第1章 绪　论

1.1 软件工程概述

1.1.1 软件工程的概念与理论

什么是软件工程？简单来说，软件工程是将工程学的思想、方法与技术应用于软件开发过程的一门学科。1968年，在北大西洋公约组织举行的一次学术会议上，人们为了解决当时出现的软件危机问题，首次提出了软件工程这个概念，并将其定义为“为了经济地获得可靠的和能在实际机器上高效运行的软件，而建立和使用的健全的工程规则”。经过半个世纪的发展，软件工程的定义、思想与方法等理论在不断丰富和完善，相应的技术也在飞速进步。今天，软件工程已成为计算机科学与技术中一门独立的学科，对各行各业产生了深远的影响。

由于软件工程一直以来没有一个统一的定义，本书仅引用IEEE对软件工程的定义作为标准。

(1) 将系统化的、严格约束的、可量化的方法应用于软件的开发、运行和维护，即将工程化应用于软件。

(2) 对(1)中所述方法的研究。

软件工程的内容非常丰富，包括软件工程原理、过程、方法、模型、管理、环境及工具等，其中过程、方法和工具是软件工程的三要素。

具体而言，在*Guide to the Software Engineering Body of Knowledge*(2004)中，软件工程知识体系划分为以下10个知识领域。

(1) 软件需求(software requirements)。

(2) 软件设计(software design)。

(3) 软件构造(software construction)。

(4) 软件测试(software testing)。

(5) 软件维护(software maintenance)。

(6) 软件配置管理(software configuration management)。

(7) 软件工程管理(software engineering management)。

(8) 软件工程过程(software engineering process)。

(9) 软件工程工具和方法(software engineering tools and methods)。

(10) 软件质量(software quality)。

同时,该指南也指出了8个相关的学科领域,这些学科领域和软件工程学科存在一定的交集,能够帮助该学科的发展,以及帮助和该学科相关的一些工作。8个相关学科如下。

(1) 计算机工程(computer engineering)。

(2) 计算机科学(computer science)。

(3) 管理学(management)。

(4) 数学(mathematics)。

(5) 项目管理(project management)。

(6) 质量管理(quality management)。

(7) 软件人类工程学(software ergonomics)。

(8) 系统工程(systems engineering)。

1.1.2 软件工程的发展

软件工程是一门相对年轻的学科,它的发展伴随着软件和软件开发技术的更新。软件是由计算机程序的概念发展演化而来的,是在程序和程序设计发展到一定规模且逐步商品化的过程中形成的。软件开发大致经历了从程序设计阶段、软件设计阶段到软件工程阶段的演变过程。

1968年,“软件工程”概念首次被提出。20世纪70年代,人们开始在软件开发中运用一些工程思想,将工程技术与软件开发技术相结合,提出了简单的“瀑布模型”。然而随着软件规模的增大和开发难度的日益增加,“瀑布模型”遇到了瓶颈。

20世纪70~80年代,人们在不断的软件实践中逐渐提出了更多的模型,以解决遇到的各种问题。COCOMO、CMM等模型相继被提出,软件体系结构相关的研究和技术日益成熟。

与此同时,需求和设计工具、开发工具、测试工具和配置管理工具等也不断发展。随着计算机辅助软件工程(computer aided software engineering,CASE)的出现,软件开发的效率得到了进一步的提高。CASE是一套方法和工具,可使系统开发商规定应用规则,并由计算机自动生成合适的计算机程序。CASE工具和技术可提高系统分析和程序员的工作效率,其重要的技术包括应用生成程序、前端开发过程面向图形的自动化、配置和管理及生命周期分析工具。CASE的出现极大地促进了软件工程的发展。

20世纪90年代,面向对象思想渗透到软件工程中。基于面向对象的分析和设计模式、方法、技术和工具极大地丰富了软件工程的理论,使其日趋成熟。其中,典型的代表是UML设计方法和技术。

20世纪90年代末,随着《敏捷软件开发宣言》的发布,敏捷开发思想逐渐兴起。极限编程(XP)、Scrum、特征驱动开发等概念和过程也相继被提出,在实践中不断被完善。

21世纪以来,随着互联网时代的高速发展,以大数据、云计算、移动互联网和智能信息技术等为代表的新技术冲击着软件工程,带来了前所未有的挑战。新的技术变革必将引发软件工程领域更多的概念、思想和工具的出现。敏捷开发、精益思想、构件化软件开发、模型驱动体系结构等都可能引领未来的软件工程发展方向。

1.1.3 软件生命周期

一个软件产品或软件系统要经历孕育、诞生、成长、成熟、衰亡等阶段，一般称为软件生命周期(也称软件生存周期)。可以把整个软件生命周期划分为若干阶段，每个阶段有明确的任务，这样一来，规模宏大、结构复杂的软件及其开发变得容易控制和管理。通常，软件生命周期主要包括以下阶段。

可行性研究：主要目的是定义问题，确定软件的开发目标和分析其可行性，制订初步的开发计划。

需求分析：在确定软件开发可行的情况下，对目标软件系统需要解决的问题和需要实现的功能进行详细分析，形成需求规格说明书。

软件设计：根据需求分析的结果，对整个软件系统进行设计，分为概要设计和详细设计。概要设计旨在建立系统的总体架构，详细设计关注每个子系统和模块的内部实现细节。形成的软件设计说明书将为后续编码实现提供依据。

编码实现：根据软件设计说明书，将设计结果转换成计算机可运行的程序代码。在编码实现过程中必须要制订统一、符合标准的编码规范，以保证程序的可读性、易维护性，提高程序的运行效率和整个系统的稳定性。

软件测试：主要目的是发现软件产品中存在的缺陷，进而保证软件产品的质量。可以划分为单元测试、集成测试、系统测试和验收测试。

运行与维护：软件产品交付后，随着用户需求的增长或改变，以及市场环境的变化，软件产品的功能需要不断完善。为了保证软件产品的正常运行，需要进行一定的维护。

ISO 12207 标准(2020 版)为软件的生命周期过程提供了设计参考，并规定了 2 个协议过程(agreement process)、6 个组织项目支持过程(organizational project-enabling process)、8 个技术管理过程(technical management process)和 12 个技术过程(technical process)。

协议过程包括：

(1) 获取过程(acquisition process)。

(2) 供应过程(supply process)。

组织项目支持过程包括：

(1) 生命周期模型管理过程(life cycle model management process)。

(2) 基础设施管理过程(infrastructure management process)。

(3) 资产组合管理过程(portfolio management process)。

(4) 人力资源管理过程(human resource management process)。

(5) 质量管理过程(quality management process)。

(6) 知识管理过程(knowledge management process)。

技术管理过程包括：

(1) 项目计划过程(project planning process)。

(2) 项目评估和控制过程(project assessment and control process)。

(3) 决策管理过程(decision management process)。

(4) 风险管理过程(risk management process)。

(5) 配置管理过程(configuration management process)。

(6) 信息管理过程(information management process)。

(7) 测量过程(measurement process)。

(8) 质量保证过程(quality assurance process)。

技术过程包括:

(1) 业务和任务分析过程(business or mission analysis process)。

(2) 涉众需求和需求定义过程(stakeholder needs and requirements definition process)。

(3) 系统/软件需求定义过程(system/software requirements definition process)。

(4) 架构定义过程(architecture definition process)。

(5) 设计定义过程(design definition process)。

(6) 系统分析过程(system analysis process)。

(7) 实现过程(implementation process)。

(8) 集成过程(integration process)。

(9) 验证过程(verification process)。

(10) 过渡过程(transition process)。

(11) 确认过程(validation process)。

(12) 运营过程(operation process)。

软件生命周期的另一种表示如图 1-1 所示。

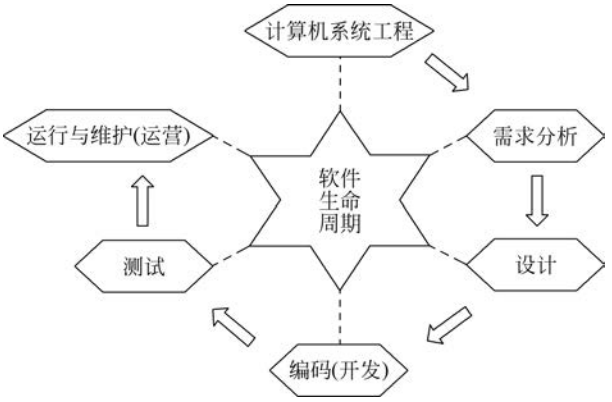

图 1-1 软件生命周期

1.1.4 软件生命周期模型

根据软件生命周期的概念,软件从产生到消亡都要经历需求、设计、编码、测试和维护等过程。这样的一个完整过程称为“生命周期模型”。常见的软件生命周期模型包括瀑布模型、增量模型、螺旋模型、快速原型模型、统一软件开发过程模型、敏捷过程与极限编程模型等。各个模型的特点如表 1-1 所示。

表 1-1　软件生命周期模型及特点

生命周期模型	主 要 特 点
瀑布模型	瀑布模型将软件生命周期中各活动(制订计划、需求分析、系统设计、编码、测试和维护)规定为依线性顺序的若干阶段,各项活动按照自上而下、相互衔接的固定次序,如同瀑布流水,逐级下落
增量模型	增量模型是把整个软件系统分解为若干软件构件,在开发过程中,逐个实现每个构件。如果发现问题可以及早进行修正,逐步进行完善,最终获得满意的软件产品
螺旋模型	螺旋模型将瀑布模型和快速原型模型集合起来,强调其他模型所忽视的风险,特别适合于大型复杂系统。该模型分为 4 个环节:制订计划、风险分析、开发实施和用户评价。4 个活动螺旋式地重复执行,直到最终得到用户认可的产品
快速原型模型	通过快速建立一个能反映用户主要需求的原型系统,让用户在使用和实践中提出改进意见,完善目标系统的概貌。开发人员根据意见快速修改原型系统,然后再次请用户试用直到用户认可原型的功能。至此,根据原型系统获取了用户的所有需求,可以形成需求规格说明书并进行设计与开发
统一软件开发过程模型(RUP)	统一软件开发过程模型是一种面向对象的、基于迭代思想的软件生命周期模型。按时间划分为 4 个阶段:初始、细化、构造和交付。这 4 个阶段按顺序依次进行,在每个阶段都可能有若干迭代
敏捷过程和极限编程模型	敏捷过程强调“个体和交互”胜过“过程和工具”,“可以使用的软件”胜过“面面俱到的文档”,“与客户协作”胜过“合同”,“响应变化”胜过“遵循计划”。极限编程是一种典型的敏捷过程模型,“极限”指把好的开发实践运用到极致,包括结对编程、短周期交付、测试驱动开发、集体所有等

1.1.5　软件工程实用工具

随着软件工程理论的发展,用于辅助软件开发的应用工具也大量涌现,统称为计算机辅助软件工程工具(CASE 工具)。这些工具通过自动化软件开发过程中的需求分析、系统设计、编码、测试及管理工作,大大降低了软件开发的成本,加快了软件开发的速度。下面将结合软件生命周期的各个阶段简要介绍相关工具实例。

在需求分析与系统设计阶段,软件开发团队通过多种手段收集原始业务需求,通过需求分析予以抽象提炼,转化为可用的需求规格说明,并基于需求规格说明进行系统设计与详细设计。在这一过程中常用的 CASE 工具有面向对象软件设计工具 Rational Rose(基于 UML1.0)、集成建模平台 Enterprise Architect、软件数据模型建模工具 PowerDesigner 等。这些工具通过简化 UML 图的绘制工作,以及强大的模型转换功能(如正向工程、反向工程、数据库模型转化等),大大简化了设计及从设计向编码转化的工作。

在编码阶段,软件开发团队将设计阶段的成果付诸实践,通过编写代码将设计转化为可用的软件实例。早期的编码工作使用文本编辑器进行,然而随着系统规模的扩张,其编码、调试的不便也大大提升。集成开发环境(IDE)通过提供代码高亮、补全,内置调试工具等功能,大大降低了这种不便。同时,前后端开发方式及不断涌现的前端开发框架(如 Vue.js、React、Ionic)和后端开发框架(如 Django、Spring Boot、Flask)允许开发者在短时间内开发出一款完整的软件应用产品。

在测试阶段,测试团队需要进行单元测试、集成测试等。自动化测试工具允许测试人员通过编写测试代码或测试脚本来描述测试用例,并且还可以一次性执行多个测试用例并输出对应的测试结果。自动化测试工具包括 Unit Test、Pytest、Vue Test Utils、JUnit 等,根据不同的编程语言、开发环境或测试目的,测试团队往往需要选择相对适用的测试工具。

如 1.1.3 所述,除主过程外,软件开发过程还包括诸多其他活动,而其中最重要的便是配置管理与项目管理。配置管理通常分为不同模式,每一种模式均有对应工具,较为著名的有 Microsoft VSS、CVS、SVN 等,近年来最常用的为 Git。而项目管理领域最普遍使用的为微软公司开发的 Microsoft Project,该软件提供了强大的项目管理功能,基本能够满足企业级项目管理的全部需要。

除此之外,在软件过程的其他活动中同样存在众多 CASE 工具。但是受到篇幅的限制,在此不再过多介绍,读者可自行搜索相关的内容。

1.2 网络应用程序的开发

1.2.1 网络应用程序

1. 概念

网络应用程序(又称 Web 应用程序)指的是通过浏览器联网来访问的应用程序。

通常,浏览器需要运行一个程序,主要由 HTML、CSS 和 JavaScript 代码组成,用于获取并在用户浏览器上面展示信息,供用户查阅或交互。浏览器在运行程序的过程中,会与服务器不断地通信并交换数据,实现业务操作,例如提交表单、修改密码等。这种通过浏览器和服务器通信来实现完整功能的方法,称为 B/S 架构(Browser/Server,浏览器-服务器架构)。

B/S 架构是 C/S 架构(Client/Server,客户端-服务器架构)的一种改进。在 C/S 架构之中,一款完整的应用是通过客户端和服务器通信来实现的。而在 B/S 架构之中,浏览器将会代替客户端负责和服务器进行通信。

2. 请求和响应

浏览器和服务器的每一次通信,都称为 HTTP 请求(request)或 HTTP 响应(response)。请求和响应通常是成对出现,首先由浏览器发出请求,然后服务器针对浏览器的请求做出响应,这样就完成了一次信息的双向交互,如图 1-2 所示。

图 1-2 浏览器和服务器之间的信息的双向交互

3. HTTP 请求和响应

浏览器向网络服务器发出的请求为 HTTP 请求,服务器针对 HTTP 请求的响应称为 HTTP 响应。

浏览器在进行 HTTP 请求时，会向服务器发送请求信息。请求信息包含了 3 部分：请求行、请求头和请求体。

(1) 请求行(request line)：描述了请求的方法、版本及协议。

(2) 请求头(request header)：包含了和浏览器环境相关的及和请求体相关的一些信息，例如 Cookie。

(3) 请求体(request body)：包含了请求的正文，例如，POST 请求会把请求的参数放在请求体中。

同样，服务器在进行 HTTP 响应时，也会向浏览器发送响应信息。响应信息同样也包含了 3 部分：响应行、响应头和响应体。

其中，HTTP 请求共有 9 种方法，如表 1-2 所示。其中，最常用的是 GET 和 POST 方法。

表 1-2　HTTP 请求的 9 种方法

请求方法	特　点
GET	通常用于检索一些信息或是获取内容，例如，搜索内容。GET 方法不建议对服务器的状态造成影响，这个状态可以是服务器上保存的数据。浏览器输入网址获取网页的请求方法就是 GET 方法。 此外，GET 方法比 POST 方法高效，其请求的参数通常附在请求的网络地址之后
HEAD	与 GET 方法类似，只不过在响应的时候服务器会忽略响应体的内容。一般用于获取一些响应头的信息以确定后续请求的方式
POST	通常用于向服务器提交数据，例如，提交表单。与 GET 方法相反，POST 方法可能会改变服务器状态。 此外，POST 方法的请求参数通常放在请求体之中，请求参数的格式根据需求可以有 Form Data 或 JSON 等
PUT	向服务器上传最新的数据以取代旧的内容
DELETE	请求服务器删除 Request-URI 所标识的资源
CONNECT	HTTP 1.1 协议中预留给能够将连接改为管道方式的代理服务器，可以在 SSL 加密服务器连接时使用
OPTIONS	用于试探服务器是否能够正常运作及服务器是否支持某些请求的方式。例如，在某些跨域请求时，浏览器会在发出正式请求前向服务器发送 OPTIONS 请求，以通过请求结果来判断服务器是否支持跨域请求
TRACE	用于询问服务器收到的请求，这样浏览器可以得知信息在到达服务器时被改变了什么内容，主要用于进行测试或诊断
PATCH	用于更改服务器状态。但是请求中只包含需要更改的信息，这样可以节省传输的内容

每个 HTTP 响应都有其状态码(status code)。状态码是一个三位数，表示在请求发出后请求的处理状态。状态码说明如表 1-3 所示。

表 1-3 状态码说明

状态码开头	状态码说明
1	表示请求已经被接收,需要等待请求继续处理
2	表示请求成功。其中常见的是 200,此时响应内容包含了请求所需要的内容
3	表示重定向,需要客户端(服务器)进行进一步的操作才能完成整个请求
4	表示客户端错误。其中常见的是 403、404 和 405。 403:禁止(forbidden)。表示服务器已理解请求,但拒绝执行。可能是权限不足,也可能是其他的原因,服务器通常会在响应体内描述拒绝执行的原因。 404:不存在(not found)。表示请求的资源不存在。 405:方法不允许(method not allowed)。表示请求使用的方法不被服务器所允许
5	表示服务器错误。其中常见的是 500,表示服务器在内部执行处理请求的时候发生了出乎意料的错误,以至于无法完成请求的处理

1.2.2 前端和后端

在 B/S 架构中,前端(front-end)通常指在浏览器上面运行的程序,即由 HTML(网页)、CSS(样式)和 JavaScript(脚本)代码组成的程序。后端(back-end)通常指在服务器上面运行的程序,该程序对外提供接口;每个接口类似于一个服务,允许前端通过 HTTP 请求来使用这些服务;后端在接到请求后,就会向前端做出响应,帮助前端获知服务的结果。

例如,一个用户通过浏览器搜索信息的活动图如图 1-3 所示。

(1) 用户单击搜索按钮后,前端浏览器会开始请求搜索,并在请求中附带了搜索关键词等信息。

(2) 后端服务器接到请求后,根据关键词在数据库搜索对应的内容,并整理好搜索的结果,向前端浏览器发出响应,响应中附带了搜索的结果。

(3) 前端浏览器在获得响应后,根据响应中搜索的结果,将信息展示到网页上,方便用户查看,此时关于搜索的业务完成。

1. 保存用户身份

在前后端通信的过程中,通常需要根据用户的身份来做出不同的响应。例如,用户登录之后,需要查询自己的信息,这时需要后端服务器能够识别出请求发出者的身份。解决这个问题的一个可行的方法是通过 Cookie,Cookie 中保存了用户身份码,会在每次发出请求时自动附在请求信息中,方便后端进行身份识别。

在服务器中,通常可以使用 Session 来保存用户通话信息,包括用户 ID 等。在用户登录时,如果登录成功,后端会创建一个 Session 并保存,然后在响应信息中附上 Set-Cookie 字段,该字段包含了 Session 的 ID。浏览器收到响应后会自动检测 Set-Cookie 字段并将该字段的内容存入 Cookie 中。例如,一个用户登录时浏览器和服务器之间交互的活动图如图 1-4 所示。

在登录后,浏览器每次发出请求时都会附带上 Cookie 字段。后端可以提取该字段的内容,获取 Session ID,然后检索得到 Session,以此来识别用户身份,如图 1-5 所示。

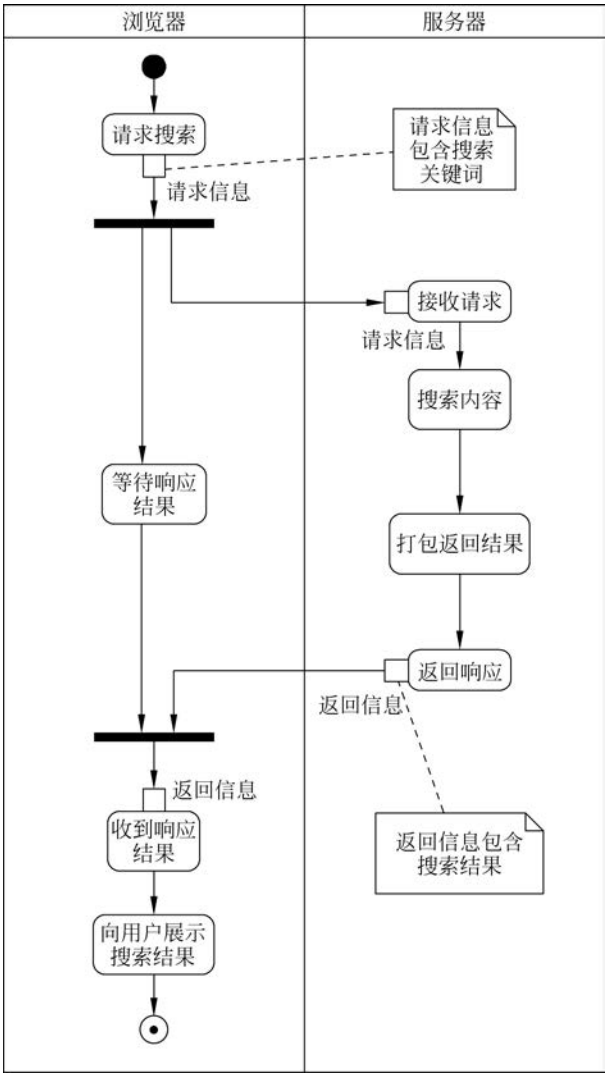

图 1-3　用户通过浏览器搜索信息的活动图

当然，这种方法也有一些缺点，例如，有人猜出了别人的 Session ID，即可操作别人的账号。因此，Session ID 应该经过加密且足够复杂，同时 Session 也应该有一定的有效期，在过期后用户需要重新进行登录，这样才能在一定程度上防止别人破解出 Session ID。

2. 前后端结合开发

细心的读者可能会有疑惑，前端的文件究竟由哪个服务器提供，这就涉及是前后端结合开发还是前后端分离开发了。

当浏览器在地址栏输入一个网址后，浏览器会向该网址发出 HTTP 的 GET 请求。如果响应结果是 HTML 文件，浏览器将会按照 HTTP 代码将内容渲染在浏览器上面，这就是用户平常看到的网页。

在前后端结合开发的过程中，后端服务器也担任了提供前端网页文件的任务。后端

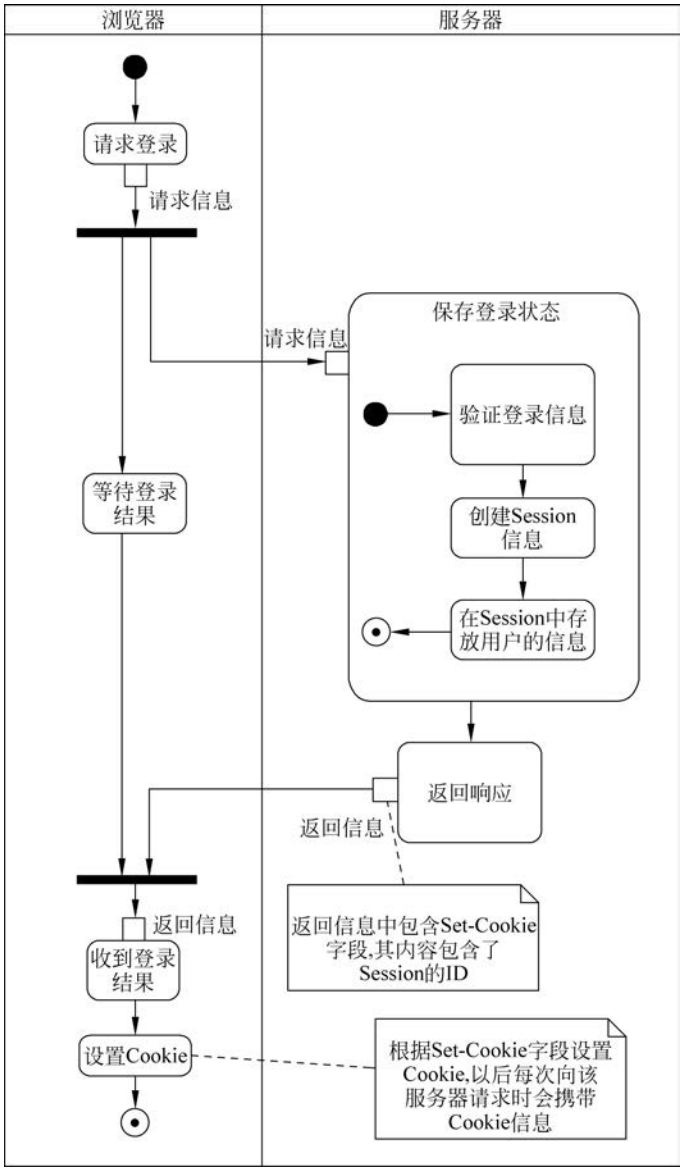

图 1-4　用户登录时浏览器和服务器之间交互的活动图

服务器会根据用户访问的网址,在后端生成一个 HTML 文件,然后返回该文件,如图 1-6 所示。在这种模式下,后端除了需要提供服务接口,还需要渲染并提供网页文件。同时,浏览器可以直接收到一个信息较全的网页,不需要网页做出额外的请求来初始化一些信息。

在开发过程中,由于后端服务器也负责渲染网页文件,因此前后端开发都在同一个程序中进行,称为前后端结合开发。

3. 前后端分离开发

前后端结合开发会带来一个明显的缺点,就是开发的耦合较大。

前后端只需定义好需要和提供的接口,就可以完全分离地进行开发。前端开发者只需

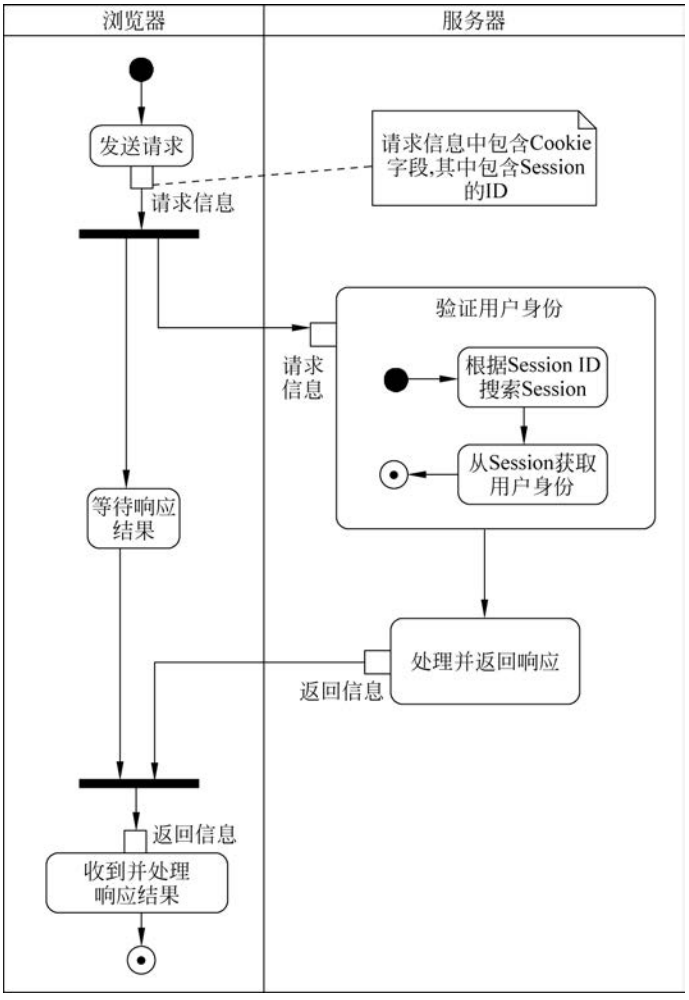

图 1-5　服务器通过 Cookie 验证用户身份的活动图

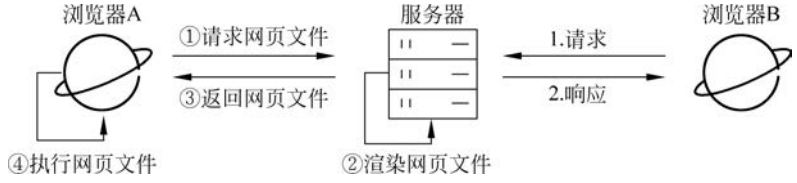

图 1-6　后端服务器同时负责渲染网页和响应业务请求的任务

考虑如何把响应的结果放到网页中,以及如何把用户提交的信息准确地发送给服务器即可。后端开发者只需考虑如何组织服务器的数据,以及实现前端需要的接口来为前端提供必要的信息和业务即可。在这种开发模式下,前后端开发的耦合大大降低。这种模式就称为前后端分离开发。

同时,前后端分离开发还允许将前后端的代码文件分别放置在两个服务器中,前端服务器只需要提供静态的网页文件,后端服务器只需要提供一些必要的接口,如图 1-7 所示。浏览器可以从前端服务器收到网页文件,这个网页文件内容通常是不完整的。浏览器在执行

该网页文件的脚本(JavaScript 代码)时,会向后端服务器请求初始化所需要的信息。JavaScript 会将响应的信息填充到页面上,这样就形成了一个完整的页面。

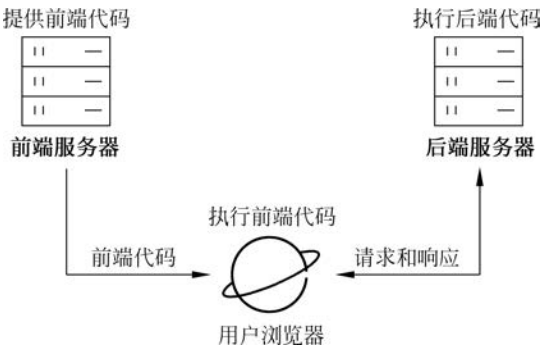

图 1-7　前端服务器只需要提供静态的网页文件,后端服务器只需要提供一些必要的接口

注意　前端服务器始终没有执行前端的代码文件,前端的代码文件是由用户浏览器执行的,前端服务器只是起到提供前端代码文件的作用。而后端服务器会执行后端代码文件,以向外界提供业务相关的接口和服务。

当然,前后端代码文件也可以放置在同一个服务器上。在这种情况下,后端服务器不需要渲染网页,直接返回前端开发者编写好的网页文件即可。

在本书后续的例子中,都将采用前后端分离开发模式。

1.3　“论文检索系统”案例介绍

本书所有章节实验均以同一案例——“论文检索系统”为基础,以保障各阶段实验的连贯性和一致性。下面对案例做简要介绍。

1.3.1　项目背景

随着学术交流需求的提升,论文的发布与分享成为了学术促进发展的一种主流的方式。然而,论文的发布与分享需要一个强大的系统支撑,提供包括搜索、学者认证、论文引用数统计等服务,允许学术科研人员在系统上高效地搜索和公开学术成果,并且保证他们的版权。考虑到这种情况,需要建立一个“论文检索系统”,为学术交流活动提供便利。

1.3.2　需求说明

根据需求调研的结果,系统的使用者主要包括检索学术文献的用户与管理和维护系统后台的管理员。前者应能够通过系统自由检索,找到所需的学术文献或感兴趣的学者。后者应能够管理系统后台,能够更新学术文献和学者信息。系统还应针对学术文献的内容主题提供学科领域分类,使用户可以通过感兴趣的领域来分类检索相关文献,并可以结合领域的论文量等数据了解当下的热门领域。除此之外,系统应提供学者关系网络,系统的用户能够根据感兴趣的学者的关系网络找到类似的学者或领域,拓宽检索同领域学者的渠道。

1.3.3 系统要求

“论文检索系统”所面向的用户群体主要为 PC 端用户，为保障便捷性，系统应该为 Web 应用。由于用户所使用设备分辨率各有不同，系统应保障在不同分辨率下均能提供友好的交互服务。系统适用范围遍布全国，但使用人员大多是科研机构的工作人员及在校大学生。在此基础上，应保证即便出现可能发生的最大同时访问量，系统仍然能够正常运转，响应时间应在可接受范围内。除此之外，系统还应在安全性、可靠性上予以保障，不应出现数据丢失、隐私泄露等事故。

1.4 小　　结

本章对软件工程的基本概念与发展、软件生命周期各个阶段的活动及软件生命周期模型予以讲解，介绍了软件生命周期各阶段常用的 CASE 工具。此外，本章也对 Web 应用、HTTP 请求和前后端进行了一定的介绍。最后，本章介绍了“论文检索系统”案例的需求和相关的内容，该案例将成为本书演示所使用的项目。

1.5 习　　题

思考题

1. Web 应用与其他的客户端应用相比，具有哪些优势和劣势？

2. 如果用户需要在 Web 应用上注册一个账号，浏览器和服务器之间的交互活动大概是怎样进行的？

1.6 参 考 文 献

[1] BOEHM B. A view of 20th and 21st century software engineering[C]//Proceedings of the 28th international conference on Software engineering. 2006：12-29.

[2] Software engineering-Wikipedia[EB/OL]. [2022-3-12]. https://en. wikipedia. org/wiki/Software_engineering.

[3] BOURQUE P，DUPUIS R. Guide to the software engineering body of knowledge[J]. Swebok Guide to the Software Engineering Body of Knowledge，2004，16(6)：35-44.

[4] ISO/IEC/IEEE 24748-3：2020（E），ISO/IEC/IEEE International Standard-Systems and software engineering-Life cycle management-Part 3：Guidelines for the application of ISO/IEC/IEEE 12207 (software life cycle processes)[S]. 2020.

[5] Hypertext Transfer Protocol-Wikipedia[EB/OL]. [2022-3-12]. https://en. wikipedia. org/wiki/Hypertext_Transfer_Protocol.

[6] MCDERMID J，ROOK P. Software development process models[M]//Software Engineer's Reference Book. Butterworth-Heinemann，1993，15：26-28.

第2章 项目管理工具 Microsoft Project

2.1 概 述

Microsoft Project(简称 Project)是由微软公司开发的项目管理软件,主要用于协助项目经理制订项目计划、资源分配、进度跟踪、预算管理等工作。Project 能根据关键路径法制订项目任务的日程安排,并通过任务进度可视化,允许项目管理人员通过图表来查看项目的进展,分析项目任务关键路径。

Microsoft Project 的主要功能点如下。

(1) 关键路径管理:Project 可以基于关键路径法生成工作计划,将项目分解成为多个独立的活动,并确定活动的工期、开始时间、结束时间及活动间的依赖关系。

(2) 可视化任务管理:在 Project 中,项目任务计划以甘特图表示。

(3) 企业级资源管理:Project 提供了面向企业级用户的资源管理功能,允许项目经理定义企业资源,并将资源分配给企业的不同项目。资源包括人力资源、设备资源与经济资源等。

(4) 成本估算:Project 能够根据资源消耗率与任务量自动估算出项目在任意时间产生的资源消耗。项目管理人员可以以此为凭据估算项目成本,以调整各个任务所占用的资源配给,降低整体项目消耗。

(5) 任务报表管理:在 Project 中,团队成员可以按照预设格式获取或提交工作报表,并将所获得的报表合并到项目状态报表中。

本章以 Microsoft Project 2021 为例,对使用 Project 绘制甘特图进行介绍。

2.2 基 本 操 作

2.2.1 界 面 说 明

在进入 Project 后,选择新建一个空白项目,即可进入主操作界面,如图 2-1 所示。

Project 操作界面分为功能区和主视图。

(1) 主视图:用于呈现不同视角下的项目视图,默认情况下会包括甘特图和日程表两个视图。

(2) 功能区:包含了一系列操作,主要分为任务、资源、报表、项目、视图等。

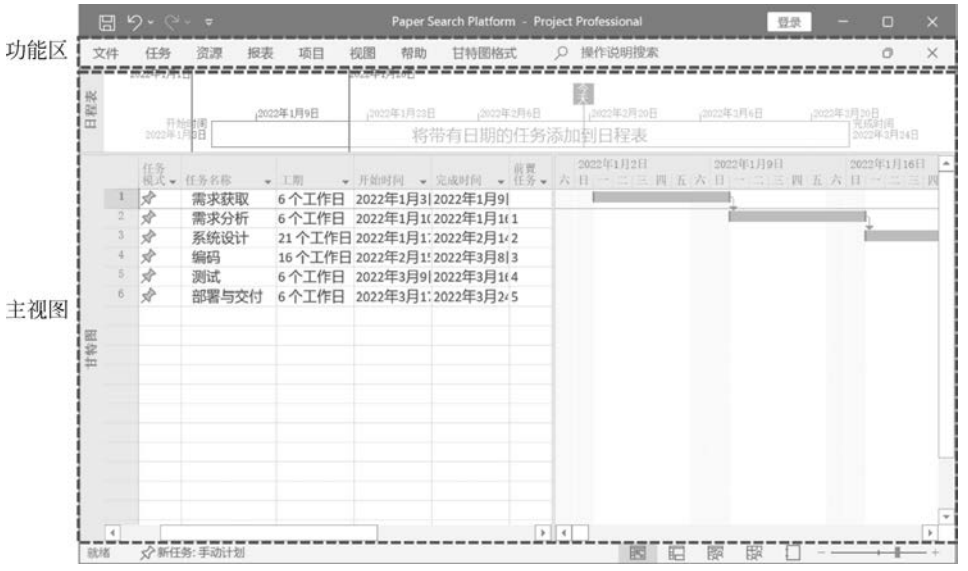

图 2-1　主操作界面

2.2.2　视图

在 Project 中,项目管理工作的任务和资源等数据可以通过不同视图呈现,并集中以某种视角向用户呈现特定的数据。

Project 提供了多种不同的视图,主要有日程表、甘特图、任务分配状况、网络图、日历等,在功能区的“视图”中可以进行视图的切换,还可以将这些视图组合,共同显示更多的信息。在本章中,仅需要使用甘特图和日程表。

在“视图”中,单击“甘特图”按钮可以在主视图中打开甘特图视图;勾选“拆分视图”中的“日程表”复选框,可以在主视图中打开日程表;在主视图中选中某个视图后,单击“窗口”→“隐藏”选项,可以隐藏该视图。上述的操作如图 2-2 所示。

图 2-2　切换视图操作

2.2.3　日程表

日程表能够提供以图形化的界面编辑甘特图的可见范围。拖动日程表深色框的位置可以改变甘特图的视野区域,改变深色框大小可以调整甘特图的视野大小。

2.2.4 甘特图

甘特图视图主要包含左侧的工作表与右侧的图表,如图 2-1 所示。该视图主要用于完成以下工作。

(1) 通过添加任务,并对所添加的任务进行调整来创建一个项目。

(2) 通过调整任务的前置任务,明确任务与任务之间的依赖关系,决定关键路径并采取相应情况下的决策。

(3) 对任务进行拆分,将单项任务拆分成多份,分派到不同时间段完成;还支持实时跟踪任务进展,以百分比来表示。

甘特图视图中工作表的结构与 Microsoft Excel 的表格类似,包含各项任务的基本信息,例如,任务模式、任务名称、工期、开始时间、完成时间、前置任务等。

(1) 任务模式:设置任务为自动计划还是手动计划。

(2) 工期:计算完成任务所需的工作日。

(3) 前置任务:前置任务的序号,用英文逗号隔开。

甘特图视图中的图表则使用甘特图呈现工作表中的任务信息,用户从甘特图中可直观地看到任务的起止时间及任务间的依赖关系。左侧的工作表与右侧的图表共享数据,修改任何一侧的数据后,另一侧的信息会立刻发生相应的变化。

2.3 绘制“论文检索系统”的甘特图

以“论文检索系统”为例,本章将对使用 Project 绘制甘特图的方法进行介绍。该项目的工期约为 6～7 个月,根据工期限制与团队讨论,确定了大致的项目任务规划,如表 2-1 所示。其中,由于交付软件后的维护时间相对模糊,所以在这里略去。

此外,工作日确定为**每周的周一至周六**,周日和国家法定节假日为休息时间。

表 2-1 项目任务规划

任务编号	任务名称	开始时间	预计工期	前置任务编号
1	需求调研与获取	2021-01-03	10 工作日	无
2	技术调研	2021-01-03	15 工作日	无
3	需求分析	/	15 工作日	1
4	技术选型	/	5 工作日	2、3
5	界面设计	/	5 工作日	3
6	系统设计	/	10 工作日	4
7	接口设计	/	5 工作日	5
8	数据库设计	/	5 工作日	6、7
9	编码	/	30 工作日	5、6、7、8
10	单元测试	/	15 工作日	9
11	集成测试	/	10 工作日	10
12	Alpha 测试	/	5 工作日	11
13	Beta 测试	/	5 工作日	12
14	部署	/	5 工作日	13
15	文档整理与交付	/	5 工作日	14

注意 Project 会自动计算开始时间和完成时间。在最终确定计划前，你可能会根据绘制出来的甘特图对计划内容进行调整。但在本章中不再演示调整的过程，直接以最终的计划来绘制。

2.3.1 设置项目信息

设置项目信息包括设置项目开始时间及设置工作日时间。

(1) 在功能区单击“项目”→“项目信息”按钮，打开“项目信息”窗口，如图 2-3 所示。

图 2-3 打开“项目信息”窗口

(2) 设置项目的开始时间，并单击“确定”按钮保存项目信息设置，如图 2-4 所示。

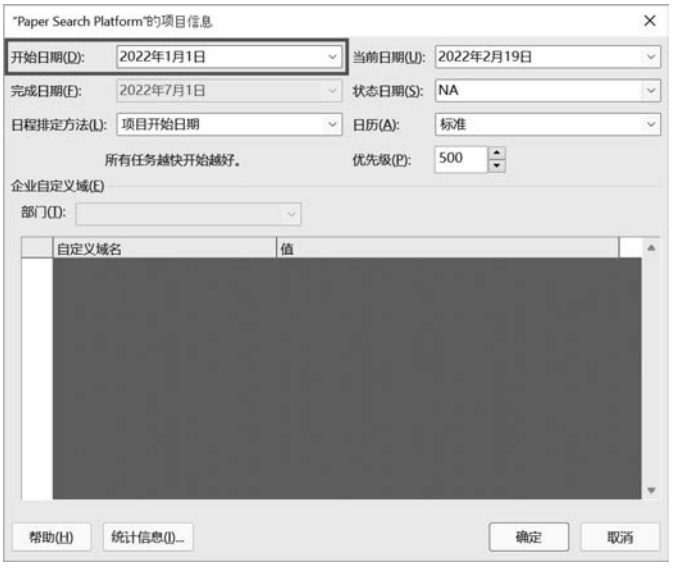

图 2-4 保存项目信息设置

(3) 在功能区单击“项目”→“更改工作时间”按钮，打开“更改工作时间”窗口，如图 2-5 所示。

图 2-5 打开“更改工作时间”窗口

(4) 单击“选项”按钮,如图 2-6 所示。

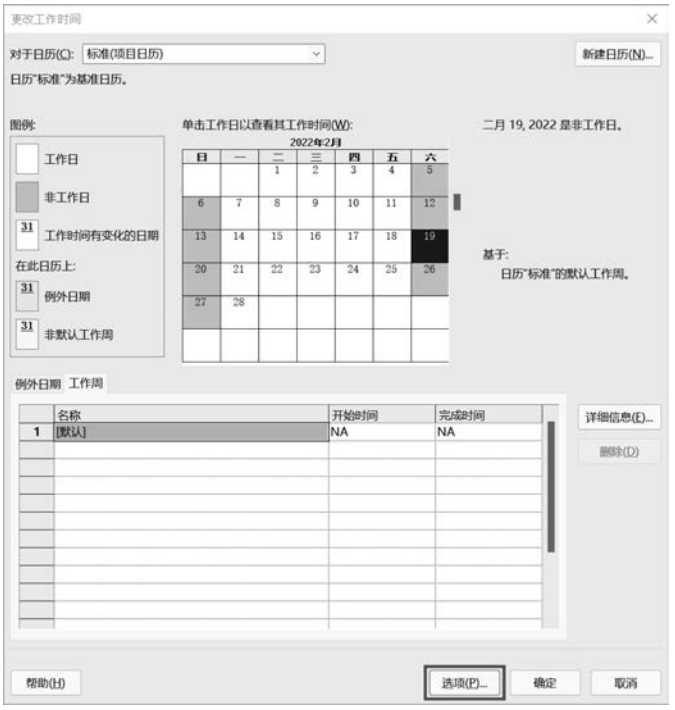

图 2-6 单击“选项”按钮

(5) 单击“日程”→“每周开始于”→“星期一”选项,设置每周的时间安排开始于星期一,单击下方的“确定”按钮保存,如图 2-7 所示。

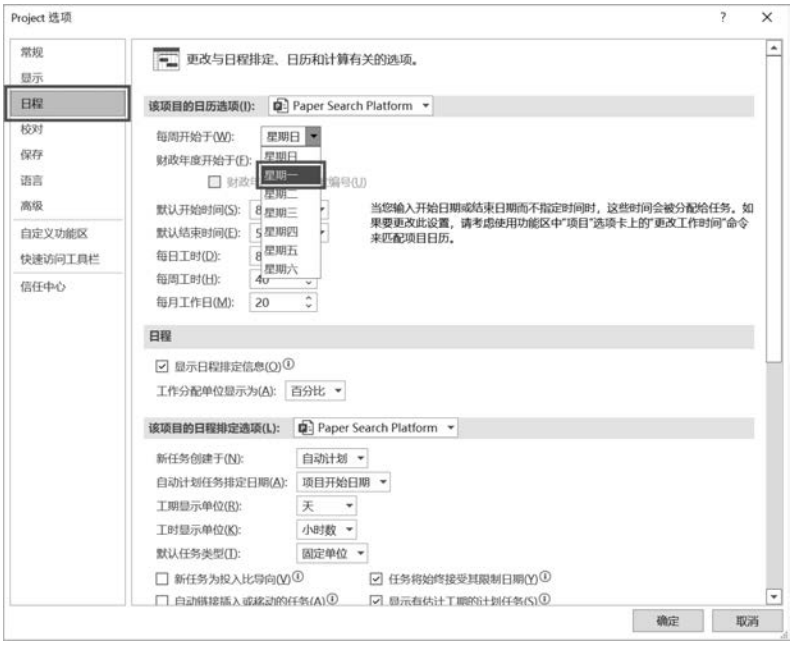

图 2-7 设置每周的时间安排开始于星期一

（6）在“更改工作时间”窗口下方的“工作周”标签页中，选中第一行“[默认]”，单击“详细信息”按钮，设置“[默认]”工作周的安排，如图 2-8 所示。

图 2-8　设置“[默认]”工作周的安排

（7）单击“星期日”选项，选中非工作日（按住键盘的 Ctrl 键后单击选项可选中多个），然后再选择“将所列日期设置为非工作时间”单选按钮，单击“确定”按钮保存，如图 2-9 所示。

图 2-9　设置工作日信息

(8) 在“更改工作时间”窗口,单击“例外日期”按钮,在下方的表格中填写节假日及节假日调休的名称,设置节假日信息,如图 2-10 所示。

图 2-10　设置节假日信息

(9) 选中填写的节假日信息,单击“详细信息”按钮,可以设置是非工作日还是调休(工作时间),以及具体的时间,对节假日进行详细设置,如图 2-11 所示。设置后单击“确定”按钮保存。

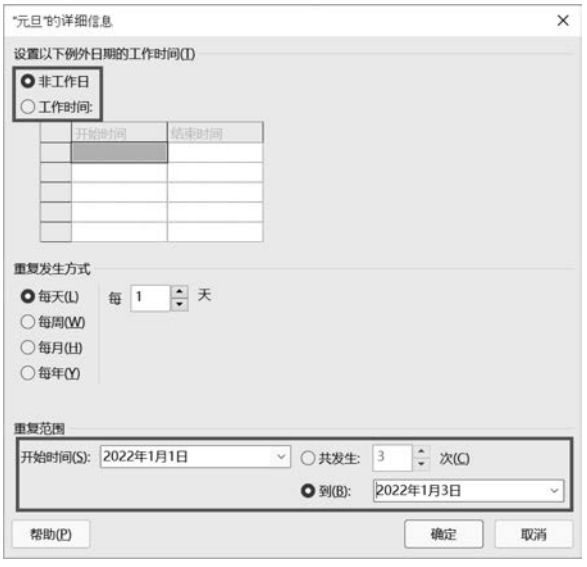

图 2-11　对节假日进行详细设置