生词

Players in the Systems Game

1. 六种主要的**利益相关者（stakeholders）**包括：
    - 系统业主(System Owners): 原文提到他们是信息系统的赞助人和主要倡导者，通常负责资助项目的开发、运营和维护。
        - 系统业主关注资金投资回报和项目成果。
        - 系统业主负责**决策和资金支持**。
    - 系统用户(System Users): 原文描述他们定期使用或受信息系统影响的人，负责数据的捕获、验证、输入、响应、存储和交换。
        - 系统用户关心系统功能、易用性和信息质量。
        - 系统用户提供**需求和反馈**。
    - 系统分析师(System Analysts): 原文指出系统分析师研究组织的问题和需求，确定如何利用人员、数据、流程、通讯和信息技术改善业务。
        - 系统分析师关注**业务问题**解决和系统改进。
        - 系统分析师进行**需求分析和系统推荐**。
    - 系统设计师(System Designers): 根据原文，他们将系统用户的需求和约束转化为技术解决方案，设计文件、数据库、输入、输出、屏幕、网络和程序。
        - 系统设计师注重技术实现和系统架构。
        - 系统设计师**制定技术解决方案**。
    - 系统构建者(System Builders): 原文说明他们基于系统设计师的规格构建信息系统组件。
        - 系统构建者侧重于编码和系统集成。
        - 系统构建者**执行设计并构建系统**。
    - 信息技术供应商和顾问(IT Vendors and Consultants): 原文表示他们开发、销售和支持信息技术，作为购买其产品和服务的企业伙伴。
        - IT供应商和顾问聚焦于技术产品和服务的价值提供。
        - IT供应商和顾问提供**技术和咨询**服务。
2. 系统分析师的职责和需要的技能包括：
    - 职责：研究业务问题，提出解决方案，确保信息技术满足业务需求。
    - 需要的技能：具备信息技术知识、编程经验、商业知识、解决问题能力、人际沟通技巧、人际关系处理能力、灵活性和适应性、品格和道德，以及系统分析与设计的专业技能。这些技能帮助系统分析师理解业务需求，设计解决方案，并有效与团队成员及客户沟通。

Information System Building Blocks

- **三个信息系统构建模块**：数据（DATA）、处理（PROCESS）、界面/接口（INTERFACE）。
- **数据与信息的区别**：数据是关于组织及其业务交易的原始事实，大多数数据项孤立时几乎没有意义和用途。**信息是经过加工和有目的**智能处理后的数据，其中目的性智能至关重要——人们提供了产生真正信息的目的和智能。
- **前台系统与后台系统的区别**：前台信息系统支持面向客户的业务功能；后台信息系统支持内部业务运作，并与供应商（材料、设备、供应和服务）互动。
- **五类信息系统**：

1. **交易处理系统**：捕捉和处理有关企业交易的数据，如航空预订、银行存取款等。
2. **管理信息系统**：提供管理层所需的信息，以进行日常管理和决策。
3. **决策支持系统**：辅助高级管理人员做出战略决策。
4. **专家系统**：模拟人类专家的知识和推理过程，解决复杂问题。
5. **办公自动化系统**：提高办公室工作效率，处理文档、日程安排等。

- **六个利益相关者视角下的关注点**：
  1. **系统业主**关注的是信息，而非原始数据，他们需要能够增加业务知识的信息。业务知识是从及时、准确和相关的信息中获得的洞察力。
  2. **系统用户**是描述业务数据的专家，他们关心数据在当前存储方式或期望存储方式下的表现，系统开发的挑战在于正确识别和验证用户的业务数据需求。
  3. **系统分析师**促进信息系统和计算机应用程序的开发，通过架起非技术人员与技术人员之间的沟通鸿沟。
  4. **系统设计师**关注技术设计，包括用户和系统间接口，必须关注接口规范。
  5. **系统构建者**关注构建、安装、测试和实施用户和系统间接口，使用接口技术，如应用开发环境或标记语言。
  6. **IT供应商和顾问**关注销售硬件、软件和服务，使它们可以融入到企业信息系统中。

Information Systems Development

- **CMM（能力成熟度模型）**：
  - CMM是一个评估组织信息系统开发和管理过程成熟度的框架，它由五个级别组成，从初始级、可重复级、定义级、已管理级到优化级，衡量关键过程领域以**确定成熟度水平。**
    - Initial 1
    - Repeatable 2
    - Defined 3 （方法论
    - Manged 4
    - Optimizing 5
- **系统开发的生存周期**：
  - 包括系统开发和
  - 系统运行与支持两个阶段。
- **系统开发的8条基本原则**：
  - 涉及所有者和使用者。
  - 使用问题解决方法。
  - 确立阶段和活动。
  - 制定标准。
  - 将系统视为资本投资。
  - 不害怕取消或修订范围。
  - 分而治之。
  - 设计系统适应增长和变化。
- **FAST框架**：
  - 是系统开发方法论，不意味着快速开发或仅使用原型法，
  - 它包含预研、问题分析、需求分析、决策分析、设计、构建和实施等阶段。
- **PIECES框架**：
  - P代表提升性能，I代表提升信息，E代表提升经济效益，C代表提升控制，E代表提升效率，S代表提升服务。
- **系统开发的不同路线和方法**：
  - 模型驱动开发，快速应用开发，商用现成软件采购，混合路线，维护与再工程。
- **CASE（计算机辅助软件工程）和ADE（应用开发环境）**：

- CASE工具是自动化或支持系统模型绘制和分析的软件，能够将系统模型转换为应用程序。ADE是一套集成的软件开发工具，用于以最大速度和质量开发新应用软件，包括编程语言、接口构建工具、中间件、测试工具等设施。

Project Management

- **项目开发失败的原因**：包括未能建立高层管理对项目的承诺，组织缺乏对系统开发方法论的承诺，对项目期望管理不当，过早承诺固定预算和时间表，估算技术不当，对问题过于乐观，增加人力解决问题的误区，人员管理技能不足，未能适应业务变化，资源估计不足，以及偏离原定计划。
- **Project management与Process management的区别**：
  - 项目管理是范围界定、计划、人员配置、组织、指导和控制开发可接受系统的过程，
  - 过程管理是**记录、管理并改进**组织选定的系统开发方法论的持续活动，关注所有项目适用的活动、交付物和质量标准。
- **项目管理的8个功能**：项目管理功能包括
  1. 确定项目边界、
  2. 识别任务、
  3. 评估所需资源、
  4. 安排任务进度、
  5. 确保人员理解角色和职责、
  6. 指导项目组成员活动、
  7. 控制项目开发过程和
  8. 总结经验和教训
- **PERT图与Gantt图**：
  - PERT图是一种图形网络模型，描绘项目任务及其相互关系，在任务调度前明确任务间的依赖关系。
  - 甘特图是一种简单横条图，显示项目任务与日历的关系，每个条形代表一个命名的项目任务，垂直方向列出任务，水平轴为日历时间线，优势在于清晰展示重叠任务。

需求分析

- 需求通常被分为两大类：功能需求与非功能需求。
  - 功能需求指的是系统必须具备的具体功能或特性，这些功能或特性是为了满足业务需求并且使用户能够接受系统。例如，处理一笔活期账户存款、计算学生GPA以及收集账户持有者身份信息等都是功能需求的例子。
  - 非功能需求则描述系统的属性、特性和约束条件，它们可能限制解决方案的范围。非功能需求通过PIECES框架来分类，包括性能、信息、经济、控制与安全、效率等方面。
- Ishikawa图，也称作鱼骨图或因果图，是一种图形化工具，用于识别、探索和描绘问题及其原因和影响。它通常包括材料(Materials)、机器(Machines)、人力(Manpower)、方法(Methods)、地点(Places)、程序(Procedures)、政策(Policies)、人员(People)、周围环境(Surroundings)、供应商(Suppliers)、系统(Systems)和技能(Skills)作为潜在的原因类别。
- 七种重要的需求发现方法包括：
  1. 从现有文档、表格和数据库中采样。
  2. 进行研究和实地考察。
  3. 观察工作环境。
  4. 使用问卷调查。
  5. 开展访谈。
  6. 原型设计。
  7. 联合需求规划（JRP）。

- 用例(use case)是描述功能需求的一种方式，可以用UML用例图来可视化表达。用例是一种文本形式，详细描述系统如何响应外部事件或行为者的行为，以便实现特定的功能目标。。

数据建模和分析

- 系统对数据的要求可以通过模型来表达，模型是现实世界的一种表现形式。在系统分析中，我们主要使用逻辑模型（logical model），而在设计阶段则转向物理模型（physical model）。
    - 逻辑模型独立于技术实现，专注于展现系统本质，它帮助我们避免由于当前系统实施方式或个人预设想法带来的偏见，降低忽视业务需求的风险，还能用非技术语言与最终用户沟通。
    - 物理模型则展示了系统如何在物理和技术上得以实现，它受到技术选择及这些选择的局限性影响。

- 描述数据需求的模型是实体关系图（Entity Relationship Diagram，简称ER图）。ER图揭示了系统中数据的存在形式和它们之间的关联关系，其绘制过程遵循一系列步骤：
    1. **构建上下文数据模型**：首先定义项目范围，确定哪些实体将被纳入模型。
    2. **绘制基于键的ER图**：在此模型中消除不具体的关联，引入关联实体，同时加入主键和备选键。
    3. **构建全属性数据模型**：这个阶段包括所有剩余的描述性属性和子集标准。
    4. **规范化**：最后一步是对数据模型进行规范化验证，确保其符合第三范式（3NF），以消除数据冗余，提高数据一致性。

总结一下，模型是表达需求和设计的核心工具，逻辑模型和物理模型分别在分析和设计阶段发挥作用。ER图作为数据需求模型，其构建流程包括多个阶段，最后需要通过规范化验证来完善。熟悉和掌握这些模型和绘制流程对系统分析与设计非常重要。

模型之间的关系类型

在软件工程和数据建模中，不同模型或模型内的元素之间可能存在多种关系，包括但不限于：

继承（Inheritance）

继承是一种类之间的关系，其中一个类（子类）可以继承另一个类（父类）的属性和方法。在数据建模中，这表现为超类型（supertype）和子类型（subtype）实体之间的关系，其中子类型继承超类型的属性，并可能添加自己的额外属性。

依赖（Dependency）

依赖表示一个模块或实体依赖于另一个模块或实体的定义或存在。例如，在数据库设计中，一个实体的外键依赖于另一个实体的主键。

聚合（Aggregation）

聚合是实体之间的关系类型，其中一个复合实体（容器）可以包含多个部分实体。这种关系是弱关系，部分实体可以独立存在。例如，一个“课程”实体可以包含多个“学生”实体。

组合（Composition）

组合是聚合的一种特殊形式，其中部分实体不能独立于复合实体存在。当复合实体消失时，部分实体也随之消失。

泛化（Generalization）

泛化与继承类似，但更常用于UML等面向对象设计中，表示类之间的关系。在数据建模中，这通常表现为实体之间的超类型和子类型关系。

关联（Association）

关联描述了两个实体之间的简单关系，可以是一对一、一对多或多对多的关系。在ER图中，通过连线表示实体之间的关联，并标注关系的基数。

实现（Realization）

实现是接口和类之间的关系，表示一个类实现了某个接口所定义的行为。

在设计和理解复杂系统时，这些关系帮助我们组织和描述实体之间的交互和依赖，从而构建出稳定且易于维护的系统架构。

Process Modeling

数据流图（DFD）是结构化方法中的一种重要模型，用于描绘系统中数据流动和处理的过程。在绘制DFD时，需要掌握以下四个关键元素及其含义：

1. **外部实体**（External Agent）：表示系统边界外的参与者，可以是人、设备或其他系统，它们与系统交互，提供或接收数据。
2. **数据存储**（Data Store）：代表系统内部存储数据的地方，这些数据可以被多个过程访问和更新。
3. **处理过程**（Process）：指对数据进行操作或转换的活动，它接受输入数据，并产生输出数据。
4. **数据流**（Data Flow）：连接外部实体、数据存储和处理过程的箭头，表示数据从一个地方到另一个地方的移动路径。

绘制DFD的一般步骤如下：

1. **绘制上下文DFD**（Context DFD）：首先建立项目的初步范围，识别系统与外部实体之间的主要数据流。
2. **创建用例或事件响应列表**（Use Case or Event List）：定义系统必须响应的所有事件，这些事件触发系统的工作流程。
3. **进行功能分解**（Functional Decomposition）：将系统分解成更小的子系统或子过程，以便更好地理解和管理。
4. **为每个事件绘制DFD**（Event DFD）：为列表中的每一个事件或用例绘制详细的DFD，展示该事件下的数据流和处理过程。
5. **合并事件DFD**：将所有事件的DFD整合成一个系统图或子系统的图，形成完整的系统视图。
6. **绘制详细原始DFD**（Primitive DFD）：如果有必要，可以为复杂事件处理器绘制更细致的DFD，以清晰地显示数据流和处理细节。

最后，要确保所有数据流和过程都在数据字典中记录下来，这有助于维护模型的完整性和一致性。在整个过程中，基于事件划分的现代结构化分析方法更为常用，它侧重于业务事件及其响应来组织和设计系统。

Feasibility Analysis and the System Proposal

可行性分析

- **定义**：可行性分析是用来测量开发信息系统对组织有多大的益处或实际性的过程。
- **类型**：主要包括技术可行性、操作可行性、进度（时间）可行性、经济可行性。

技术可行性

- 衡量特定技术解决方案的实际性和可用的技术资源与专长。
- 考虑技术是否可在本组织的信息系统环境中使用，以及是否能获取到所需技术。
- 需要评估团队是否具备必要的技术专长，并且项目时间表是否合理。

操作可行性

- 测量解决方案在组织中的运行效果，也反映了人们对系统/项目的态度。
- 要解决的问题是否值得，或者解决问题的方案是否可行。

- 结束用户和管理层对解决方案的感觉。

进度可行性

- 给定技术专长的情况下，项目截止日期是否合理。
- 区分截止日期是强制性的还是期望的，以提出替代时间表。

经济可行性

- 许多项目的关键在于经济可行性。
- 成本效益分析是识别每个备选方案的成本和收益的过程。

成本效益分析(Cost-Benefit Analysis)

- 不要求进行具体的数学计算，但需要理解分析的基本原理。
- 成本包括开发成本（一次性成本）和运营成本（在整个系统生命周期中重复发生的成本）。
- 成本分为固定成本和变动成本。
- 收益分为有形收益（可以量化）和无形收益（难以量化）。

投资回报率(Return On Investment, ROI)

- 比较备选方案或项目的生命周期盈利能力。
- ROI是一个百分比，衡量业务从投资中收回的金额与投资额之间的关系。

净现值(Net Present Value, NPV)

- 许多管理者认为是首选的成本效益技术。
- 需要将所有成本和收益调整回到当前的美元价值。
- NPV是正数则投资良好，否则不良。

货币时间价值(Time Value of Money)

- 应用于上述每种技术的概念，认识到今天的1美元比一年后的1美元更有价值。

成本分析

- 发展成本（如人员成本、计算机使用、培训等）是一次性成本。
- 运营成本（如租赁支付、员工薪资、计算机使用费等）是整个系统生命周期内重复发生的成本。

通过以上信息，我们可以看出，文件中详细解释了可行性分析的各个方面，以及如何进行成本效益分析，包括投资回报率和净现值的计算方法和重要性。这些知识点直接对应于您的复习提纲要求。

Application Architecture and Modeling

- 系统架构：集中的（不用管这个）、分布式的

- 将系统划分成5层，这5层在分布的时候，有5种方式，就形成了不同的体系结构的模式，比较经典的是c/s(2层、3层) 、b/s。需要搞清楚这5层是哪5层——数据、数据操作、应用逻辑、显示逻辑、显示层，每一层是干什么的，在分布的时候，这些层怎么分就形成了不同的分布式系统的架构，怎么把一个logical的DFD转换成physical的DFD。

应用架构概念

- 应用架构定义了用于实施一个或多个信息系统的技术，涵盖数据、流程、接口及其在网络上的交互。
- 它为详细设计和实施提供了蓝图。
- 概念源自土木建筑学。

物理数据流图(PDFD)

- PDFDs是技术决策和设计决策的模型，用于信息系统的实现。
- 它们作为系统构建和实施的技术蓝图。
- 包含物理过程，即处理器或技术实现的具体工作，例如计算机程序或手动过程。

分布式与集中式系统

- 分布式系统将数据、流程和接口组件分散到网络中的多个位置，工作负载在网络中分布。
- 集中式系统中，所有组件集中在一台多用户计算机上，用户通过终端或终端模拟器交互，处理主要在主机上完成。

分布式计算的层次

- 展示层：用户界面。
- 展示逻辑层：如输入编辑。
- 应用逻辑层：业务规则、政策和程序。
- 数据操作层：存储和检索数据库中的数据。
- 数据层：实际的业务数据。

put

13 14 15这三章讲的都是interface，可能在选择题里面考概念。

分成三部分讲——输出、输入、GUI

输出种类：internal external turnaround 要知道这些是什么

输出设计时的基本原则

知道不同控件能用来解决什么输入就行

了解UI设计的基本原则

针对你提到的三章内容——输出设计、输入设计和图形用户界面设计，以下是复习重点的总结：

输出设计 (Output Design)

输出种类：

- **内部输出 (Internal Outputs):** 面向组织内部的系统所有者和用户，包括详细报告、摘要报告和异常报告。
- **外部输出 (External Outputs):** 面向客户、供应商、合作伙伴或监管机构。
- **周转文档 (Turnaround Documents):** 开始时作为外部输出，但最终会作为输入再次进入系统，例如账单和发票上的存根。

输出设计基本原则：

- 明确输出的目标受众，确保信息呈现清晰、准确且符合需求。
- 使用图表和图形展示趋势和关系，以便于理解和消化大量数据。
- 设计输出时考虑到不同的分发和交付方式，如打印机、屏幕、微缩胶片、网络链接、语音输出和电子邮件。

输入设计 (Input Design)

输入设计关注点：

- 数据捕获、录入和处理的方法和技术。
- 选择正确的控件来匹配输入数据的特性，如文本框、复选框、下拉列表等。
- 使用标准和已知的输入设计模式，减少用户的学习成本。
- 设计输入原型，进行验证和测试，使用布局工具和原型设计工具辅助设计。

内部控制对于输入的重要性：

- 监控每个输入和总输入数量，防止交易丢失。
- 验证所有数据，包括存在检查、数据类型检查、域检查、组合检查等。

图形用户界面设计 (GUI Design)

GUI设计的基本原则：

- 理解用户及其任务，使设计符合用户需求。
- 涉及用户在设计过程中，确保界面的实用性和易用性。
- 在实际用户上测试系统，观察和倾听他们的反馈。
- 实践迭代设计，因为界面设计可能永远不会有真正意义上的完成。
- 始终告知用户下一步该做什么，确保用户清楚系统期待的行动。
- 屏幕应格式化，使各种类型的信息、指令和消息始终出现在相同的显示区域。

GUI设计中的人机工程学指导方针：

- 尽量少地使用高亮、闪烁等视觉效果，避免分散用户注意力。
- 显示信息足够长的时间，以便用户阅读。
- 规定字段和用户需输入答案的默认值。
- 预测用户可能犯的错误，并在用户执行可能导致灾难性后果的操作时锁定键盘，指示其联系分析员或技术支持。

复习时应特别注意这些概念，因为它们可能会出现在选择题中，要求你识别或解释特定的输出、输入和GUI设计原则。

OO

划分三类：entity object interface object control object 得知道他们是什么，用来刻画什么样的对象的职责。也需要UML模型。

- 面向对象的概念方面，类是定义一组具有相同属性和行为的对象的蓝图，对象是类的一个实例，方法是类中定义的操作或功能，继承是指一个类可以继承另一个类的属性和方法，而多态则是指一个接口可以有多种不同的实现方式。

- UML建模中，
    - 用例图用于描述系统功能需求，它由参与者、用例和它们之间的关系组成，用于展示系统的行为视角。
    - 类图则用于静态结构的建模，描述系统中的类、接口、对象和它们之间的关系。类图上的元素包括类名、属性和方法，以及类与类之间的关系如关联、聚合、组合和泛化。
    - 状态图用于描述对象在其生命周期内的状态变化，以及导致状态转换的事件。活动图则用于展示业务流程或用例中活动的顺序和并发性，它由节点、边、泳道等元素组成，用于描绘工作流或算法的执行流程。