0835 软件工程一级学科研究生核心课程指南

01 软件工程理论基础

一、课程概述

软件工程既关注构造软件的理论、模型与算法及其在软件开发与维护中的应用，也关注求解问题的数学理论与方法及其在软件建模、分析、设计和验证中的应用。为了使学生深入理解软件的行为特征，提高软件开发和问题求解中的抽象思维与逻辑思维能力，更好地掌握软件分析、设计、测试等方法，本课程主要讲述软件工程所需要的数学理论和计算理论，为软件工程研究生培养建立坚实的理论基础。

二、先修课程

计算机程序设计、线性代数、离散数学（含抽象代数）、数据结构、算法设计与分析、概率论与数理统计。

三、课程目标

本课程的目标是让软件工程一级学科的研究生掌握与未来软件工程学术研究和软件项目开发密切相关的逻辑学、抽象与建模、形式化方法基础理论，培养学生严密的数学思维能力和抽象、推理能力，为其后续课程学习以及未来的研究与工程实践建立坚实的理论基础。

四、适用对象

本课程适用于软件工程及相关学科、具有先修课程知识基础的博士研究生和硕士研究生。

五、授课方式

本课程具有内容广泛、理论性强的特征，在教学实施中具有较高难度。各高校在具体开设本课程时，可以考虑选择以下适合的方式。

1. 可以根据本校特色、培养目标定位以及后续专业课程设置，在本课程中选择部分内容进行讲授，或者对不同知识点采用不同的详略程度进行授课。

2. 将本课程细分为几门不同的课程，例如数理逻辑/计算机逻辑学、形式语言与自动机、形式语义学、形式化方法/形式化验证等，供学生根据培养方案进行选择。

3. 将本课程作为理论基础概述性课程，覆盖课程内容的各个知识点，学生在了解相关概念的基础上，根据兴趣选择某部分内容，通过阅读教材和文献进行深入学习。

尽管本课程是理论基础课,但目前主要内容都有很多学术界的开源软件工具,在授课过程中结合工具实践,将会提高学生对理论知识的理解与运用能力。

六、课程内容

第一部分　数学理论基础

1.1　经典逻辑

1.1.1　命题逻辑

主要内容:命题逻辑公式语法、语义;典型命题逻辑公理系统及其元性质;命题逻辑的可满足性判定——消解法。

■ 重点和难点:公理系统完全性与独立性的证明方法。

1.1.2　一阶逻辑

主要内容:一阶逻辑公式语法、语义;一阶逻辑公理系统及其元性质;一阶逻辑的消解法及Herbrand 定理。

■ 重点和难点:公理系统完全性与独立性的证明方法。

1.2　非经典逻辑

1.2.1　直觉主义逻辑

主要内容:直觉主义逻辑的语法、语义;直觉主义逻辑的表达能力及典型推理系统。

■ 重点和难点:直觉主义逻辑的语义结构。

1.2.2　模态逻辑

主要内容:模态逻辑/动态逻辑/模态 μ-演算的语法、语义、表达能力的比较;典型的公理系统及其元性质;各类模态逻辑的判定算法。

■ 重点和难点:μ-演算的语法、语义。

1.2.3　时序逻辑

主要内容:LTL、CTL 的语法、语义、表达能力的比较;LTL、CTL 的典型公理系统及其元性质;LTL、CTL 可满足性判定。

■ 重点和难点:时序逻辑间表达能力的比较,可满足性判定算法。

1.3　计算理论

1.3.1　图灵可计算

主要内容:图灵可计算的定义;递归函数、Ackermann 函数。

■ 重点和难点:丘奇-图录论题。

1.3.2　可计算理论

主要内容:对角线方法、典型不可判定问题;判定性问题归约及转化;通用图灵机、S-M-N 定理及递归定理。

■ 重点和难点:对角线方法的使用;判定性问题归约。

1.3.3　时间复杂性

主要内容:时间复杂性计量;P 类、NP 类、NPC-问题、列文-库克定理;典型 NPC 问题;EXP-问题、(时间)层次定理。

■ 重点和难点:列文-库克定理的证明;时间复杂性归约方法。

1.3.4 空间复杂性

主要内容：空间复杂性计量；PSPACE-问题类、萨维奇定理；典型 PSPACE-完全问题及其归约；L 类、NL 类、NL 与 coNL 的关系；(空间)层次定理。

■ 重点和难点：萨维奇定理的证明；空间复杂性归约方法。

第二部分 抽象与建模

2.1 形式语言与自动机

2.1.1 正规文法与有穷自动机

主要内容：正规文法、有穷自动机、正规语言；有穷自动机的确定化、极小化；Pumping 引理；正规语言的封闭性。

■ 重点和难点：文法、自动机、语言之间的转换；Pumping 引理的使用；正规语言的封闭性。

2.1.2 上下文无关文法与下推自动机

主要内容：上下文无关文法/语言、下推自动机及其相互等价性；Chomsky 范式；迭代定理；上下文无关语言的封闭性质。

■ 重点和难点：迭代定理的使用；上下文无关语言的封闭性。

2.1.3 图灵机

主要内容：图灵机与递归可枚举语言；递归可枚举语言的封闭性。

■ 重点和难点：图灵机及其变种的定义。

2.1.4 上下文相关文法与线性界限机

主要内容：上下文相关文法/语言、线性界限机；线性界限机的判定问题；上下文相关语言的封闭性。

■ 重点和难点：上下文相关文法的构造。

2.2 λ-演算

2.2.1 无类型 λ-演算

主要内容：无类型 λ-演算的语法、规约、λ-等价；Church-Rosser 性质；递归函数与 λ-演算的表达能力。

■ 重点和难点：Church-Rosser 性质、从递归函数到 λ-项的转化。

2.2.2 带类型 λ-演算

主要内容：Church 和 Curry 型 λ-演算的语法、规约；λ-项的类型推导与合一化。

■ 重难和难点：带类型 λ-演算的类型推导与合一化。

2.2.3 组合逻辑

主要内容：K、S 算子的完全性；组合逻辑规则的完全性。

第三部分 形式验证

3.1 形式语义与演绎验证

3.1.1 三种基本程序

主要内容：流图型程序、while 型程序、函数式程序的范型及特点；递归程序论域的 ω-扩张。

■ 重点和难点：while 型程序及函数式程序的范型。

3.1.2 操作语义

主要内容：流图型程序的格局及迁移的定义；while 型程序格局迁移以及计算序列的定义；基

于定义计算函数式递归程序的操作语义。

■ 重点和难点:while 型程序格局迁移以及计算序列的定义。

3.1.3 指称语义

主要内容:完全偏序集、完备格、完备格上的连续函数、连续函数的不动点;λ-记号及其意义;基于递归语义泛函最小不动点计算程序指称语义;指称语义与操作语义的等价性。

■ 重点和难点:完备格、连续函数、不动点;指称语义与操作语义的等价性。

3.1.4 公理语义

主要内容:程序部分正确性、停机性、完全正确性;程序最弱前置条件、最强后置条件;Hoare 逻辑系统。

■ 重点和难点:程序最弱前置条件、最强后置条件;Hoare 逻辑及其使用。

3.2 模型检验

3.2.1 系统模型与规约

主要内容:模型的表示——显式及符号化;模型的规约——时序逻辑;Z、B、VDM 或 Petri Net 等形式规格说明语言。

■ 重点和难点:典型时序逻辑的语法、语义。

3.2.2 显式模型检验

主要内容:ω-自动机、从 LTL 到 ω-自动机的转化;基于自动机 LTL 的显示模型检验算法;从 CTL 到 μ-演算的转化;显式 CTL 模型检验算法。

■ 重点和难点:从 LTL 到 ω-自动机的转化及模型检验算法。

3.2.3 符号化模型检验

主要内容:BDD 的定义、约简、相关操作;基于 BDD 的 CTL 符号化模型检验;基于 BDD 的 LTL 符号化模型检验;基于 SAT 的 LTL 符号化模型检验(BMC)。

■ 重点和难点:BDD 及其相关操作;LTL 符号化自动机的构造及其编码。

3.2.4 概率系统验证

主要内容:随机过程、Markov 链、Markov 决策过程;PCTL、PLTL 语法与语义;PCTL 模型检验算法、PLTL 模型检验算法。

■ 重点和难点:Markov 性质及相应随机过程;基于确定自动机、非歧义自动机的 PLTL 模型检验算法。

七、考核要求

建议将闭卷考试、专题研讨与报告、工具实践应用等几方面相结合,各学校可按照自己的需要进行安排并设定成绩比例,综合形成最终的考核成绩。

八、编写成员名单

廖湘科(国防科技大学)、董威(国防科技大学)、刘万伟(国防科技大学)。

02 基础软件与开源系统

一、课程概述

基础软件系统(Infrastructure Software System)包括操作系统、数据库管理系统、中间件等支撑应用核心功能和业务的软件系统。基础软件系统规模大,通常扮演着行业和应用信息服务基础设施的角色。

“互联网+”深刻地改变了诸多领域服务和运行方式,而云计算、大数据和人工智能等技术的飞速发展则对业务的功能、性能提出了新的要求。这些变化进而要求基础软件系统的功能、架构和实现方法与之相适应。

本课程旨在介绍基础软件系统设计与实现的基本方法,介绍操作系统、数据库管理系统、中间件等基础软件系统的架构和实现技术,介绍基础软件系统的开发方法,特别是开源系统的设计和开发方法,介绍基础软件系统研究与开发的最新进展,为学生从事相关领域的研究与工程开发工作奠定理论基础,培养学生的动手实践能力。

本课程是软件工程一级学科硕士研究生和博士研究生的基础课程,是分布式系统等课程的先导课程。

二、先修课程

学习本课程之前应具备以下基础知识。

1. 计算机系统:从整体上了解计算机系统的各个部分以及它们之间的关联,理解处理器、存储层次、系统 I/O、网络的基本原理,理解程序的编译、运行,以及并行编程的基本原理和方法。

2. 操作系统:掌握操作系统的功能,了解其结构,掌握操作系统主要模块和功能的设计和实现技术,掌握系统程序设计方法,理解操作系统实现技术对应用程序的性能影响。

3. 数据库管理系统:了解和掌握数据库管理系统的基本概念、原理和技术,学会使用常用的数据库管理系统(包括 SQL 数据库管理系统、NoSQL 数据库管理系统和数据库开发的常用工具)开发应用软件。

4. 计算机网络:了解计算机网络与通信的基本原理与技术,掌握计算机网络的概念、组成和体系结构,掌握数据通信、各层网络协议和网络互联方面的基本问题和主要算法。

三、课程目标

通过本课程的学习,学生应掌握基础软件设计与实现的一般方法;掌握操作系统、数据库管理系统、中间件等基础软件系统的架构和实现技术;掌握基础软件系统的开发方法,特别是开源系统的设计和开发方法;了解基础软件系统研究与开发的最新进展;理解在“互联网+”的背景下,云计算、大数据、人工智能等技术对基础软件系统功能、架构、实现等方面的要求与影响;掌握并具备参与基础软件系统开发和研究的基础理论知识和动手实践能力。

四、适用对象

本课程适用于博士研究生和硕士研究生。

五、授课方式

本课程采用三种教学方法相结合的方式进行授课。

1. 课堂讲授:对基础软件设计与实现的一般方法,操作系统、数据库管理系统、中间件等基础软件系统的架构和实现技术等内容,采用课堂讲授的方式进行教学。

2. 实验实践:本课程通过实验实践,教授开源系统设计与开发方法,在实践中使学生具备大型基础软件设计与工程实现能力。

3. 课堂研讨:对基础软件系统的最新研究进展,采用学生课外阅读相关文献、课堂研讨的方式进行教学。

六、课程内容

第一部分　基础软件系统与软件栈概述(3 课时)

主要内容:基础软件系统基本概念,系统软件和基础软件发展历史,软件工具、系统栈及其主要示例,开源的基本概念。

第二部分　开源系统开发方法(9~12 课时,含实践)

主要内容:开源协议(GPLv2、GPLv3、Apache License 等);版本控制与 Git/Github;Bug 管理与 Bugzilla;开源社区及其工作方式;协同开发、调试方法与实践;开源系统实例(选讲,包含 Linux、PostgreSQL、MySQL、Chromium、Hadoop、Spark 等)。

第三部分　操作系统进阶(9 课时)

主要内容:操作系统基础知识回顾(操作系统结构、进程管理、并发与死锁、内存管理、存储与文件管理、I/O 管理);虚拟化与云操作系统;资源管理(Zookeeper、Yarn);分布式共识机制概念与实现(Paxos、Raft);存储系统(RAID、新型存储介质及其对存储系统的影响);分布式文件系统。

第四部分　数据库系统实现(6 课时)

主要内容:行存与列存;内存数据库管理系统;查询优化(Volcano 优化、Orca 优化);查询的并发执行。

第五部分　数据密集型计算系统(9 课时)

主要内容:OldSQL、NoSQL、NewSQL 的概念、功能与架构以及它们之间的区别与联系;Google File System、MapReduce、BigTable;批处理大数据系统(Hadoop);批流融合大数据系统(Spark、Flink);流式大数据系统(Storm、Kafka);图处理系统(Giraph)。

第六部分　中间件(6 课时)

主要内容:中间件架构,包括中介(mediator)方式和仓库(warehouse)方式;应用服务器;队列系统;BPM 和工作流;Web 服务。

第七部分　事务处理(选讲,3~6 课时)

主要内容:事务处理理论基础;乐观并发控制;多版本并发控制;日志与检查点机制;索引结

构的并发控制。

第八部分　基础软件系统的其他问题与新进展(选讲,3~6 课时)

主要内容:支撑互联网级应用的基础软件系统,基础软件系统安全,区块链系统。

■ 重点:

1. 开源软件系统开发模式与实践;
2. 虚拟化与云计算;
3. 数据密集型计算系统服务模式、功能接口、架构;
4. 分布式环境下的基础软件系统:共识与容错;
5. 基础软件系统实现技术。

■ 难点:

1. 分布式共识机制;
2. 数据密集型计算系统的异同;
3. 事务处理理论及其实现方法;
4. 大型基础软件系统的协同开发与调试。

七、考核要求

本课程的考核包括两部分,比例可按需调整:

1. 书面考核。对基础软件设计与实现的一般方法,操作系统、数据库管理系统、中间件等基础软件系统的架构和实现技术采用书面考核方式。

2. 实践考核。通过实际开源系统实践,包括 Bug 解决、新功能实现、新数据结构、算法和方法实现等具体任务,考核学生开源系统设计与开发的能力,同时通过研究和测试报告,考核学生对基础软件系统研究进展的掌握情况。

八、编写成员名单

周傲英(华东师范大学),钱卫宁(华东师范大学),周烜(华东师范大学),翁楚良(华东师范大学),蔡鹏(华东师范大学),徐辰(华东师范大学),胡卉芪(华东师范大学)。

03　大规模领域软件系统

一、课程概述

大规模领域软件系统是指在社会信息化过程中特别是互联网大规模应用以来,网络和软件技术与各行各业创新业务应用深度融合、综合集成的大型软件系统。

本课程以实践及案例为基础,系统地介绍大规模领域软件系统的规划、设计、实施和运营服务所涉及的原理、方法、技术和应用,包括软件系统的总体规划,软件运行所需的物联网布线、感

知、网络基础设施,大规模应用软件需求分析和架构设计,数据集成、应用集成,大规模领域软件创新应用,软件运营服务等内容。

二、先修课程

数据库原理、计算机网络。

三、课程目标

本课程的目标是通过基础知识及设计案例的学习,学生能够掌握大规模领域软件需求分析和设计理论、复杂软件架构和系统集成技术的实际应用方法,培养学生系统集成的创新能力。

四、适用对象

本课程可作为软件工程专业硕士研究生必修课或选修课、博士研究生选修课。

五、授课方式

本课程应以课堂讲授与设计实践相结合的方式来组织教学。

在教学过程中,要与领域软件规划、设计和实施的企业或系统集成的软件企业合作。

建议加强课程的教学团队建设,结合合作的软件企业、重大系统类课题研究,介绍综合案例,设置系统设计实训单元。

六、课程内容

本课程可大致分为两个部分来组织教学:第一部分讲授大规模软件的硬件和网络运行环境、通用系统架构、主要技术及相关标准、基本设计方法;第二部分介绍和练习典型设计案例,实现综合集成。

本课程的教学内容大致按照以下六个单元来划分。

第一单元　大规模领域软件系统的组成、规划和实施过程

1.1　大规模领域软件系统的组成和创新

主要内容:介绍大规模领域软件系统的特点,所涉及的技术系统,典型的行业应用创新和创新企业;介绍一些典型的国际标准、国家标准、行业和地方标准分布及特点(如 ISO、GB、DB、GA 等)。

1.2　系统或项目的规划和实施

主要内容:介绍在软件开发和集成工程实践中,典型的大规模领域软件的实施过程,一般包括项目规划、初步设计、工程设计、开发实施、运营服务;重点讲授信息系统项目可行性研究报告的编制要点。

作业设计:

1. 设计基本知识练习,回顾软件工程的主要概念和生命周期;

2. 设计一类综述作业,应用场景图、体系结构图来了解一个行业的大规模软件系统,提炼出系统目标、主要功能、应用效果。

本单元建议讲授课时:2~4 课时。

第二单元　领域软件系统的运行环境:设备系统和网络集成

本单元可根据实际需求,只讲授一部分,或全部不讲;也可在后续实际软件系统案例中讲授相关模块。

2.1　综合布线系统

主要内容:简要介绍基本概念、系统组成、相关标准和创新应用;介绍综合布线作为物联网、智慧城市、大数据领域软件创新应用的信息基础设施的重要性,重点是综合供电、综合管线以及联网的可行性。

作业设计:

1. 创新布线案例或产品综述;

2. 场地观察布线设计作业,培养创新意识。

2.2　智能建筑与智慧社区系统

主要内容:介绍智能建筑的概念;介绍典型弱电系统组成及特点,特别是与物联网和智慧城市相结合的创新。

作业设计:

1. 某一类弱电系统综述;

2. 以"面对绿色建筑、智慧社区等的挑战"为主题设计创意或创新作业。

2.3　物联网系统

主要内容:介绍物联网的基本概念,感知层、网络层、应用层的分层结构及组件,新一代物联网感知设备和联网技术、产品,物联网中间件,典型创新应用。

作业设计:综述一类物联网技术、设备或创新应用(任选其一)。

2.4　网络集成

主要内容:介绍局域网、广域网、数据中心网络设计的主要网络工程和设备技术。

作业设计:给定需求或应用案例,完成局域网、广域网、SAN 网的方案设计(任选其一),给出主要设备选型。

本单元建议讲授课时:4~8 课时。

第三单元　大规模领域软件需求分析与架构设计

3.1　大规模领域软件的需求分析

主要内容:针对大规模领域软件,介绍从系统总体上进行需求分析的软件工程技术、工具(图)和主要需求分析内容及要求。

作业设计:给定一个系统(有可选项),应用典型的需求分析工具描述其总体需求。

3.2　应用集成的架构设计

主要内容:面向工程应用,介绍企业级应用软件架构设计的基本概念、模式、主要设计任务、交付物和主要分层结构;介绍国内外主流的技术路线、厂家和产品。

作业设计:给定项目案例或已有系统的创新需求,总体描述系统的分层结构,设计各层的主要组成以及所选择的商业软件套件和组件支持。

3.3　Web 网站系统的框架设计

主要内容:介绍基于 SOA 分层架构完成典型的 Web 网站系统(后端系统)的组成和框架设计所需的技术;介绍 App 客户端设计、微信小程序设计和相应的后端支持环境;可结合后面要出

现的案例,引入公有云或私有云。

作业设计:

1. 给定项目案例需求或已有系统的创新需求,设计典型的 Web 网站系统。

2. 结合已有资源(用于软件工程教学的 Web 平台),扩展某一层的组件。

本单元建议讲授课时:6~8 课时。

第四单元　数据集成

本单元可结合后面的案例来讲解。

4.1　数据集成的基本技术

主要内容:介绍大规模领域软件系统和集成项目中数据采集、处理(清洗)、存储与管理、访问、发布的主要过程和技术手段、主要技术产品及平台。

4.2　数据分析与可视化技术

主要内容:介绍 OLAP,数据仓库,数据可视化技术的基本概念、主要过程、分析模型,典型的可视化形式、创新的系统应用。

4.3　大数据处理及分析系统

主要内容:介绍大数据采集、管理、计算、分析的基本流程、系统组成、主要功能及设计方法。

本单元建议讲授课时:4~8 课时。

作业设计:

1. 数据处理技术与系统的综述、典型的系统(如数据共享与交换平台)。

2. 结合物联网系统、智慧城市系统、智慧校园系统等典型应用,进行数据集成的分析和设计。

第五单元　大规模领域软件系统创新案例

本单元可根据教学资源、实践基地、合作企业和校外企业指导专家的实际情况,介绍 2~4 个典型的大规模领域软件系统;可以与数据集成、应用集成单元的内容相结合来讲解案例。

主要介绍的案例可以是:大型行业软件产品或应用、电子政务应用、智慧校园软件系统、智慧城市综合服务系统、电子商务系统等。

本单元建议讲授课时:4~8 课时。

作业设计:

1. 行业软件应用创新的案例分析,分析其中所涉及的架构、集成技术。

2. 与应用架构设计、数据集成单元相结合,完成子系统的分析与设计。

第六单元　软件运营服务

主要内容:介绍软件运营的理念、重要性、主要任务、主要技术,强调好的大规模软件系统是可运营的系统;介绍 ITSS 服务标准、ITIL 软件技术服务体系及实现平台。

本课程建议讲授课时:2~4 课时。

七、考核要求

建议按需设置考核方式及比例。

八、编写成员名单

赵沁平(北京航空航天大学)、熊桂喜(北京航空航天大学)、胡春明(北京航空航天大学)。

04 软件体系结构

一、课程概述

软件体系结构研究软件系统的基本组成元素、元素的外特性以及元素之间的相互关系。软件体系结构师是软件团队中最重要的技术角色,从宏观和全局的角度做出重要的软件设计决策。软件体系结构是软件工程一级学科研究生核心课程,是一门关于复杂软件系统整体高层结构设计和分析的课程。

随着信息系统与软件产品规模的急剧扩大,软件体系结构已经成为软件工程领域的热点及关键技术。本课程的目的是通过对软件体系结构内涵、软件体系结构建模、软件体系结构设计方法、分析和评审等原理和方法的介绍,并通过案例分析,培养学生对具有一定规模的软件系统的体系结构设计和分析能力,使学生能针对现实中的具体系统做出最佳的体系结构设计决策,从而充分培养学生的抽象思维能力、面向全局的系统分析与设计能力、运用知识求解实际问题的能力、独立思考与创新能力,为学生进入软件企业工作和从事软件工程领域科研工作奠定良好的技术基础。

本课程强化对系统的复杂性、效率、演化、抽象层次、复用、折中等计算学科核心概念的理解。

二、先修课程

高级语言程序设计、软件工程。

三、课程目标

软件体系结构课程的目标是培养学生的软件体系结构意识,理解软件体系结构在复杂系统开发中的重要性,掌握软件体系结构设计、分析、评估的原理和方法,了解常用的体系结构风格和策略;培养学生在体系结构层次的抽象思维能力、面向全局的系统分析与设计能力,以及针对功能和非功能需求进行折中分析并做出决策的能力。

四、适用对象

软件体系结构课程适用于软件工程一级学科的博士研究生和硕士研究生,也可以作为与计算机有关的学科研究生的选修课。

五、授课方式

软件体系结构是一门实践性很强的课程,建议采用课堂授课、案例分析、课堂讨论、实验相结合的授课方式。

六、课程内容

本课程建议学分为2~3学分(30~46学时),可根据各个学校的培养特点,合理选择知识模块或知识点。

软件体系结构课程的知识模块和知识点包括:

第一模块　软件体系结构概述(2学时)

主要内容:介绍软件体系结构的内涵、重要性,介绍基于软件体系结构的软件开发过程、软件体系结构业务周期(ABC),架构师的能力需求等;并通过一个简单的案例初步展示软件体系结构。

第二模块　软件体系结构设计(12~18学时)

主要内容:介绍软件体系结构的设计方法。首先讨论面向体系结构设计的软件需求分析,重点介绍软件质量属性及其描述方法;然后讨论软件设计模式、软件体系结构风格、软件体系结构策略/战术,并在此过程中结合大量的案例进行分析和研讨。

这部分要结合经典和现代两方面进行讲授:经典设计模型和最新的设计模式;经典的体系结构风格和最新的进展(如微服务架构)等。在案例的选择中,要考虑学生的知识背景,如对Windows操作系统各个版本的体系结构分析等;也可以介绍目前主流的一些软件体系结构,分析其设计特点和优势。

本模块为课程核心内容,旨在培养学生细化需求并以此为依据进行体系结构设计的能力,在此过程中充分利用已有的体系结构风格、策略和设计模型,培养学生分析问题和综合权衡的能力,培养系统的全局观。

第三模块　软件体系结构描述和编档(4学时)

主要内容:介绍软件体系结构文档的编写,包括软件体系结构建模语言(如UML等及其相关工具)、软件体系结构描述语言ADL、软件体系结构文档的编写规范,并结合具体案例进行分析和讨论。

第四模块　基于软件体系结构的软件开发(4学时)

主要内容:讲解基于软件体系结构的软件开发特点,构件化、模型化、增量迭代方法;重点介绍模型驱动的方法,讲解四层元模型、建模方法和模型转化方法。本部分内容可以结合第二模块中软件体系结构建模讲解。

第五模块　软件体系结构分析、演化和维护(4~10学时,选讲)

主要内容:介绍软件体系结构的分析评估方法,包括软件体系结构度量和评估、软件体系结构仿真与形式化验证、软件体系结构分析和评审、案例分析和研讨等。可根据情况设定学时。

软件体系结构演化和维护可作为选讲内容,包括软件体系结构恢复和重构、软件体系结构腐蚀和对策、软件体系结构"坏味道"、案例分析和研讨。

第六模块　领域软件体系结构(选讲)(2~4学时)

主要内容:介绍软件体系结构重用和软件产品线中软件体系结构的内容,重点是可配置体

系结构;讨论面向特定领域的软件体系结构,包括软件体系结构建模语言的定制、特定的分析方法等,可结合特定领域的体系结构案例、体系结构框架进行分析和讨论。

第七模块　软件体系结构领域的最新进展(选讲)(2~4 学时)

本部分内容可以融入前面的章节介绍,也可以最后单独介绍和讨论。可以教师讲解,也可以组织学生自主查资料并进行研讨。

七、考核要求

本课程成绩建议采用实践作业和课程考试相结合的方式评定。建议课程实践作业和课程考试各占总成绩的 50%,可根据各个学校的情况适当调整。

实践作业可以是典型系统的体系结构设计或者典型系统的体系结构分析等;课程考试可采用闭卷考试的形式,也可以采用课程论文与命题报告的形式。

八、编写成员名单

张莉(北京航空航天大学)、李波(北京航空航天大学)。

05　软件分析与测试

一、课程概述

软件质量是贯穿软件整个生命周期的关键指标,程序分析和软件测试是保障软件质量的主要技术手段。本课程主要讲解软件质量模型、程序分析和软件测试的方法和技术、软件分析和测试工具以及若干应用案例分析,并配合课程实验,力图使学生掌握软件质量的概念、软件分析和测试的理论知识并具备实践能力。所有软件都需要进行分析和测试,因此,本课程在软件工程一级学科研究生课程体系中具有基础性的地位和作用,建议列为必修课或学位课程。

二、先修课程

学习本课程之前,应具备软件工程理论基础知识和至少一种编程语言开发能力,建议先修课程为软件工程基础、程序设计等。

三、课程目标

学生通过本课程的学习,应了解软件质量的概念和软件质量保证的过程;重点掌握软件质量保证的两个主要手段,即程序分析和软件测试;掌握常用程序分析和软件测试技术;熟悉软件测试各个阶段和过程;掌握目前主流的软件分析和测试工具。课程实验培养学生具备一定的独立进行软件分析和测试的实践能力,为其今后更深入地学习或从事软件工程相关工作打下坚实基础。

四、适用对象

本课程适用于软件工程一级学科的博士研究生和硕士研究生,对软件工程一级学科的所有学科方向均适用。

五、授课方式

本课程的授课方式主要为课堂教学和课程实验。

六、课程内容

第一部分　软件质量与软件度量

1.1　软件质量模型与软件标准

1.2　软件度量

1.2.1　GQM 模型

1.2.2　规模度量方法

1.2.3　成本度量方法

1.2.4　复杂性度量方法

第二部分　程序分析的技术与工具

2.1　代码评审与静态分析

2.1.1　检查表技术

2.1.2　形式化代码静态分析技术

2.2　程序静态分析工具

2.2.1　静态分析工具原理

2.2.2　主要工具介绍:当前流行的 Findbug、checkStyle 和 PMD

2.3　代码分析案例

主要内容:通过静态分析工具,利用检查表技术对案例代码进行静态分析;同时讲解如何结合单元测试、设计模式等技术提升代码质量。

第三部分　软件测试技术

3.1　探索式测试

3.1.1　全局探索式测试法

3.1.2　混合探索式测试技术

3.1.3　基于场景的探索式测试

3.2　基于模型的测试(MBT)

3.2.1　基于业务建模的 MBT 方法

3.2.2　基于 UML 的 MBT 方法

3.2.3　基于分层有限状态机的测试技术

3.3　测试用例的自动生成技术

3.3.1　基于路径的测试用例自动化生成

3.3.2　基于遗传算法的测试用例自动生成

3.3.3 基于规则提取的自动化测试用例生成

3.4 云测试技术

3.4.1 云测试原理、类型及使用场景

3.4.2 云测试案例分析:服务商

第四部分 自动化测试框架与工具

4.1 功能测试工具

4.2 性能测试工具

4.3 安全测试工具

4.4 移动 App 测试工具

4.5 自动化测试框架

第五部分 全程软件测试

5.1 测试过程模型

5.2 全程静态测试

5.3 全程安全测试

第六部分 测试案例分析

主要内容:以一个包含 Web 前端、移动 App 和服务器端的软件系统作为测试目标,分析一个完整的测试案例。

6.1 利用 Selenium+TestNG 实现功能测试

6.1.1 数据驱动的测试代码的编写

6.1.2 业务模型驱动的测试代码的编写

6.2 利用 JMeter 实现性能测试

6.2.1 性能测试代码的编写

6.2.2 性能测试报告解读

6.3 利用 Appnium 实现移动 App 测试

6.3.1 功能遍历的测试用例编写

6.3.2 多手机测试

6.4 利用 AWVS 实现安全测试

6.4.1 安全测试用例的设计

6.4.2 安全测试报告解读

6.4.3 系统加固

七、考核要求

本课程考核方式为笔试和实验考核结合,笔试主要考核学生掌握课程知识的程度,实验主要考核学生进行程序分析和软件测试的动手能力;考核比例按需设置。

八、编写成员名单

陈纯(浙江大学)、杨小虎(浙江大学)、张亶(浙江大学)。

06 软件工程管理

一、课程概述

软件工程一级学科研究生核心课程包括理论、系统、开发、质量四个类别共10门课程,软件工程管理属于质量类别,本课程教学的主要内容覆盖软件工程管理经典理论和方法、前沿的软件工程研究和实践以及软件工程管理的相关知识。

首先,课程结合软件工程本质难题来讨论软件工程管理的必要性;然后进一步介绍前沿软件的历史发展,尤其是软件工程本质难题在不同历史时代的演变,引出软件工程管理中的重要理论和方法的历史演变,探讨其发展趋势以及背后的驱动力。在此基础上,本课程围绕两条线索详细讲解软件工程管理关键知识:项目管理,包括立项和组织、估算、计划、跟踪以及总结等;过程管理,包括过程定义、执行、诊断和改进等。在上述知识教学过程中,本课程同时介绍软件工程管理有关概念,例如瀑布生命周期模型、迭代式开发、CMMI、敏捷软件开发方法、精益软件开发、DevOps等。此外,本课程还涉及开源软件开发方法学等内容。

本课程通过理论与案例教学,让学生掌握软件工程管理的概念、方法,从而培养学生综合运用所学知识进行软件工程管理的能力。

二、先修课程

软件工程本科专业或计算机科学与技术本科专业合格毕业生具备的知识。

三、课程目标

通过本课程的学习,学生应理解软件工程管理的概念和方法;掌握项目组织、立项与范围管理、估算与计划、计划执行、评审与回顾、度量等具体实践与工具;在面临特定上下文环境时,能够定义和使用适合的软件工程管理实践和工具来实现项目特定目标。

四、适用对象

软件工程专业的博士研究生和硕士研究生。针对本科学习过软件工程管理相关课程的学生,建议突出案例教学;针对没有学习过相关课程的学生,教学内容应当覆盖课程内容的所有知识点。

五、授课方式

本课程采用课堂讲解、课堂讨论、案例教学、课后阅读和课程项目等形式进行授课;在课程项目中强调软件工程工具的使用。

教师可以根据授课人数及学生背景的不同,灵活安排上述教学活动的比重和形式。

六、课程内容

第一章　软件工程历史和软件工程本质难题

第二章　软件工程管理概述

主要内容：软件工程管理与软件过程、软件生命周期的关系；软件项目管理三大典型目标（成本、进度、质量）、三大目标统一于质量目标的原因。

第三章　项目组织

主要内容：TSP 角色；Scrum 人员角色；自定义角色。

第四章　项目立项与范围管理

主要内容：需求获取（用例与用例图、用户故事与用户故事地图等）；需求管理。

第五章　软件项目估算与计划

主要内容：软件项目估算的本质；基于历史数据的估算方法；风险识别；故事点；扑克牌估算法；计划会议；迭代周期（短周期迭代）。

第六章　计划执行

主要内容：软件项目跟踪活动；风险跟踪；挣值管理方法；每日站立会议；燃尽图；Kanban；简单设计；重构；测试驱动开发；持续集成；DevOps。

第七章　质量管理

主要内容：面向用户的质量观；质量目标识别和优先级；质量建模、估算和计划；质量过程（评审、测试）控制。

第八章　评审与回顾

主要内容：里程碑评审；进度评审；一般评审会议；回顾会议。

第九章　软件工程度量

主要内容：GQM 和 GQM+度量体系；过程度量；产品度量。

第十章　软件过程管理

主要内容：多维度过程特征（个人与小组、敏捷与规范、框架与实践等）；过程融合和裁剪；过程改进基础设施；过程改进参考模型（PDCA 和 IDEAL 模型）。

第十一章　敏捷软件开发宣言和原则及 CMMI 框架

第十二章　企业软件工程

主要内容：Google、百度。

第十三章　开源软件开发方法

主要内容：大教堂与市集；Linus 定律。

第十四章　端到端工具链

主要内容：协同开发支持工具、持续集成工具、配置管理工具、测试工具、代码扫描工具、APM 监控工具等。

七、考核要求

本课程考核由作业、课程项目和期末考试构成，建议三部分占比分别为 20%、40%和 40%。（考核方式及比例由授课教师按需设置。）

1. 作业:阅读软件工程经典书籍与论文,并提交读书笔记5篇。

2. 课程项目(分组完成):

项目来源:教师指定或者学生自选;

项目管理实践:分为两个迭代周期,要求学生按照Scrum方法来完成项目(包括人员角色定义、需求获取、估算、计划、进度跟踪、评审会议、回顾会议,学生应当提供这些工作的过程证据);

项目工程实践:要求学生采纳极限编程中的一些工程实践原则,主要是测试驱动开发、持续集成;

软件工程工具要求:项目管理工具,代码控制工具,持续集成工具(要求学生提供相应工具使用截图),如果是互联网项目,要求搭建并使用基本的DevOps工具链。

3. 期末考试。

八、编写成员名单

邵栋(南京大学)、荣国平(南京大学)、张贺(南京大学)、李宣东(南京大学)。

07 软件安全

一、课程概述

软件工程一级学科研究生核心课程包括理论、系统、开发、质量四个类别共10门课程,软件安全属于质量类别,本课程教学的主要内容覆盖软件漏洞的攻击和防御机制,应用于软件安全的程序分析技术,引导学生对一些前沿的攻防技术问题做深入讨论。

本课程首先讲述软件安全的历史、概念和表现形式,然后讲述各种安全威胁和防御机制在不同时期的演变,探讨其发展趋势以及背后的驱动力。在此基础上,本课程围绕两条线索详细讲解软件安全关键知识:软件漏洞和攻击方式,包括各种漏洞类型的成因和危害、攻击方式和实施流程等;软件安全防御机制,包括各种代码审查机制、攻击平缓机制、程序行为管理等。在上述知识教学过程中,同时介绍在软件安全领域中广泛使用的程序分析技术,例如染色分析、符号执行、指针分析等。

本课程通过理论学习与上机实践,指导学生掌握软件安全领域的概念、方法和应用,并通过阅读相关研究论文,引导学生讨论热点问题可能的解决思路。

二、先修课程

C/C++程序设计语言、X86汇编程序设计、计算机系统结构、操作系统、编译技术。

三、课程目标

通过本课程的学习,学生应理解软件安全问题的本质;熟练掌握针对基础的内存破坏型漏

洞的攻击及相应的防御技术;应用程序分析技术实现基于控制流和数据流追踪的漏洞发现;在对若干问题作深入讨论的基础上,阅读最新的研究成果,思考这些问题可能的解决思路。

四、适用对象

软件工程一级学科的博士研究生和硕士研究生。

五、授课方式

授课方式包括课堂教学、演示和讨论。学生在阅读指定论文的基础上,系统性地理解软件安全的本质问题和难点,尽量提出自己的解决思路;课后在助教的指导下熟悉必要的工具,完成相应的实验项目。

六、课程内容

本课程主要讲述软件安全的基本概念、软件安全问题的本质、常见的攻击与防御技术以及挖掘软件安全问题所利用的各类程序分析技术及工具。

第一部分　软件安全概述

主要内容:信息安全和计算机系统安全的发展;什么是软件安全;软件安全的由来;软件安全威胁的广泛表现形式——程序漏洞;威胁防不胜防的原因——操作系统、库函数、数据库、Web、动态链接机制、内存分配与回收机制等;如何应对——代码审查、攻击平缓、程序行为管理等。

第二部分　内存破坏(Memory Corruption)型漏洞以及攻击原理

主要内容:

1. 内存破坏型漏洞是普遍存在、被广为利用的软件漏洞;漏洞的本质是提供了空间(Spatial)或时间(Temporal)维度越界访问的可能;越界访问的具体表现形式包括缓冲区溢出(Buffer Overflow)、格式化字符串(Format String)溢出、整型溢出(Integer Overflow)、UAF 漏洞(Use-After-Free)等。

2. 最常见的攻击方法是通过越界窜改如代码指针(Code Pointer)等敏感数据,以实现控制流劫持(Control Flow Hijacking),并通过代码注入技术(Code Injection)、返回库函数(Return-into-Libc)、面向 Return 的编程(Return Oriented Programming,ROP)等技术实现恶意语义。

3. 针对内存破坏型漏洞的攻击除了劫持控制流以外,还包括信息泄漏(Information Leak)、非控制流攻击(Non-Control Data Attack)等。

第三部分　相关的防御技术

主要内容:在软件漏洞的形成、被攻击以及恶意操作的各个阶段,都可以有针对性地构建防御机制,包括以下 3 种方法。

1. 编写安全的程序或通过代码审查、漏洞挖掘等技术,发现和修复程序中易受攻击的安全弱点。

2. 通过监控某一类漏洞攻击明确的特征,达到阻止攻击或平缓攻击结果的目的,如保护返回地址(StackGuard),格式化串警卫(FormatGuard),栈不可执行(DEP),避免泄露地址先验知识(ASLR)等。

3. 通过规范化地描述程序的安全行为来对攻击进行防范,如控制流完整性(CFI)检查,数据

流完整性(DFI)检查等。

第四部分　程序分析技术在软件安全领域的应用

主要内容:软件安全运行特征模型的建立需要一系列程序分析技术的支持,包括控制流分析、数据流分析、数据依赖关系、染色分析(Taint Analysis)、符号执行(Symbolic Execution)、指针分析(Point-to Analysis)等。程序分析技术应用于软件安全保障机制,例如利用控制流和数据流分析实现 CFI 和 DFI;利用染色分析实现漏洞定位;利用指针分析消除悬空指针(UAF 防御);利用符号执行技术提高漏洞挖掘的路径覆盖率和自动化构造攻击。

补充说明:在上述内容的基础上,可以选择性地介绍其他相关的软件安全技术,如软件安全工程、Web 安全、恶意软件(Malware)和程序后门、安卓系统的安全等。

1. 软件安全工程是从软件开发和软件工程的角度讲解构建安全软件的实践方法,具体内容包括软件安全的构成、安全软件的需求、安全软件的架构和设计、安全编码和测试、系统集成、安全管理等。

2. Web 安全包括 SQL 注入、DDoS、流量劫持、XSS、CSRF 等攻击及应对措施。

3. 恶意软件和程序后门包括病毒、蠕虫、间谍软件、木马、后门等。

4. 安卓系统安全包括代码混淆和保护、访问控制、侧信道攻击及防御等。

除此之外,也可以围绕 1~2 个主题,每个主题提供 3~5 篇论文,引导学生阅读和讨论,建议由易到难,例如:基于 Canary 的栈保护技术;控制流完整性(CFI)检查技术;染色分析及 UAF 防御。

另外,学生需要了解软件安全研究领域常用的工具,这不仅有助于增强其对软件安全技术的理解,也是完成本课程的课程实验所必需的。安全研究常用工具包括:软件漏洞和攻击资源库,如 CVE 等;二进制反汇编和调试工具,如 IDA Pro、GDB 等;攻击辅助工具,如 Metasploit、ROPgadget 等;程序分析和插桩工具,如 PIN、Valgrind、QEMU、Angr 等。

七、考核要求

本课程采用论文阅读、实验和期末考试综合评定的方式进行考核,以上各部分建议占比分别为 20%、40%和 40%。

八、编写成员名单

茅兵(南京大学)、李宣东(南京大学)。

08　分布式系统

一、课程概述

云计算、移动计算、大数据等新兴计算模式和服务背后的支撑系统,几乎都采用或依赖于分

布式系统和相关技术，例如 Web 服务、云存储、电子商务、在线支付、数字金融、社交和游戏平台等。本课程从分布式系统的基本概念和关键技术出发，讲述分布式系统的设计原则与实现技术，并对当前分布式存储和计算系统中的重要问题，通过对比经典和前沿系统中设计和技术进行学习。课程内容包括分布式存储和分布式计算两大板块，既包含重要理论（如一致性协议、分布式提交等）又涉及真实系统的分析（如 GFS、Pregel 等）。本课程旨在给计算机科学与技术和软件工程专业的研究生在分布式系统方面提供理论准备，使其掌握有关的设计方法及思想，能够运用这些技术和方法构建分布式系统，为后续系统开发奠定基础。

二、先修课程

学生应具备基本的计算机系统相关知识，掌握必要的计算机系统设计原理，具体先修课程包括程序设计、计算机系统基础和操作系统等。

三、课程目标

本课程为学生在分布式系统领域的研究打下坚实的理论基础；使学生掌握核心概念和解决问题的基本手段和方法，了解本领域研究前沿与产业前沿；培养学生拥有较强的实践能力和自学能力。

四、适用对象

本课程适用于软件工程一级学科的硕士研究生。

五、授课方式

本课程强调基本原理和设计分析能力并重，因此在教学上除了课程讲授外，还要求学生以团队的方式对一个有代表性的分布式系统进行全面案例分析（包括论文研读、系统介绍、方法模拟、功能演示等）。课程内容分为分布式存储和分布式计算两大板块，以解决实际分布式系统中的关键问题为单元进行讲授和学习。每个单元从基础理论和技术出发，通过案例系统的分析对比来强化对理论和技术的掌握和应用，明确在实际设计过程中必须面临的取舍问题。最后在课程实验和案例讲授之外，增加论文阅读和书面问题解答部分，帮助学生进一步培养自学能力。学生在研读论文、搜索背景资料和组织表达的过程中巩固所学，同时也了解分布式系统发展历程和领域前沿。

六、课程内容

课程内容按照 16 学时安排。

第一学时　课程介绍、分布式系统概要

- 重点和难点：什么是分布式系统、分布式系统设计的主要考虑因素和评价方法等。

第二学时　顺序一致性

- 重点和难点：一致性协议的多面性、顺序协议与释放协议的关系和比较、分布式时间等。

第三学时　最终一致性

- 重点和难点：冲突检测和处理、函数更新、因果关系维护、稳定状态算法等。

第四学时　原子性与日志方法

■ 重点和难点:All-or-Nothing 概念、REDO-UNDO 日志算法和恢复算法、文件系统的日志应用等。

第五学时　分布式事务处理协议

■ 重点和难点:乐观和悲观并发控制协议、行为异常的分类、多版本控制和镜像隔离协议等。

第六学时　分布式提交协议

■ 重点和难点:二段式提交、中止算法、分布式提交的算法变种等。

第七学时　分布式协同算法

■ 重点和难点:PAXOS 协议、Replicated State Machine、FLP 理论。

第八学时　分布式文件系统

■ 重点和难点:传统分布式文件系统、大规模分布式文件系统、重要考量因素和设计比较等。

第九学时　分布式系统安全

■ 重点和难点:安全模型、安全挑战、沙盒等基本保护方法等。

第十学时　数据并行编程模型

■ 重点和难点:MapReduce 编程模型和运行时系统、Dryad 系统、不同系统的设计比较等。

第十一学时　图计算编程模型

■ 重点和难点:图计算问题的难点、基于数据并行计算系统的缺陷、图计算系统典型设计,同步异步计算模型等。

第十二学时　分布式任务调度

■ 重点和难点:分布式系统模型、去中心化调度方法、负载平衡技术,尾时延优化等。

第十三学时　分布式数据划分

■ 重点和难点:图数据划分的关键问题、边划分和点划分方法、系统负载均衡方法等。

第十四学时　分布式外存计算

■ 重点和难点:内存计算与外存计算比较、硬件环境对系统设计的影响等。

第十五学时　分布式流计算系统

■ 重点和难点:流计算的关键问题、流计算的扩展性、流数据的容错机制等。

第十六学时　分布式计算容错方法

■ 重点和难点:基本容错方法、数据和计算特性对容错和恢复方法的影响等。

七、考核要求

本课程考核由课程考试、团队案例研究和演示、书面作业等组成,各部分所占比例可按需调整。

1. 课程考试:主要考核学生对课程内容的掌握程度,建议采用开卷或半开卷形式,注重实际问题的解答和系统的设计思考。

2. 团队案例研究和演示:主要考核学生对分布式系统某个具体问题采用的设计和技术以及背后原因的分析,并通过动手开发来演示该工作的某方面特性。

3. 书面作业:主要考核学生对分布式重要问题的自学能力,通过论文总结和回答问题的方式帮助学生进行课前预习。

八、编写成员名单

臧斌宇(上海交通大学)、陈榕(上海交通大学)、夏虞斌(上海交通大学)、王肇国(上海交通大学)。

09 软件需求工程

一、课程概述

软件需求工程是软件工程专业系列核心课程之一,主要培养学生的问题定义与系统分析能力。课程内容主要包括需求工程过程、需求抽取技术、需求建模与分析方法、需求文档化、需求验证与磋商、需求工程典型案例与需求工程最新研究进展。本课程围绕各教学重点介绍本领域的最新研究进展,带领学生实践工业界采用的最新技术、过程、方法和工具,目的是为后续的软件设计、软件测试、软件演化与维护等软件工程活动提供上游输入。

本课程教学内容具有基础性、实践性和前沿性的特点。教学活动与企业最新技术紧密结合,教学内容涵盖软件生命周期中的需求活动,需求获取技术,企业与组织机构建模方法,系统目标与行为建模,非功能性需求建模,需求不一致性处理,需求规约、测试、验证与演化。

二、先修课程

学生在学习本课程之前应完成至少一门程序设计课程。

三、课程目标

本课程旨在让学生理解需求工程基本原理、建模技术、经典方法和最新最佳实践。学生通过本课程所能掌握的知识和技能如下:

1. 了解软件需求工程在软件工程和系统工程中的重要地位;了解软件需求工程活动的性质;

2. 了解和应用软件需求工程的概念、方法、过程和工具;

3. 掌握常见的需求工程典型方法,包括面向目标的方法、面向主体的方法、面向情景的方法以及问题驱动的方法等;

4. 熟悉非功能性需求的分析技术、形式化方法等工作难点与重点;

5. 学习软件需求工程领域当前最新研究成果和最佳实践,通过课堂讨论及课后作业,给学生提供软件需求工程的实践机会。

四、适用对象

软件工程一级学科的硕士研究生。

五、授课方式

本课程采用授课与课程项目实践相结合的方式授课,将讲授、思考与交流、实践三个环节进行有机结合。

课程讲授:在软件需求工程理论知识的讲解中注重贯穿实际的软件系统分析案例,使学生真正理解这些理论知识,建立软件需求工程过程的系统化与工程化观念和质量意识,掌握软件需求工程的最新技术。

启发思考:注重鼓励和引导学生进行探索式学习,学生通过文献查阅以及与软件企业人员的接触交流,真正体会当前软件工程业界的真实案例和最佳实践;同时,在教学过程中营造轻松活跃的课堂气氛,开展互动式的讨论,鼓励学生结合课程实践中的问题进行专题报告和软件演示。

六、课程内容

本课程主要包括四部分:

第一部分　需求工程基础

主要内容:讲授需求工程的基本原理、建模技术和需求工程活动过程。

1.1　课程介绍、软件需求工程导论

1.1.1　课程设置的目的及基本要求

1.1.2　软件需求工程的重要性

1.1.3　软件需求工程的基本概念及其应用

1.1.4　软件需求工程的性质

1.2　软件需求工程基础

1.2.1　什么是软件需求

1.2.2　软件需求的种类

1.2.3　不良软件需求的实例列举

1.2.4　高质量需求的标准

1.2.5　需求质量检验工具的研发与应用

1.3　软件工程周期模型中的需求工程活动

1.3.1　瀑布模型中的需求

1.3.2　原型法中的需求

1.3.3　螺旋模型中的需求

1.3.4　V 型模型中的需求

1.3.5　演化及增量开发模型中的需求

1.3.6　敏捷模型中的需求——用户故事

第二部分　需求工程方法

主要内容:介绍需求工程的几种典型方法,包括面向目标的方法、面向主体的方法、面向情景的方法、问题驱动的方法等;在介绍经典方法的基础上,教师可以灵活开设与个人研究领域和兴趣相关的进阶课程,例如面向大规模群体的需求调研、面向流程密集型企业的需求获取方法、

面向安全攸关型系统的需求规约方法等。

2.1 软件需求获取技术(Ⅰ)

2.1.1 什么是软件需求获取

2.1.2 高效的获取过程及其结果

2.1.3 薄弱的获取过程及其结果

2.2 软件需求获取技术(Ⅱ)

2.2.1 现有技术、工具

2.2.2 需求获取的难点及重点

2.3 企业及组织机构建模

2.3.1 软件系统服务于企业目标

2.3.2 软件系统配置与组织机构关联

2.3.3 建模方法及工具

2.3.4 实例分析

2.4 信息及行为建模

2.4.1 信息/数据建模的方法

2.4.2 软件功能/系统行为的建模方法

2.4.3 工具及实例

第三部分 需求工程过程需要涉及的几个重要方面

主要内容:介绍典型非功能性需求的分析技术,形式化的需求规约和验证技术,以及领域工程和基于领域建模的需求开发技术等;在以上专题的基础上,教师可以灵活开设与个人研究领域和兴趣相关的进阶课程,例如安全需求工程、可用性测试、用户体验设计等。

3.1 非功能性需求建模

3.1.1 非功能性需求的重要性

3.1.2 非功能性需求驱动的软件设计

3.2 规约描述

3.2.1 规约描述语言介绍

3.2.2 规约描述过程

3.3 形式化分析

3.3.1 基于图结构的分析

3.3.2 基于语义的分析(OO、逻辑)

3.4 形式化方法

3.4.1 为什么使用形式化方法

3.4.2 形式化语言介绍(Z、进程代数)

3.4.3 形式化语义介绍(一阶逻辑、模态逻辑)

3.5 规约验证

3.5.1 模型检查

3.5.2 定理证明

3.6 需求变化及不一致性的管理

3.6.1 多视角方法

3.6.2 需求磋商及优先权定义

3.6.3 不一致性检查

3.6.4 工具支持

第四部分 需求工程热点和未来发展趋势

主要内容:其他最新涌现的需求研究问题和最新实践案例(建议案例基于移动应用类、大型企业应用类、新型通用系统类等分类进行介绍,可根据教师实践经验和所掌握的案例素材定制),以及需求工程的未来发展趋势。

4.1 案例介绍与项目作业

教师命题案例:移动应用类、大型企业应用类(含工业及控制系统类)、新型通用系统类。

需求研究型课题:文献调研、新方法建议(游戏化、模型、过程等)、经验验证。(可三选一)

4.2 课程总结

教师需考查学生发现问题、明确定义问题的能力;

教师需考查学生向他人陈述问题和建议方案的能力;

教师需考查学生对需求分析方法的理解和运用能力。

课程实践设计方案
第一类选题:有实际应用背景、较大规模、需求复杂。 要求:完成需求分析、概念设计。 重点:需求抽取、领域相关业务知识的学习和获取、概念建模与分析、面向对象的分析与设计。 实例:电子海图与航行辅助系统、中国电信储值卡、伦敦救护车调度系统。
第二类选题:有应用背景、中小规模、需求相对直观、设计与部分实现可在学期内完成。 要求:完成需求分析、概念设计与详细设计、部分编码。 重点:需求抽取、概念建模与分析、面向对象的分析、设计与实现。 实例:课程注册系统、小型图书管理系统。
第三类选题:小规模、经典概念问题、需求明确、设计与实现可在学期内完成。 要求:完成需求的形式化、设计与完全编码实现、测试与验证。 重点:需求形式化、分析与验证、面向对象的分析、设计与实现。 实例:电梯控制系统、会议日程安排系统。

七、考核要求

考核方式为平时作业成绩加项目实践和(或)考试成绩,其中平时作业成绩占总成绩的40%~60%,项目实践和(或)考试成绩占40%~60%。要求学生能够熟练掌握需求工程的基础理论、方法与技术,能够应用典型的需求工程方法对具体的项目进行实践,并了解需求工程的未来发展趋势。

补充说明:

软件需求分析的重要性与相关理论方法的有效性很大程度上是在软件开发的其他活动中体现的,没有软件工程基础的学生无法通过其中的关联关系掌握需求工程的基本原理,例如由于不了解基于模型的测试方法,所以学生对建模语言与方法的理解不到位等。建议教师在教授

各知识点时，展示相关理论方法与软件开发各环节的关联关系，通过正面或反面的实例阐述其由来与合理性，真正让学生从研究者的角度审视需求工程中的基础理论与实践经验。

八、编写成员名单

金芝（北京大学）、刘璘（清华大学）、陈小红（华东师范大学）、李童（北京工业大学）。

10 软件开发方法学

一、课程概述

软件是人类制造的最复杂的一类制品，是人类大脑思维活动的体现，开发和演化这类系统需要有系统化的方法。软件开发方法一般指在软件开发过程中需要遵循的办法和步骤，如何高效、低成本地构造高质量的软件，是软件工程学科的基本科学问题。

软件开发方法学是软件工程专业系列核心课程之一，主要培养学生关于软件系统的“世界观”和“方法论”。这里，所谓世界观是指理解设计和构造软件的时候如何抽象其将来所处的环境及其应用目标，所谓方法论是指软件系统本身的抽象，包括：结构模型，即软件的组成元素、组合方式等静态构成形式；运行机理，即软件各组成元素动态运行及其间交互的机制和原理；构造方式，即如何通过层次化的问题分解和步骤分解来使系统构造满足完成高效可行任务的要求；质量保障，即如何定义并改善所构造软件满足目标。

本课程教学内容侧重基础性和前沿性，重点介绍模型驱动开发、与其有关的过程管理以及技术工具支撑，同时介绍业界流行的敏捷开发方法和实施方式。

二、先修课程

1. 修完软件工程、面向对象程序设计等课程；
2. 掌握统一建模语言（UML）；
3. 掌握至少一种面向对象的程序设计语言。

三、课程目标

本课程第一部分讲授模型驱动软件开发方法学的基本原理、方法、过程、管理、技术和工具等知识体系，使学生能够以模型为软件开发的第一制品，掌握软件开发的全过程以及在过程的各阶段中所采用的方法、技术和工具；使学生在传统软件工程的基础上认识通过模型驱动软件开发的意义，并更好地理解软件开发方法本身发展的规律和特点。

本课程第二部分讲授敏捷开发方法，学生将获得对敏捷管理原则和实践的理解，将学习如何协调敏捷开发过程的各个方面，包括运行设计冲刺、管理团队以及培养实验文化，并结合具体的项目，掌握如何应用学到的知识来完成真实的软件开发项目。学生按组完成项目，每个组按

分布式的模式进行软件开发,重点理解敏捷开发过程。

四、适用对象

软件工程一级学科的硕士研究生。

五、授课方式

本课程采用理论教学和实践教学两部分结合的方式进行授课。

理论部分教学任务是通过课堂讲解、讨论、专题介绍等方式讲授模型驱动软件开发的基本原理、方法和技术,使学生对模型驱动软件开发知识体系有一定层次的理性认识。

在讲授模型驱动软件开发理论知识的同时,要求学生参与开发规模适当的软件项目(例如,5~10人开发团队可以在2个月内完成的项目),熟悉主流的(元)建模、模型转换等工具,了解基于传统软件工程流程进行模型驱动软件开发的方法,进一步提高实践动手能力。

结合项目实践,掌握分布式环境下的敏捷开发方法。

六、课程内容

本课程主要包括五部分:

第一部分　传统软件开发方法回顾

主要内容:介绍软件开发方法的演化历史和形成原因,从传统结构化开发方法到主流的面向对象开发方法,再到目前普遍应用的模型驱动开发方法,以及相对独特的敏捷开发方法;介绍软件工程经典过程模型。

第二部分　模型驱动开发方法导论

主要内容:模型驱动开发方法概述;从技术角度剖析模型驱动开发方法的特点,所需的基础性方法、技术和工具;介绍工业界的相关标准和框架性规范,包括MDA、MOF、UML、OCL以及XMI,以及主流的实现技术和平台。

第三部分　模型驱动工程的技术架构

主要内容:介绍元模型和元建模理论;以OMG的元模型规范MOF和建模标准UML为具体案例,详细介绍经典的四层元-元模型体系;以事实上的工业标准Ecore为例,详细介绍简化版的三层元-元模型体系。

第四部分　模型转换理论和模型转换标准

主要内容:以MDA规范中的QVT标准为具体案例,详细介绍模型驱动开发的核心环节,双向模型转换的理论和技术;以基于Ecore实现的转换技术为例,详细介绍工业应用中使用的转换技术和工具;模型驱动开发案例介绍,包括元建模、业务建模以及模型转换的实践案例。

第五部分　敏捷开发方法

主要内容:敏捷开发方法概述;介绍敏捷软件开发宣言的提出及敏捷联盟,与传统软件开发方法相比,敏捷方法的独特之处;介绍敏捷开发的原则,以及开发过程的管理和协调;敏捷开发实践案例介绍。

七、考核要求

本课程的最终成绩由三个部分组成：课堂参与讨论的情况，占比 20%；平时课后练习及实验情况，占比 30%；课程结束后一个月内提交课程大报告，占比 50%。

八、编写成员名单

金芝（北京大学）、张天（南京大学）。

0836 见《学术学位研究生核心课程指南（一）（试行）》

0710/0836 生物学及生物技术一级学科研究生核心课程指南。