



如何搞掂一本  
书？

书掂

**BOOKDONE**

AI学练阅读器

2024.9.8 Hackathon in Canton

# 痛点自测

“知识点又细又密”  
“逻辑不清”  
“乱”

“全是国字，拼一  
起不认识了”  
“抽象”

“一道例题也没有”  
“看了=学了”  
“忘了”



知识点碎片化

理解难度高

学习路径单一

数据来源

《第二十一次全国国民阅读调查成果发布》 2023

《从碎片化数据到全面知识网络：知识图谱的构建与应用》 2022

《人工智能赋能个性化学习：E-Learning推荐系统》 2022

《基于知识图谱的碎片化知识整合与推送研究》 2021

《大数据时代成人碎片化学习:趋势、痛点与提升路径》 2020



#一句话介绍

# 书掂——AI学练阅读器

通过知识可视化技术  
将复杂文本拆解为可视化知识树

以AI自生成式教学  
配合自适应难度试题  
为用户提供高效、个性化的阅读学习体验

## 用户校准

# 针对结构化内容的学习群体

产品适用于具有客观分析，情境依赖，经验归纳，静态结构特性的文本内容，如经济学，教育学，历史学，医学，管理学等；

产品不适合学习具有强主观感受，公式计算，艺术创作，逻辑推演特性类的阅读作品，如数学、哲学，文学，艺术学等。

# 产品功能1\*MVP

## 「拆开书，长出树」

清晰，简洁  
结构化知识树

- 基于自然语言处理技术和图像识别技术，系统提取上传文本信息，拆解构建知识树。
- 知识提取Agent深度分析用户资料，精准识别文本中的关键信息和知识点，转换成半结构化的JSON数据，为教育过程提供清晰、有组织的知识基础。

文本分级可视化  
层层递进

- 通过知识树，每一节点皆可分级展开。“摘要”、“原文”或“自生成”三种选择，灵活调整学习深度

可选程度	摘要	原文	自生成
使用场景	快速浏览回顾	回到文本，深度理解	将复杂内容以个人偏好形式重新讲解
介绍	提供文本精简版，突出关键信息。	展示完整的原始文本，保留细节和深度信息。	利用自然语言处理技术，基于个人偏好形式，文本内容，提供更丰富的讲解视角。

# 产品功能2

## 「讲点你想听的」

### 定制化内容生成

- 通过深度学习算法理解用户偏好，系统采用多维度教学策略，动态生成个性化知识内容。

### 多种教学策略设定

- 包括视觉、言语、主动、直觉、反思和全局策略，用户可以根据个人偏好选择不同的授课方式、文本风格和语气，实现个性化学习体验。

维度	授课方式	文本风格	语气	策略
可选 & 说明	视觉 (Visual)	正式 (Formal)	鼓励性 (Encouraging)	演绎 (Deductive)
	言语 (Verbal)	教科书式 (Textbook)	中立 (Neutral)	归纳 (Inductive)
	主动 (Active)	通俗易懂 (Layman)	信息性 (Informative)	溯因 (Abductive)
	直觉 (Intuitive)	故事讲述 (Story Telling)	友好 (Friendly)	类比 (Analogical)
	反思 (Reflective)	苏格拉底式 (Socratic)	幽默 (Humorous)	因果 (Causal)
	全局 (Global)	综合 (Comprehensive)	专业 (Professional)	混合 (Hybrid)

## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来

### 知识点细化

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银  
户数据的保密性和系统的快速响应。

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银  
户数据的保密性和系统的快速响应。

### 什么是软件体系架构中实现质量属性的策略？

## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改  
结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来

### 知识点细化

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银  
户数据的保密性和系统的快速响应。

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银  
户数据的保密性和系统的快速响应。

## 产品功能3

# 「今天，你闭环了吗？」

### 自适应题目反馈

- 如有练习需求，产品支持自适应  
题目反馈功能。
- 通过分析用户的学  
习进度和测试  
结果，系统从网络题库中智能筛选出相匹配题目，实现即时反  
馈。

### 知识点回溯推荐

- 基于用户对知识点的掌握程度，  
系统可以上溯提供相关的前置知  
识点，帮助用户构建完整的知识  
体系。

# 差异化优势

“**垂**”

够聚焦，够专业

- 大模型总结框架 (chatgpt、kimi chat、claude)；
- 拆书大师/AI思维导图；
- AI阅读工具 (chatdoc、chatpdf、humata、pandagpt)

“**灵**”

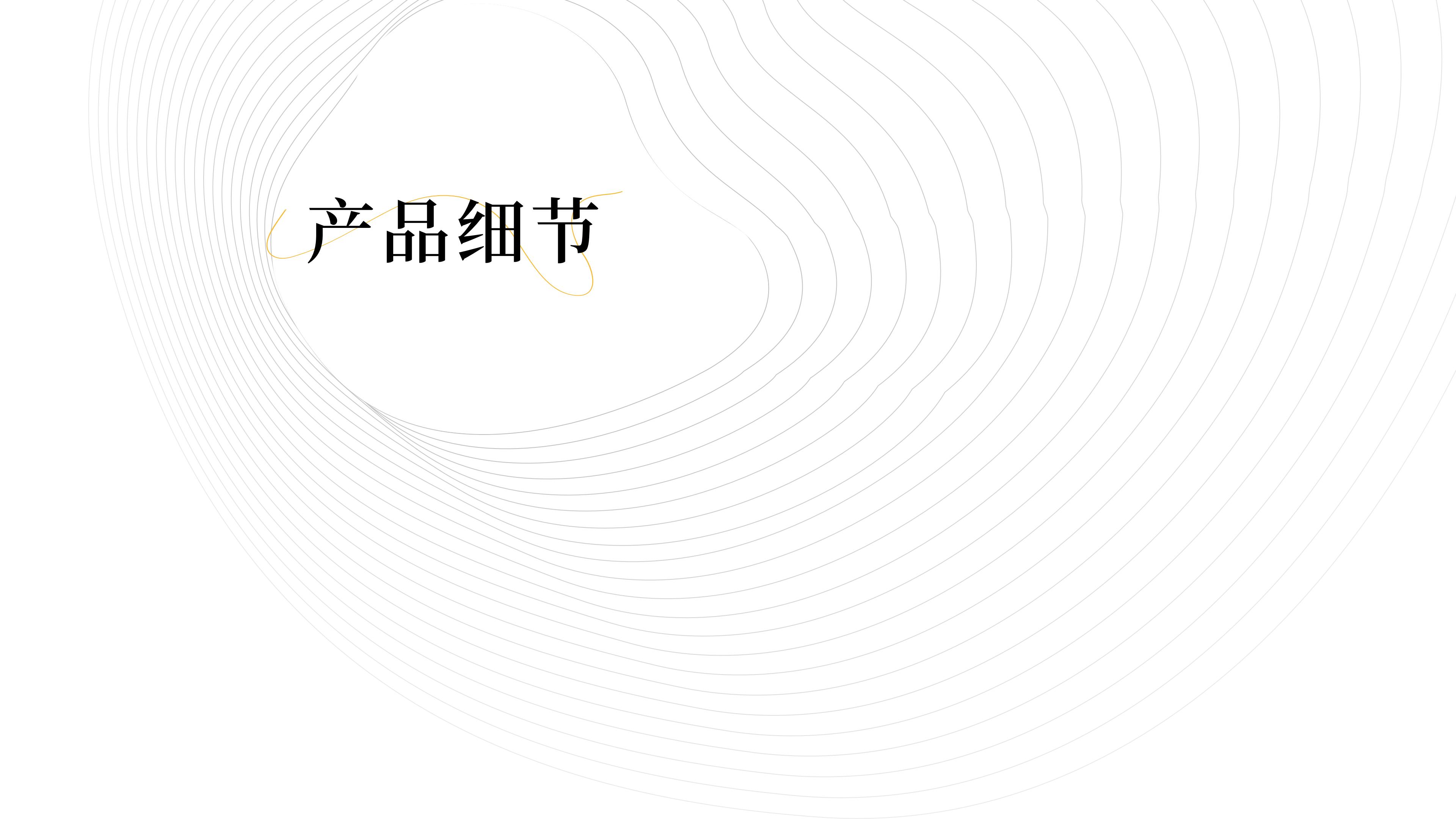
交互，可视化

- 知识内容管理体系 (Perplexity、Korra/Zendesk)；
- 可视化潜在竞品 (Flowwith/podwise)

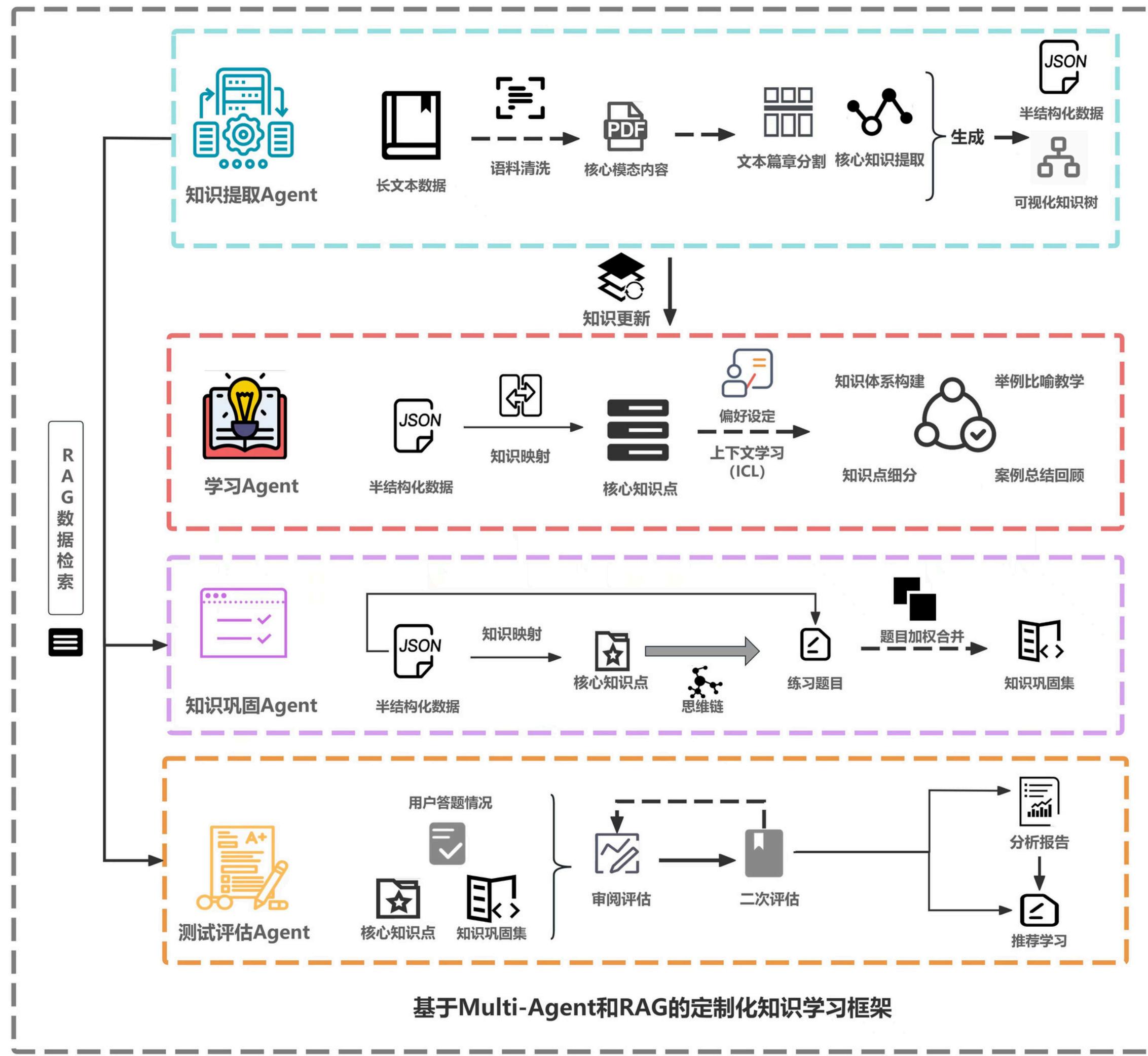
“**广**”

不只K12

- AI自适应试题教学 (Khanmigo、Duolingo、Adaptemy)；
- AI教育类知识图谱 (松鼠AI)；
- AI定制化内容生成 (MATHia、Cognii)



# 产品细节



分支

## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改性、可用性和可扩展性等。实现这些质量属性的策略被称为体系结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来帮助理解。

### 知识点细化

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

### 什么是软件体系架构中实现质量属性的策略？

## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改性、可用性和可扩展性等。实现这些质量属性的策略被称为体系结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来帮助理解。

### 知识点细化

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

#### 1.质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

输入提问内容...

Software Architecture in Practice  
Barry Shaw

Software Architecture in Practice  
Barry Shaw

Steve Jobs  
Walter Isaacson

1984  
George Orwell

thus, the set of architectural structures is not fixed or limited. What is architectural is what is useful in your context for your system.

### Architecture Is an Abstraction

Because architecture consists of structures and structures consist of elements and relations, it follows that an architecture comprises software elements and how the elements relate to each other. This means that architecture specifically omits certain information about elements that is not useful for reasoning about the system in particular, it omits information that has no ramifications outside of a single element. Thus, an architecture is foremost an abstraction of a system that selects certain details and suppresses others. In all modern systems, elements interact with each other by means of interfaces that partition details about an element into public and private parts. Architecture is concerned with the public side of this division; private details of elements details having to do solely with internal implementation are not architectural. Beyond just interfaces, though, the architectural abstraction lets us look at the system in terms of its elements, how they are arranged, how they interact, how they are composed, what their properties are that support our system reasoning, and so forth. This abstraction is essential to taming the complexity of a system we simply cannot, and do not want to, deal with all of the complexity all of the time.

1. In this book we use the term "element" when we mean either a module or a component, and don't want to distinguish.

### Every Software System Has a Software Architecture

Every system can be shown to comprise elements and relations among them to support some type of reasoning. In the most trivial case, a system is itself a single element an uninteresting and probably non-useful architecture, but an architecture nevertheless.

Enterprise architecture is a description of the structure and behavior of an organization's processes, information flow, personnel, and organizational subunits, aligned with the organization's core goals and strategic direction. An enterprise architecture need not include information systems clearly organizations had architectures that fit the preceding definition prior to the advent of computers but these days, enterprise architectures for all but the smallest businesses are unthinkable without information system support. Thus, a modern enterprise architecture is concerned with how an enterprise's software systems support the business processes and goals of the enterprise. Typically included in this set of concerns is a process for deciding which systems with which functionality should be supported by an enterprise.

An enterprise architecture will specify the data model that various systems use to interact, for example. It will specify rules for how the enterprise's systems interact with external systems.

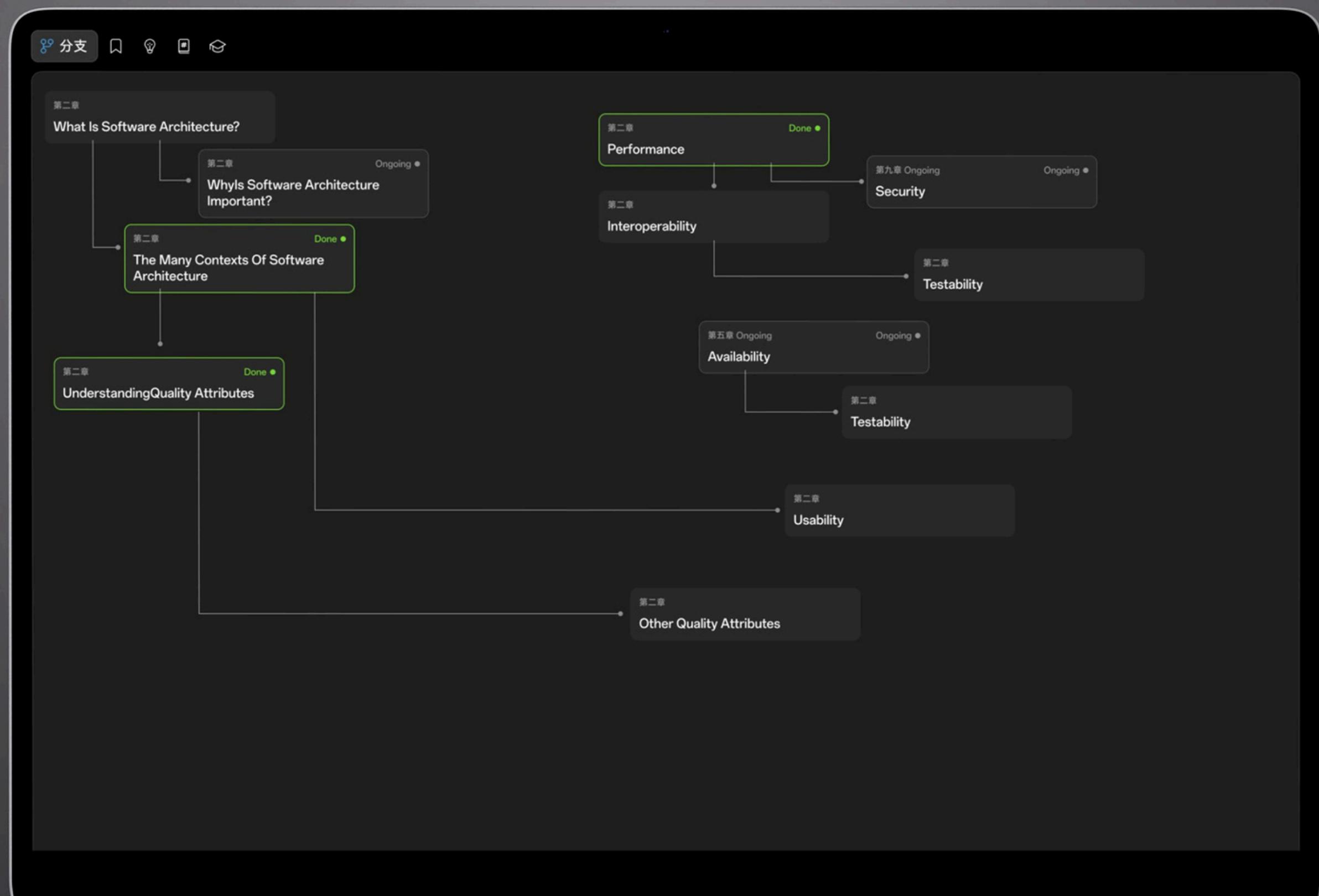
Software is only one concern of enterprise architecture. Two other common concerns addressed by enterprise architecture are how the software is used by humans to perform business processes, and the standards that determine the computational environment. Sometimes the software infrastructure that supports communication among systems and with the external world is considered a

输入提问内容...

Software Architecture in Practice

目录

- 1. What Is Software Architecture?
  - 1.1 巴黎象的家
- 3. The Many Contexts Of Software Architecture
- 4. Understanding Quality Attributes
  - 前置要求
  - 4.1 Architecture and Requirements
  - 4.2 Functionality
  - 4.3 Quality Attributes Consider ion s
  - 4.4 Specifying Quality Attributes
  - 4.5 Achieving Quality Attributes
  - 4.6 Guiding Quality Design Decisions
  - 4.7 Summary
  - 4.8 For Further Reading
  - 4.9 Discussion Questions
- 5. Availability
- 6. Interoperability
- 7. Modifiability
- 8. Performance
- 9. Security



## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改性、可用性和可扩展性等。实现这些质量属性的策略被称为体系结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来帮助理解。

### 知识点细化

#### 1. 质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

#### 1. 质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

### 什么是软件体系架构中实现质量属性的策略？

## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改性、可用性和可扩展性等。实现这些质量属性的策略被称为体系结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来帮助理解。

### 知识点细化

#### 1. 质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

#### 1. 质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

输入提问内容...

## Software Architecture in Practice

### 目录

#### 1. What Is Software Architecture?

#### Chapter Title

#### 3. The Many Contexts Of Software Architecture

#### 4. Understanding Quality Attributes

##### 前置要求

① ② ③

##### 4.1 Architecture and Requirements

##### 4.2 Functionality

##### 4.3 Quality Attributes Considerations

##### 4.4 Specifying Quality Attributes

##### 4.5 Achieving Quality Attributes

##### Section Title

#### 5. Availability

#### 6. Interoperability

#### 7. Modifiability

#### 8. Performance

自生成

## 知识点细化

性能一般场景  
并发  
事件到达模式  
响应度量

### 知识点细化讲解

性能一般场景

详细解释：性能场景涉及事件进入系统，需要消耗资源（包括时间）。系统可能同时处理其他事件，引入并发性。

例子：考虑一个处理多个请求的网络服务器。每个请求都需要处理时间，服务器可能同时处理多个请求，这展示了一个具有并发处理的性能场景。

并发

详细解释：并发涉及并行操作。例如，如果两个线程执行 `x := 1; x++;`, `x` 的最终值可能是2或3，取决于执行顺序。并发发生在多个CPU、多线程或并行算法中。

例子：在一个多线程应用程序中，一个线程更新计数器，而另一个线程读取它。如果没有适当的同步，最终计数可能由于竞争条件而不正确。

### 什么是软件体系架构中实现质量属性的策略？

## 软件体系架构中实现质量属性的策略

软件体系架构的质量属性对于满足系统需求至关重要，包括但不限于性能、安全性、可修改性、可用性和可扩展性等。实现这些质量属性的策略被称为体系结构策略。接下来，我将通过一个详细的框架来解析资料中提到的各个点，并且提供实例来帮助理解。

### 知识点细化

1. 质量属性需求与业务目标的关联

质量属性需求明确了系统在特定条件下的期望反应，它们与业务目标直接相关。例如，在银行系统中，安全性和性能是关键的质量属性，因为它们保证了客户数据的保密性和系统的快速响应。

1. 质量属性需求与业务目标的关联

输入提问内容...

Software Architecture in Practice

目录

- 1. What Is Software Architecture?
- 1.2
- 3. The Many Contexts Of Software Architecture
- 4. Understanding Quality Attributes
  - 前置要求 1 2 3
  - 4.1 Architecture and Requirements
  - 4.2 Functionality
  - 4.3 Quality Attributes Consider ion s 1
  - 4.4 Specifying Quality Attributes
  - 4.5 Achieving Quality Attributes
- Section Title
- 5. Availability
- 6. Interoperability
- 7. Modifiability
- 8. Performance

## 第五题

在软件体系结构设计中，质量属性（Quality Attributes）如性能（Performance）、可扩展性（Scalability）、安全性（Security）、可维护性（Maintainability）等对于系统的成功至关重要。请针对以下场景，描述如何在架构设计中应用具体策略以实现所需的质量属性。请包括至少两种不同的质量属性，并说明这些策略如何在实际应用中权衡。

输入提问内容...

## Software Architecture in Practice

### 目录

1. What Is Software Architecture?
2. Why Is Software Architecture Important?
3. The Many Contexts Of Software Architecture
4. Understanding Quality Attributes
  - 前置要求
  - 4.1 Architecture and Requirements
  - 4.2 Functionality
  - 4.3 Quality Attributes Consider ion s
  - 4.4 Specifying Quality Attributes
  - 4.5 Achieving Quality Attributes
  - 4.6 Guiding Quality Design Decisions
  - 4.7 Summary
  - 4.8 For Further Reading
  - 4.9 Discussion Questions
5. Availability
6. Interoperability
7. Modifiability
8. Performance

## 第五题

以下是一个选择问题：

"心理学中术语的操作性定义的重要性在于："

- A. 使心理学家能够用日常语言交流
- B. 让心理学概念能够被普通人轻松理解
- C. 保证概念在科学理论中的准确用法和可测量性
- D. 增加心理学论文的阅读难度'

## 解析：

在心理学中，操作性的定义是指对研究术语进行的具体的、可观测的操作定义，这确保了概念在实验和观察中的可测量性。这种定义方式的重要性在于它能够保证概念在科学理论中的准确用法和可测量性，使得不同的研究者可以在相同的基础上进行交流与合作。选项A和B虽然提到了交流和理解的方面，但并不是操作性定义的主要重要性。选项D与操作性定义的目的无关，反而是一种负面效果，因此不是正确答案。选项C准确描述了操作性定义的核心重要性。

输入提问内容...

发送

# 更多关注

群聊: 书掂天使用户群



书掂  
**BOOKDONE**

AI学练阅读器

2024.9.8