具箱, 就像一个工具库一样创建了基于本机操作系统的 Java 编码方式的 GUI(Graphical user interface)用户界面控件。不同于 SWING, 它的 JFace 外观、行为与操作都是本机操作系统的反映<sup>[12]</sup>。用来在 SWT 库顶部提供常见的应用程序用户界面功能, 它对 SWT 进行了扩展, 把常用的组件进行了封装。SWT/JFace 是 Eclipse 的根基, Eclipse 的工作台 Workbench 就是建立在 SWT/JFace 之上的, 是整个编辑软件的平台基础。

### 3.2.2 Eclipse RCP 提供基本工作台功能

Eclipse 是一个基于 Java 的源码开放的可扩展开发平台，其中包括 Eclipse Platform、Eclipse JDT(Java development tooling)、Eclipse CDT(Java development tooling)和 Eclipse PDE( Plug-in development environment )四个组成部分<sup>[2]</sup>。Eclipse Platform 为开发者提供了通用开放的可扩展平台，其中 JDT 支持 Java 开发，CDT 支持 C 开发，PDE 支持插件开发。Eclipse RCP 的全称是 Rich Client Platform（富客户端平台），为桌面应用程序开发人员提供了一个基本的工作台。这个工作台拥有与本地一致的外观特性，窗口管理、更新管理、帮助与选择等一系列的服务管理，标准化的组件模型与 Workbench 工作台，以及 Runtime 运行平台<sup>[3]</sup>。建立在 Eclipse RCP 上的桌面应用程序，减少了对于界面图形的修饰，能够使开发人员的注意力能够更多得放到系统逻辑实现上。

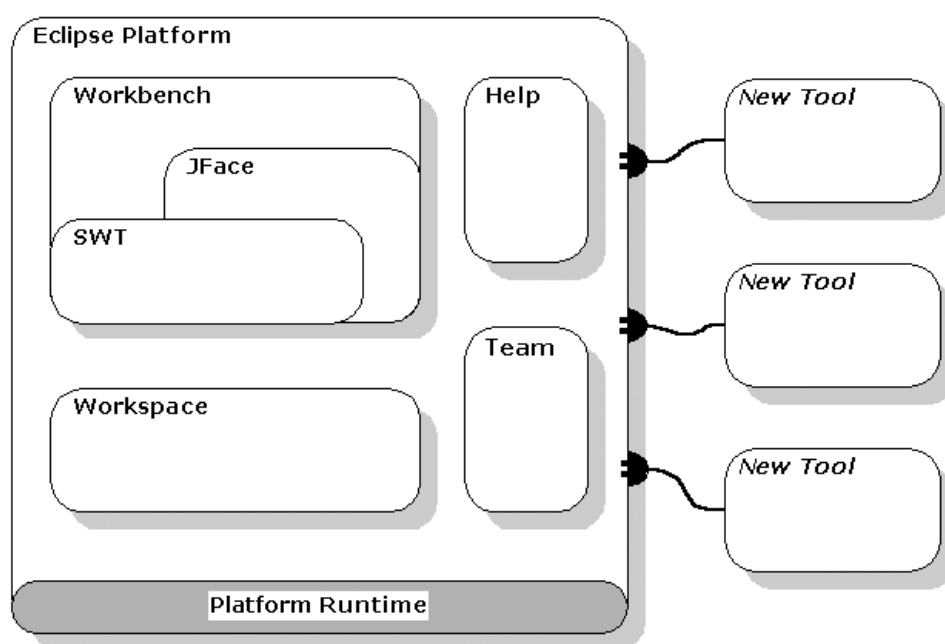


图 3-1 Eclipse RCP 基本工作台架构示意图

### 3.2.3 高内聚低耦合的 PLUG-IN 插件

脱去 RCP 骨架的 Eclipse 拥有的丰富功能是由数以千计的插件程序完成的。如上文的图 3.1 所示，一个个像插头形状插入到 RCP 工作台上的 New Tool 组件就是插件程序。它遵循了 RCP 对于应用程序的接口规范，必须依赖于主程序运行的程序<sup>[4]</sup>。就像 WEB 浏览器安装插件程序处理如 flash、pdf 等特定类型文件一样，本项目的插件程序实现图

形编辑、工程文件管理、数据通信和脚本编写功能都是通过插件实现的。插件技术为项目的设计开发、产品功能扩展、设计软件模式等拥有很多优势：

- 1.结构清晰、易于理解；各插件之间相互独立，互不干扰，项目结构清晰易懂；
- 2.可维护、可扩展性强；功能的增加与修改仅在插件内部调整，方便软件功能扩展；
- 3.高内聚、低耦合；插件与宿主程序通信而与其他模块无关，典型的高内聚低耦合；

### 3.2.4 GEF 图形编辑框架实现核心功能

GEF（Graphical Editing Framework）图形编辑框架被设计用来以图形而不是文本的方式来编辑模型数据，例如数据库 schema 编辑器、UML 建模工具和工程作业设计工具，都很好的展示了可视化图形编辑器的强大功能<sup>[7]</sup>。它的这些功能需要模块化的结构，选用合理的设计模式和独立于控制端的组件才能够构成完整的编辑框架。它为用户提供了一个编辑区域、一个大纲视图区域与显示树状模型结构区域，提供了一个工具条、创建图形控件、查看修改控件属性、连接节点的功能，所有的模型控件都能够被创建、移动、编辑与删除。这些良好的操作得益于 GEF 遵循了标准 MVC(Model-View-Control)模式。与其他以 MVC 为框架的系统相比，GEF 主要的设计目的是尽量减少模型和视图之间的依赖，提供不受框架局限的模型与视图。一个完整的 GEF 框架一般由以下几种类组成：数据模型 Model 类、控制部件 EditPart 类、控制策略 Policy 类、执行命令 Command 类、展示图形 Figure 类。

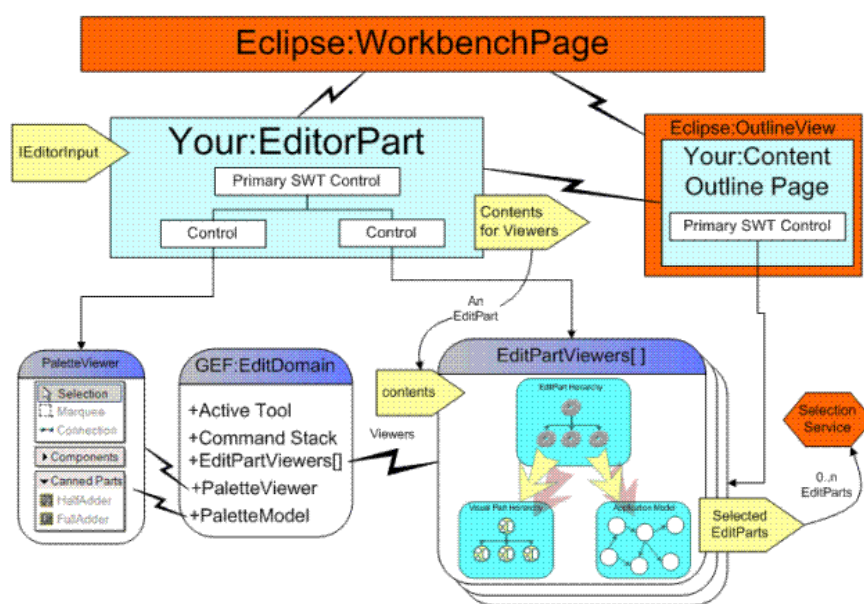


图 3-2 GEF 框架结构示意图

### 3.2.5 MVC 模式详解

GEF 是具有标准 MVC（Model-View-Control）结构的图形编辑框架<sup>[6]</sup>。根据项目的业务需求定义 Model 模型，其中提供了模型控制器事件监听机制将模型的改变通知 Control 控制层。Control 控制层是模型与视图连接的桥梁，由 EditPart 类实现监听模型并把结果发送给 View 视图层显示的作用。View 层是模型以可视化图形方式显示实现。

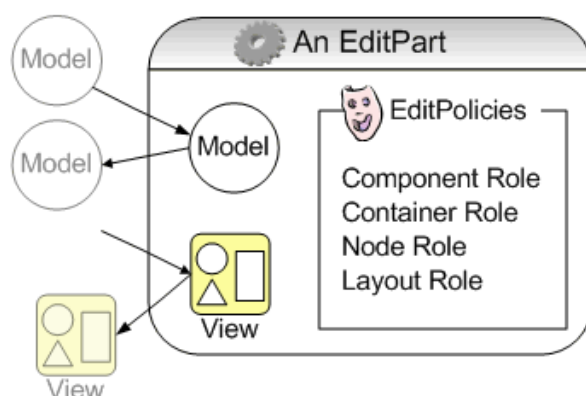


图 3-3 MVC 模式

**Model-模型：**不直接与 View 建立联系的任意一个 Java 对象。定义图形控件的属性作为成员变量，运用 PropertyChangeSupport 类型的成员变量作为控制器监听成员。

**Control-控制器：**作为模型与视图的桥梁，是整个框架的核心部分。一个模型对应的一个 EditPart 对象统一由 EditPartFactory 工厂创建出来，操作一个 EditPolicy 策略模式。EditPolicy 接收一系列称作 Role（角色）的用户编辑操作并调用 Command（命令）直接操作模型对象。

**View-视图层：**利用 Draw2D 图形包（IFigure 接口）创建可视化的图形控件，其中 GraphicalEditViewer 和 PaletteViewer 这两个视图类，分别实现编辑面板和调色面板的绘制与显示。

## 3.3 开发环境

操作系统：Windows 7

开发平台：Eclipse MARS.1

开发环境：Plugin development environment

服务器平台：Zookeeper

开发进程管理：Microsoft Project 2007

## 4. MON 屏编辑软件功能模块设计与实现

### 4.1 工程管理模块

#### 4.1.1 设计描述

工程管理模块提供给用户操作工程文件的导航视图。

该模块将创建一个 navigator 资源导航视图插件，实现 CNF（Common Navigator Framework）资源视图框架。模块设计为在 RCP 平台左侧加入资源导航栏视图，内容的显示以树状形式图标加题目的形式显示。右键弹出式菜单提供新建、删除、复制、粘贴、剪贴、刷新和导入导出功能。新建文件类型以分隔符分类为普通类、图形编辑类和实例类文件。文件的打开模式根据文件类型确定。若是 OPI 文件则提供编辑模式打开和监控模式打开两种模式。其他类型文件直接以编辑模式打开。导入文件功能重点实现导入 BOY Example 和 LogicExample。导出文件需要选择导出格式和内容来导出到指定位置。

最后，导航视图作为插件注册到 RCP 工作平台上实现的这些功能。

#### 4.1.2 时序图

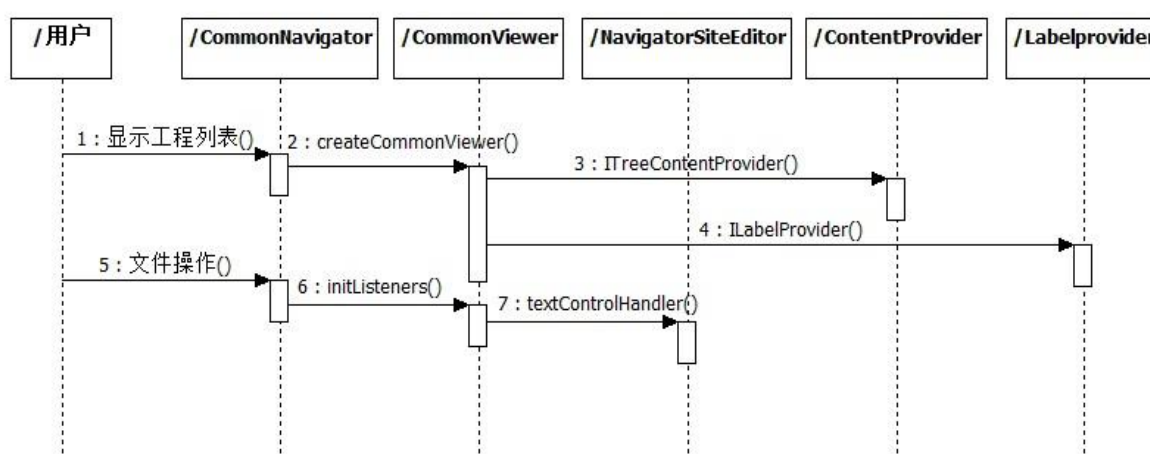


图 4-1 工程管理模块实现时序图

时序图说明：

用户在编辑软件工作台左侧的资源导航视图中会看到所有创建的工程文件的列表，由 CommonNavigator 类调用 createCommonViewer()方法通知消息给 CommonViewer 类，CommonViewer 类调用 ITreeContentProvider 与 ILabelProvider 通知 ContentProvider 与 LabelProvider 类为视图添加文件名与图标内容；

用户在点击右键触发操作相应时，CommonNavigator 类的 initListeners()方法监听到事件变化通知 NavigatorSiteEditor 类，执行 textControlHandler()方法操作相应的动作。

#### 4.1.3 核心类的结构设计

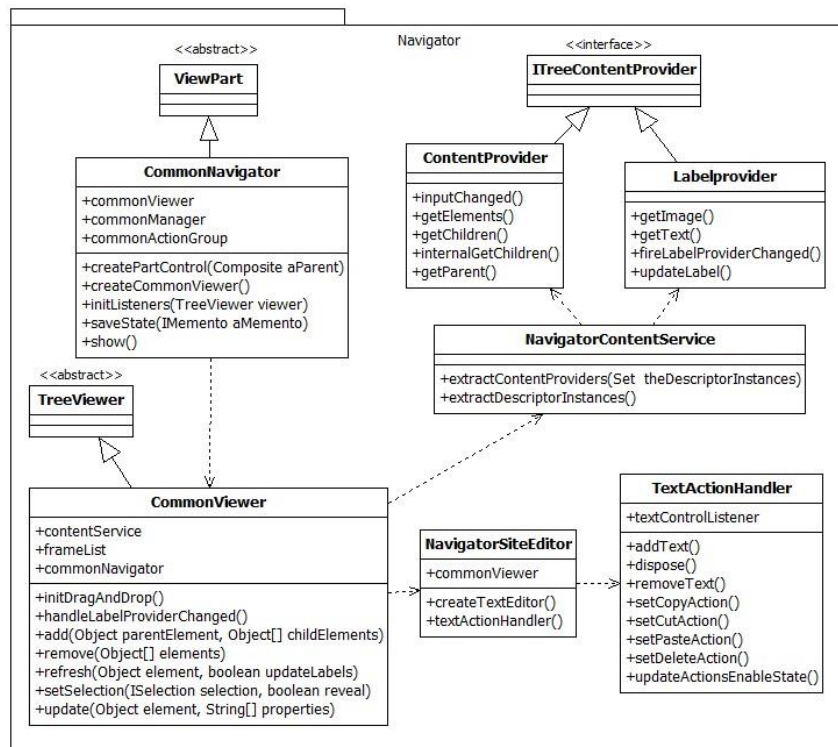


图 4-2 工程管理模块核心类设计图

类结构说明：

Navigator 是 CNF 框架的直接表现形式，主要由四大部分组成：

**View Part:** 定义为整个导航资源视图的控制中心，负责视图内容的调用显示和事件监听并控制操作的执行。它装载了控制器的核心类 CommonNavigator，该类继承了 Eclipse 提供的 ViewPart（定义在 org.eclipse.ui.part 包）基本抽象类使其作为 Eclipse 的视图扩展点显示在工作台。createPartControl()方法调用 createCommonViewer()方法创建视



图显示器，根据默认文件夹下创建的工程文件形成工程列表到结构列表中，并且添加事件监听器 `initListeners()` 方法监听用户操作行为以即时作出响应；`NavigatorSiteEditor()` 方法是实现了模块要求的文件操作功能，`textControlListener()` 是监听用户操作的监听器，`createActionHandler()`、`copyActionHandler()`、`pasteActionHandler()`、`cutActionHandler()`、`deleteActionHandler()`、`updateActionHandler()` 分别是对文件的新建、复制、粘贴、剪切、删除、刷新功能的实现；

**Viewer:** 装载了 `CommonViewer` 类，该类继承了 Eclipse 提供的 `TreeView` (源自于 `org.eclipse.jface.viewers` 包) 抽象类基类。类初始化时调用 `NavigatorContentService` 变量获取 `ITreeContentProvider` 与 `ILabelProvider` 提供文件标题与图标内容。从 `CommonNavigator` 中获取的工程列表通过 `createFrameList()` 方法将文件以树形结构显示在导航视图中。其中 `handleLabelProviderChanged()` 方法对文件标签内容改变做出响应，`update(Object, String[])` 方法更新文件的变化；

**Navigator Content Extensions:** 是资源模型与 `Viewer` 联系起来的桥梁，包含 `ContentProvider` 和 `Labelprovider` 两大类。前者是提供内容容器的类，实现 `ITreeContentProvider` 接口，`getElements()`、`getChildren()`、`internalGetChildren()` 方法获取资源文件中的内容；后者是提供标签显示的类 `getText()`、`getImage()` 等方法获取导航视图中所显示的文件名称和图标等内容，`updateLabel()` 方法更新视图显示的内容。

**Resource Support:** 提供了多种资源的支持。由于项目没有用到架构中的此部分，因此不做知识性的解释，有兴趣的读者可以自行学习。

#### 4.1.4 实现效果图

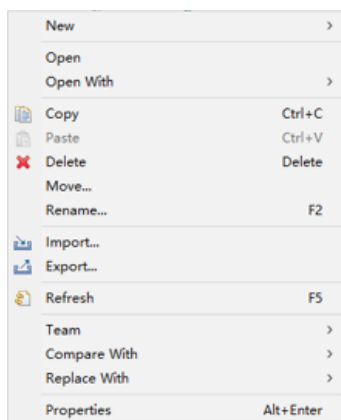


图 4-3 工程管理模块右键菜单栏实现效果图

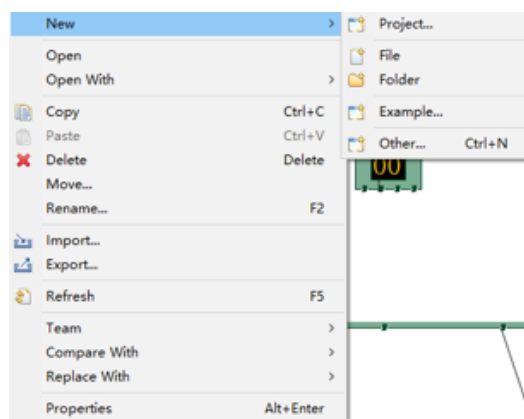


图 4-4 工程管理模块新建工程实现效果图

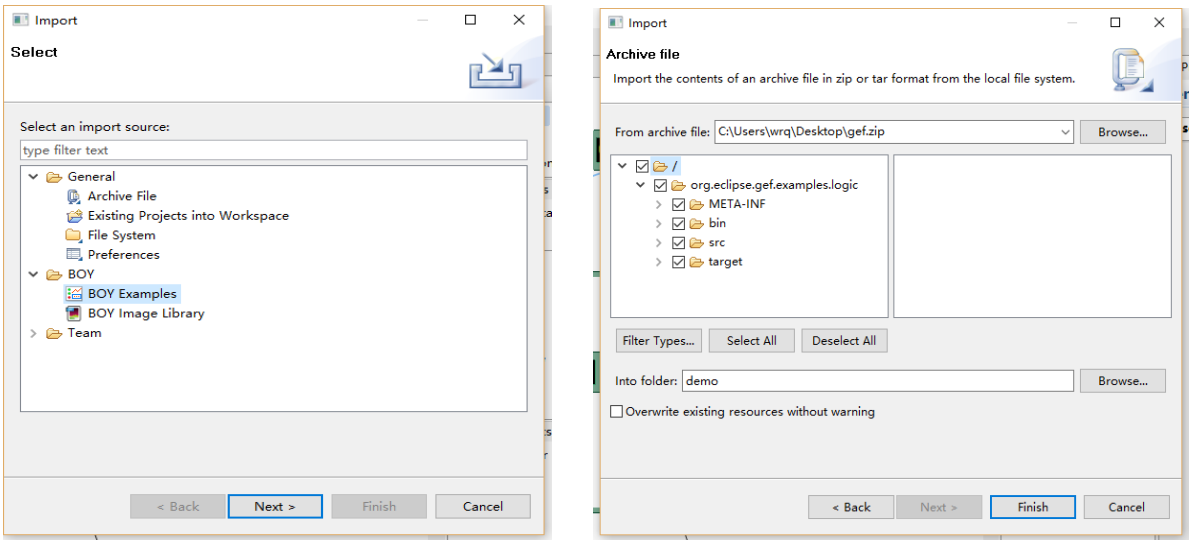


图 4-5 工程管理模块导入工程实现效果图

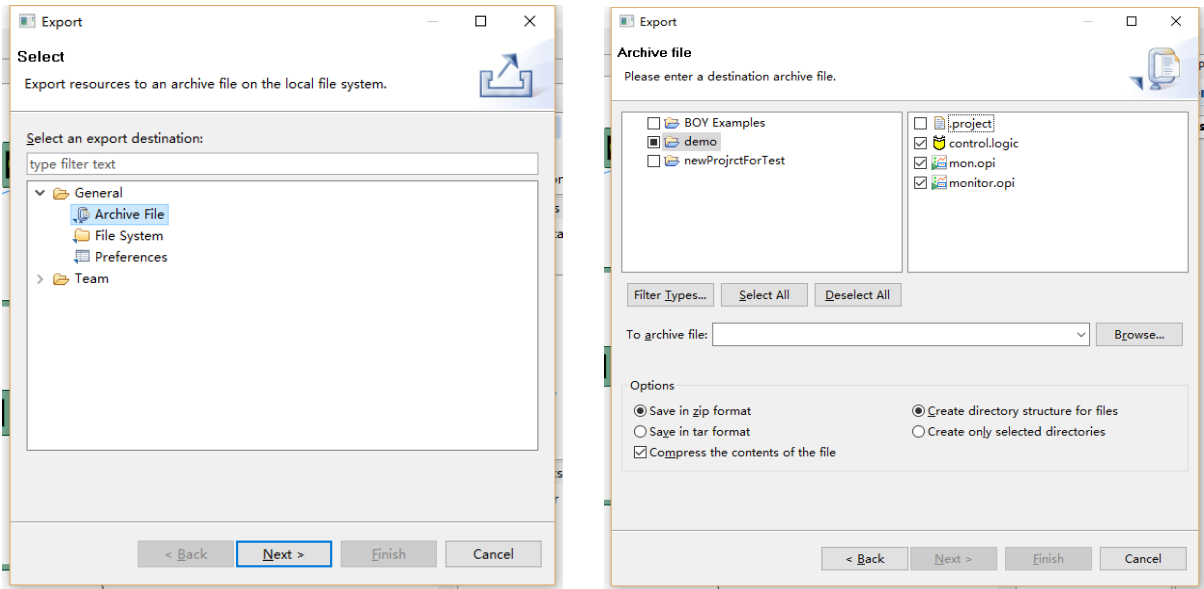


图 4-6 工程管理模块导出工程实现效果图

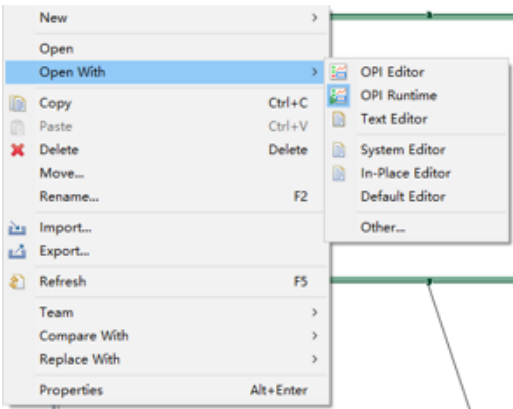


图 4-5 工程管理模块显示

## 4.2 图形编辑模块

### 4.2.1 设计描述

图形编辑模块是整个项目的核心模块，承担起实现用户主要需求的任务。

该模块实现用户通过在编辑器面板中选择调色板内的类型控件创建图形，并且可以对图形的名字、大小、颜色等一系列的属性编辑，属性列表显示控件属性的变化。控件的创建利用模型来实现，控件绘制的形状用视图中的形状类实现，用户对于图形的操作由控制类实现并且通过控制类通知模型和视图两端的变化。基于 GEF 框架的设计理念，该项目的设计实现将实现 MVC 设计模式，从五个方面实现，分别是：Model（模型类实现），Figure（控件图形类实现），EditPart（控制类实现），EditPartView（图形编辑类实现），Action（操作类实现）五大类包。

### 4.2.2 时序图

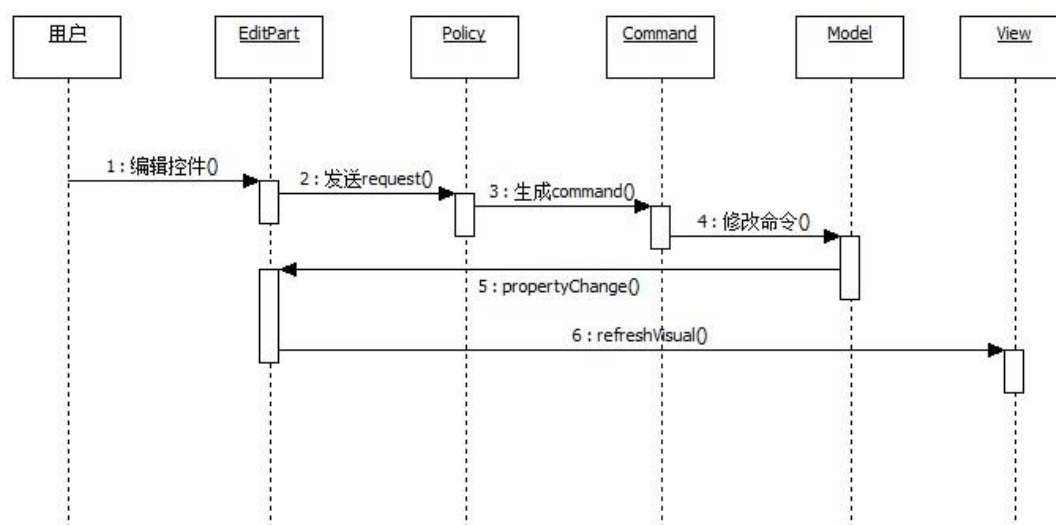


图 4-7 图形编辑模块时序图

时序图说明：

用户在编辑工作面板的图形时，可以拖拽调色板提供的模型，包括基本类图形控件、选择型、按钮型、布尔型图形控件。当用户在编辑区用鼠标拖拽或属性列表更改属性内容时，控件所对应的 EditPart 控制器会监控到用户的编辑行为并发送更改请求 request 给相应的 Policy 策略类，选择操作对应的 Command 命令方法对图形控件的 Model 模型

执行修改语句。模型的修改会触发控制器的 `propertyChange()` 方法，通过 `refreshVisual()` 更新视图方法控制视图显示修改后的内容。

### 4.2.3 核心类的结构设计

#### 1) MODEL 类

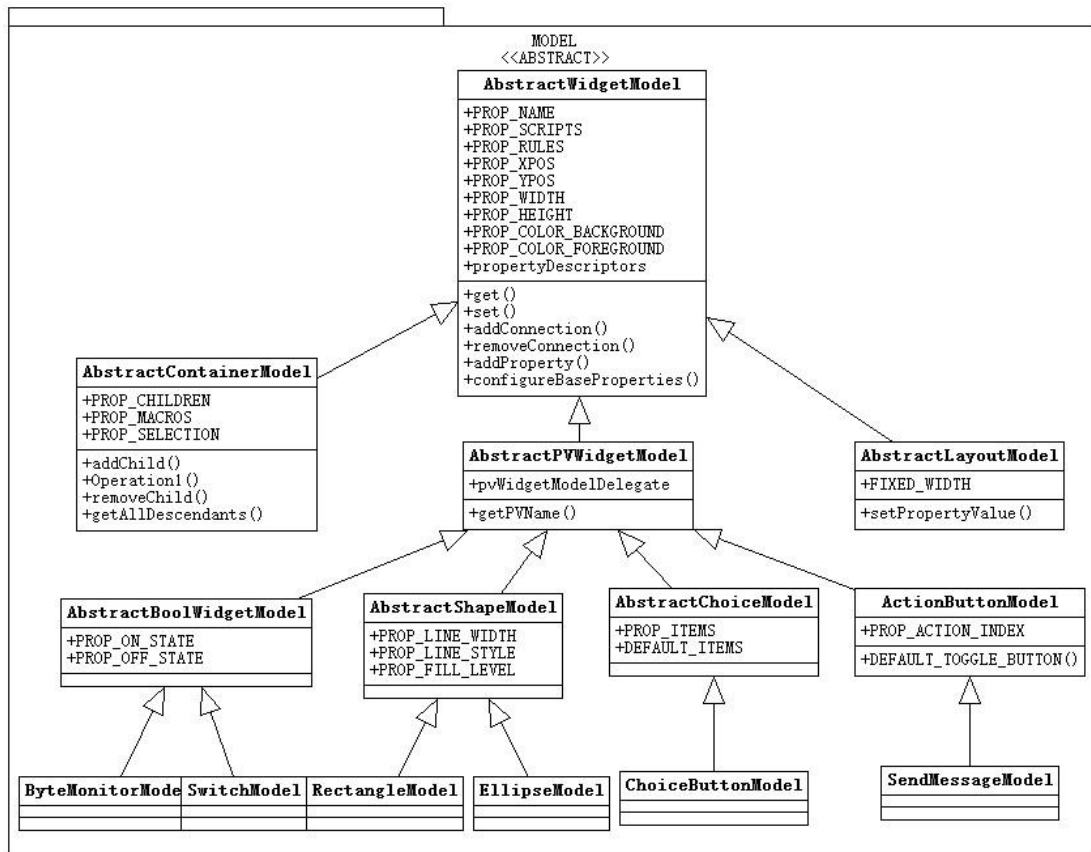


图 4-8 图形编辑模块 MODEL 类图

类图说明：

Model 类实现了对所有控件对象的创建工作，AbstractWidgetModel 抽象类定义了图形控件拥有的基本属性与功能，它实现 IPropertySource 接口使继承它的所有子类都可以将属性内容显示在侧栏的 Properties 属性列表中。AbstractContainerModel，AbstractLayoutModel，AbstractPVWidgetModel 分别继承了 AbstractWidgetModel 父类，实现了容器抽象类、布局抽象类和允许关联 PV 值的控件抽象类。在项目创建的所有控件中，最多的是继承父类：AbstractPVWidgetModel，其添加了 PV Name 这一特殊的属性值，使得控件将会实现下一模块所论述的数据通讯功能。AbstractBoolWidgetModel、

AbstractShapeModel、AbstractChoiceModel、ActionButtonModel 分别实现了构建布尔型、基本图形、选择型、响应按钮型抽象类，具体的控件模型，例如：ByteMonitorModel、SwitchModel、RectangleModel、EllipseModel、ChoiceButtonModel、SendMessageModel 分别继承相对应的抽象模型，添加属性与功能实现模型对象的构建。

2) FIGURE 类

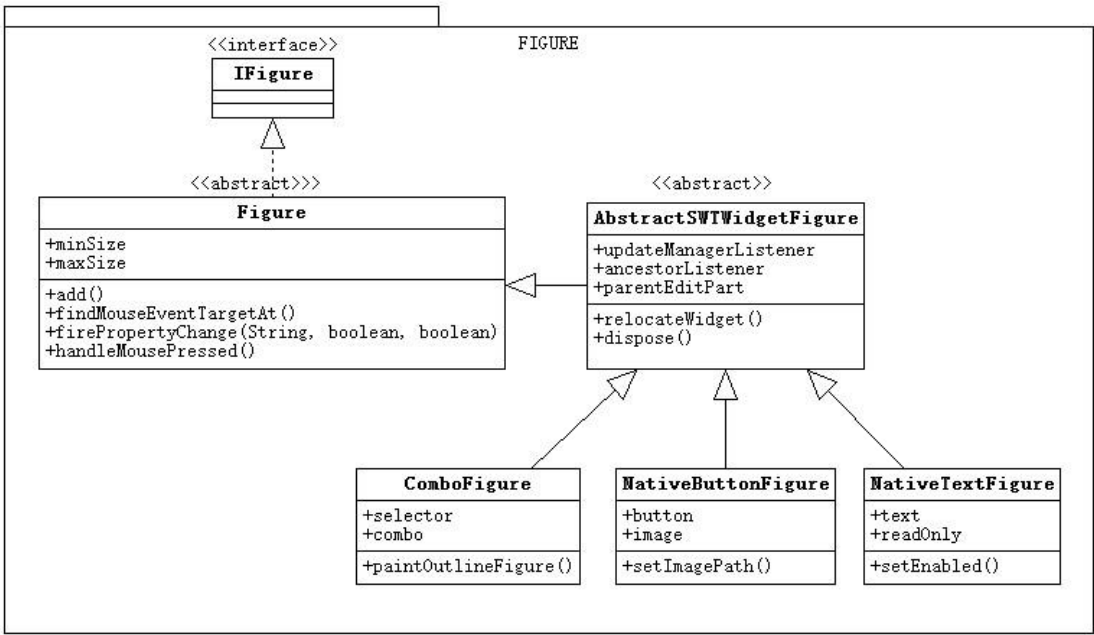


图 4-9 图形编辑模块 FIGURE 类图

类图说明：

Figure 类是 Eclipse Draw2D 类库中自带的图形绘画类，它实现了 IFigure 接口的属性和方法，构建了一个基本的图形类供开发者使用。AbstractSWTWidgetFigure 作为项目所有视图的父类继承了 Figure 基类，是控件显示在编辑区域时拥有可编辑、回显、工具提示等基本功能。ComboFigure, NativeButtonFigure, NativeTextFigure 分别实现了下拉菜单、按钮、文本类型控件的特殊方法实现。

### 3) EDITPART 类

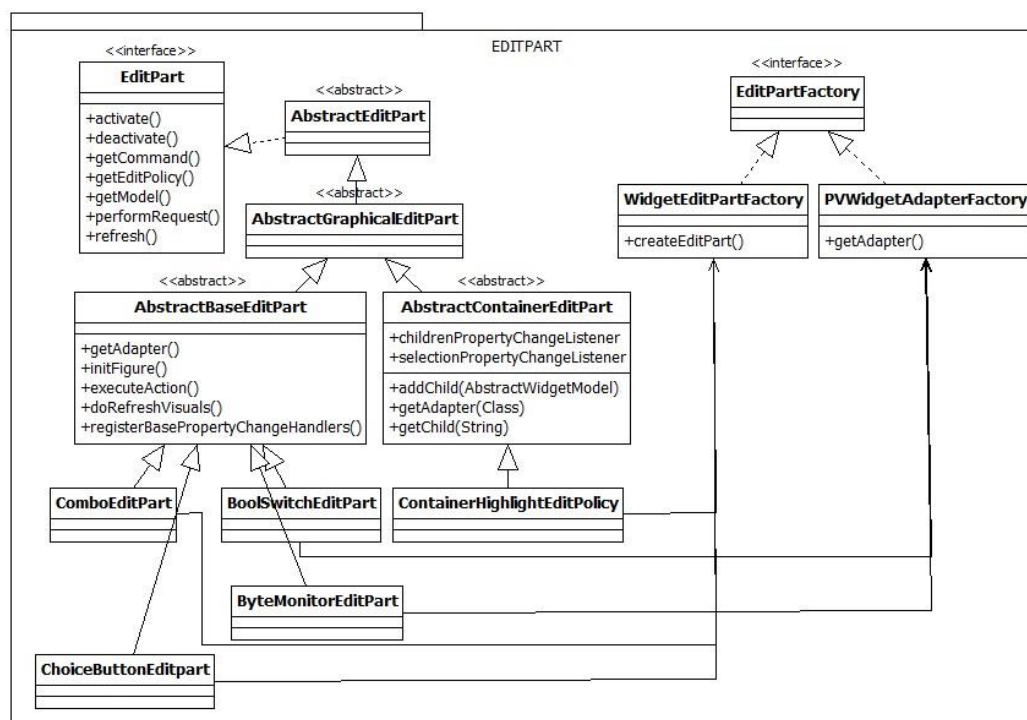


图 4-10 图形编辑模块 EDITPART 类图

类图说明：

在模型与视图之间，控制器作为两者的桥梁。不仅实现将图形控件显示在编辑器上，而且当视图发生变化时，要把编辑结果映射到模型上。一组 **EditPart** 对象继承实现了 eclipse 自带的 **EditPart** 接口的抽象类 **AbstractGraphicalEditPart** 构建各自特殊属性的类实现。**EditPartFactory** 负责为每一类型创建具体的实现类，体现了设计模式中的简单工厂模式。用户的编辑动作被 **EditPart** 转换为一系列的请求（叫做 **Role**），其中 **active()** 方法将控制器自身作为监听器注册到对应的模型中。当监听器返回模型变化时，**PropertyChange()** 方法调用 **refreshVisuals()** 方法将视图进行改变。

### 4) EDITPARTVIEW 类

类图说明：

创建 **EditorViewer** 类作为编辑面板的生成类。这个类继承 **GraphicalEditorWithPalette** 类生成一个带有调色板的图形编辑器。其中的 **configureGraphicalViewer()** 方法利用 **setEditPartFactory()** 工厂模式生产编辑器，**initializeGraphicalViewer()** 方法初始化编辑器包含的控件及属性内容，**hookGraphicalViewer()** 方法将编辑器对象同步到 RCP 框架的工作台上。**Palette** 调色板中的内容就是自定义绘制出的一些图形控件。

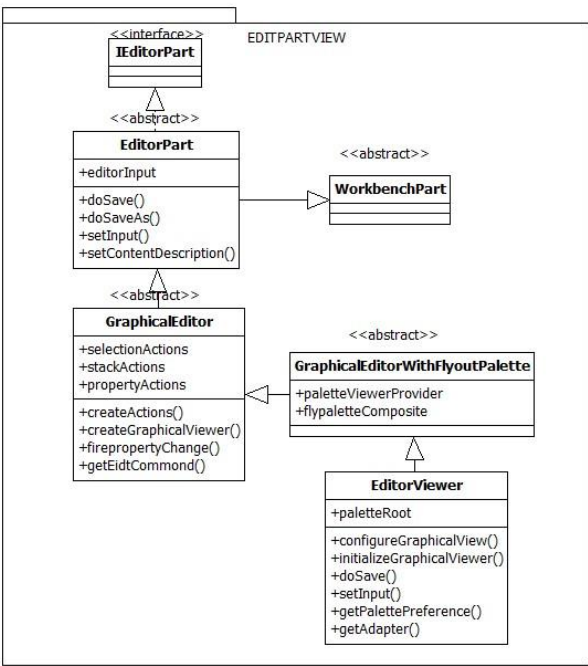


图 4-11 图形编辑模块 EDITPARTVIEW 类图

5) ACTION 类

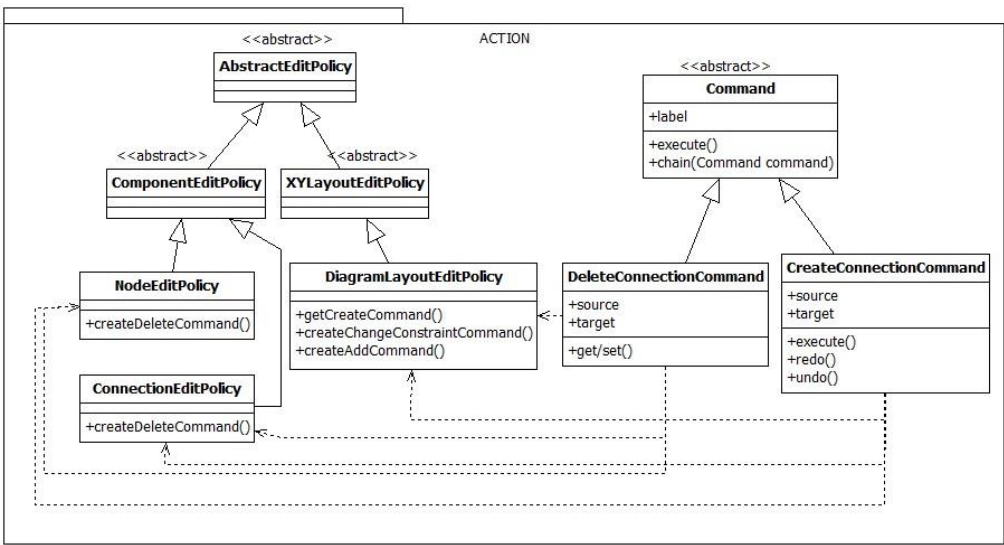


图 4-12 图形编辑模块 ACTION 类图

类图说明：

用户在操作模型时，每一步操作都有对应的 Command 执行命令，这些命令都继承自 Command 抽象类，undo(),redo()操作实现撤销或重做步骤指令，execute()执行命令的入口点，所有的 command 都将保存在 CommandStack 中。Policy 是图形编辑的策略类，在策略类中调用 Command 的各项命令去执行，所有的策略任务都将在相对应的 EidtPart 控制器中被调用。



#### 4.2.4 核心功能实现

图形控件的实现分为四大部分：

表 4-1 图形控件详细分类实现

图形类别	包含内容	总数
普通类控件（Graphics）	直线、带箭头连接线、自定义曲线、矩形、椭圆形、自定义图形、标签、图片	9 个
监控类控件（Monitor）	可更新文本框、LED 监控指示灯、进度条（OPI）、电路板、传感器、示波器（Logic）	6 个
控制类控件（Control）	按钮、多选按钮、可输入文本框、复选框（OPI）、开关、或门、与门、非门（Logic）	8 个
其他类控件（Others）	表格、组合容器、标签容器、树形容器、表格形容器	5 个

由于每一个类型的控件实现有很多个，论文不能详细叙述每一个控件的实现过程，因此，论文挑选了每一类控件中的一个作为典型详细叙述：

##### （1）监控类控件——Circuit 电路图形控件实现示例

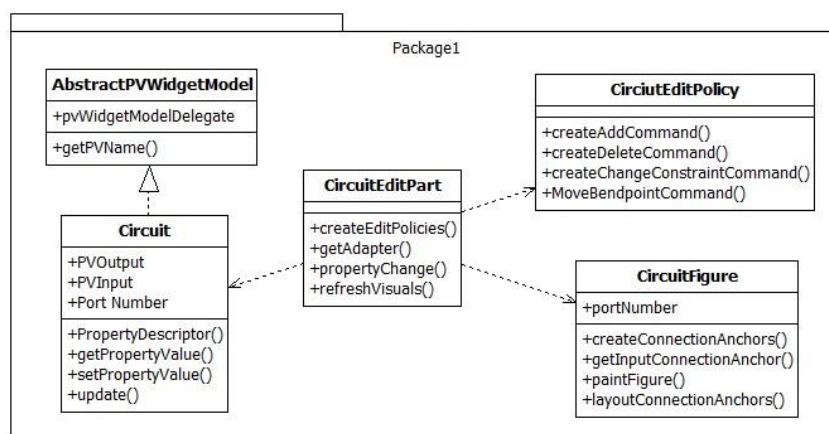


图 4-13 Circuit 图形控件实现类图

类图详解：

Circuit 模型继承 AbstractPVWidgetModel 抽象父类实现基本属性：Basic 属性：Name, PV Name, Widget Type; Behavior 属性：Action, Enabled, Rules, Script, Visible; Border 属性：Border Color, Border Style, Border Width; Display 属性：Background Color, Font, Foreground, Tooltip; Position 属性：Height, Width, Scale Option, X, Y; 同时，模型类定义了控件特殊的属性 PV Input(输入值), PV Output(输出值), Port Number(出入口数)，通过



PropertyDescriptor 属性接口显示在属性列表中。

CircuitFigure 图形视图实现了图形在编辑面板中的显示方式。createConnectionAnchors()实现了图形与其他图形连线以传递监控数据，drawBorder()方法实现了根据用户输入的 Port Number 属性值数据更改出入口数量。

CircuitEditPart 通过 FigureFactory 工厂类创建 CircuitFigure 图形，createEditPolicies()方法装载了图形的操作策略，propertyChange()方法监听操作动作对视图进行更新操作。控制类继承 LogicEditPart 类实现了监听器注册与注销，连线寻找数据源与数据靶方法；

CircuitEditPolicy 策略类调用一系列 Command 方法实现策略的命令。createAddCommand()调用添加命令实现添加控件到编辑面板，createDeleteCommand()实现从编辑面板删除控件的命令，createChangeConstraintCommand()实现将属性列改变的内容显示更改在编辑面板的控件上，MoveBendpointCommand()实现鼠标操作编辑面板的控件通知属性列表的内容改变；

当用户操作 Circuit 控件时，CircuitEditPart 控制器监听用户的操作在编辑面板上创建图形控件。用鼠标拖拽控件或在属性列表中更改控件时，模型通过控制器通知视图对用户操作做出相应的改变。用连线方式联通两个图形控件时，传入的数据会根据连线的靶向点确定传出的数据。靶向点的数量根据用户的输入可在 1-64 位进行改变，若有 4 个靶向点，数据按照二进制的位置从右向左分别为  $x_1, x_2, x_3, x_4$ ，根据公式

$$y = x_1 * 2^0 + x_2 * 2^1 + x_3 * 2^2 + x_4 * 2^3$$

得出输出的值。

## (2) 普通类控件——Polyline 自定义曲线图形控件实现示例

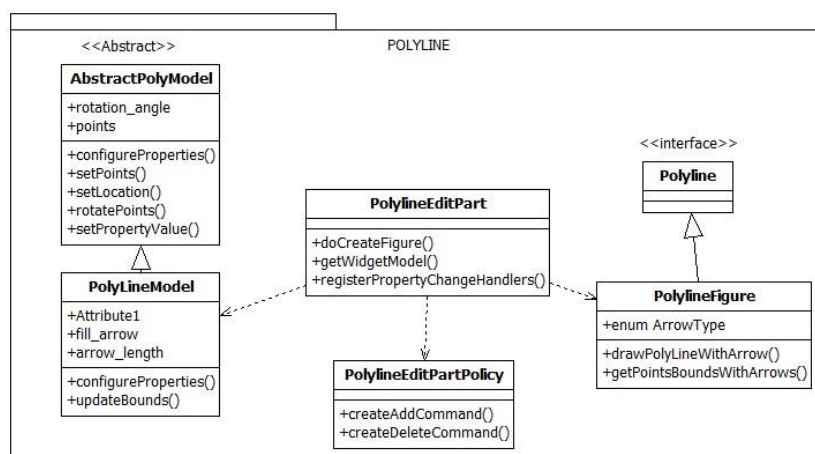


图 4-14 Polyline 图形控件实现类图

类图详解：

Polyline 模型继承 AbstractPolyModel 抽象父类实现鼠标单击编辑面板构成图形的一个点，双击完成多线条图形的绘制的功能。AbstractPolyModel 抽象类用 setPoints()方法调用 setPropertyValue 将鼠标点击的每一个坐标点加入到 PointList 自定义链表中。子类增加了 Arrows, Fill Arrow, Arrow Length 属性列分别实现为多线条增加 To, From, Both, None 箭头朝向的功能。

PolylineFigure 继承 Polyline 抽象类，给控件定义一个枚举类型的 ArrowType，提供 To, From, Both, None 四个方向的图形绘制方式：To：末端添加箭头；From：开端添加箭头；Both：两端都添加箭头；None：两端都不添加箭头。drawPolyLineWithArrow()方法从鼠标点击第一个坐标点并记录到最后一个坐标点双击鼠标完成，调用 Eclipse 库提供的 graphics.drawLine()方法划线并加上坐标。getPointsBoundsWithArrows()方法实现通过比较点击点的坐标值，最小的坐标值为图形的 x,y 坐标，x 值最大差为图形的宽度，y 值的最大差为图形的高度。

PolylineEditPart 控制器调用 doCreateFigure()方法绘制图形并显示在编辑器面板中，getWidgetModel()方法获取控件对应的 Polyline 模型。PropertyChangeHandlers()方法为控制器的监听器，对模型以及视图的改变做出响应并通知对方。

PolylineEditPolicy 为模型的策略类，负责为对控件的操作选择合适的 Command 命令。实现方法同其他控件一样，因此不再赘述。

### (3) 控制类控件——OrGate 或门图形控件实现示例

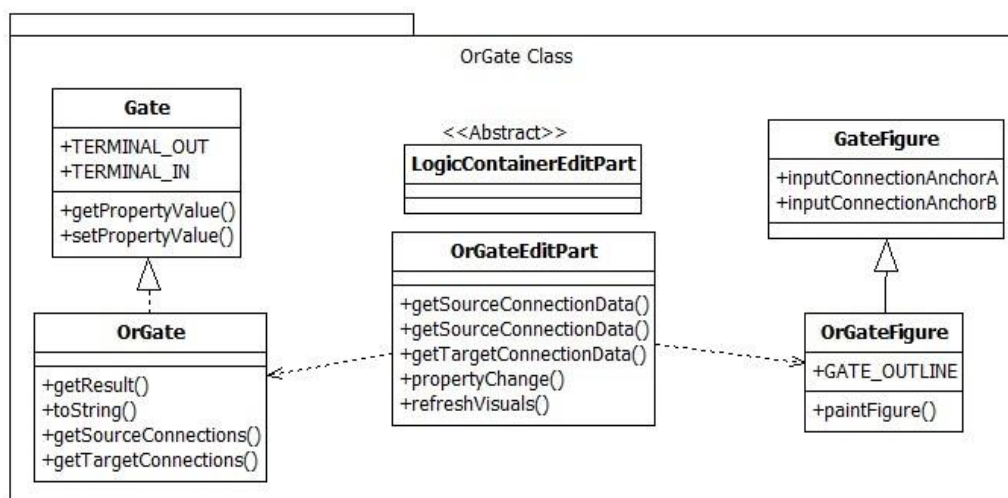


图 4-15 OrGate 图形控件实现类图

类图详解：

OrGate 模型同 AndGate, XOrGate 共同继承父类 Gate 逻辑门模型。Inputs\_a 变量存放输入通道获得的一个数据，inputs\_b 变量存放输入通道获得的另一个数据。getSourceConnections()方法获得控件与输入控件的数据连线，getTargetConnections()方法获得控件与输出控件的数据连线。

OrGateFigure 视图与 AndGateFigure, XOrGateFigure 组成了控制类控件中常用的或、与、非三大逻辑判断图形控件，供本项目 Logic 逻辑界面布局时使用。三个控件共同继承于 GateFigure 逻辑门图形父类。GateFigure 对逻辑门的输入门与输出门的 4 个连线进行定义。OrGateFigure 图形在输入门定义了两条通道，两条数据源可以输入或逻辑门，定义了一条输出通道，经过或门的数据被计算都输出这条输出通道。

OrGateEditPart 继承 LogicContainerEditPart 父类实现 getSourceConnectionData()方法获取数据源的输入数值，getTargetConnectionData 实现数据输出给相连控件的功能。createEditPolicies 调用控件策略类为控件的增加、删除、修改、查找确定合适的 Command 命令。实现逻辑以及调用方法与上述控件类似，因此不再赘述。

#### (4) 其他类控件——GroupingContainer 组合容器图形控件实现示例

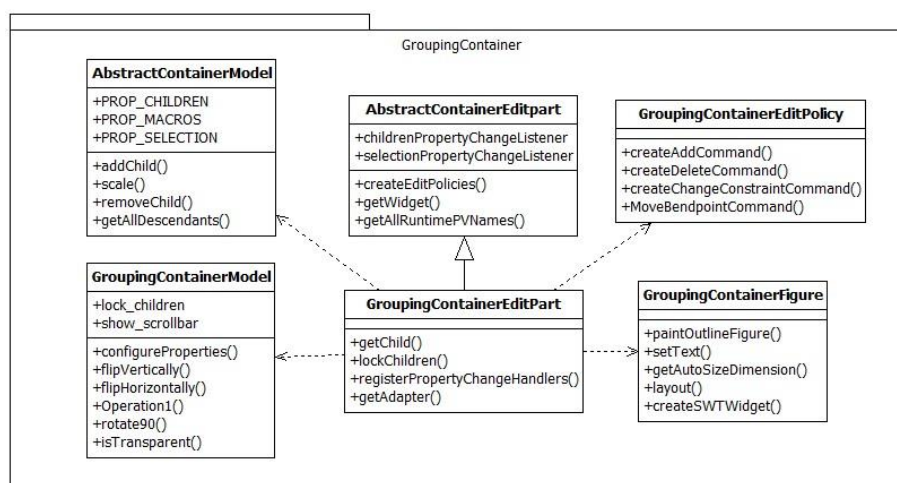


图 4-16 GroupingContainer 图形控件实现类图

类图详解：

GroupingContainer 是最常用来布局界面的工具，放在其他类控件中是方便用户组合与拆分多个控件。

GroupingContainerModel 模型继承了抽象的 AbstractContainerModel 组合模型。

flipHorizontally, flipVertically()通过传入的图形 x 中点与 y 中点数值在 for 循环中设置组合中包括的控件以水平显示或垂直显示。rotate()方法通过传入的旋转类型与旋转角度，改变图形的坐标，实现顺时针 90 度、顺时针 180 度、逆时针 90 度、逆时针 180 度这四项操作。isTransparent()设置布局控件的背景和边框颜色，实现需要透明或者显示。GetWidget()方法定义了 GroupingContainer 通过顺序结构存放子类图形控件。

GroupingContainerFigure 对控件的背景色、前景色、边框类型及颜色、是否透明显示等方面进行配置。

GroupingContainerEditPart 控制类继承 AbstractContainerEditpart 抽象类，重写了父类 getChild()方法不同于树形或者表形组合控件的实现方法，将控件按顺序写入到组合容器中。lockChildren()可以将子类控件上锁，被组合控件调用 setSelectable()传入 lock 变量锁定，解除锁可以释放控件的操作权。

GroupingContainerEditPolicy 策略类实现为控件的增加、删除、修改、查找确定合适的 Command 命令。实现逻辑以及调用方法与上述控件类似，因此不再赘述。

#### 4.2.5 实现效果图

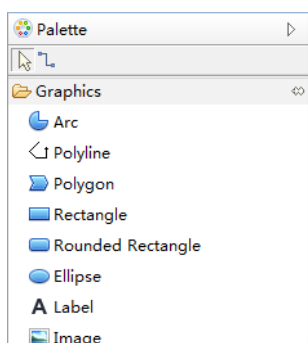


图 4-17 普通类控件实现效果图

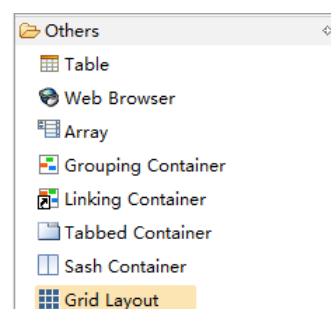


图 4-18 其他类控件实现效果图

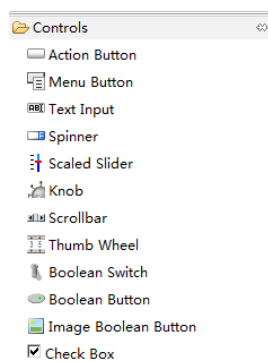


图 4-19 控制类控件实现效果图

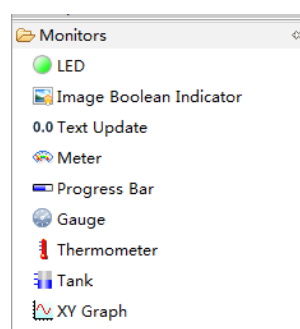


图 4-20 监控类控件实现效果图

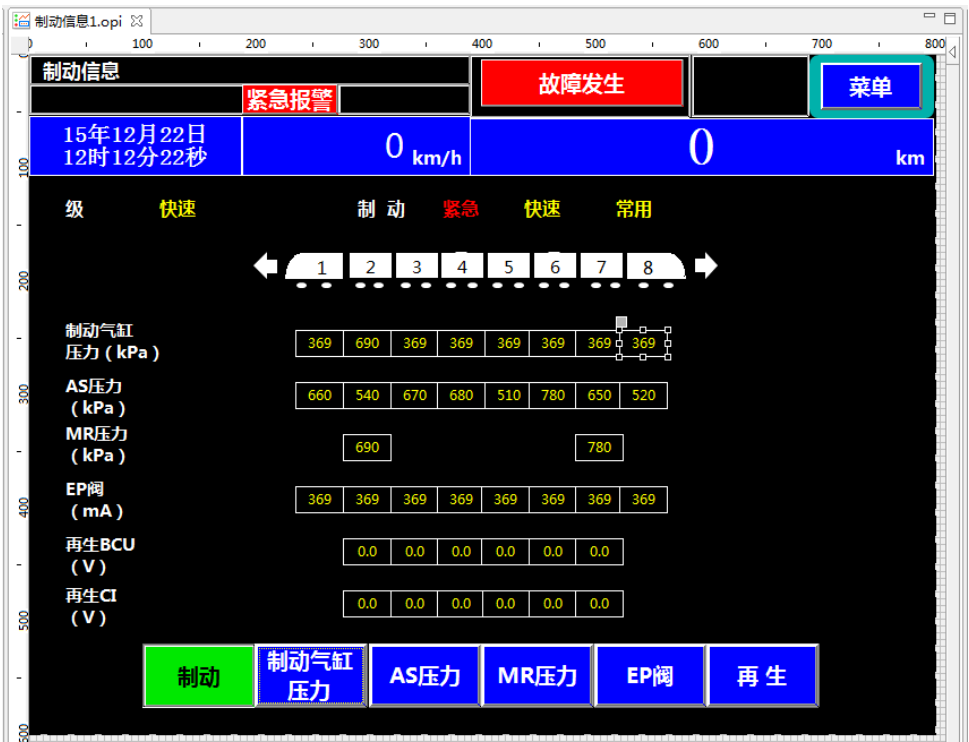


图 4-21 图形编辑模块 OPI 界面实现效果图

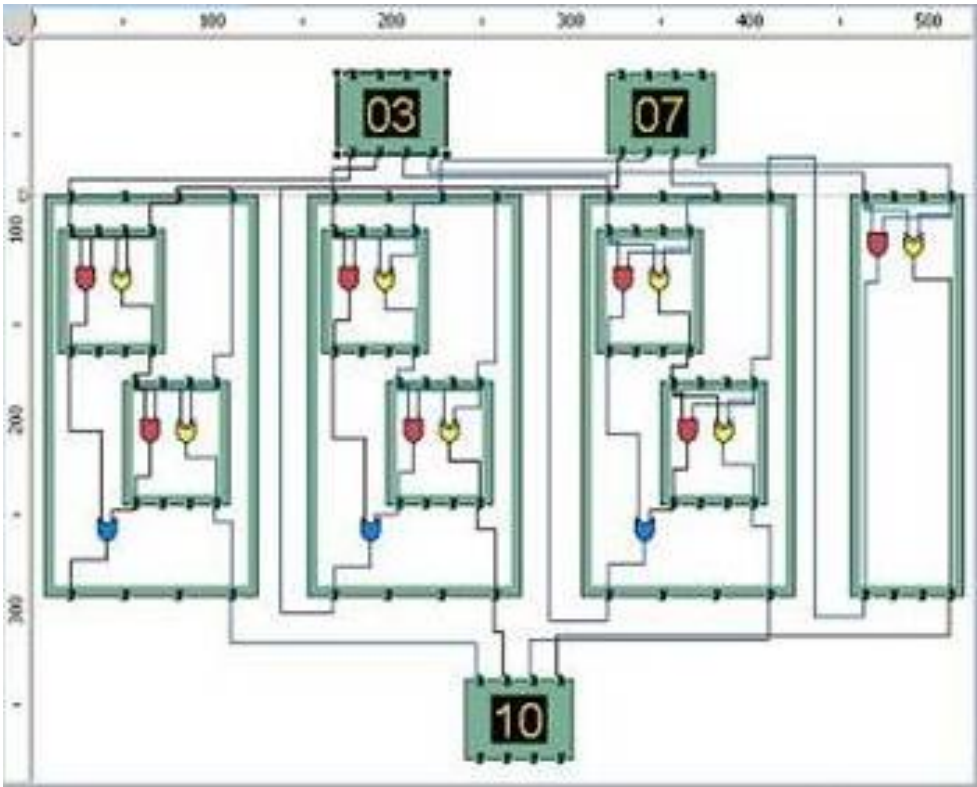


图 4-22 图形编辑模块 Logic 界面实现效果图

## 4.3 数据通信模块

### 4.3.1 设计描述

数据通信模块为实现与监控平台的实时数据传输，需要通过控件绑定 PV Name 属性值，后端 PV Manager 的数据分发与 DataSource 数据源管理，最终实现实时监控数据在仿真机、Zookeeper 服务器与 MON 屏编辑软件之间的传输。

### 4.3.2 时序图

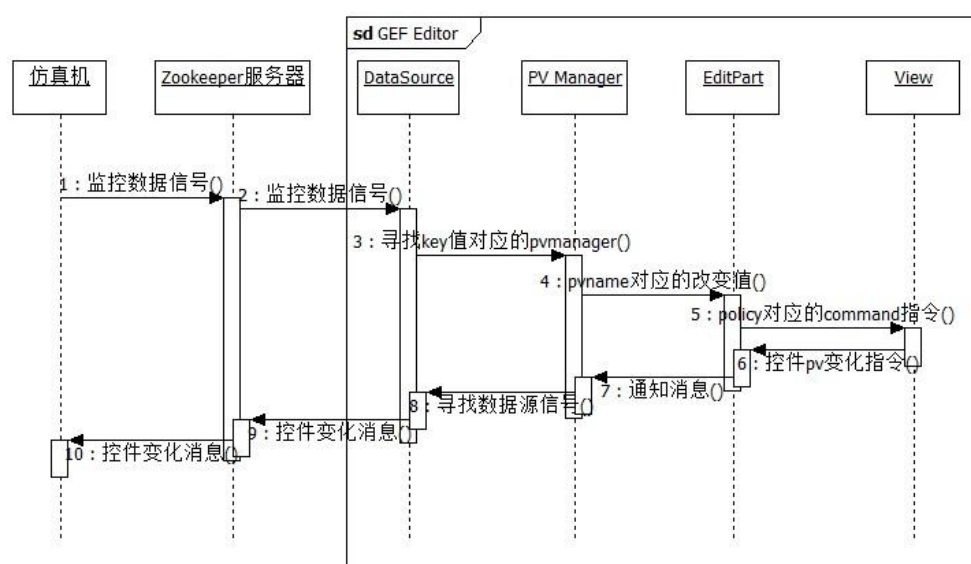


图 4-23 数据通信模块时序图

时序图说明：

仿真机模型在绑定的实体部件发生变化时，将会通过数据通信中心发送监控数据到 Zookeeper 服务器端。在 Zookeeper 服务器端将监控数据发送给编辑软件时，系统采用 PV Manager 作为显示控件与数据源关联的中间桥梁。当显示控件被指定一个 PV Name 属性值时，系统自动创建一个 PVManager 对象且指向 DataSource 数据源库中与制定 PV Name 具有相同 key 值的 DataSource 对象。PV Manager 接受到指令将通知被绑定 PV 值的图形控制器调用对应的策略对视图层的控件显示监控数据<sup>[15]</sup>。



4.3.3 核心类的结构设计

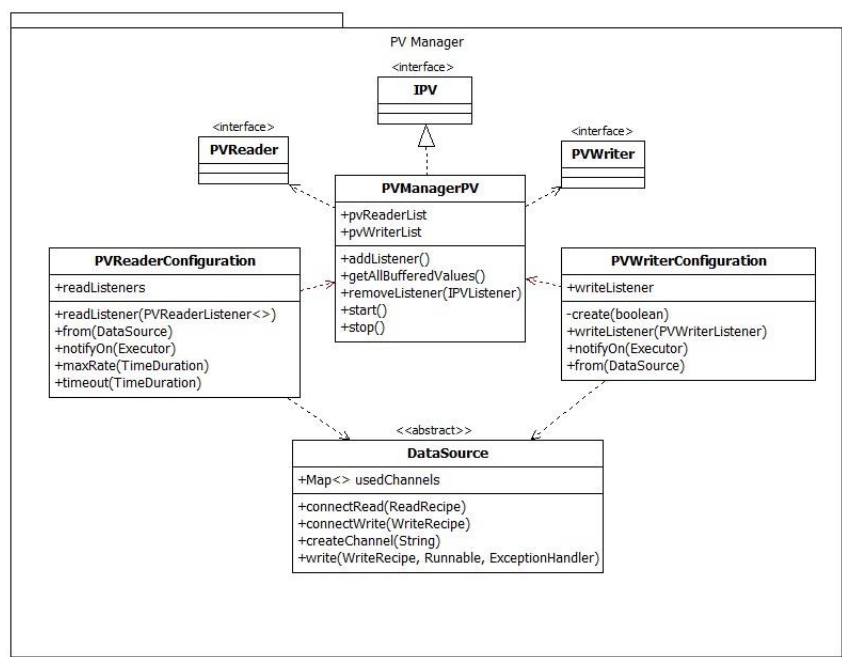


图 4-24 数据通信模块类图

类图说明：

PVManagerPV 类实现了 IPV 接口，是 PV 通信机制管理的核心类。它拥有一个 PVReaderList 和一个 PVWriteList 用来存放读入和写入的数据流。addListener()方法是管理器对外界传入监控数据变量的监听器，PVReaderConfiguriation 和 PVWriterConfiguration 两大类是数据流读写的注册方法类，实现对数据通道的监听、注册操作，DataSource 类即数据源，负责创建数据通道和传递数据源。

4.3.4 实现效果图

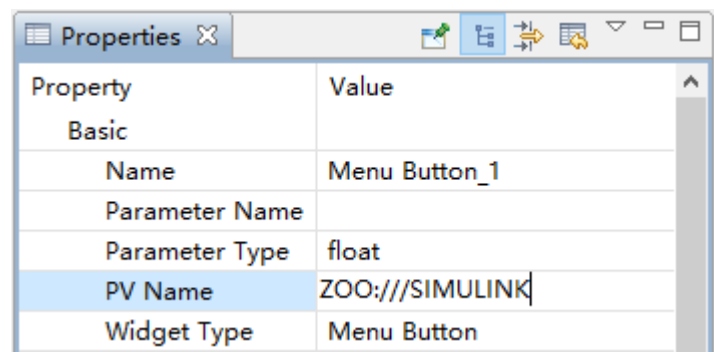


图 4-25 数据通信模块 PV 变量值绑定实现图

## 4.4 脚本关联模块

### 4.4.1 设计描述

脚本关联模块实现了用户创建并编辑 JavaScript 或者 Python 类型的脚本文件，将脚本文件关联到需要控制的图形控件上。在 OPI 文件的运行模式下，根据 PV 值的变化触发脚本文件的执行，从而改变控制的显示。

### 4.4.2 时序图

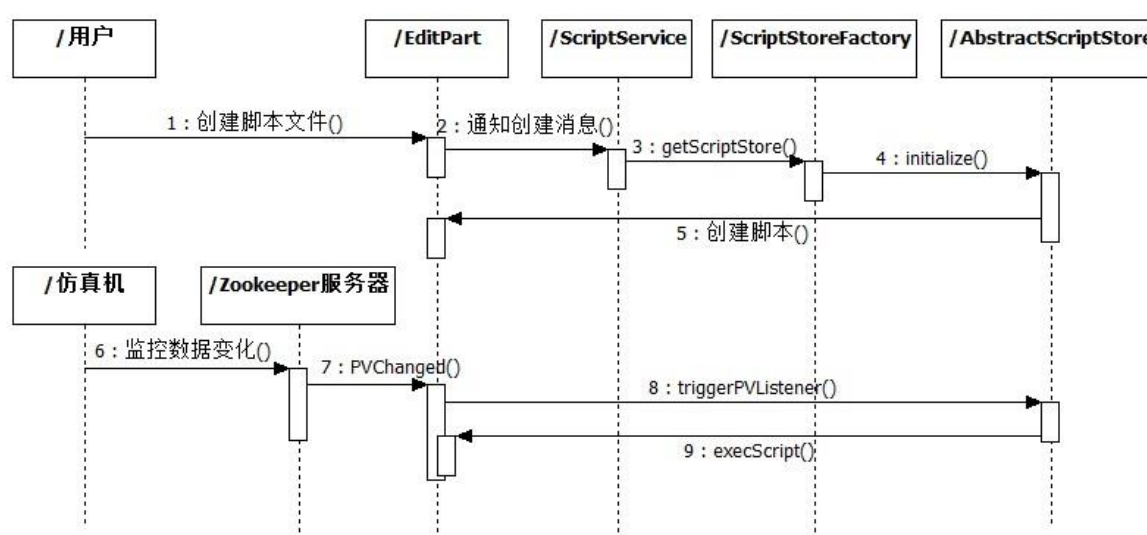


图 4-26 脚本关联模块时序图

时序图说明：

用户通过创建脚本文件选择 js 或者 python 脚本，EditPart 控制器通知 ScriptService 创建脚本文件的消息，ScriptStoreFactory 工厂类根据传入的消息类型判断调用 AbstractScriptStore 方法创建脚本<sup>[11]</sup>。一创建好的脚本文件需要在控件的属性列表 Script 属性中关联其对应的控件。当控件监听的 PV 值在仿真机端发生变化时，Zookeeper 服务器接收到消息通知触发 PVChange()方法传递给控制器变化消息。AbstractScriptStore 方法拥有 trigger PV Listener 方法监听到 PV 值变化，以 exeuteScript 方法触发关联脚本程序的运行，脚本程序将会控制前端视图的内容显示。



## 4.4.3 核心类的结构设计

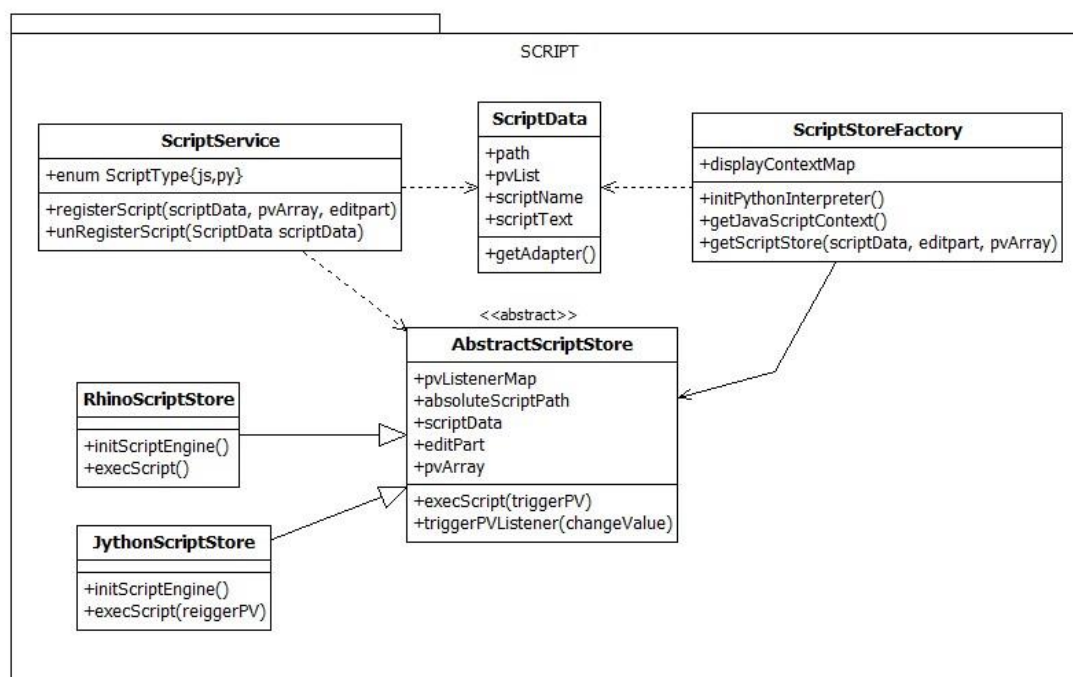


图 4-27 脚本关联模块类图

SCRIPT 类图说明：

ScriptService 类是为脚本的关联与执行服务的类。它定义了两种固定的脚本类型：JAVASCRIPT 与 PYTHON。用户通过编辑这两类脚本文件，ScriptService 类调用 registerScript()方法，registerScript()方法调用 ScriptStoreFactory 中的 getScriptStore()方法创建一个关联一个 PV 控件的脚本文件。ScriptData 类提供脚本的存储地址、文件名、文件内容以及关联的 PV 值列表。AbstractScriptStore 抽象类利 ScriptService 类传入的 scriptData, editpart, pvArray 变量值，在 PV 在通信机制中的 PV Name 变量发生变化时，triggerPVListener()方法监听到变化消息，及时触发所关联脚本的 execScript()方法的执行。JythonScriptStore 与 RhinoScriptStore 方法都继承了 AbstractScriptStore 抽象类，分别实现了 PV 控制变化的 js 或 pyhton 脚本和自定义脚本执行的方法类。

4.4.4 实现效果图

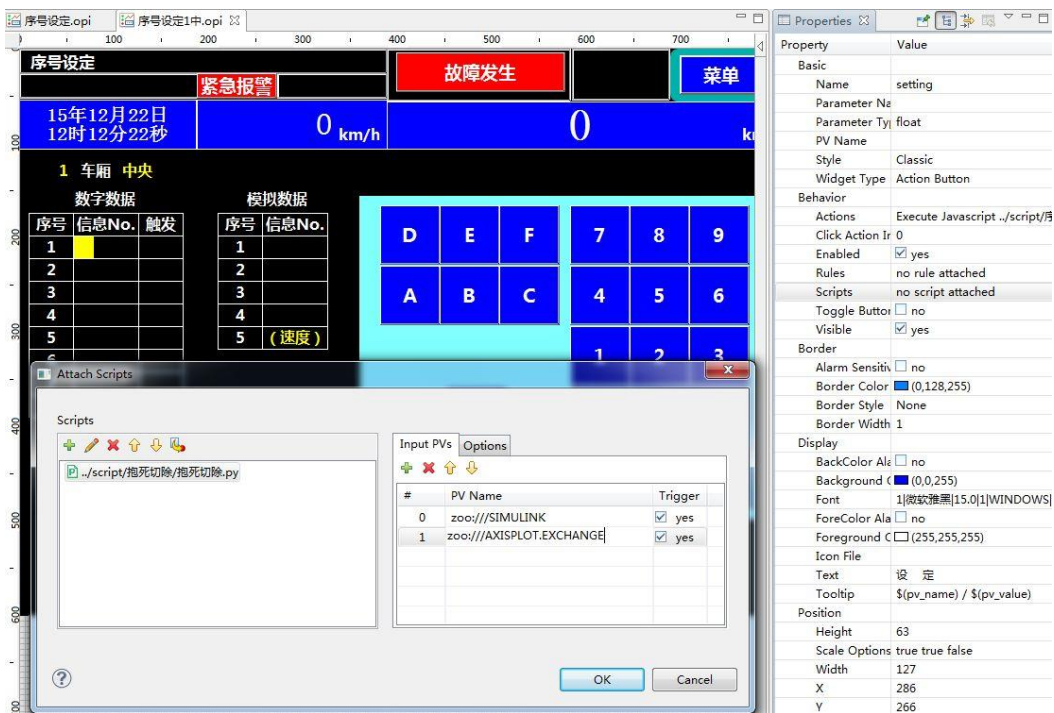


图 4-28 脚本关联模块实现效果图

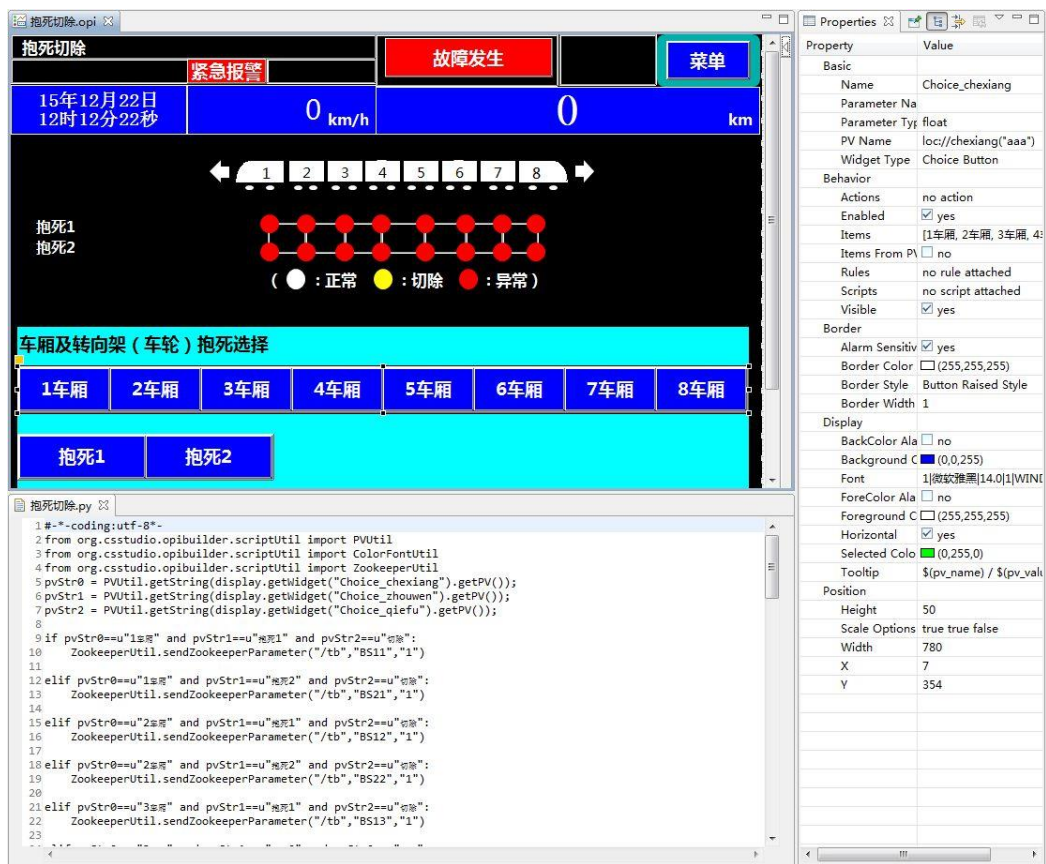


图 4-29 被关联脚本编辑实现效果图

## 5. 项目整体优化方案

### 5.1 数据通信功能优化

#### 5.1.1 通信变量分渠道绑定 PV 值

经过前期系统的试运行反馈，每一个控件使用一个特定的 PV Name 关键字，Zookeeper 服务器端也同时要为每一个 PV 变量创建一个 path 数据通道。由于项目的业务需要，动车组列车故障有上百个，仿真机的列车控件数以万计。为每一个仿真机控件分别绑定一个数据通道，当仿真机发送监控故障信息给服务器时，将会使 Zookeeper 服务器少则产生数十秒的延迟，严重时会导致服务器崩溃，系统不能继续运行。

优化方案：

为通信渠道分别绑定 2-3 个读取、写入变量，从仿真机传递的监控数据都通过 Zookeeper 中的写入流与编辑软件通信，编辑软件端产生的监控变量都通过 Zookeeper 读取流读取数据并传递给仿真机模型。为每一个控件与仿真机提供一个 parameter 变量，在属性列中以 Parameter Name 属性的方式编辑，硬件与软件的映射就通过 parameter 实现。当仿真机发送监控数据给编辑软件时，绑定了同一个 parameter 的图形控件的变化利用其关联的脚本文件接收变化消息并作出相对应的显示处理。

#### 5.1.2 绑定变量的列表显示

由于每一个编辑页面中需要监控的控件变量非常多，分布无规律，因此，想要查找页面中的一个 PV 值必须要遍历每一个控件的属性列表。这项工作对于人工操作来说非常繁琐造成了不太友好的用户体验。

优化方案：

为编辑软件创建一个 PV 值列表视图，每一个 PV 值被绑定在所有页面的所有控件都可以一一对应的显示出来。用户可以从列表视图中拖拽变量值到编辑器面板中，直接实现变量的绑定。在列表选中任何一个变量对应的图形控件名，编辑软件自动打开控件所属的页面，并且定位到选中的控件上方便用户修改。

## 5.2 脚本编写功能优化

在用户为其 OPI 页面编写 JavaScript 或 Python 脚本时，软件只提供了编辑面板为用户编辑使用。但是，根据用户实际使用反馈，没有良好的编程提示与实时语法纠正将会导致脚本编写的错误率迅速增长。这一问题不仅加重了使用人员的工作量、降低了工作效率，并且造成了糟糕的用户体验。

优化方案：

为脚本编辑器提供如同其他脚本开发平台一样友好的编程语言提示、实时语法检查、控制台错误提示的功能。

## 5.3 基本图形控件功能优化

在软件版本测试阶段，用户根据实际要求需要编辑 45 度矩形。但是，出现了可以编辑创建一个平置状态的矩形，仅可以进行 90 度倍数旋转，不能进行任意角度旋转。出现了图形编辑瓶颈。图形的背景只提供了颜色的变换，没有提供纹图案的背景选择。

优化方案：

基本图形往往比其他专业控件的使用频率更加高，因此丰富基本图形控件，如：矩形、三角形、圆形，为其增加任意角度旋转和背景图案选择功能。或者可以在条件允许情况下为图形控件视图的父类添加任意角度旋转的功能，继承自父类的所有控件都可以在编辑面板使用鼠标控制旋转角度。基本图形控件背景图案添加纹理背景功能，这样，就可以通过颜色和纹理图案区别专业领域的硬件设备，例如，传感器状态综合显示中区别传感器的种类。

## 6. 结论

本论文是以北京经纬恒润科技有限公司参与开发的实际项目——动车组列车全仿真模型故障分析与研究实验台系统的设计与实现作为基础的。在分析了铁路运输行业故障应急处理体系不完善、列车相关人员缺乏基本的紧急处理能力的情况下，得出了列车全仿真模型故障分析与研究实验台开发的必要性与存在的意义。论文主要完成了系统中 MON 屏编辑软件开发从需求功能分析、系统总体架构设计与核心技术点、软件详细设计与实现到整体优化方案四大部分的论述。

首先，通过与列车相关人员以及故障研究人员的多次沟通，将实际业务需求转变为软件开发中的需求模型。运用统一建模语言创建需求功能用例图，从用户角度给出专业的功能需求分析。作者发现，将用户的需求变为开发人员所能理解并使用的方式就是建立 UML 用例图，不仅能从用户角度阐述问题，而且方便系统架构师为软件设计合理高效的架构模型。其次，在系统架构设计阶段，项目不使用 AWT 图形工具包，而是选择 Eclipse RCP 搭建基本工作台框架，采用 GEF 框架实现图形编辑的核心功能。不得不承认，这是一个非常明智的选择，具体内容在论文第三章给出了项目核心技术的简述。再次，软件的详细设计与实现分为了工程管理模块、图形编辑模块、数据通信模块、脚本关联模块四个部分，每一部分都通过时序图展示了数据传输过程，通过类图展示了代码实现的核心类设计。要实现一款软件，涉及纷繁复杂的业务逻辑以及方方面面软件设计模式，本项目的实现也是在不断设计与修改中完成的软件设计，这是使软件具备高耦合、低内聚、可扩展、安全性等优质性能的前提，是软件的骨架。最后，每一款软件的完成不管是在功能上还是非功能上，总会伴有或多或少的设计缺陷。因此，在软件开发完成后，还要进行后续的维护与扩展。论文记录下个人对于软件优化方案的一些浅见，以供后续开发或有兴趣参与后期修缮的同仁使用。

这次论文的撰写，不仅是对本人此次实习期间所做的整个工作的回顾、梳理与总结，更是将作者大学四年所学的知识运用在真正的项目实践中，使本人深刻理解了一套完整的软件开发流程的理念。这不仅是一个阶段的终点，更是作者的人生迈向下一个阶段坚实的基础与有力的起点。

## 参考文献

- [1]. 陈振虹. CRH 高速动车组技术原理与趣谈. 第一版. 中国铁道出版社. 2013.06. P67-112
- [2]. 周婷, 董海棠. Eclipse 平台架构及其插件实现. 第一版. 甘肃科技纵横. 2007.3. P12-P37
- [3]. 陆阳. Eclipse RCP 与 Spring OSGi: 技术详解与最佳实践. 第一版. 机械工业出版社. 2012.12
- [4]. 张鹏, 姜昊, 许力. Eclipse 插件开发学习笔记. 电子工业出版社. 2008.07
- [5]. 那静. Eclipse SWT/JFace 核心应用. 清华大学出版社. 2007.03
- [6]. 弗里曼. Head First 设计模式. 第一版. 中国电力出版社. 2007. P113-P117
- [7]. 刘旺晖. 基于工业以太网的动车组列车级网络仿真研究[学位论文].西南交通大学. 2013
- [8]. 金星善. 基于 Eclipse RCP 的应用系统的研究与实现[学位论文].武汉理工大学.2011. P10-P45
- [9]. F. 荣凯拉(美), B.里德(Reed)(美). 谢超(译). ZooKeeper: 分布式过程协同技术详解. 第一版. 机械工业出版社. 2016. P15-P23
- [10]. Eric Clay berg, Dan Rubel (著). 周忠良(译). Eclipse 插件开发. 人民邮电出版社. 2006.10
- [11]. Robert Bill. Jython for Java Programmers .第一版. Sams Publishing. 2001. P54-P119
- [12]. Jackwind Li Guojie. Professional Java Native Interfaces with SWT / JFace. 第一版. Wrox. 2004.06. P66-P104
- [13]. Stephen Holder, Matthew Scarpino, Laurent Mihalkovic. SWT/JFace in Action. 第一版. Manning Publications. 2004.10. P87-P255
- [14]. Jeff McAffer, Jean-Michel Lemieux, Chris Aniszczyk. Eclipse Rich Client Platform. 第二版. Addison-Wesley Professional. 2007. 09. P5-P398
- [15]. Kay Watford. The Zookeeper. 1<sup>st</sup> Edition. Authorhouse. 2007.03. P84-P123

## 致 谢

衷心感谢作者的毕设导师北京交通大学软件学院 XXX 老师。从项目的开始到论文的撰写结束，老师以严谨的学术理念、认真的教学态度和孜孜不倦的教诲给予了作者很大的帮助。毕业设计能够顺利的完成，离不开老师的督促和引导。

感谢给予作者宝贵实习经验的北京经纬恒润科技有限公司，以及在项目实践中给予作者莫大帮助的同事和领导，你们给予了很多实际的经验与工作帮助。

感谢给予作者帮助的其他老师与同学，是你们的热心帮忙替作者解决了很多顾虑与烦恼，为作者的生活增添了丰富的色彩，使作者拥有了一个充实快乐、积极向上的大学生活，作者将倍加珍惜，心念不忘。

最后，感谢给予作者生命的母亲，感恩生命。

## 附 录

## 附录 A 外文翻译

文章源自:

Eclipse Rich Client Platform (Second Edition)

Jeff McAffer

Jean-Michel Lemieux

Chris Aniszczyk

Library of Congress Cataloging-in-Publication Data

Copyright . 2010 Pearson Education, Inc.

翻译正文:

### Eclipse 富客户机平台 第二版

#### 1. Eclipse 作为富客户机平台

在 20 世纪 90 年代初，富客户端的术语被创造出来，用以建立客户应用程序使用视觉基本和专家。这些客户端应用程序的数量和普及的急剧增加，部分原因是由于“丰富”的用户体验的愿望。

丰富的客户支持高质量的终端用户体验，通过提供丰富的用户界面（UI）本地域以及高速的地方处理。丰富的 UI 支持本地桌面隐喻如拖放，系统剪贴板，导航和定制。做得好，有钱的客户是几乎透明的终端用户和他们的工作重点放在工作，而不是系统。富客户端被用来区分这样的客户从终端客户应用程序，或简单的客户，他们所取代。

客户技术的兴起伴随着发展环境的改善。所见即所得的界面设计者造富客户端应用程序的方便和乐趣。这些开发工具允许客户端程序员重用常用的构建块，以减少开发时间。

早期的富客户端平台（RCP）用于胶客户的业务逻辑的操作系统（OS）。他们消灭了很多粗活的编程任务需要创建用户界面和访问数据库。中间件提供框架和基础设施开发商而不是重新发明轮子，花更多的时间在编程领域逻辑。

最终用户很高兴与由此产生的丰富的客户端应用程序，因为他们功能和使用方便。信息技术（信息技术）管理者，然而，发现了许多隐藏的成本。部署和升级的客户是一个人工任务，用户经常调整其安装、移动文件，或者安装了其他客户对共享库。

然后沿着互联网和网络为基础的应用程序，或瘦客户。瘦客户承诺解决许多部署和管理问题与丰富的客户。由于应用程序在服务器上，更新集中。用户只需要一个网络浏览器。这大大减少了成本的部署和维护企业应用的费用用户体验瘦客户端没有提供的用户界面功能和高速相互作用的客户已经期待。

可用性和瘦客户端的应用与发展进行了增强富 Internet 应用程序或 RIA 的。区域一体化协定带来了常用的桌面特性和隐喻通过改进的用户界面技术的基于 Web 的应用程序。当前的 RIA 正在使用的技术构建例如 Adobe Flash / AIR，谷歌网页。工具包（GWT），微软的 Silverlight 和 Mozilla 的 XULRunner 的。在节约成本和简化部署很受欢迎，但这一举动瘦客户机和区域一体化协定仍然是一个倒退的功能，能力，开发效率。瘦客户机应用程序，使用请求 - 响应模型中，需要更多的网络功能，以确保最佳的互动性能。典型的 RIA 技术缺乏模块化很强的模型，挫折用于，例如，在 Eclipse 模块化模式



带来的功率组。Eclipse 富 Ajax 平台（RAP）是这里一个明显的例外。今天的用户和问题，继续推动利用从富客户端网络到桌面。域正变得越来越复杂，以及量数据可视化和操纵正在增加，因为是需要与整合其他系统。对于富客户端的需求超出了丰富的用户界面的愿望。用户需要移动，脱机工作，整合自己的内容和 workflows，协作，并采取本地硬件的优势。该技术此处列出至少部分解决这些课题，但最终结果在越来越多的功能的浏览器，使薄客户不要这么薄和振兴的许多老胖客户端主题。这些办法交易的操作和窗口系统的浏览器，但仍留下模块化和版本控制的挑战。但关于引起的部署和维护问题什么早前转移到瘦客户机摆在首位？已经改变的东西，以减轻这些问题？是；现在，愈来愈多的分量的机制，例如像 Eclipse，可用于富客户端应用程序开发人员。组件化系统由绝缘和隔离同时解决的部署和维护问题从变化的组件。新的部署机制，例如春分 p2 和企业管理系统，如 Yoxos 根本上消除需要个性化的企业管理软件。总之，这个新品牌的丰富客户需要能够为当今的动态场景应用集成的类型并提供了两全其美。

### 1.1 Eclipse

如果你是新 Eclipse，你可能会想，“什么是 Eclipse？”首先，eclipse 是人们建立基于 Java 的工具和基础设施来帮助你解决你的问题的一个开源社区。社会的最明显的输出是 eclipse java 集成开发环境（IDE）。这个世界级的 java IDE 常常超过图表开发者满意度和使用。它也没有 <http://eclipse.org>。在 IDE 是一个通用的工具，支持多种平台对语言和系统工具，从 java 到 C 到 Python 的 Web 技术，数据处理和报告。蚀相元件模型意味着这些工具可以组合和集成所需的。工装平台下的 Eclipse RCP。这是一个通用的平台运行应用程序。Eclipse 是一个这样的应用。这本书的重点是如何利用 Eclipse RCP 构建你的应用程序。

### 1.2 Eclipse 富客户机平台

为什么 Eclipse 特别适合建立丰富的客户端应用程序？要回答这个问题，让我们看看历史上的有钱和瘦的客户，并观察日食的特点如何保持的好处和解决过去的方法的陷阱。这些特点总结在这里：

组件：eclipse 包括一个强大的组件模型。基于日食的系统是由组成部分称为插件。插件是版本可以被多个应用共享。同一个插件的多个版本可以并排安装，应用程序可以配置为运行与他们需要的精确版本。这种方法是有吸引力的，因为它允许应用程序的发展，随着时间的推移，加入和更换组件。

中间件和基础设施：在这个组件模型的顶部是一组编写客户端应用程序的框架和设备容易得多。Eclipse RCP 是中间件的功能，你不想写，因为它是一种手段，而不是一个结束，为您的域。它包括一个灵活的用户界面范例，可扩展的 UI 的设施，可扩展应用程序，帮助支持，上下文敏感的帮助，网络更新，错误处理，以及更多。

本地用户体验，在对比的是什么是提供的瘦客户端，用户想要一个丰富，舒适，和本地用户体验。这包括一个平滑的，集成到桌面的响应用户界面。月食标准件 6 章 1，Eclipse 作为一个丰富的客户端平台工具包（SWT）提供了一个图形用户界面（GUI）为 java 工具包允许高效和可移植的访问操作系统的本地用户界面。

可移植性：瘦客户机的优势是到处跑。日食提供开源软件和客户端异构环境的支持，从传统的个人电脑，如平板电脑和更薄的设备，到移动和嵌入式设备如 PDA 和智能手机。只要你能找到一个 java 虚拟机（JVM）与 J2ME 基础库或更高（例如，J2SE 6），你可以运行你的客户。23 章，“RCP 无处不在”，“如何使用 Eclipse 的设计，基本上运行客户无处不在。

智能安装和更新：控制与部署和维护丰富的客户端应用程序的成本是在早期的问题。Eclipse 的插件组件框架能够将部署和使用任何数量的机制更新：HTTP，java web start，库文件，复制简单，或复杂的企业管理系统。26 章，“最后一公里”，“细节的任务让你的 Eclipse RCP 应用程序到你的用户。

非连接操作：因为丰富的客户端应用程序在本地运行机器，它们可以独立运行，无需网络连接。

这是一个过瘦客户的主要优势。应用程序，本质上是断开的，可以使用本地缓存，副本和存储和转发机制，以适应网络中断。

开发工具支持：开发人员在第一次浪潮中的丰富客户喜欢思想，帮助他们建立自己的应用程序。Eclipse 提供了一个一流的 java IDE 包含集成开发工具，测试和包装丰富的客户端应用程序。

构件库：没有一个组件框架是不完整的构建应用程序的组件集。这个 Eclipse 社区产生了建筑的可插拔的 UI 插件，管理帮助内容，安装和更新支持，文本编辑，游戏机，产品介绍，图形编辑框架，建模框架，报告，数据操作，以及更多。一些讨论在章 29，“食生态系统”。

### 1.3 Eclipse RCP

Eclipse 项目没有启动建设的 RCP 的意图。相反，它的目标是创建一个集成开发工具的平台。Eclipse 作为一个 RCP 开始在 Eclipse 2.1 发布时间框架作为一个黑客活动。这个词指出，基于 Eclipse 的 IDE 是专业、好看，抛光，他们表现得很好。一些勇敢的开发商进一步观察，相同的框架，使工具更容易写和更具吸引力的可能用于构建更通用的应用程序。他们是正确的，但也有许多挑战。最明显的这些都是交织在一起的基于 Eclipse 作为工具的假设平台和由此产生的无法改变环境的某些元素看起来和感觉。

Eclipse 3.0 是一个重大的有利步骤 Eclipse RCP。几乎所有的 DE 相关的依赖关系被淘汰，和许多不同的部分的用户界面进行了定制。动态的基础介绍了插件的安装、拆卸和更新一种基于 OSGi 运行时（<http://osgi.org>）。这 2 项工作项目达一个大规模的重构平台的主要方面。有了这些改进，兴趣在 RCP 大幅上涨和商业应用开始出现。IBM 改进了其产品套件是基于莲花 RCP；美国宇航局开始使用 RCP 管理，建模，分析空间任务；和 RCP 的出现在各个领域的应用被忽视。今天 RCP 作为从银行和保险的软件平台的基础上卫生保健和地理信息系统。

这本书的发行紧密结合在 3.5（伽利略）的释放。日食 3.5 包含了无数的改进，改进和批发的飞跃支持多种应用场景。创建和编辑工具产品定义是一个这样的飞跃。RCP 开发人员现在能够定义品牌、内容，以及使用 purposebuilt 产品部署策略编辑和向导。

### 1.4 RCP 的利用

在 Eclipse RCP 的生活，我们已经看到了相当数量和应用范围采用技术。一些在 RCP 应用程序页指出在 Eclipse RCP 网站 <http://eclipse.org/rcp>。然而，也有许多到这里或那里的细节，因为这件事。相反，我们强调两点用途是什么 RCP 引人注目的例子是关于：IBM Lotus Expeditor 和 NASA 大师项目。

#### 1.4.1 IBM Lotus 和 Eclipse RCP

在 IBM 的 Lotus 团队是一个在使用 Eclipse 的领先创新者对于一般的应用程序。他们正在寻找建立下一代的莲花平台和生产力工具（例如，电子邮件，邮件，文件管理）为企业和解决以下问题：

- 管理成本
- 部署相关组件化的系统
- 基于角色的功能交付
- 丰富的用户体验
- 合作

莲花车队选择建立在 Eclipse 作为一个跨平台的，industrialstrength 支持基础，使支持应用程序，如消息，文档管理，合作，和其他业务应用。一个额外的驱动程序是可扩展的组件化的 Eclipse 平台提供了集成 onramps 对于现有技术。这些要求不能合理地满足其他的技术，如 Flash 和 JavaScript 客户端平台。这种努力的最终结果是 Lotus Expeditor 平台，服务器管理在桌面上运行的应用程序容器。在这项技术的一个使用，用户通过一个基于服务器的门户系统与系统进行交互。门供应不同的应用程序取决于用户和他或她的角色。应用程序的实现技术从 portlet 和 JavaServer Pages (JSP) 运行变化在 Eclipse RCP 应用，动态下载服务器在客户端上运行。可供用户使用的一组应用程序管理由服务器，如部署这些应用程序。在客户端上运行的应用程序容器基本上是一个独立的额外的中间件，安全，UI Eclipse RCP 的外壳，和管理支持。这两个平台的可选组件和最终用户的应用程序增量由服务器配

置；即，服务器规定的 eclipse 插件客户端。这些插件是动态安装和运行，从而给用户应用程序访问。这种模式的一个关键好处是，只有平台扩展和应用程序的用户有权限使用的配置，导致在一个最小配合的目的配置。容器本身是高度可配置的。图 1-1 显示了一个典型的特派员配置。左边是一个选择器，列出可用的应用程序给用户。此列表是基于用户的角色，并在中央配置。同样，一些应用程序可以在服务器上运行，一些在桌面上。那些在桌面上运行，只有在桌面上使用的是物理上安装的；其他人被下载和安装在需求。

右边是一组视图显示工具，如即时通讯联系人和数据相关的应用程序在屏幕中间。在这个例子中该应用程序是一个三维产品生命周期管理工具显示，汽车与零部件组装。IBM 的 Lotus Expeditor 完全可插拔，和布局是高度可定制的使用声明性标记。不同的用户，在不同的角色，可能会体验到不同的外观和感觉一组不同的应用程序。这是所有配置，并管理通过，服务器。Eclipse 和特派员被证明是一个强大的平台，所以在 2007 Lotus Notes V8 宣布，Lotus Notes 客户端在基于 Eclipse 畅通的环境，如图 1-2 所示。在那之上，莲花队修正了客户端是基于 Sametime V7.5 Expeditor 技术也。说到外观，注意畅通和笔记看起来不一样标准的 Eclipse。畅通和笔记的团队使用的可扩展性 Eclipse RCP 创造独特的用户交互范式和模型，以适应他们的需要。他们反过来又揭露了一个允许消费者的人格机制特派员来进一步自定义和品牌的 UI。今天，莲花技术如 Expeditor 和笔记是用于各种企业金融服务的范围从银行到政府和公众部门。使用场景范围从桌面到呼叫中心和银行出纳工作站亭。

#### 1.4.2 美国宇航局和 Eclipse RCP

在上世纪 90 年代末，美国宇航局开始了一个软件项目，最终成为被称为大师。这是写的基于 java 的工具，一个雄心勃勃的项目管理远程车辆和空间任务，如火星的实验极地登陆者。他们开始使用各种 java 技术，包括应用程序，并逐步提高他们的应用程序的能力和可用性移动到更强大的丰富客户端的方法。这个过程在一系列表面任务中使用的科学分析和规划工具，如精神和探索火星的机会。在 2004 年，大师团队转向了 Eclipse 作为他们的发展环境和 Eclipse RCP 作为大师的基础。图 1-3 显示基于 Eclipse RCP - 大师操纵一些火星探测器的数据。任务软件有其独特的特性，但它的主题是同样的要求，作为你写的的应用：不同的功能要求，任何给定的任务涉及到大量的从数据的工具，从数据浏览到目标捕获的活动生成航天器监测与报告。这些特殊的工具可能不应用于你的领域，但潜在的技术（例如，复杂的用户交互、复杂渲染、数据管理、报表等）相关的日食插件是有趣在各种各样的情况下。从根本调整系统的需求的累积集合很快。每个任务都不同，每一个实验都是独一无二的。在效果，每个任务都需要一套新的工具集。日食组件模型可以共享和重新配置功能。

多样化的用户基础任务软件具有广泛的用户从计算机一害羞的领域专家技术工程师的普通成员公共。日食框架简化了世界级的，可访问的创建 UIS，规模，满足用户的需求。任务软件有其独特的特性，但它的主题是同样的要求，作为你写的的应用：不同的功能要求，任何给定的任务涉及到大量的从数据的工具，从数据浏览到目标捕获的活动生成航天器监测与报告。这些特殊的工具可能不应用于你的领域，但潜在的技术（例如，复杂的用户交互、复杂渲染、数据管理、报表等）相关的日食插件是有趣在各种各样的情况下。从根本调整系统的需求的累积集合很快。每个任务都不同，每一个实验都是独一无二的。在效果，每个任务都需要一套新的工具集。日食组件模型可以共享和重新配置功能。

多样化的用户基础任务软件具有广泛的用户从计算机一害羞的领域专家技术工程师的普通成员公共。日食框架简化了世界级的，可访问的创建 UIS，规模，满足用户的需求。中央管理与任务软件所做的大部分工作是名称暗示，任务关键。因此，部署的软件必须密切合作，以确保它是一致的和正确的。EquinoxP2 更新设施，确保正确的组件部署。超越大师及其使用 Eclipse RCP 设施迁移到 EclipseRCP 有更深远的影响。例如，利用日食组件模式已经从根本上改变了美国宇航局团队建设的方式系统。组件是独立开发的地理和组织上分散的团队。这些组件必须集成在一个连续的依据。事实上，所有组件被定义在共同的术语（例如，插件和功能），以及它们之间的依赖关系，带来了结构并控制这一过程。美国宇航局在效果上创造了一个空间平台探索任务软件，并将团队正在堵漏。优

质的插件可从蚀工程的财富在其他地方，美国航空航天局的团队把巨大的块现有的代码。例如，他们曾写过自己的更新机制。现在他们可以使用一个在月食。早期的软件有它自己的用户界面框架和机制。这些已经取代了月食工作台模型。整个团队现在可以更加关注的问题任务软件，较少的基础设施和中间件的写作复杂的应用程序。在 NASA 的 Eclipse RCP 的使用迅速蔓延。球队现在为未来的火星探测车任务的一个更为雄心勃勃的工具工作作为在月球和地球上使用的先进的机器人系统的许多工具。

## 2 Eclipse RCP 概念

月食环境很丰富，但也有一些概念和机制这是必不可少的日食。在本章中，我们将介绍这些概念，定义一些术语，并将这些概念和术语在技术上细节。最终的目标是让你展示如何在身体上与之相适应和概念。

即使你很熟悉的月食，你可能想通过这一翻转第一章，以确保我们有一个共同的理解和术语的基础。写作是从 RCP 应用程序只是写插件微妙的不同。你有机会来定义更多的外观和感觉，品牌，和其他基本要素。了解这些基本原理，使您得到最出的平台。有了这样的理解，你就可以读懂其他的了这本书，看看日食如何适合你的世界。

### 2.1 一个社区的插件

在 1 章中，“月食作为一个丰富的客户平台”，我们描述的本质月食及其作为一个组件框架的作用。功能的基本单位该框架被称为插件（或在 OSGi 术语一束），模块化单元在 Eclipse。每一个东西都是一个插件。一个 RCP 应用程序是一个收集插件和它们运行的框架。一个 RCP 开发组装一个集合的插件，从日食基地和其他地方，并补充说在他或她写的插件。这些新的插件包括应用程序一个产品定义以及它们的域逻辑。除了解如何蚀管理插件，重要的是要知道它的存在 16 章 2•Eclipse RCP 概念插件来使用，以及如何使用它们，以及如何构建自己如何建立它们。

小套插件很容易管理和讨论。作为插件池然而，在应用程序中，分组抽象是需要帮助的隐藏一些细节。日食小组定义了几个粗集的插件，作为图 2-1 所示。在图表的底部是 Eclipse RCP 插件上的一小部分前一个 java 运行环境（JRE）。对自己的 RCP 是一样的基本操作系统或 java jre 本身是等待被添加的应用。扇形向上的图是一个 RCP 应用一些书面由你，一些人，和一些由日食小组。Eclipse 平台，传统的蚀作为一种发展环境，它本身就是一个高功能的 RCP 应用程序。如图 2-1 所示，IDE 平台需要在 Eclipse RCP 插件。插入 IDE 平台是 Eclipse 软件开发工具包（SDK）的 java 和插件工具和公司和开源社区编写的其他数百个工具。这种模式继续。在 Eclipse RCP 和你一般的形状产品是一样的都是刚刚套的插件，使一个连贯整个。这些主题的一致性和一致性复发整个日食。为 Eclipse RCP 插件设置内部细节图 2-2 所示。这些插件形式的 RCP 应用的基础。在这里我们看到了一系列相互依存的插件提供的各种功能，如在标注框指出。我们可以放大任何图 2-1 和插件集看到相同基本一致结构。你事实上是免费的切片和切块的 RCP 本身或任何其他插件的设置以满足您的需要只要相关插件依赖关系满意。在这本书中，我们集中在 RCP 应用程序完全使用 RCP 插件设置。

管理的依赖关系是一个很大的组成部分，建立一个日食应用。插件是自我描述，并明确列出了其他插件或功能必须为他们经营。OSGi 的工作是解决这些依赖关系和编织的插件一起。需要注意的是，这些关系是很有趣的不是因为月食而是因为他们代码中隐含和结构的插件。月食让你的依赖明确，从而有效地管理它们。

### 2.2 内插插件

现在，你已经看到了 10000 和 1000 英尺的意见，月食，让我们下降下降到 100 英尺，并期待在插件，基本构建块。一插件是一个集合的文件和一个清单，描述了插件和它的关系其他插件。图 2-3 显示 org.eclipse.ui 插件布局。第一件事注意的是，该插件是一个 java 文件（JAR）。作为一个瓶子，它有一个 manifest.mf。清单中包含一个插件的描述及其与其他的关系到全世界。插件可以包含代码以及只读内容，如图像，网络网页，翻译邮件文件，文档等。例如，用户界面图 2 中的插件在组织/

用户/用户界面/界面/……目录结构中有代码在 `about.html` 图标/和其他内容。注意，插件也有一个 `plugin.xml` 文件。从历史上看，这是执行的相关信息已经储存在 `manifest.mf` 家。这个其在任何扩展和扩展点声明的家由插件贡献。

### 2.3 把一个系统放在一起

所有的这些插件都漂浮着，一个日食系统的样子磁盘吗？图 2-4 显示了一个典型的 RCP SDK 的安装。最上面的目录是安装位置。它包括一个插件存储，一些引导代码和一个发射器，`eclipse.exe`，这是用来启动 Eclipse。插件存储（插件目录）包含一个目录或为每个插件。按惯例，文件系统名称匹配的标识符插件，并遵循其版本号。每个插件包含它的文件和如前面所述的文件夹。配置位置（配置目录）包含配置定义。这个定义描述了要安装哪些插件和运行。配置位置也可用于存储设置的插件和其他数据，如偏好和缓存索引。默认情况下，配置位置是安装位置的一部分。这是方便标准的单一用户安装在用户有完全控制的机器上。产品和共享，或多组态，在 UNIX 系统上安装的可能，然而，把配置在其他地方，如当前用户的主目录。

### 2.4 OSGi 框架

日食插件组件模型是基于点的执行 OSGi 框架的 r4.2 规范（<http://www.osgi.org>）。你可以看到它图 2-5 底。简而言之，OSGi 规范形成一个框架用于定义、组合和执行组件或束。认为束作为插件的实现。术语插件是用来指在月食的组成部分，并使用整个文档和工具。有没有基本或功能上的差异之间的插件和日食束。两者都是分组，提供和管理的机制代码。事实上，传统的日食插件类只是一个薄的，可选的层便利功能上 OSGi 束。月食，一切都是束。因此，我们使用的术语和走动呗，”插件是一个捆绑。捆绑是一个插件。他们是一样的东西。”这是方便想 OSGi 框架提供的组件模型 java；就是把它作为一个设施为基础的 JRE 相同的水平。OSGi 框架管理束和代码管理和分割每一个包的类装入都有它自己的类装入器。一束的路径基于其清单中声明的依赖关系动态构造的。清单定义插件是什么和它取决于什么。所有插件都是自我描述的。图 2-5 所示的 `manifest.mf` 给 `org.eclipse.ui` 插件插入式标识，或捆绑符号名称和版本。常用的做法是使用 java 包的名称如标识符和 `org.eclipse.ui` [专业。小。服务。预选赛]元组的版本号。身份证和版本配对到唯一标识插件。然后，使用对表示依赖关系。你可以看到这需要的束头清单的 UI 插件需要运行时，JFace 和 SWT 插件。在 Eclipse 中，OSGi 的主要作用是结合在一起的安装插件，让他们进行互动和协作。严格管理依赖关系和类路径使紧，显在束相互作用的控制从而创造更灵活、更容易的系统由。

### 2.5 春分

从历史上看，日食运行时包括插件模型和各种功能元素。正如你所看到的，插件或捆绑模型已经移动了以 OSGi 层。这是由彼岸项目实施的。大多数先前由 Eclipse 运行时提供的 `otherfunctionality` 现在也部分的 Equinox 项目。所以我们区分基础标准 OSGi 实现之间和本节讨论的附加值函数元素。

#### 2.5.1 应用

像 JVM 和标准的 java 程序，OSGi 系统一定要告诉你做。月食，某人必须定义一个应用程序。一个应用程序是非常就像在普通 java 程序 `main()` 方法。后点开始，它查找并运行指定的应用程序。应用程序使用扩展定义。应用扩展标识一类以用作主要入口点。当你运行月食，你可以指定一个应用程序运行。一旦被调用，应用程序是完全控制的蚀。当应用程序退出，月食关闭。

#### 2.5.2 产品

一个产品是一个应用程序的层次。你可以通过只指定一个应用程序，但产品品牌背景（例如，飞溅屏幕和窗口图标）和各种自定义（例如，喜好和配置文件）就会错过。一个产品的概念捕捉到这个扩散信息用户理解和运行的一个概念。

#### 2.5.3 扩展注册表

OSGi 提供了定义和运行单独的组件和机制一种支持跨束协作的服务机制。点增加了插件扩展注册处之间的关系的机制。插件可以通过声明一个扩展或配置来打开自己的配置扩展点。这样的插件基本上是说，“如果你给我以下信息，我会做……”其他插件，然后贡献所需的信息扩展点的扩展

形式。这个典型的例子是 UI 插件和扩展其 `actionsets` 点。简化一些，动作集合是如何在菜单和工具栏的用户界面的对话条目。Eclipse 用户界面暴露扩展点 `org.eclipse.ui.actionSets` 说，“插件可以 `actionsets` 扩展定义行动的 ID 标签，图标，和一个实现该接口的类 `IActionDelegate`。用户界面将呈现给用户的标签和图标，当用户点击项目，UI 将实例化特定动作类，丢给 `IActionDelegate`，`run()`并调用它的方法。”图 2-6 显示这种关系的图形。扩展到扩展点关系的定义在文件中使用称其。每一个参与式插件都有一个文件。在这种情况下，`org.eclipse.ui` 的 `plugin.xml` 包括以下：`actionsets` 扩展点的合同具有如下：界面介绍标签“调试聊天”随着 `debug.gif` 图标。当用户点击的作用，`debugaction` 实例化类及其 `run()`方法称为。这个看似简单的关系是非常强大的。用户界面有效打开菜单系统的实现，允许其他插件贡献菜单项。此外，用户界面插件不需要知道关于提前的贡献，并没有运行代码的贡献——一切都是陈述和懒惰。这些变成了关键特性作为一个整体的注册表机制和月食。其他的一些特点值得注意的是这些：

○扩展和扩展点是在 Eclipse 的广泛应用一切从贡献的意见和菜单项目连接帮助文件并发现过程资源变化的建设者。

○机制可以用来贡献代码或数据。

○机制说明插件没有加载任何连接他们的代码。

○机制是懒惰，没有代码加载到需要的时候。在我们的例如 `debugaction` 类加载，只有当用户点击了行动。如果用户不使用该操作，则该类没有加载。

○这种方法的尺度，使呈现的各种方法，范围和过滤的贡献。

## 2.6 Standard Widget Toolkit (SWT)

坐在 OSGi 和春分是 SWT。SWT 是一个低级别的图形提供标准的用户界面控件，如列表、菜单、字体和颜色等，也就是说，一个库，公开了底层的窗口系统所提供的。作为 SWT 团队指出，“SWT 提供快捷的便携式访问 UI 设施该源码对它实施。”

这相当于 SWT 被放在现有的窗口系统，薄薄的一层设施。SWT 不哑下来或粉饰底层窗口系统而暴露出它通过一致的，便携式的 `java API`。SWT 是可用的在各种各样的窗口系统和操作系统。应用程序使用 SWT 是可移植的所有支持的平台。

SWT 真正的诀窍是使用原生部件尽可能。这使外观和感觉的 SWT 应用相匹配的主窗口系统。因此，基于 SWT 的系统是便携式和本土。注意，SWT 并不取决于春分或 OSGi。这是一个独立的图书馆可以 Eclipse RCP 之外使用或。

## 2.7 JFace

而 SWT 提供了访问控件的窗口系统定义，JFace 添加 UI 概念结构和设施。用户界面团队描述 JFace 如下：“JFace 是一类用于处理许多常见的 UI 工具包用户界面编程任务。JFace 窗口系统在它的 `API` 和独立实施，并设计使用 SWT 而不隐藏它。”它包括了一系列的用户界面工具包组件，从图像和字体注册，支持文本、对话框、数据绑定、和框架的偏好向导向运行报表的长期运行操作。这些和其他的 JFace 用户界面结构，如行动和观众，形成的基础上的日食用户界面。

## 2.8 UI 的工作台

正如 JFace 添加结构，SWT，工作台增加表达和协调在 JFace。对用户来说，工作台包括一些视图和编辑器布置在一个特定的布局。特别是工作台

○提供基于 UI 的可扩展性的贡献

○定义了一个强大的用户界面范例的 Windows，观点，看法，编辑，和行动

### 2.8.1 基础性贡献

而 JFace 介绍行为、偏好、巫师、窗口，等等，这工作台提供的扩展点，允许插件来定义这样的用户界面元素以声明方式。例如，向导和偏好页扩展点只是薄贴面在有关 JFace 构建。然而，更多的是，使用扩展建立一个用户界面有一个基本的在复杂性和性能方面对用户界面的可扩展性的影响。声明扩展使集合的描述和操作如我们讨论过的行动集的贡献。例如，工作台的功能机制支持逐步披露功能通过过滤操作，直到触发其定义动作集合。你的应用程序可能会有大量的行动，但用户只看到他们的感兴趣的用户界面的增长与用户的需求。由于所有这些扩展处理懒洋洋地，应用程序也变得更好。随着你的用户界面变得更加丰富，它包括更多的视图，编辑和操作。没有声明的可扩展性，这样的增长需要额外的加载和执行代码。这增加了代码块和启动时间，而应用程序不规模。随着扩展，在它的时间之前没有加载代码。

### 2.8.2 的观点，意见，和编辑

工作台显示为窗口的集合。在每一个窗口工作台允许用户以同样的方式组织他们的工作你会整理你的桌子，你把类似的文件放在文件夹里把它们堆在桌子上。透视是一组视图的可视容器和内容编辑器显示给用户的一切是在视图或编辑器和从一个角度看。

用户在以下方式组织内容的观点：

○栈的编辑与其他编辑。

○堆栈视图与其它视图。

○分离的观点主要从工作台窗口。

○调整视图和编辑器和最小化/最大化编辑和查看堆栈。

○快速创建视图，停靠在一边的窗口。的角度来看，支持一组特定的任务，提供一个限制集对相关内容创建的视图和支持操作集以及快捷方式向导、其他相关视图和其他相关观点。用户可以例如，转换之间的观点，在开发代码，交易股票，工作的文件，和即时通讯。这些任务可能有独特的布局和内容。

### 工作台

该工作台提供的用户界面构建块，使日食应用很容易写的，易于使用，可扩展的，可扩展的。使用的主要优点之一 Eclipse RCP 是你得到重用 UI 积木让你受益要集中在你的领域，而无需重新设计车轮。这是显而易见的事实上，在第二部分中开发应用程序所需的双曲线相对较少的代码。但这项工作只是划伤了什么是可能的表面。工作台更强大。它由 50 多个扩展点和 350 多个接口组成类。这将需要一整本书，以公正的功能提供。而比尝试一个广泛的部分覆盖，下几章重点的部分工作台，RCP 应用程序是必不可少的。他们潜入他们的原料药和使用他们解决问题的动机在各种真实场景的双曲线。你应该远离这本书的一部分，有一个坚实的理解工作台的结构和如何控制和自定义操作。

## 3 工作台的顾问

在第二部分中，你得到了一个方法，在该方法中使用的工作台顾问在实际应用中，但你可能已经注意到，许多顾问的方法未使用。本章探讨了这些额外的原料药，并解释了如何他们可以在你的应用程序中使用。另一个你开始得到的区域

熟悉的是工作台；我们结束本章的概述工作台及其扩展点。

### 15.1 工作台顾问

你也许还记得在第二部分，当双曲线应用开始，其主要工作是电话 `platformui createandrunworkbench`（显示，`workbenchadvisor`），在下面的代码片段所示。这个有点不起眼的法的力量你的用户界面开始工作台。这导致了创造，配置，和应用程序窗口打开。然而，工作台，不知道如何应用程序的行为或者看一下，工作台顾问进来的地方。正如他们的名字所暗示的，顾问给工作台建议。在这样做时，他们会影响用户界面包含和它的外观和感觉。而不是改变工作台的行为说，通过子类化，这工作台聚集的顾问，让他们参与的运行工作台。正如你看到的双曲线，当工作台打开窗口，它要求一个顾问，以确定它是否应该包括一个菜单酒吧 `workbenchwindowadvisor` `prewindowopen()`方法调用。有三种类型的顾问，并且每一个是其特征的一部分它给了建议的工作台：`workbenchadvisor` 这个顾问提供的应用水平的建议。它参与在工作台本身的启动和关闭过程中，有一个运行工作台/运行的日食应用。`workbenchwindowadvisor` 这个顾问提供窗口水平的建议。它参与在显示或隐藏菜单、工具栏、状态行和配置窗口中显示的控件。有一 `workbenchwindowadvisor` 实例为每个窗口。

`actionbaradvisor` 这个顾问提供建议和帮助窗口定义在菜单、工具栏和状态行中出现的动作窗口。有一 `actionbaradvisor` 每个窗口实例。每个顾问都有一个相关的配置者提供特权访问的工作台，工作台窗口和窗口的动作栏（例如，菜单，工具栏，和状态线）。那里的每个顾问型配置器。由于 `configurers` 提供特权访问的工作台，他们不应该被传递到其他插件。三种不同类型的 `configurers` 是 `iworkbenchconfigurer`，`iworkbenchwindowconfigurer`，和 `iactionbarconfigurer`。下面的代码片段演示如何在其相关联的顾问中使用：顾问们有另一个非常重要的角色，他们定义了顶级的整合产品点。例如，顾问决定的名称在顶级菜单、工具栏和状态行组和分隔符。插件贡献的产品必须使用这些名称作为积分点时定义和放置自己的行为。这是第 17.2.2 进一步讨论，“允许捐款。”

顾问是非常密切相关的概念的产品。虽然这关系不是技术上强制执行的，例如，产品扩展点不包括来自外界的细节，他们通常被视为一个。您通常有一组顾问为您的产品。当你看，产品定义的产品延伸点指的是一个应用程序（例如，`iapplication`），每个应用程序是一组相关的顾问。

#### 15.1.1 平台生命周期

要了解如何使用顾问来配置工作台的全部范围，让我们来看看工作台的生命周期和点不同的顾问参与。

在后台，当 `platformui createandrunworkbench()`叫的。

工作台执行以下高级步骤：

- 初始化一个异常处理程序来捕获未捕获的异常。
- 打开主窗口，默认控件（工具栏、状态栏，视角切换、角度），并恢复任何已保存的状态。
- 取下屏幕。
- 打开一个或多个窗口和初始化与界面设置，被保存从上一次会话。
- 运行 SWT 的事件循环。
- 当 `iworkbenchwindow`。（）称，Windows 的状态被保存，然后窗户被关闭。

图 15-1 显示工作台在典型的运行时交互和各种顾问。工作台运行时，它允许 `workbenchadvisor` 参加通过调用 `workbenchadvisor` 开始。`initialize()`。这之前发生的任何窗口打开。每个窗口打开时，一个 `workbenchwindowadvisor` 创建和关联窗口实例。这 `workbenchwindowadvisor` 实例在整个窗口的生命周期中进行协商。是的 `actionbaradvisor` 由 `workbenchwindowadvisor` 窗口打开之前，参与在填充菜单，工具栏，并且每个窗口的状态行。下一个部分显示了每个类型的顾问的生命周期和原料药的更多细节。

#### 15.2 workbenchadvisor



双曲线的 `workbenchadvisor` 在整个应用程序只有一小部分。它初始化设置的方法和 `workbenchadvisor initialize()` 然后连接聊天的侦听程序来处理传入的聊天记录。这是非常典型的 `workbenchadvisor` 初始化应用程序生命周期的设置和清理当工作台关闭。这是很有诱惑力的使用在您的 `workbenchadvisor` 生命周期方法来触发您的应用程序的附加部分的初始化。抵制这种冲动！工作这里所做的是在显示应用程序的第一个窗口的路径上——你做的工作越多，你的用户就越等着看你的申请。这不能强调不够：尽量在 `workbenchadvisor` 初始化代码。`workbenchadvisor` 的 API 方法分为三类：生命周期允许顾问参与启动和关闭的应用例外和懒惰使顾问可以参加一个例外当应用程序空闲时发生配置允许的顾问，以确定全球的默认值应用程序设置

### 15.2.1 生命周期 API

你应该使用生命周期方法来初始化，恢复，然后保存和关闭您的应用程序的下降方面。在表 15-1 每个方法的说明为每个方法中典型的工作类型提供一个提示。

#### 15.2.1.1 `iworkbenchconfigurer`

当 `workbenchadvisor` 方法初始化（`iworkbenchconfigurer`）是称，它提供了一个 `iworkbenchconfigurer`。用于配置的配置器工作台，并提供特权访问它，例如，告诉它要保存工作台设置或注册图像与工作台。的配置器可在 `initialize()` 方法被调用

`workbenchadvisor` 方法 `getworkbenchconfigurer()`。

的 `iworkbenchconfigurer` 主要用于

○设置和获取属性的配置器使用 `setData`（字符串对象）

和 `GetData`（`String`）方法。这是有用的传递数据从

`workbenchadvisor` 的 `workbenchwindowadvisor` 或 `actionbaradvisor`。

○关闭工作台的情况下，急救，看看它的关闭通过

`emergencyclose()` 和 `emergencyclosing()` 方法。

○确定工作台应退出时关闭最后一个窗口，通过的 `setexitonlastwindowclose`（布尔）。如果工作台不退出和事件循环保持运行，您可以打开另一个窗口使用 `iworkbench.openworkbenchwindow`（`String`，`IAdaptable`）或关闭工作台 `iworkbench()` 调用。由于顾问有特权访问的工作台，它是最好的保持无论是 `configurers` 和顾问私人并没有传递给其他插件。

#### 15.2.1.2 关闭工作台

工作台可以关闭 `workbenchadvisor` 方法后的任何时间 `initialize()` 叫做。关闭工作台有三种方法：`iworkbench()` 这个方法执行正常关机。双曲线的 `actionfactory.quit_action` 调用此方法退出工作台。`iworkbench.restart()` - 此方法执行正常关机使应用程序立即重新启动。`iworkbenchconfigurer.emergencyclose()` - 这种方法会导致 Eclipse 优雅地退出，但立即。它被称为一个致命的错误发生时应用程序不能正常关机。工作台尝试在至少保存用户的设置。

#### 15.2.1.3 Workbench 的偏好

除了配置设置可在 `iworkbenchconfigurer`，还有额外的工作台的喜好来控制的外观和感觉应用。这些措施包括设置透视栏的位置，快速查看酒吧的位置，并与传统的弯曲的标签的使用。表 15-2 列出对 RCP 应用程序最受大众欢迎的偏好。看到 `org.eclipse.ui` 一个完整的清单 `iworkbenchpreferenceconstants` 接口。的 `workbenchadvisor` 可以设置在 `initialize()` 方法这里显示的：

`Platformui.getpreferencestore()`

`setValue (.iworkbenchpreferenceconstants.show_traditional_style_tabs, 真的)；`

正如 13.4 节所述，“添加帮助内容，”偏好也可以使用产品偏好定制文件初始化。例如，添加以下为您的产品的喜好定制文件显示的观点在左栏快速查看酒吧和使用弯曲的标签。的优势使用自定义文件，它允许可以构建不同的产品从您的应用程序中有不同的值，这些偏好。

`org.eclipse.rcp.hyperbola/preferences.ini`

`org.eclipse.UI / dock_perspective_bar = 左`

```
org.eclipse.UI / show_text_on_perspective_bar = false  
org.eclipse.UI / show_traditional_style_tabs = false
```

### 15.2.2 例外和懒惰的 API

工作台运行一个标准的 SWT 的事件循环。这可能是错误的应用程序，一个组成插件，或关键的情况（例如，运行内存）在该系统中，可以将异常引发为事件循环代码。当这发生时，工作台的电话 `workbenchadvisor eventloopexception()` 事件循环，在下面的代码片段所示。表 15-3 还列出了例外和懒惰在 `workbenchadvisor` 可用的方法。虽然这通常意味着该应用程序处于无效状态，但 `workbenchadvisor` 的默认实现日志的例外，允许应用程序继续运行。您的顾问可以将此异常处理扩展到例如，提示用户采取一些行动，如记录错误报告，或通过关闭应用程序。在紧急情况下关闭工作台 `workbenchadvisor` 应该叫 `iworkbenchconfigurer` 方法 `emergencyclose()`。为了避免造成这样的突发事件，你应该写你的代码的防守一本地捕获和处理异常。这可能是具有挑战性的，因为你做在调用第三方代码时抛出的异常不具有控制权。以的帮助下，运行时提供了平台的方法运行（`isaferrunnable`），方便机制的安全运行不受信任的代码。的 `isaferrunnable` 接口包含两个方法：`run()` 和 `handleexception`（错误）。你只需把电话给不受信任的代码在 `run()` 方法。当 `isaferrunnable` 执行的平台，并可以捕获所有异常把 `handleexception()` 方法。自平台自动日志的例外，你 `handleexception()` 应该侧重于恢复从问题如果可能的话。下面的代码段显示了一个典型的使用 `isaferrunnable` 当通知听众。如果侦听器失败，该代码将移动到下一个侦听器，相信失败已被记录。

## 4. 观点，观点和编辑

在第二部分中描述的双曲线的 UI 是故意简单，但它仍然触及工作台的用户界面范例的大部分中心功能，特别是观点，意见和编辑。如果您需要在 Eclipse IDE 的 UI 仔细一看，你可以看到更多的可以做这些建筑块比什么我们已经使用过双曲线。例如，在图 16-1 所示，工作台允许多个窗口打开，每个窗口有多个窗口视角。您还可以定义为完全可定制的观点用户视图可以被移动、最小化、关闭或安排为快速视图。RCP 应用可以利用这个力量提供丰富的用户体验和多层次的没有压倒用户的功能。在这一章中我们将双曲线有多角度、多窗口，并使用高级视图操作技术。具体，我们向你展示如何

- 添加透视
- 使用编程的视角和 API
- 使用多个实例的视图和粘性的观点
- 开放和跟踪多个工作台窗口
- 连接部件
- 添加支持拖放到编辑区

### 16.1 个方面

透视组在一起，并组织与特定任务相关的用户界面元素或工作流。例如，在 Eclipse IDE，java 透视图包含的观点编辑 java 源文件，在 Debug 透视图包含视图 java 程序的调试。您可以在同一窗口中有多个视角，允许您切换任务而不必更改窗口。显示角度可用于 RCP 应用程序，让我们走过增加新的观点，“自由移动”和“调试”的步骤，以双曲线。自由移动的角度允许用户移动联系人视图。Debug 透视图，如图 16-2 所示，显示控制台与所有传入和传出消息的 XMPP 输出。

#### 16.1.1 添加透视

第一步是定义新的透视扩展和透视工厂类。开放的双曲线的插件编辑和添加两个新视角扩展对现有 `org.eclipse.ui.perspectives` 扩展点，如图所示 16-3。步骤是相同的在 4 章，“双曲线的应用”，“5，从双曲线的原型。”注意到原来的双曲线视角没有图标。一个图标是不需要的，因为透视的概念是从来没有暴露用户刚刚看到双曲线的窗口。在这个新的双曲线视角暴露；因此，你需要图标。对

于每一个新的观点和默认透视，提供一个有意义的名称和图标。类的属性定义，生成初始的 `iperspectivefactory` 页面布局和可见的操作集。你可以使用通常的伎俩在类的属性上，启动新的类向导。名字的角度类 `perspectivefreemoving` 和 `perspectivedebug`。自由移动的视角是原双曲线角度相同除联系人视图具有标题栏和可移动。所以，而不是调用 `ipagelayout.addstandaloneview()`，叫 `ipagelayout.addview()` 如下所示：默认情况下，Eclipse 视图是关闭的。重写此行为，自由移动角度检索 `iviewlayout` 和电话 `setcloseable`（假）。决定允许视图关闭取决于用户界面是如何构造的。防止视图被关闭，让用户简单。如果你决定为了让视图被关闭，确保用户知道如何让它们重新获得。看到那打开的视图操作实例 16.2.1 节。请注意，`createinitiallayout()` 方法传递一个 `ipagelayout`。透视使用布局对象来控制页面的外观和在何处的观点去。

#### 16.1.2 添加调试角度和控制台视图

这个日食平台包含一个非常有用的控制台插件，可以用来添加一个控制台 RCP 应用程序。在这里，我们使用控制台视图通过这个插件在调试的角度。控制台支持高级功能，如行着色和超链接。添加控制台双曲线，按照第 13 章的指令，“增加帮助”，增加插件到你的目标，但没有添加的帮助插件，添加以下：

- org.eclipse.text
- org.eclipse.jface.text
- org.eclipse.ui.console
- org.eclipse.ui.workbench.texteditor

### 16.2 视图和编辑器

在开发第二双曲线，你学会了使用基本知识和观点编辑与两者的区别。通常，编辑器显示主您的应用程序的内容，而视图支持此附加导航或上下文相关的任务正在与编辑完成的任务。如果你熟悉 Eclipse IDE，你已经知道，双曲线我们有管理视图隐藏的四个标准 IDE 机制和编辑。例如，用户打开任意视图的想法是不暴露，而在 Eclipse IDE 有一个窗口>显示视图菜单条目允许用户打开任何视图。正如许多与观点、观点和编辑有关的事情，这取决于个人的 RCP 应用程序来决定如何让这些给最终用户。这个本节的目的是回顾一些先进的 RCP 相关视图和编辑器特征。

#### 16.2.1 多个实例相同的观点

正如我们已经看到的双曲线，打开一个编辑器的多个实例，这是可能的每一个独特的编辑输入。它也可以打开多个实例一个视图。想象一个更先进的双曲线，允许你连接到多个服务器一次。用户可以管理不同的联系人列表或不同的在同一应用程序的登录。目前，然而，双曲线会每个连接需要一个联系人视图。在同一视图中打开多个实例有 2 种技术同样的观点。无论使用的技术，第一步是编辑双曲线的插件定义和改变的 `enablemultiple` 财产触点视图扩展点为真。一旦完成，多视图可以直接打开的角度看工厂，如下图所示。该 `addview()` 方法看身份的争论以一个合格的视图格式为主要身份：`secondary-id`。次 ID 必须是独特的观点。

#### 16.2.5 连接部件

有一个很好的机会，你的用户界面部分的意见和编辑需要沟通。例如，双曲线联系人视图可以显示联系人在大胆的时候，一个聊天编辑器，该联系人是活动。在工作台上有三种技术用于通信部分。首先让我们回顾一下这些技术，然后我们再看看如何使用这些方法实现联系人条目加粗技术。使用选择的 `iselectionservice` 允许视图和编辑登记他们的选择与工作台，从而使其他部分聆听对选择的变化和适当的反应。要发布选择，使用 `iworkbenchsite.setSelectionProvider` (`iselectionprovider`) 的方法，和订阅选择，使用 `iselectionservice` 方法 `addSelectionListener` (`iselectionlistener`)。这是一个很好的技术因为它允许分离的部分。我们已经使用这种技术将双曲线行动联系人视图，确保当选择的变化，正确的动作。部分听众可以通过听事件来连接部分当一个部分被关闭，打开和隐藏。使用 `ipartservice` 注册部分事件。再次，这是一个很好的技术，因为它保持部分解耦。这是用下面的技术

来实现的双曲线加粗接触的例子。

直接沟通，而选择和部分服务允许任何一部分听和反应的变化，也可以用直接连接有特定的视图调用或打开其他视图或编辑器（例如，使用在 `iworkbenchpage` 打开或关闭件的方法）。例如，双曲线加粗功能可以让 `chateditor` 实施通知 `contactsvie` 打开时。这种技术并不理想，因为它放置在部件之间的紧密耦合。实施大胆接触当聊天的过程中，接触查看要知道何时一个聊天编辑器是打开和关闭和大胆的接触，因此。最好的技术是为联系人视图注册一部分听众使用 `ipartservice` 记住聊天编辑器打开。当一个部分被打开或关闭时，标签提供程序被刷新，并且该标记提供程序聊天列表是协商决定是否应该是一个接触的大胆。

代码的变化很简单，这里描述的是：

- 添加一部分听众的联系人视图跟踪聊天编辑打开和关闭。
- 登记和注销部分听众。
- 改变标签装饰设置项的字体，和使用活动清单聊天编辑要确定一个联系人的字体。部分听众是作为一个领域的 `contactsvie`。它记得那套打开聊天 `openeditors` 和刷新标签当聊天编辑器打开或关闭。

### 16.3 多工作台窗口

工作台支持打开多个顶层窗口。这些窗口通过平台管理，和 `workbenchwindowadvisor` 参与每个窗口的生命周期。打开一个新的工作台有 2 种技术从一个 RCP 应用程序窗口：

- 使用 `actionfactory.open_window` 或 `openinnewwindow` 行动。这些动作打开一个新窗口，使用透视当前显示的透视行动是运行的窗口。
- 叫 `iworkbench` 方法 `openworkbenchwindow` (`String`, `IAdaptable`)。此方法打开一个新窗口，并显示标识透视。

#### 16.3.1 窗口导航菜单

当应用程序允许创建多个窗口时，它应该提供一个方便的方式在窗口之间导航。工作台提供了可重用的显示一个打开窗口列表的菜单。选择一个窗口使它得到集中。要为应用程序添加“窗口列表”菜单，请添加“贡献”—`itemfactory.open_windows` 项目菜单，如下面的代码片段所示在图 16-7：

### 5. 行动

在我们的例子中，行动的概念要比我们所涵盖的要多的多到目前为止。6 章，“加入行动”，“以解释为什么双曲不使用声明性行为，一个标准的技术在月食。总的来说，行动双曲线的定义和编程的 `actionbaradvisor` 放置。这种技术是简单和理想的小型，封闭的应用程序。在 14 章，“添加软件管理，”我们谈到更新部署的应用程序增加新的插件。这些新的插件如何增加他们的行为？这个答案在于工作台的支持，声明性的行动。本章分为 2 个部分。第一次说明何时以及如何使用声明性的动作，并给出了有用的提示和技巧的写作产品—质量行动。具体地说，我们将向你展示如何

- 决定使用声明性和纲领性的作用
- 添加声明性行动双曲线
- 使用重定向操作定义可插拔的行为
- 写正确轨道的选择行为
- 显示进度为长期的行动
- 用行动中的替代方法，如显示文本，创建多列工具栏，并将控件添加到工具栏
- 添加贡献状态行

### 17.1 概述

在介绍声明性操作之前，让我们先退一步看看行动都是关于。行动的主要作用是向用户公开应用程序行为。当你点击一个菜单项或工具栏项或调用一个键序列，一个动作是运行。此外，你可以

建立一个系统，以便相同动作是放置在一个菜单和一个工具栏，并绑定到一个关键的序列动作让你从布局中分离行为。行动也决定他们是否被启用，找出哪些元素来工作，运行该工作，并显示结果。

下面是一个清单的基本职责的行动。当你探索各种各样的定义和使用行为的方法，请记住这些责任不管它们是如何定义和使用的，都适用于语法不同。放置操作可以放置在工作台的许多领域：在视图中，编辑上下文菜单和顶层菜单和工具栏。当他们被创造，它们被给予一个参考的容器中，他们被放置，允许他们查询选择或访问其他信息。渲染有几种类型的动作，例如，菜单，工具栏按钮，无线按钮，下拉菜单按钮等。当你创建一个行动，你必须决定它是什么类型的行动，以及它将在用户界面中呈现。行动还包含基本信息，如标签、图标，和一个工具提示，它显示和描述对用户的作用。输入启用操作的一组对象通常要做的工作。在相关的用户界面部分中定义了一个动作的输入。对于例如，如果一个行动是在一个视图中，相关的选择是小部件在视图中。工作台窗口中的一个动作使用选择来自活动视图或编辑器。作为一个行动的上下文的变化（例如，它的输入变化或容器状态更改），该操作必须更新其启用状态。行为当一个动作最终运行时，它执行一些操作它的输入。如果该动作是长时间的，则应该向用户显示进度。绑定一个动作可以通过键盘来控制它显式用户界面元素。行动也可以被绑定到其他控制的机制当他们运行。当你添加一个动作，你必须执行每一个责任。

图 17-1 显示了如何建立行动从 IDE 产品实现这些责任。将构建动作放在最顶层的工具栏和菜单中，它将被呈现为一个按钮的名称和图像，它的启用时，至少有一个项目被选中，其输入是选定的项目。其行为是建立选定的项目。

### 17.3 标准工作台操作

在 6 章中，你看到工作台定义了一组可重用的操作。这些行动被定义为 `org.eclipse.ui.actions.actionfactory` 内部类和实例化和作为普通的行动。`exitaction = actionfactory`。退出。创建（窗口）；可重复使用的行动已经预先配置的名称，图标，入侵检测，和行动的定义入侵检测系统，但这些属性可以被重写后的行动了。`exitaction.setText（“退出”）；//从出口离开`。从 `actionfactory` 创建的所有行动都 `iworkbenchaction` 实例。`iworkbenchaction` 延伸作用，加 `dispose()` 方法自当关联的窗口关闭时，必须删除实例。在执行应用程序的操作之前，请检查 `actionfactory` 定义相关的行动。再利用这些标准操作应用更一致的外观和感觉。