Stack Overflow 中架构相关帖子的特征及其有用性

Musengamana Jean de Dieu[a]、Peng Liang[a,*]、Mojtaba Shahin[b]、Arif Ali Khan[c]

[a]武汉大学计算机学院，中国武汉 *430072*[b]皇家墨尔本理工大学计算技术学院，澳大利亚墨尔本 *3000*[c]奥卢大学 *M3S* 实证软件工程研究组，芬兰奥卢 *90014*

arXiv:2301.00943v2 [cs.SE] 5 Jan 2023

摘要

背景：Stack Overflow（SO）吸引了软件工程师（例如架构师）前来学习、实践和运用开发知识，例如架构知识（AK）。然而，人们对 SO 中交流的 AK 知之甚少，而 AK 是开发过程中一种层次较高但十分重要的知识。

目标：本研究旨在从 SO 用户的角度，调查 SO 帖子中的 AK，包括其类别和特征，以及这些帖子的有用性。

方法：我们通过定性分析来自 SO 的 968 条架构相关帖子（ARP）这一具有统计代表性的样本，开展了一项探索性研究。

结果：主要发现如下：（1）架构相关问题可分为 9 个核心类别，其中“架构配置”最为常见，其次是“架构决策”；（2）提供清晰描述并附有架构图的架构相关问题，更有可能获得多个回答，而结构不佳的架构问题往往只能获得一个回答。

结论：我们的研究结果表明，未来的研究可以着眼于开发自动化方法和工具，以促进对 SO 中架构知识的搜索和（再）利用。Stack Overflow 用户可以参考我们提出的指南，撰写更有可能在 SO 上获得更多回复的架构相关问题。

*Key*词语：架构知识，架构层级元素，架构解决方案，Stack OverOverflow，有用性

1. 引言

技术问答网站（例如 Stack Overflow（SO））彻底改变了用户在互联网上获取知识的方式 [1]。SO 已成为软件开发领域最重要的社区问答网站，促进知识共享与学习。SO 借助用户（例如开发人员）的知识和技能，通过提出各种与开发相关的技术问题并提供回答，分享他们的想法和经验。此外，Stack Overflow 用户还可以从 SO 学习新颖的技术和工具 [2]。SO 主要用于解决编码问题 [3], ，而这些问题往往与架构师无关，或对他们吸引力较低，因为它们侧重于较低层次的实现细节 [3]。然而，自该网站开始发展并逐渐普及以来，架构师也开始通过提出架构相关问题或提供架构解决方案（例如架构策略），分享他们的能力、经验和设计问题。在我们最近开展的一项关于开发人员如何搜索架构信息的工业调查中 [4], ，从业者指出，问答网站（例如 SO）是最有用的架构信息来源（例如，了解架构解决方案的优缺点）。因此，与搜索并（重新）使用 SO 上已有的、用于解决编程相关问题的编码相关回答类似，软件工程师（例如架构师和开发人员）也会搜索并（重新）使用 SO 上已有的架构解决方案，以解决他们的设计关注点。因此，SO 不仅

*通讯作者地址：+86 27 68776137中国武汉武汉大学计算机学院。电话：+86 27 68776137；传真：+86 2768776027。电子邮箱：mjados@outlook.com (Musengamana Jean de Dieu), liangp@whu.edu.cn (Peng Liang), mojtaba.shahin@rmit.edu.au (Mojtaba Shahin), arif.khan@oulu.fi (Arif Ali Khan)

积累了代码示例，还整理了大量针对各类架构相关问题或设计问题提供的架构解决方案 [5] [6]。

尽管 Stack Overflow 用户会在 SO 中讨论高层次知识，例如架构策略和质量属性知识 [5], 以及用于技术决策的架构知识 [6], ，但截至目前，现有研究大多集中于从不同角度分析 SO 帖子中的编程相关知识。例如，Diamantopoulos 和 Symeonidis [7] 利用源代码信息改进 SO 中的问答，Zhang 等人 [8] 研究了 SO 编程相关帖子中代码示例的质量。很少有研究关注分析 SO 中架构相关帖子（ARP）所提供的架构知识。例如，Bi 等人 [5] 从 SO 中挖掘帖子，并构建了实践中使用的架构策略与质量属性之间的设计关系。Liu 等人 [9] 提取了 SO 帖子，并挖掘了设计模式的使用场景及相关设计模式对。Soliman 等人 [10] 开发了一种搜索方法（即领域专用搜索方法），用于搜索 SO 中的架构知识。在另一项研究中，Soliman等人 [11] 对 50 名软件工程师开展了实证研究。这些工程师使用 Google，并采用属性驱动设计 [12], 进行设计决策；研究人员据此确定了网络搜索引擎从各种来源（包括 SO）查找相关架构信息并获取架构知识的有效程度。Malavolta 等人 [13] 从五个开源软件仓库（包括 SO）中提取数据，并挖掘了从业者在实际机器人项目中应用的节能架构策略。Tian 等人 [14] 利用 SO 帖子研究了 SO 用户对架构异味的理解。上述研究从 SO 中提取了架构相关帖子，并从不同方面研究了架构知识（高层概念）。然而，以往工作仅基于架构相关问题及其相关回答。相比之下，我们的工作涵盖完整的架构相关帖子，包括其中的问题、问题下的所有评论、与问题相关的所有回答，以及回答下的所有评论。此外，尚无研究专门考察架构相关帖子（回答）中所提供的架构知识及其有用性。此外，也没有全面研究从类型、设计情境、特征和有用性等方面探究 SO 用户所传达的架构知识，而这正是本研究的重点。分析并理解 SO 用户如何在在线开发者社区（如 SO）中处理架构设计问题，可以带来三项优点：（1）深入了解 SO 用户在架构设计过程中面临的设计问题类型、所讨论的架构解决方案类型及其有用性；（2）有助于了解提出架构问题时所处的设计情境；（3）有助于了解所讨论的架构问题和解决方案的特征。这些优点为开发新技术和工具提供了机会，从而帮助 SO 用户搜索和（重新）使用在线开发者社区中共享的架构知识。因此，本研究旨在通过从 SO 用户的角度分析 SO 中架构相关帖子的特征和类别及其有用性，补充以往研究。在本研究中，我们依据 [15] 中的定义，将有用性（问答网站中帖子（例如回答）的一项质量标准 [15]）视为“回答是否有助于解决问题？”

为了实现本**研究**的目标（见第3.1节），我们开展了一项探索性研究，调查 Stack Overflow 上架构相关帖子的各个方面（例如类别和特征）。具体而言，我们从 Stack Overflow 提取了 32,182 篇帖子。随后，我们手动筛除无关帖子，得到 10,423 篇候选 ARP。由于 10,423 篇候选 ARP 构成的数据集相当庞大，人工分析如此规模的数据集并获得准确、全面的结果并非易事，因此我们利用统计功效计算出这 10,423 篇 ARP 的代表性样本量 [16] 。在置信水平为 95%、误差幅度为 3% 的条件下，计算得出的最终代表性样本量为 968 篇 ARP。随后，我们从 10,423 篇 ARP 中随机抽取 968 篇，并对其进行分析，以回答一系列研究问题（见表1）。具体来说，我们运用扎根理论（GT）中的开放式编码和持续比较方法，对这 968 篇 ARP 进行了人工分析 [17] ，以回答这些研究问题。本研究的**主要结果与发现**如下：（1）Stack Overflow 用户提出的问题涵盖广泛的架构相关问题，从架构工具到架构配置，从架构实现到架构部署。（2）有用的架构解决方案被归为七个类别，形成一个分类体系（见图4），例如架构配置解决方案、架构实现解决方案、架构策略和架构模式。一项观察结果是，这些帖子的类别（问题和回答）几乎涵盖了系统生命周期中架构设计的所有活动，包括架构创建的初始阶段（即架构分析与综合 [18]），以及后续阶段（即架构实现、维护与演进 [19]）。因此，我们识别出的架构相关帖子类别能够支持架构生命周期中的上述架构设计活动，而 Stack Overflow 可被视为架构知识的来源之一 [6]。我们发现

那些提供清晰描述并附有架构图的架构相关问题，更有可能获得多个回答；而结构不佳的架构问题往往只能获得一个回答。（3）Stack Overflow 用户经常使用两个与有用性相关的词语（即有用和有帮助），明确表达他们对针对相应架构相关问题所提供的特定架构解决方案的有用性的看法。

本研究作出以下三项贡献：(1) 对 Stack Overflow 用户（例如开发人员）在 SO 上提出的架构相关问题进行分类和特征描述；(2) 列出已识别出的提出架构相关问题的设计情境；(3) 对 SO 上有用的架构解决方案进行分类（即提出一个分类体系）和特征描述。本研究的发现对各类利益相关者都有帮助。例如，研究人员可以将我们提出的 SO 有用架构解决方案分类体系作为指导，以开发新的自动化方法和工具，从而挖掘并定位架构解决方案（例如，架构配置解决方案，见图4），以解决类似的设计关注点（例如，询问架构配置的问题，见表5）。这有助于 Stack Overflow 用户查看与其设计关注点相关的问题和解决方案。SO 可以利用我们的研究结果改进其答案和评论组织机制，并促进对 SO 上有用架构解决方案的搜索和（重复）使用。

本文其余部分安排如下：第2 节介绍本研究的背景。第3 节阐述研究方法，第4节详细介绍研究结果。第5节分析研究结果并讨论其影响。第6 节介绍研究结果有效性的威胁，第7节总结相关工作。最后，第8节总结本文，并展望未来研究方向。

2. 背景

本节介绍本研究中使用的背景概念，包括 Stack Overflow、架构知识、架构问题、设计上下文和架构解决方案。

2.1. Stack Overflow

Stack Overflow 是 Stack Exchange[1] 网络中的一个网站，为用户提供问答平台，以便他们分享不同领域（如编程、设计、统计学和数学）的知识。Stack Overflow 用户通过提出问题或回答已有问题，交流与软件开发相关的知识。除其他开发知识外，Stack Overflow 上也分享了架构知识，例如特定应用领域中架构解决方案（如模式和策略）的缺点与优点，以支持架构设计活动 [20][21]。近年来，架构研究社区一直关注在问答网站（尤其是 Stack Overflow）上挖掘架构知识，例如用于支持架构设计过程的技术决策相关架构知识 [6] ，以及架构策略和质量属性 [5],。

2.2. 架构知识

软件架构（SA）是一组结构，由软件元素、元素之间的关系，以及元素和关系的属性组成 [22]。构建软件系统的架构通常需要知识，尤其是架构知识 [23], 和技能。架构知识包括架构决策及其决策依据 [24], ，以及架构解决方案的优缺点 [6], ，是软件开发中最重要的知识类型之一 [22]。架构知识通常以各种格式进行描述，例如文本和图形表示 [25]；这些知识也记录在各种来源中，例如书籍 [22], 、技术博客和教程 [11], 、开发者邮件列表（例如 ArgoUML [26]）以及问答网站（例如 SO [5]）。在本研究中，我们从多个方面调查了 SO 中讨论的架构知识，例如 SO 中架构相关帖子的类别和特征，以及 Stack Overflow 用户对 SO 中提供的架构解决方案有用性的讨论。

2.3. 架构问题

架构问题（例如“是否有适用于 *MFC* 应用的可测试架构或设计模式？”[2]）会在开发过程中出现，通常是为了应对特定的架构设计问题（例如质量

[1] https://stackexchange.com/sites
[2] https://tinyurl.com/2z69uzs5

属性）及其权衡 [22]。SO 上有人提出了各种与架构设计相关的问题。在本研究中，我们考察了架构问题/问题的类别，具体而言，是 SO 上提出的架构相关问题的类别（见第4.1节）。

2.4. 设计上下文

软件系统的设计上下文包含架构师需要了解的系统预期运行环境（例如，硬件平台）的相关知识 [27]。设计上下文可视为“会影响设计决策、但未被明确列为需求的条件”[28]。Harper 和 Zheng 指出，设计上下文是影响利益相关者关注事项的力量[29]。一些研究对设计上下文进行了分类。Bedjeti 等人确定了架构视点的四种上下文类别（即平台情境、用户上下文、应用情境和组织情境） [27]。Petersen 和 Wohlin 提供了一份清单，从六个视角记录设计上下文：产品、流程、实践与技术、人员、组织和市场 [30]。Groher 和 Weinreich 研究了影响架构决策的环境因素 [31], ，并确定了八个类别：公司规模、业务因素、组织因素、技术因素、文化因素、个体因素、项目因素和决策范围。在本研究中，我们考察了 Stack Overflow 上架构相关帖子中讨论的设计上下文，并参考了两项现有研究提出的设计上下文分类 [27][30]（见第4.2节）。

2.5. 架构解决方案

架构解决方案是现代软件设计中的基本构建块，用于解决架构设计问题 [22]。架构解决方案种类繁多，例如模式、策略和框架，可用于解决不同的设计关注点。 架构模式（例如，模型-视图-控制器、客户端-服务器、发布-订阅模式）是在特定情境下针对架构设计中常见问题的可复用解决方案 [22]。与更改实现（例如低级代码）不同，一旦采用并实现某种架构解决方案（例如架构模式），要对其进行更改就会相当困难且代价高昂 [22]。架构模式决定软件系统的整体结构和行为 [32] ，通常会在开发早期选定，以满足多项系统需求（例如质量属性） [22]。在本研究中，我们研究了 Stack Overflow 用户关于架构解决方案（例如模式、策略和框架）有用性的讨论（见第4.5节），以及他们认为在 Stack Overflow 上有用的架构解决方案的类别和特征（见第4.6节）。

3. 研究设计

我们针对 Stack Overflow 中架构相关帖子的各个方面（例如类别和特征）开展了一项探索性研究。在以下小节中，我们将介绍本研究研究设计的详情，包括第3.1 节中的目标和研究问题，以及第3.2节中的研究实施过程。

3.1. 目标和研究问题

本研究基于目标-问题-度量方法所设定的总体**目标**为 [33] “从***Stack Overflow*** 用户的视角出发，在实际软件开发的背景下，分析 *Stack Overflow* 中的架构相关帖子（问题和回答），以研究其类别、特征和有用性”。根据本研究的目标，我们提出了六个研究问题（见表1），旨在考察能够体现 Stack Overflow 帖子问答讨论串的四个方面，包括：（1）架构相关问题的分类；（2）提出架构相关问题时所处的设计情境；（3）对有多个回答的架构相关问题进行特征描述，以及对只有一个回答的架构相关问题进行特征描述；（4）被认为有用的架构解决方案的分类和特征描述。

3.2. 研究执行

在本 小节中，我们介绍架构相关帖子的数据收集和分析流程。图 s
概述了 两个流程（即数据收集和分析）中的.....。

表1：研究问题及其决策依据

研究问题	决策依据
RQ1. 哪些架构相关问题会在 Stack Overflow 上被提出?	架构相关问题（设计问题）主要是向 ad-在架构设计活动中关注某些设计问题（例如系统的质量属性），例如架构分析。SO 整理了不同类型的架构相关问题，这些问题涉及各种设计问题。回答这一 RQ 有助于研究人员了解 Stack Overflow 用户在架构设计方面感兴趣的领域，并帮助从业者深入了解 SO 上提出的架构相关问题。它们能够提供切实可行的贡献。
RQ2. 设计情境是提出架构相关问题的背景?	设计情境包含有关系统预期运行环境的知识系统预期运行的环境 [27]。设计情境不可或缺能够驱动系统架构设计的要素 [27]。一个系统即使功能相似，在不同情境下也可能以不同方式运行 [30]。尽管人们已认识到在架构设计过程中考虑设计情境的重要性，但对于哪些设计情境会在架构设计中得到考虑，人们的了解仍然有限。回答这个研究问题可以帮助研究人员和从业者了解 Stack Overflow 上提出了哪些架构相关问题?
研究问题3. 其特征是什么的架构相关问题在 Stack Overflow 上有多个答案?	架构设计中的一项主要挑战，是选择恰当的架构解决方案来满足系统的需求 [34]。尽管不同的架构解决方案可作为解决类似架构问题的替代方案，但它们在质量方面各不相同 [34]。因此，为架构问题／相关问题提供多个答案（例如替代方案）十分重要，因为这能为制定架构设计决策提供丰富的可能性。通过这一研究问题（RQ），我们识别并考察获得多个答案的架构相关问题的特征。所谓架构相关问题的特征，是指问题内容或问题表述中的某些特征，例如架构图 [35], ，这些特征使该架构相关问题区别于其他问题，或吸引 Stack Overflow 用户关注并获得多个答案。该研究问题的答案有助于研究人员和从业者了解是什么促使 Stack Overflow 用户为这些问题提供更多解决方案，进而改善或减少 Stack Overflow 上未获回答的架构相关问题。
研究问题4. Stack Overflow 上只有一个回答的架构相关问题有哪些特征?	一些架构相关问题未能持续获得 Stack Overflow 用户的回答。与研究问题3类似，我们希望考察这一现象背后的因素。我们研究了那些只有一个回答的架构相关问题的特征。此研究问题的答案可以帮助研究人员和从业者了解哪些因素会降低 Stack Overflow 用户继续回答这些问题的积极性，并制定通用指南，帮助 Stack Overflow 用户撰写更有可能在 Stack Overflow 上获得更多回答的架构相关问题。
研究问题5. Stack Overflow 上提供了哪些被 Stack Overflow 用户认为有用的架构解决方案类型?	Stack Overflow 上提供了许多架构解决方案（例如策略和模式），用于解决架构相关问题。然而，研究人员和从业者一直十分关注 Stack Overflow 上提供的解决方案/回答的质量。正如相关工作中所阐述的（见第 7 节），这一点从日益增多的研究中可见一斑；这些研究从不同角度分析 Stack Overflow 帖子内容的质量，例如代码和文本。此研究问题的结果可以帮助研究人员和从业者了解在 Stack Overflow 上被认为有用的架构解决方案类型（分类体系）。
研究问题6. Stack Overflow 中的架构解决方案有哪些特征会被 Stack Overflow 用户认为有用?	Zhang 等人。 [8]指出，在 Stack Overflow 中，已被采纳、获得高票且经常被浏览的帖子并不总是可靠或有用的。识别被认为有用的架构解决方案的特征，有助于更好地了解 Stack Overflow 用户在判断架构解决方案是否有用时会考虑哪些因素，从而为改进当前的架构相关问题回答机制提供见解，并帮助 Stack Overflow 用户检索所需的架构解决方案。

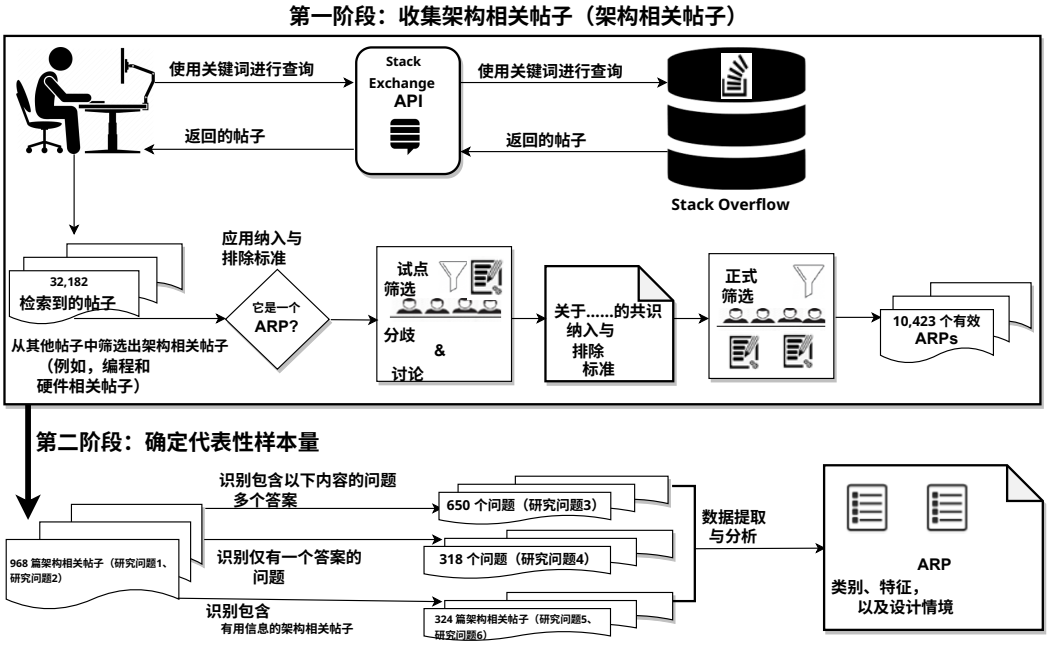


图1：数据收集与分析概览

3.2.1. 数据收集

我们的数据收集分为两个阶段，即第一阶段：收集架构相关帖子和第二阶段：确定代表性样本量，详见下文：

第一阶段：收集架构相关帖子

a) 搜索词：在确定最适合用于捕捉与架构设计相关帖子的词语之前，我们先在 Stack Overflow 上使用若干词语进行了试点搜索，即“architect*”（即“architect”“architecture”“architectural”和“architecting”）以及“design*”（即“design”和“designing”）。我们通过 StackExchange Data Explorer 提供的查询界面执行 SQL 查询来完成这一过程[3]，。该界面是一个 Web 界面，允许对来自问答网站（包括 Stack Overflow）的数据执行 SQL 查询。使用上述词语进行试点搜索后，我们发现，Stack Overflow 用户在 Stack Overflow 的编程语境中大多使用“design*”（即“design”和“designing”）一词，例如“singleton design pattern”[4]。此外，我们了解到 Soliman 等人 [6] 从其研究的 Stack Overflow 帖子样本中识别出了架构相关帖子与纯编程帖子之间的区别性词语。然而，在我们的研究中，我们没有使用这些区别性词语来搜索 Stack Overflow 上的架构相关帖子，原因有以下两点：

(1) 我们的研究与 Soliman等人在 [6] 中的研究目的不同。 [6] 中的研究旨在从 SO 提取技术相关的架构知识。具体而言， [6]中的作者识别并分析了主要讨论技术决策相关架构知识的架构相关帖子，例如某项技术方案在特定应用程序中的优缺点。此外， [6]中的作者称，他们在架构相关帖子的研究数据集中未发现许多纯粹的架构概念（如架构模式或策略）。相比之下，我们的研究涵盖的问题范围更广。具体而言，我们旨在通过考察各种架构信息（包括架构模式和策略），识别并分析 SO 上的架构相关帖子。因此，使用诸如对比、替代方案、优点、缺点、XMPP 等特定的区别性术语（这些术语在 [6] 中被发现），可能会遗漏其他相关的架构相关帖子，从而影响检索结果的完整性。

（2）使用 Soliman 等人工作中发现的独特术语 [6] 可能会导致搜索结果产生偏差。提取出的架构相关帖子的相关性和完整性可能会影响我们对六个研究问题的回答是否正确（见表1）。因此，在搜索查询中纳入 [6] 中发现的特定独特术语，可能会导致搜索结果偏向这些术语。

[3] https://data.stackexchange.com/stackoverflow/query/new
[4] https://tinyurl.com/8yks7nhm

因此，我们选择了通用术语“architect*”（即“架构师”“架构”“架构的”和“架构设计”）用于搜索。值得一提的是，我们没有仅使用搜索词在标签中进行搜索，因为标签有时信息量较少且效果不佳 [36]。仅将标签作为判断帖子是否与某个主题相关的依据存在若干缺点。这是因为创建帖子的用户可能不确定讨论最适合使用哪个标签，这可能导致使用错误或不相关的标签 [37]。例如，在这篇架构相关帖子中[5]，提问者询问了可用于设计单个 Web 表单应用程序的架构模式，一名开发人员使用了“jc#”“asp.net”和“web”这些标签，而这些标签无法直接说明它们实际用于哪些语境（例如架构语境或编程语境）。用户定义的标签还有另一个问题：用户可能会尽可能多地添加标签（SO 最多允许添加 5 个标签），以提高浏览量，并可能增加快速获得回答的概率 [38]。因此，虽然标签有助于捕捉与架构设计相关的帖子，但仅使用标签可能会漏掉有关该主题的重要帖子。因此，我们决定将问题的标题和正文也纳入搜索。例如，按照 Stack Exchange为“architect”一词提供的查询界面标准[6]，，我们使用以下查询在帖子标题、标签和正文中进行搜索： `SELECT p.Id, p.Tags, p.Title, p.Body as "Questions Body", p.Score as "Questions Score", p.Answercount as "Answer Count" FROM Posts p WHERE (p.Body like '%architect%' or p.Title like '%architect%' or p.Tags like '%architect%') AND p.Score >0 and p.AnswerCount <>0 ORDER BY p.Score DESC`。在我们的复现包 [39], 中，我们提供了用于在 SO 中搜索架构相关帖子的完整 SQL 查询，例如搜索过程中如何组合帖子的标题、标签和正文。搜索过程共得到 32,182 个帖子（见图1）。请注意，我们使用上述搜索词（即“架构师”“架构”“架构的”和“架构设计”），并非为了收集 SO 中的所有架构相关帖子，而是为了获取足够的数据，以便进行较为全面的分析，从而实现本研究的目标。

b）从其他帖子（即编程和硬件相关帖子）中筛选出架构相关帖子：我们发现，Stack Overflow 用户使用“架构”一词时，不仅是在软件架构的语境中，也会出现在其他语境中，例如硬件架构语境（如 ARM6 CPU 架构[7]）和编程语境（如数组架构[8]），用以描述他们在 Stack Overflow 帖子中关注的问题。因此，我们需要对检索到的 32,182 个帖子进行筛选，并排除与编程和硬件架构相关的帖子。为此，我们进行了上下文分析，并应用了所定义的纳入和排除标准（见表2），以准确筛选并区分软件架构相关帖子与上述其他类型的帖子。

在正式帖子筛选（人工检查）之前，为了就纳入和排除标准（见表 2）达成一致，我们进行了试筛选：第一作者从 32,182 个帖子中随机抽取 1,000 个帖子，并依据所定义的标准（见表 2）逐一进行人工检查。其余三位作者对结果进行了检查和审阅，以便本研究的全部四位作者对所定义的纳入和排除标准达成共识。之后，我们发现筛选结果中有 51 个帖子存在争议和理解分歧。本研究的四位作者就这些争议和分歧进行了讨论，直至达成共识。随后，第一作者依据纳入和排除标准开展正式帖子筛选，并持续进行，直至人工检查完全部 32,182 个帖子。此步骤共得到 10,423 个候选 ARP（见图 1）。本轮结果由本研究的其余三位作者检查和核实。我们历时整整二十一天，才从这 32,182 个帖子中识别并区分出架构相关帖子与编程和硬件架构相关帖子。

第二阶段：确定代表性样本量

10,423 条候选 ARP（从上一阶段（即第一阶段）筛选得出）构成了一个相当大的数据集，不现实通过人工分析如此规模的数据集并获得准确、全面的结果。因此，为了获得统计显著结果，我们使用统计功效计算了这 10,423 条架构相关帖子的代表性样本量 [16] 。在 95% 的置信水平下，我们将整个 10,423 条架构相关帖子的误差幅度（即我们预期分析结果能够反映整个数据集观点的程度）设为 3%。最终计算出的代表性样本量为 968

[5] https://tinyurl.com/mb9y37z4
[6] https://data.stackexchange.com/stackoverflow/query/new
[7] https://tinyurl.com/f8sjvzz2
[8] https://tinyurl.com/3ps7b3ek

表 2：从编程和硬件相关帖子中筛选架构相关帖子的纳入和排除标准

纳入标准
I1. 架构相关帖子应包含关于软件架构的讨论，例如架构设计和架构策略。
I2. 架构相关帖子应至少有一个针对其架构相关问题的回答，因为我们旨在研究导致这些问题有多个回答或只有一个回答的因素，以及其回答的有用性。
I3. 架构相关帖子应至少包含一个可根据数据项提取的数据项定义于表4。
排除标准
E1. 评分（即赞成票与反对票的中位数）低于 1 的架构相关帖子将被排除，因为我们希望确保所有研究的帖子都获得了社区的足够关注 [40]。

架构相关帖子。随后，我们从 10,423 条架构相关帖子中随机选取 968 条，并对其进行分析，以回答六个研究问题（见表 1）。具体而言，除研究问题1和研究问题2使用 968 条（架构相关帖子的代表性样本量）来回答外，对于其余研究问题（即研究问题3、研究问题4、研究问题5和研究问题6），我们使用这 968 条帖子中符合既定标准（见下文步骤）的子集，以满足相应研究问题的研究目的（见表 1）。我们通过以下步骤，将计算得出的架构相关帖子代表性样本量（968）进一步划分为与回答其余研究问题相关的子集：

步骤 ***1***：识别有多个回答的问题（研究问题*3*）和只有一个回答的问题（研究问题*4*）。如这两个研究问题的决策依据中所述（见表 1），我们希望考察哪些因素会使这类架构问题获得多个回答或仅获得一个回答。因此，我们参考了这两项研究，考虑了针对架构相关问题发布的评论[41][42], ；这两项研究中，作者认为回答的质量由回答及其相关评论共同构成，因为评论可能会提供关于回答的补充信息，例如对回答的改进 [42] 以及指出回答已过时 [41]。因此，在本研究中，我们纳入了针对架构相关问题发布的评论，并研究这些帖子为何会获得多个回答（研究问题3）或仅获得一个回答（研究问题4）。具体而言，第一作者人工检查了 968 条架构相关帖子及其评论，并筛选出 650 个架构相关问题（每个问题都有多个回答）。这些问题用于回答研究问题3（见图 1）。另一方面，第一作者采用相同的流程（针对研究问题3，对 968 条架构相关帖子进行人工检查），筛选出 318 个架构相关问题（每个问题只有一个回答）。这些问题用于回答研究问题4（见图 1）。

步骤 ***2***：识别包含有用知识的架构相关帖子。为回答研究问题5和研究问题6，我们需要从架构相关帖子的代表性随机样本（968 篇架构相关帖子）中识别包含有用知识的架构相关帖子。我们参考了以下两项研究（即 [41, 42]），以识别具有有用知识的架构相关帖子。Zhang 的这些研究 等 [41, 42] 认为，回答的质量由回答及其相关评论共同决定，因为评论可能会提供补充信息来支持回答，例如改进回答 [42] 以及指出过时的回答 [41]。因此，在本研究中，我们纳入评论中的信息，以深入了解 Stack Overflow 用户如何讨论 Stack Overflow 上针对架构相关问题所提供的架构解决方案的有用性。在本研究中，我们没有将投票分数作为回答研究问题5和研究问题6的依据，因为正如 Zhang 等人 [8]所指出的，即使投票数很高，Stack Overflow 帖子也未必可靠或有用。我们观察到，Stack Overflow 用户偶尔会在评论中使用与有用性相关的词语（如“有帮助”），以表明针对架构相关问题所提供的某些架构解决方案是有用的（见图 2）。因此，基于这一观察并参照表 3中规定的筛选标准，我们人工检查并筛选了这 968 篇架构相关帖子，以识别包含有用知识的解决方案讨论串（每个解决方案讨论串包括一个问题的所有解决方案（即已采纳和未采纳的解决方案），以及与这些解决方案相关的所有评论）。具体而言，为筛除未讨论架构解决方案有用性的架构相关帖子，并就表 3中规定的标准达成一致，两位作者（即第一作者和第二作者）进行了架构相关帖子的试筛。他们从这 968 篇架构相关帖子中随机抽取 20 篇，并分别进行人工检查。

架构相关帖子。近期研究也采用过类似流程（即从大型数据集中抽取随机数据样本，随后进行人工筛选），例如 [43]。为衡量前两位作者之间的评审者一致性，我们计算了科恩卡帕系数 [44]，得到的一致性为 0.898。请注意，在最终将某个解决方案纳入相关解决方案（即有用的解决方案）之前，第一作者和第二作者会先阅读被评论为有用／有帮助的解决方案，以确认它是否确实有助于解决问题。两位作者会讨论架构相关帖子中的分歧，直至达成共识。随后，第一作者继续检查并筛选其余的架构相关帖子。最终得到的、用于回答两个研究问题（研究问题5和研究问题6）的包含有用知识的架构相关帖子共 324 篇（见图 1）。

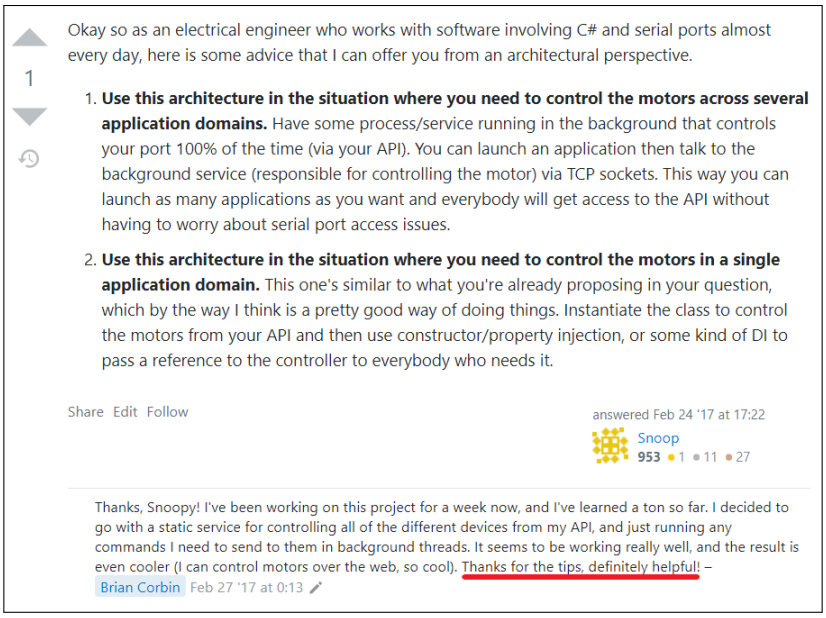


图 2：一个被评论称为有帮助的回答示例

表3：用于识别包含有用知识的架构相关帖子的纳入和排除标准

纳入标准
I1. 回答帖中的评论必须包含与有用性相关的关键词，例如“有用”“有帮助”“有益”“实用”和“有效”，且该评论用于表明该回答的有用性。
排除标准
E1. 与有用性相关的关键词，例如“有用”“有帮助”“有益”，被用于谈论其他内容（例如，问题或回答本身与“有用性”这一主题相关，而不是作为该回答可能有用的标志）。
E2. 如果架构相关帖子在回答中存在争议讨论（即同一回答帖中有两条评论，一条指出回答有用，另一条则指出回答无用），则不纳入分析。

3.2.2. 数据提取与分析

(1) 数据提取：我们通过从 968 篇架构相关帖子中识别需要提取的相关信息，开展数据提取流程，以回答我们定义的研究问题（见表 1）。表 4列出了从候选 ARP 中提取相关信息的数据项，并展示了预期使用提取的数据回答的研究问题。随后，我们对提取的数据进行了数据分析，并借助 MAXQDA（一种定性数据分析工具）开展和记录了这两个流程[9]。

[9] https://www.maxqda.com/

Tabl表 4：从架构相关帖子中提取的数据项及其描述、分析方法和相关研究问题 s

#	数据项	描述	数据分析方法	RQs
D1	内容 of the 问题	问题的主要内容 架构相关帖子中的问题	开放式编码与持续比较	RQ1
D2	设计上下文	设计上下文阐 根据……的内容评分 架构相关帖子中的问题	预定义分类在 [27] 以及 [30]	RQ2
D3	内容 of the 问题 and its 评论	问题的主要内容 问题以及对 架构相关帖子中该问题评论的总结 架构相关帖子	开放式编码与持续比 比较	研究问题3、研究问题4
D4	分析内容 swer	分析的主要内容 架构相关帖子中的回答	开放式编码与持续比较 比较	RQ5
D5	回答内容及其 回答及其相关性- 意见	回答的主要内容 回答内容摘要 回答中的评论 ARP	开放式编码与持续比- 较	RQ6

（2）数据分析：

与若干已有研究（例如 [4]）类似，我们在本研究中采用开放式编码和持续比较来回答研究问题1和研究问题3至研究问题6。开放式编码和持续比较是扎根理论中两种广泛使用的定性数据分析方法 [17]。扎根理论是一种自下而上的方法，侧重于理论生成，而非扩展或验证现有理论 [17]。开放式编码会为事件生成编码，之后可进一步将其归类为概念和类别 [17]。持续比较是一个持续验证所生成概念和类别的过程。概念和类别都会不断演变并趋于饱和，直至与数据相契合 [17]。因此，本研究采用扎根理论中的开放式编码和持续比较法，生成用于回答研究问题1和研究问题3至研究问题6的概念和类别。具体而言，我们对研究问题1和研究问题3至研究问题6所提取的数据项（见表 4）进行开放式编码，以生成编码。随后，我们运用持续比较法，将一份数据中识别出的编码与其他数据中出现的编码进行比较，从而找出语义含义相似的编码。接着，我们将相似的编码归纳为高层概念和类别。另一方面，我们采用 [27] 和 [30] 中预定义的设计情境分类来回答研究问题2。我们遵循相同的流程（对编码进行编码并将相似编码归纳为高层类别）来回答研究问题2。

正式数据分析之前，第一作者针对每个研究问题进行了试点数据分析。具体而言，该分析过程包含以下步骤：（1）第一作者从计算出的代表性样本量（即 968 条架构相关帖子）中随机选取了 100 条架构相关帖子。（2）第一作者对每个研究问题提取的数据进行编码（见表 4）。如果帖子内容不清楚，导致第一作者在对提取的数据进行编码时感到困惑，就会安排与第二作者进行线下会议，以解决这些疑惑。（3）第一作者运用持续比较法，将所有编码归纳为更高层次的概念，并将其整理为类别和子类别。归纳过程是迭代进行的，第一作者不断在概念、类别、子类别以及问题、回答和评论的内容之间来回比照，以修订和完善这些概念、类别和子类别。（4）随后，另外三位作者（第二、第三和第四作者）对试点数据分析的结果（即概念、类别和子类别）进行了检查和验证。研究团队采用协商一致法，在一次会议中解决了分歧 [45]，以提高试点数据分析结果的可靠性。第一作者随后继续进行正式数据分析，并遵循了试点数据分析期间所采用的类似步骤。下文将详细介绍正式数据分析过程：

a）用于分析研究问题*1*、研究问题*3*和研究问题*4*

如上所述，我们采用开放式编码和持续比较法 [17] 对提取的数据进行人工分析（即，研究问题1的问题内容，以及研究问题

3 和研究问题4的问题内容及其评论），如表4所示。围绕这些研究问题，我们从两个方面研究架构相关问题，即对这些问题进行分类（研究问题1）和特征描述（研究问题3和研究问题4）（见表1）。具体而言，在对问题进行分类时，第一作者研究了每个架构相关问题的内容（从与回答研究问题1相关的架构相关帖子中选取，见图1），以探究并识别其主要目的（例如设计关注点），如询问如何重构系统架构（例如重构循环依赖）。随后，第一作者用一个简短的句子概括每个问题的目的。首先，第一作者对概括出的句子进行编码。这一过程是迭代式的，他持续应用这一方法，直至数据集中的所有架构相关帖子都完成编码。其次，第一作者采用持续比较法，将某个概括句中识别出的编码与其他概括句中出现的编码进行比较，以检查哪些编码具有相似的语义含义。接着，第一作者将相似的编码归并为高层概念、类别和子类别。归并过程是迭代式的，第一作者不断在概念、类别、子类别与问题内容之间反复对照，以修订和完善这些概念、类别及其子类别。为减轻正式数据分析中的个人偏见，本研究的其他作者（第二、第三和第四作者）参与验证生成的编码、概念、类别和子类别。研究团队在一次会议上采用协商一致法解决了分歧 [45]，以提高研究问题1分析结果的可靠性，这与试点数据分析阶段的做法一致。最终，研究问题1得出了9个高层类别和21个子类别，相关结果在第4.1节中作了详细阐述。

如3.2.1节所述，为回答研究问题3和研究问题4，我们手动检查了968篇架构相关帖子，以识别有多个答案的架构相关帖子（研究问题3）和只有一个答案的架构相关帖子（研究问题4）。由此得到968篇架构相关帖子的两个子集（即650篇和318篇），分别用于回答研究问题3和研究问题4（见图1）。具体而言，在正式数据分析中，第一作者对问题及其附带的评论进行了分析（这些问题来自上述968篇架构相关帖子中的两个子集，即650篇和318篇），并对提取的数据进行编码，以分析这两个研究问题（即表4所示的问题内容和评论）。他希望探究是否存在某些因素，例如问题表述 [35]或问题中的某些特征（如架构图），会促使此类架构相关问题获得多个答案或仅获得一个答案。例如，在研究研究问题3（有多个答案的问题）时，一位社区成员在一个架构相关问题下发表评论：“*+1* 好问题，表述得非常清晰”[10]，。针对这条评论，他选取了一个短语（即对该评论的概括）：“表述清晰”。随后，他研究了这条评论所针对问题的内容（例如，问题中如何陈述架构信息），并据此提出了一个适用于该问题的编码。在分析这两个研究问题（研究问题3和研究问题4）时，他采用了分析研究问题1时所用的相同流程（例如，将相似编码归为高层概念和类别）。为减少个人偏见，本研究的另外三位作者对这项分析的结果进行了检查和验证。与分析研究问题1时一样，我们召开会议并采用协商一致法 [45]讨论并解决分歧，从而提高研究问题3和研究问题4分析结果的可靠性。最终分析得出了研究问题3和研究问题4的结果：有多个答案的架构相关问题具有四项特征，而只有一个答案的架构相关问题具有五项特征。研究问题3的四项特征和研究问题4的五项特征详见4.3 节和4.4节。

b）用于分析研究问题*2*

我们在 [27] 和 [30] 中采用预定义分类来回答研究问题2。具体而言，第一作者手动分析了用于回答研究问题2的提取数据（即设计情境，见表4），这些数据来自与回答研究问题2相关的架构相关帖子中的问题（见图1）。随后，第一作者检查提取的数据，以探究提出架构相关问题时所处的设计情境。参照上述研究提出的设计情境类别，我们从分析的架构相关问题中归纳出三个主要类别和八个子类别。通过与本研究的其他三位作者共同验证生成的类别和子类别，降低了个人偏见。与先前研究问题的分析一样，研究者采用协商一致法，在会议上讨论并解决了分歧 [45]，以提高研究问题2分析结果的可靠性。研究问题2的结果见第4.2节。

[10] https://tinyurl.com/dfw27h38

c) 用于分析研究问题5和研究问题6

这两个研究问题旨在探究架构解决方案在 Stack Overflow 中的有用性。如第3.2.1节所述，为了回答这两个研究问题，我们定义了一组标准（见表3），并筛选出包含有用知识的架构相关帖子。为清晰起见，我们从两个方面考察这两个研究问题：

我们研究了 Stack Overflow 用户如何讨论与其相关架构问题对应的架构解决方案的有用性。我们需要了解 Stack Overflow 用户可能会使用哪些方式（例如，术语）来表达架构解决方案在 Stack Overflow 上的有用性。此外，了解 Stack Overflow 用户对这些解决方案有用性的讨论，对于指导问答平台所有者创建相应机制至关重要，这些机制能够帮助用户高效、有效地搜索并（重新）使用此类有用的架构解决方案。为此，我们在所定义的标准（见表3）的辅助下，手动检查了 968 篇架构相关帖子中附于解决方案的评论。我们发现，Stack Overflow 用户偶尔会在评论中使用与有用性相关的词语，例如“有帮助”，并搭配其他词语（例如“非常”“超级”），明确表达他们认为某些架构解决方案有多大用处（例如，见图 2）。如第 3.2.1 节所述，用于回答这两个研究问题（研究问题5和研究问题6）的最终架构相关帖子（包含有用知识）共 324 篇（见图 1）。值得注意的是，在本研究中，我们并未依据评论中出现“有用”等词语（及类似词语）的频率来衡量针对架构相关问题所给出的架构解决方案的有用性。如上所述，我们参考了附于解决方案的评论中的信息，以考察 Stack Overflow 用户对架构解决方案有用性的讨论。这里指的是 Stack Overflow 用户看到并使用针对相关架构问题给出的架构解决方案后的反应，例如，见图 2中的一条评论。此外，在将此类架构相关帖子（解决方案）最终纳入分析之前，我们先阅读了被评论为有帮助的解决方案及其相关问题，以确认它们是否确实有用（即这些解决方案/回答是否有助于解决问题？ [15]）。第 4.5 节将进一步介绍我们从 324 篇架构相关帖子中识别出的有用性相关词语（例如“有帮助”）以及其他词语（例如“极其”“非常”）；Stack Overflow 用户会在评论中使用这些词语，明确表达他们认为针对相关问题提供的架构解决方案有多大用处。

在针对这两个研究问题（即被认为有用的架构解决方案分类体系（研究问题5）及其特征（研究问题6））进行数据分析时，第一作者采用了与分析研究问题1时相同的流程（例如，对相似编码进行编码并归入高层类别）。在分析研究问题5以构建分类体系时，需要说明的一点是，第一轮归类得出了七个主要类别。在第二轮中，本研究的所有作者基于这七个主要类别进一步提出子类别和类型，确保这些主要类别、子类别和类型遵循“是一个”关系。归类过程是迭代进行的，作者在类别、子类别、类型和解决方案之间不断往返，以完善分类体系。此次分析最终得出了一个包含7个主要类别、20个子类别（其中1个被归为“其他”，即指不属于已生成子类别的编码）和85种类型的分类体系。请注意，协商一致法 [45] 用于讨论并解决任何分歧。研究问题5的最终分类体系详见第4.5节。研究问题6的分析归纳出四项特征，详见第4.6节。

请注意，在阅读这些帖子并对 SO 中的架构相关帖子进行分类和描述时，我们发现单个架构相关帖子可能包含多种类型的架构知识。例如，在这个架构相关帖子中[11]，一位 SO 用户询问了模型-视图-控制器（MVC）模式的替代架构模式（即替代架构方案）。在问题正文中，该用户询问了促使人们决定采用这些替代架构模式而非 MVC 的原因（即架构决策及其决策依据）、这些替代架构模式通常适用于哪些类型的系统（即设计上下文），以及采用这些替代架构模式所带来的优缺点（即架构解决方案的优点和缺点）。因此，我们对这类包含多种架构知识的帖子进行了相应编码。此外，在分析数据集中的架构相关帖子时，我们记录并讨论了分析结果（例如，架构相关问题的类别和有用架构解决方案分类体系），并开展了定性数据分析。这带来了一些有趣的发现和可供各类利益相关者采取行动的启示，分别见第 4 节和第 5节。本研究收集和使用的数据集以及数据分析的详细信息（例如在 MAXQDA 中进行的编码）均已在线提供，以便复现和验证 [39]。

[11] https://tinyurl.com/2d8r6w8m

4. 结果

本节将呈现通过数据分析得出的研究问题结果（见第3.2.2节）。每个研究问题的结果均在单独的小节中介绍，并以相应结果的关键发现作为结尾。

4.1. 架构相关问题的类别（研究问题*1*）

Stack Overflow 用户在 Stack Overflow 上提出的架构相关问题类别，是通过第 3.2.2 节所述的开放式编码和持续比较法确定的。我们对 968 篇架构相关帖子中的问题进行了考察，以回答这一研究问题（见图 1）。数据分析得出了架构相关问题的 9 个主要类别和 21 个子类别。表5 展示了上述类别及其子类别、出现比例（以 968 个架构相关问题为基数）和数量信息。如表 5 所示，架构配置 （27%，968 个架构相关问题中占 261 个）、架构决策 （19%，968 个架构相关问题中占 181 个）和架构概念（15%，968 个架构相关问题中占 142 个）是最常被问及的三大类别。接下来，我们将介绍这些类别及其子类别。必要时，我们会提供一个 Stack Overflow 问题示例，以帮助理解这些类别及其子类别。

表 5：架构相关问题的类别、子类别及其数量和百分比 s

类别	子类别	数量
架构配置（27%，261）	技术支持下的架构配置	144
	架构模式配置	117
架构决策（19%，181）	技术决策	104
	行为决策	77
架构概念（15%，142）	架构概述	62
	基本架构概念	32
	架构组件功能	26
	特定架构模式	22
架构实现（12%，119）	架构组件实现	79
	架构模式实现	40
架构工具（10%，99）	架构建模工具	34
	基于模型的代码生成工具	30
	架构工具的使用	21
	基于代码的模型生成工具	14
架构演化（6%，55）	为满足新需求而进行的架构扩展	42
	为满足新需求而进行的组件扩展	13
架构重构（5%，45）	循环依赖的重构	21
	大型组件的重构	13
	大泥球的重构	11
架构部署（4%，34）	满足质量属性的应用部署	24
	满足功能需求的应用部署	10
架构文档（3%，32）		32

（1）架构配置此类别中的问题询问如何配置软件系统中的组件和连接器。这些组件和连接器的类型可能属于特定技术（例如 Windows Communication Foundation (WCF) 和 Windows Presentation Foundation (WPF)），也可能属于其他架构概念（例如架构模式）。这是 Stack Overflow (SO) 上架构相关问题中最常见的类别（占 27%，968 个架构相关问题中有 261 个）（见

表5）。我们进一步将此类别划分为两个子类别，其中技术支持下的架构配置占此类别中 Stack Overflow 用户提出的问题的一半以上（架构配置类别中的 *261* 个架构相关问题里有 *144* 个）（见表5）。

- 技术支持下的架构配置关注的是如何使用特定技术（例如 WPF、WCF）配置应用程序的架构。例如，在这个问题中[12]，一位开发人员询问了如何使用 WCF 配置或构建可扩展的基于 Web 的应用程序："有人了解使用微软的 *WCF* 构建的 *Web* 服务如何扩展以支持大量用户吗？我考虑的规模是大约有 *1000+* 个客户端用户连接到一组 *WCF* 服务，这些服务为我们的应用程序提供业务逻辑（.....）"。

- 架构模式配置旨在就设计应用程序以满足特定需求（例如功能需求）时，如何配置特定架构模式（例如模型-视图-控制器和六边形架构模式）提供实践指导。例如，在这个问题中[13]，一位开发人员询问了如何配置符合六边形架构模式的应用程序，并表示："我想了解一些指导意见或最佳实践，知道如何配置和构建一个符合六边形架构、且支持同时使用多个（驱动）适配器的应用程序（.....）"。

(2) 架构决策：Stack Overflow 用户提出这类问题时，通常是希望在设计软件系统时，在两个或多个替代架构方案之间做出选择。在这一类别中识别出的两个子类别里，技术决策子类别包含了 Stack Overflow 用户在此类别中提出的大多数问题（181 个架构相关问题中有 104 个，见架构决策类别的表5）。

- 技术决策主要涉及在两个或多个技术解决方案（例如框架、数据库）之间做出选择，以满足架构层面的特定需求。此外，做出选择时还会考虑技术特性、优点和缺点等多个方面 [34]。例如，在这个问题[14]中，一位开发人员询问了促使其决定在应用程序中使用 Cassandra 而非 HBase 的原因，并指出："*HBase* 以键值存储而闻名，可基于键使用 *.get* 和 *.put* 函数进行随机读取。*Cassandra* 是否更适合满足键值存储的需求？它是否支持基于键的随机读取？如果支持，在 *Spark Streaming* 应用程序中，什么情况下应该选择 *Cassandra* 而非 *HBase*？"。

- 行为决策关注的是如何决定系统中的某些元素如何相互作用，以提供某种功能或满足特定质量属性 [46]。例如，一位开发人员想要决定是让客户端直接连接数据库，还是让连接经由 Web 服务，因此提出了这个问题[15]："最近，我一直在开发一个用于运行高安全性数据库的系统（使用 *VB.NET* 和 *SQL Server 2005*）。我想提高数据库的安全性，因此不允许直接连接数据库，而是发送 *HttpWebRequest* 到 *Web* 服务，再由该服务连接数据库并以 *XML* 格式返回请求的数据表。我唯一担心的是性能；我无法决定是让客户端直接连接数据库，还是让连接经由 *Web* 服务。"

（3）架构概念涵盖与软件架构相关的理论性问题。我们将这一类别划分为四个子类别，其中架构概述包含了 Stack Overflow 用户在此类别中提出的大多数问题（在架构概念类别的 142 个架构相关问题中占 62 个）。

- 架构概述 类问题关注的是有关特定现有架构总体工作机制或概述的信息。例如，在这个问题[16]中，一位开发人员询问 Drupal 7 的架构概述："有人能介绍一下 *Drupal 7* 的控制流架构概述吗？或许可以用流程图说明页面是如何生成的（.....）"。

[12] https://tinyurl.com/yuxjp2su
[13] https://tinyurl.com/4kn6t27e
[14] https://tinyurl.com/k9xzkman
[15] https://tinyurl.com/4pjh3ufk
[16] https://tinyurl.com/2a9hb3ek

- 基本架构概念 指的是寻求解释软件架构基本概念的问题。例如，在这个问题[17]中，一位开发人员想了解若干架构概念，例如架构模式：“*MVC* 是一种模式、架构还是框架？什么是模式？什么是架构？（……）”。

- 架构组件功能关注的是架构中某些组件的用途、目的或功能。例如，在这个问题中[18]，一位开发人员询问了基于 Android 的应用程序中生命周期感知组件的用途或目的：“我们的 *Activity/Fragment* 已经有了 *Lifecycle*，那为什么还要使用生命周期感知组件？还请指点一下它的主要用途。如果我们使用生命周期感知组件，那为什么还要使用我们已经了解的生命周期”。

- 特定架构模式问题询问的是某些应用程序设计中常用的特定架构模式，这些模式用于满足功能需求或非功能需求。例如，在这个问题[19]中，一位开发人员询问了三维（3D）电子游戏应用程序中常用的架构模式：“开发 *3D* 游戏时，常用的设计模式有哪些？是否有一些常见的高层架构设计模式？（……）”。

（4）架构实现问题询问的是如何根据架构设计实现某个软件系统。架构设计会先细化为详细设计，然后再转化为代码。架构实现类别包含两个子类别，其中架构组件实现占该类别问题的大多数（在 S O 用户提出的 119 个架构实现类别架构相关问题中占 79 个）。

- 架构组件实现 关注的是系统中组件应如何实现。例如，在这个问题中[20]：“在使用懒加载时，如何在 *Angular 7* 的不同模块中实现单个组件共享？”。

- 架构模式实现问题关注的是如何根据基本设计原则实现特定的架构模式。例如，在这个问题中[21]：“如何在 *Swift* 中实现 *MVC*？我一直在构建 *Swift* 应用，基本上所有功能都放在视图控制器中。我知道这并非最佳做法，因为设计模式有助于扩展应用，但我并不太理解这些模式（……）。我该如何将其改造成模型-视图-控制器设计？”。

(5) 架构工具：有各种架构工具（例如 Enterprise Architect、Archi、Cloudcraft），可用于辅助软件系统的架构设计。通过我们的数据集，我们发现了一些架构相关问题，其中 Stack Overflow 用户会询问这些工具；我们将其归入架构工具 类别。我们进一步将此类别划分为四个子类别，其中架构建模工具 包含了 Stack Overflow 用户在此类别中提出的大多数问题（99 个架构工具类别的架构相关帖子中有 34 个）。

- 架构建模工具问题询问哪些工具能够在设计过程中创建或绘制架构图，以便对软件系统的架构进行建模或表示。例如，在这个问题中[22]：“我刚接触 *TOGAF*，需要开始进行第一次尝试。我正在尝试对我的架构建模。您建议我使用哪种工具来运用 *TOGAF* 对架构建模？”。

- 基于模型的代码生成工具指询问哪些架构工具能够从架构模型生成代码的问题。例如，在这个问题中[23]：“请推荐一款开源工具，用于从 *UML* 设计器生成 *C#* 代码（……）。我需要一款用于生成 *C#* 代码的代码生成工具”。

- 架构工具的使用问题寻求如何使用特定架构工具的说明

[17] https://tinyurl.com/5burxuca
[18] https://tinyurl.com/3x35jzu6
[19] https://tinyurl.com/2u2df8z5
[20] https://tinyurl.com/kp73y3wk
[21] https://tinyurl.com/mpmvwb5c
[22] https://tinyurl.com/ndf7mrnc
[23] https://tinyurl.com/jwkvtzwc

（例如 Archi、Microsoft Visio）。例如，在这个问题中[24]：“如何在 *VS 2015 Community* 的现有解决方案中添加 *UML*/层图？那里没有架构菜单吗？”

- 基于代码的模型生成工具 涉及协助从代码库生成或恢复架构模型的工具。例如，在这个问题中[25]：“我需要为一个项目绘制 *UML* 类图（……），但我不太想手动编写所有类和函数，因此尝试从源代码生成图表，却找不到实现这一点的方法或工具。（……）”。

（6）架构演化：当 Stack Overflow 用户寻求如何重新设计现有架构并对其进行扩展，以满足某些新需求（功能需求或非功能需求）的帮助时，他们会提出这类架构相关问题。在此类别中识别出的两个子类别中，满足新需求的架构扩展子类别包含了 Stack Overflow 用户在此类别中提出的大多数问题（架构演化类别的 *55* 个架构相关问题中有 *42* 个）（见表5）。

- 满足新需求的架构扩展 涉及如何扩展现有系统架构以满足某些新功能需求或非功能需求的实践指导。这些变更并非只发生在某个组件中，也可能波及系统几乎整个架构。例如，在这个问题中[26]：“我正在将一个遗留 *Web Forms* 应用程序扩展／转换为一个全新的 *MVC* 应用程序。扩展涉及技术和业务用例两方面（……）。新项目有两个主要目标：可扩展性（满足当前和未来的需求）以及性能（……）。有没有办法通过 *DDD* 同时实现 *DDD* 所提供的可扩展性和 *DBDD* 所提供的性能？”。

- 为满足新需求而扩展组件包含这类问题：询问如何扩展特定架构组件，以满足现有且正在运行的软件系统中的某些新功能需求或非功能需求。这与上述子类别不同，因为此处的变更或扩展局限于特定组件或层，而不会影响整个系统架构。例如，在这个问题中[27]：“我们目前正在评估 *CQRS* 和事件溯源架构（……）。如果应用程序运行了一段时间后，出现了新增字段的需求，需要在读取模型数据库中的视图模型里添加一个字段，会发生什么情况？比如，*CustomerList* 视图模型需要包含客户邮政编码，而此前并不需要”。

（7）架构重构：当 Stack Overflow 用户希望重组系统架构，以改善这些系统的非功能属性，同时不改变其外部行为时，他们会提出这类架构相关问题。此类别包含三个子类别，其中大多数问题（45 个架构相关帖子中有 21 个）与循环依赖重构这一子类别有关。

- 循环依赖的重构关注的是能够帮助消除模块间不必要的循环或环状依赖问题的技术和工具，从而解决分层违规并改善系统中的依赖结构。例如，在这个问题中[28]：“我正在开发一个 *MVC* 项目，并采用分层架构（……）。现在，我的业务逻辑层（*BLL*）依赖于数据访问层（*DAL*），而数据访问层又依赖于 *BLL*，因为领域对象位于 *BLL* 中。因此，两者相互引用（……）。我该如何解决循环依赖问题？”

- 大型组件的重构 关注的是能够帮助重构软件系统中大型组件的方法。例如，在这个问题中[29]：“我有一个相当大的表格组件，想把它的主体部分拆分成一个新的组件。每次尝试这样做时，表格的样式都会出问题（……）。我希望重构后页面能和现在完全一样。有人知道如何将样式传递给这个新的子组件，或者如何让样式重新正常生效吗？”。

[24] https://tinyurl.com/52zffbw3
[25] https://tinyurl.com/n7rz24mu
[26] https://tinyurl.com/ymbyzcvz
[27] https://tinyurl.com/3rh9xmhs
[28] https://tinyurl.com/475dvbp5
[29] https://tinyurl.com/475dvbp5

- 大泥球的重构：当软件系统缺乏清晰可辨、灵活且适当的架构时，就会出现大泥球。 [47]此子类别涵盖有关大泥球重构方法和工具的问题。例如，在这个问题中[30]：“要把一个大泥球般的 *CF* 应用重构成现代且易于维护的应用，你会采取什么步骤？”

(8) 架构部署 收录了询问如何在托管环境中部署特定软件系统以满足需求（例如功能需求和非功能需求）的架构相关问题。根据我们研究的数据集，我们将此类别分为两个子类别，其中满足质量属性的应用部署 包含了 Stack Overflow 用户在此类别中提出的大多数问题（架构部署类别的 34 个架构相关问题中有 24 个）。

- 满足质量属性的应用部署 包括询问可用于在托管环境中部署应用程序以满足质量属性（例如可用性和性能）的方法和工具的架构相关问题。例如，在问题[31]中，一位开发人员询问如何实现微服务系统的零停机部署：“目前，我正在开发一个基于微服务架构的应用程序。我们计划将 *Spring Boot* 和 *Docker* 作为每个微服务开发的主要技术。目标/需求之一是为用户提供零停机部署功能（.....）。关于零停机部署流程，有什么建议吗？如果您对其他架构有好的想法，或者使用过能帮助我们的工具，也请告诉我们（.....）”。

- 为满足功能需求而进行的应用部署涵盖了这样一类架构相关问题：询问有哪些方法和工具可以帮助部署软件系统，以满足功能需求。例如，在这个问题中[32]，一位开发人员询问了如何部署基于微服务的应用程序，使其在生产环境中运行，并让应用程序的各个服务能够相互调用：“我正在尝试将我的微服务架构部署到生产环境。目前我有 *15* 个服务和 *1* 个外观层。外观层会调用服务、获取数据、对数据进行聚合并生成最终结果。此外，服务之间也会相互调用（虽然不常见，但确实会相互调用）.....因此，我决定使用 *5* 台服务器（*5* 台高端服务器）：*A*、*B*、*C*、*D*、*E*。*A* 将作为 *LVS*（用于负载均衡）；*B* 和 *C* 将托管外观层。因此，当收到发往外观层的请求时，请求会先到达 *A*，再由其负载均衡到 *B* 和 *C*.....因此，*B* 和 *C* 服务器各自也会包含一个 *HAProxy* 实例，因为外观层调用服务时也需要进行负载均衡.....但我的问题是，应该如何允许服务之间相互调用？.....”。

（9）架构文档：在我们的研究数据集中，这是唯一没有子类别的架构相关问题类别。架构文档类别包含有关辅助记录软件系统架构的方法和工具的问题。例如，在这个问题中[33]，，一位开发人员询问了记录不同类型系统架构的最佳实践和工具：“基于 *Java* 或 *.NET* 的 *PC* 应用程序，以及基于 *VxWorks*、嵌入式 *Linux* 或 *Windows CE* 的嵌入式应用程序，在记录软件设计和架构方面有哪些最佳实践和软件工具？（.....）”。

研究问题1的主要发现

发现 1：Stack Overflow 用户会提出范围广泛（9 个类别）的架构相关问题，其中，架构配置（27%，968 个架构相关问题中的 261 个）、架构决策（19%，968 个架构相关问题中的 181 个）和架构概念（15%，968 个架构相关问题中的 142 个）是最常被提出的三类架构相关问题。

4.2. 设计情境的类别（研究问题*2*）

本研究问题旨在探究提出架构相关问题时所涉及的设计情境类别。如第3.2.2节所述，为回答这一研究问题，我们在分析研究问题2的提取数据（即设计情境）时，采用了 [27]和 [30]中预定义的设计情境分类

[30] https://tinyurl.com/j6rdefeb
[31] https://tinyurl.com/4tyd4yt6
[32] https://tinyurl.com/37dmd6av
[33] https://tinyurl.com/msnbc7xb

在968个架构相关问题中（见图1）。我们发现，大多数（71%，968个中有687个）经分析的架构相关问题描述了其设计情境（即关于系统预期运行环境的知识 [27]），随后回答者根据给定的设计问题和设计情境，提供了可能的解决方案及其决策依据。此外，我们确定了设计情境的三个主要类别和八个子类别。表6报告了提及的类别、各类别的子类别、出现比例（以687个架构相关问题为基数）以及计数信息。从表6 中也可以明显看出，应用情境是最常见的设计情境类别（54%，687个架构相关问题中有377个），而组织情境则是占比最低的类别（8%，687个架构相关问题中有56个）。

表6：设计情境的类别、子类别及其数量与百分比

设计上下文	子类别	数量
应用情境（55%，377）	应用领域情境	313
	外部服务情境	64
平台情境（37%，254）	软件情境	139
	硬件情境	115
组织情境（8%，56）	开发进度情境	36
	利益相关者情境	13
	资源背景	7

（1）应用情境指待设计的软件系统或产品。它通过设备（平台实体）提供服务给最终用户 [27]。此类别包含两个子类别，其中应用领域情境子类别最为常见（377 个架构相关问题中有 313 个）。

- 应用领域情境描述正在开发的应用程序所属的领域／类型（例如电子商务系统、银行系统、分布式系统） [30]。一些 Stack Overflow 用户会在架构相关问题中说明他们即将设计的应用领域，以获得适合这些应用领域的潜在且相关的架构解决方案。例如，在这个问题中[34]，一位开发人员提到他／她正在设计一个电子商务系统：“我正在使用微服务架构设计一个电子商务系统。假设我有两个上下文：产品目录、库存和定价。我觉得它们的职责划分很明确。但是，为了展示商品列表，我需要向产品目录发出请求，获取 *ID* 列表，然后用这些 *ID* 查询库存微服务，以检查库存状态（有货或缺货）。除此之外，我还需要向定价服务发出请求，获取每种产品的价格（.....）。我一直在阅读有关微服务架构的资料，了解到当涉及许多‘联接’时，这些上下文可能应该合并为一个（.....）。我们可以使用领域事件通知‘搜索’微服务有内容发生了变化。这样，我们就可以通过单个请求来展示商品列表。这看起来像 *CQRS*。有没有正确的做法？哪种更好？有哪些权衡？”

- 外部服务情境 指的是应用程序所使用的外部软件服务的规格说明 [27]。例如，在这个问题中[35]，一位开发人员提到，他/她正在设计一个需要使用 Azure 或亚马逊云服务的系统：“基本上，我的问题是关于应用程序架构。为托管进行设计很容易，但云计算带来了新的挑战（.....）。我不确定在为安全工程师设计应用程序时应该怎么做，因此高可用性非常重要。所以，如果我的应用程序是用 *ASP.NET* 编写的，并使用 *SQL Server*，那么似乎最好的选择是针对 *Azure* 进行设计，但亚马逊的解决方案会是个不错的选择吗？我该如何决定是让所有内容都放在同一个系统上，还是将数据放在亚马逊的云上，而把 *ASP.NET* 放在 *Azure* 上？（.....）我决定了编程语言，这是否就意味着我只能选择某种云解决方案？”。

（2）平台情境包括用户用于访问应用程序的硬件技术，

[34] https://tinyurl.com/2p93nesu
[35] https://tinyurl.com/2p8phmr6

其运行的软件，以及此类技术的网络能力 [27]。在我们的数据集中，我们识别出两种平台情境（即软件情境和硬件情境）。

- 软件情境包含设备软件元素的信息，例如操作系统（OS）或其他已安装的应用程序。此子类别收集描述设备软件元素（例如操作系统）的架构相关问题，说明计划中的软件系统在生产环境中需要运行于何种设备上 [48]。我们发现，一些 Stack Overflow 用户在提出架构相关问题时会提供这类信息。例如，在这个问题中[36]：“我需要开发一个从***Windows Phone 7***开始的移动应用程序，之后需要将该应用程序转换到其他平台，例如***Android***、***iOS***。该应用程序包含许多用于采集数据的屏幕，所有数据都存储在本地存储中，最后传送到中央服务器。我想了解应该如何设计架构（.....）。”

- 硬件情境包含定义用户访问和使用应用程序所经设备的平台实体，其类型多种多样，例如台式机、笔记本电脑和可穿戴移动设备 [27]。硬件情境类别收集提及硬件技术（例如台式计算机）的架构相关帖子，用户通过这些技术访问和使用计划中的应用程序。例如，在这个问题[37]中，一位开发人员提到自己正在开发桌面应用程序：“我们想开始开发一款中间层桌面软件。我们决定使用 *WPF*。我们不想使用 *MVVM* 模式，因为我们不熟悉 *MVVM*。不使用 *MVVM* 模式（采用三层架构，但不使用 *MVVM*）来开发 *WPF* 应用程序是否可行？尽管它的性能是否仍优于 *Windows Forms*？。”

（3）组织情境指可能影响软件系统开发的开发进度（例如，上市时间）、人员（即利益相关者）或资源。该类别包括两个子类别，其中开发进度情境子类别最为常见（56 个架构相关问题中有 36 个）。

- 开发进度描述了软件开发所投入的时间。例如，在这个问题中[38] 一位开发人员提到，某个软件项目的开发时间仅限三个月：“出于个人和大学研究方面的原因，我正在考虑使用面向服务的架构构建一个简单的客户关系管理系统（.....）。我设计的架构包括：- *We-bGUI*（其他服务的客户端）- *AnalyticsService*（一项接收、分析并收集数据的服务）- *CustomerCareService*（一项使用表述性状态转移 *API* 执行 *CRUD* 操作的服务.....）。对于客户端，哪种身份验证方式更合适（用户令牌、*OAuth* 或类似方式）？我只有大约 ***3*** 个月来完成它（.....）”。

- 利益相关者情境描述参与软件系统开发的人员，例如项目经理、所有者、架构师、开发人员、用户等。例如，在这个问题中[39]，提问者提到了参与开发 3D 地图应用程序的若干开发人员，并表示：“我正在尝试开发 *3D* 地图，找到了 *3* 种解决方案：使用游戏引擎（如 *Unity*）、使用 *3D* 图形 *API*（*OpenGL* 等）或使用 *Web* 应用程序。还有其他实现方式吗？这 *3* 种解决方案（设计决策）中哪一种更好？（请说明理由）（.....）开发人员：***3*** 名程序员”。

- 资源背景指开发应用程序时可用资源（例如资金或技术能力）的匮乏（或可用性） [49]。例如，一位开发人员需要在预算紧张的情况下更新一个应用程序，并在这个问题中[40] 表示：“我必须构建一个数据库/富图像应用程序，而且它只会不断增大（可扩展性）。我的预算有限，但有一台相当不错的 *3Ghz* 至强服务器和 *400 GB* 存储空间。有什么好主意吗？对于个人来说，有没有适合预算非常紧张的好办法”。

[36] https://tinyurl.com/2p8nd2kw
[37] https://tinyurl.com/yc2eyvhs
[38] https://tinyurl.com/2bu5yu65
[39] https://tinyurl.com/yra7d3y7
[40] https://tinyurl.com/4pjp3ntj

研究问题2的主要发现

发现 2：大多数 Stack Overflow 用户（71%，968 个架构相关问题中有 687 个）在提出架构相关问题时会考虑设计情境。
发现3：应用情境是架构相关问题中最常见的设计情境类别（54%，687个中有377个），而组织情境是架构相关问题中最不重要的设计情境类别（8%，687个中有56个）。

4.3. 有多个回答的架构相关问题的特征（研究问题*3*）

一些架构相关问题持续获得 Stack Overflow 用户的关注，用户会不断回答这些问题。这促使我们通过分析这些问题的特征，探究为什么它们在 Stack Overflow 上会获得多个回答。如第3.2.2节所述，我们采用了扎根理论中的两种技术（即开放式编码和持续比较）[17]，从 650 个架构相关帖子（968 个架构相关帖子中的一个子集）中考察了有多个回答的架构相关问题的特征（见图1）。如第3.2.2节所述，我们参考问题内容及其附带的评论，以了解哪些因素（例如问题表述 [35] 或问题中的某些特征（例如架构图））会促使此类架构相关问题获得多个回答。数据分析结果归纳出有多个回答的架构相关问题的四个常见特征。表7列出了这四个常见特征及其数量和占比，显示表述清晰的架构信息是最常见的特征（46%，650 个架构相关问题中有 297 个），而获赞的架构问题则是最不常见的特征（8%，650 个架构相关问题中有 51 个）。此外，图3展示了有多个回答的架构相关帖子的回答数量。

（1）问题中表述清晰的架构信息：架构相关问题能够持续获得回答的主要原因，是其架构信息表述清晰。该问题概述了计划中的软件系统及其基本原则（例如，设计情境和架构约束），帮助其他 Stack Overflow 用户理解该问题的真正内容（问题目的）。例如，有一条评论：“*+1* 很棒的问题，我想说这是一个写得非常漂亮且表述清晰的问题！）”发布在该问题下方，[41] 其中清楚地说明并解释了所遇到的架构问题（即设计一个满足可扩展性和数据可视化需求的系统）。

（2）问题中清晰的描述及架构图：架构相关问题能够持续获得回答的另一个原因，是其易于阅读和理解。例如，有些问题会清楚地说明组件和连接器，以及接口和与其他组件之间的关系等必要信息。此外，提供架构组件图等图示来描绘和阐明软件系统的逻辑架构视图，也有助于问题持续获得回答。例如，在一位回答者开始回答该问题 [42] 之前，这位回答者表示：“你的问题描述得很清楚。感谢你绘制的这张小图，它有助于阐明整体架构（.....）”。

(3) 用于回答问题的替代架构方案：尽管不同的架构解决方案可作为解决类似设计问题的替代方案，但它们的质量各不相同[34]。例如，两种架构解决方案都可能解决互操作性问题，但在解决性能问题方面有所不同。然而，这类替代架构方案非常重要，因为它们为针对特定设计问题，从候选架构方案中进行选择并作出设计决策提供了广泛的可能性。在本研究中，我们注意到，询问如何在各种架构解决方案（例如数据库、框架和编程语言等技术）之间进行选择的问题，会不断收到新的回答（替代方案）。例如，Stack Overflow 用户希望为其 Ruby 和 Java 应用程序之间的交互，选择合适的消息格式与消息传输技术组合，以实现高性能、可用性、可扩展性等质量属性，并发布了一个问题[43]：

[41] https://tinyurl.com/dfw27h38
[42] https://tinyurl.com/74fr8mwe
[43] https://tinyurl.com/44sey2a2

“我们有托管在云端（*RackSpace* 云）上的 *Ruby* 和 *Java* 应用程序，它们将按如下方式交互（……）。我们希望评估消息格式（例如 *JSON*）以及消息传输技术（*RPC*、表述性状态转移、简单对象访问协议等）。我们的标准是高性能、可用性、可扩展性（……）。您会推荐哪种消息格式与传输方式的组合？为什么？”。

（4）获赞的架构问题：通常，在 SO 中，如果一个问题持续获得赞成票，那么它得到回答或获得更多回答的可能性就会增加 [50]，，架构相关问题也不例外。在我们的数据分析中，我们发现，这一因素（获赞的架构问题）也促使架构相关问题获得多个回答。例如，在这个架构相关帖子中[44]，，一位答主在回答问题时，在回答帖的标题中写道：“由于这个问题获得了好几次赞成票，我想分享一下我最终采取的做法（……）”。

表7：回答较多的架构相关问题的特征及其数量和百分比

特征	数量
表述清晰的架构信息	297（46%）
问题中有清晰的描述和架构图	174（27%）
回答问题的替代架构方案	118（18%）
获赞的架构问题	51 (8%)

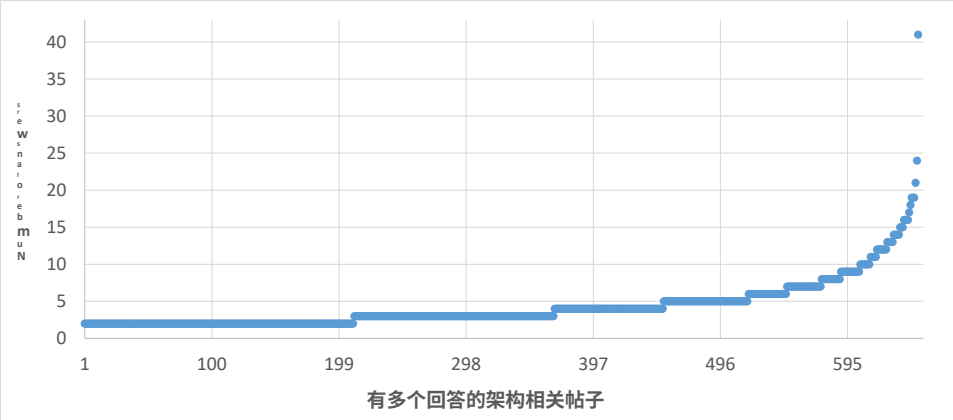


图3：答案数多于一个的架构相关帖子的回答数量

研究问题3的主要发现

发现4：表述清晰的架构信息是答案数多于一个的架构相关问题中最常见的特征（占46%，即650个架构相关问题中的297个）。

Fin现5：在架构相关问题中，架构图（例如组件图）的存在 - ture 会增加这些问题获得多个回答的可能性。

4.4. 仅有一个回答的架构相关问题的特征（研究问题*4*）

我们发现，有些架构相关问题不太能引起 Stack Overflow 用户的关注，因此后续回答较少。与先前的两个研究问题（即研究问题1和研究问题3）类似，我们采用了两种技术（开放式编码和持续比较），研究318个架构相关帖子（968个架构相关帖子的子集）中仅有一个回答的问题的特征（见图 1）。如第 3.2.2 节所述，与研究问题 3 类似，我们参考了问题内容及其附带的评论，以了解

[44] https://tinyurl.com/7bc8764a

哪些因素会打消回答者继续回答某些架构相关问题的积极性。我们归纳出这类问题的五个常见特征，并在表8中列出了各自的数量和百分比。结果显示，问题中信息不足（39%，318个架构相关问题中的125个）和结构不佳的架构问题（22%，318个架构相关问题中的69个）是仅有一个回答的架构相关问题的两大主要特征。下文将结合架构相关帖子中的示例，详细阐述这些特征。

（1）问题信息不足：缺少某些重要信息的架构相关问题（例如，缺少有关组件和连接器的信息，以及这些组件与其他组件之间的接口和关系）无法吸引社区成员提供回答。例如，一位开发人员指出，这个架构相关问题[45]缺少一些信息："如果不做出（许多）假设，就无法回答你的问题。关于模块依赖项，还需要更多信息。它们是无状态的吗？你能画出请求流程吗？"。

（2）结构不佳的架构问题：我们发现，结构不佳（例如，表述不清楚）的架构相关问题无法向社区成员清楚地阐明其目的，因此社区成员难以提供回答。这些问题听起来含糊不清、模棱两可，或难以理解。例如，在这个问题[46] 中，一位开发人员询问如何在 Android MVP 整洁架构中实现交互器，却未能清楚地组织问题；随后，另一位开发人员发表评论："那么，你的问题究竟是什么？"。提问者回来后编辑了问题，使其更加清楚，以便其他社区成员理解问题的真正内容。

（3）架构被视为离题：尽管 Stack Overflow 上有人提出架构相关问题，但 Stack Overflow 主要是为寻求编程信息的人设计的。因此，架构并非直接针对编程相关问题，而主要关注高层次的结构问题，这导致架构相关问题较少受到社区关注，因而在 Stack Overflow 上往往只有一个回答或无人回答。例如，一位开发人员询问 ZeroMQ 架构的概况，另一位开发人员在该问题下方评论[47] 道："我投票关闭此问题，因为它属于离题内容，与编程并无真正关系"。

（4）问题中涉及专有技术：我们发现，有少数问题询问使用并不广泛的专有技术，例如数据库和框架，这些问题在 Stack Overflow 上较少受到关注，因此往往只有一个回答。例如，在这篇架构相关帖子[48]，中，一位社区成员在回答问题时，在回答标题中自称是 RethinkDB 的创始人之一。其他社区成员在评论串中继续提出有关 RethinkDB（一种并不广泛使用的数据库）的问题，但这些问题一直没有得到回答。例如："免责声明：我是 *RethinkDB* 的创始人之一。抱歉，回答有点长（……）。*RethinkDB* 的设计采用了非常灵活的架构（……）"。

（5）重复的架构问题：与 Stack Overflow 上其他类型的问题（例如编程问题）类似，有些架构相关问题得到的回答很少（即只有一个回答），或者一直无人回答，因为它们是重复的架构问题。社区成员不喜欢再次回答之前已经回答过的问题 [50]。他们希望提问者在发布这类新问题之前先查看网站（即确认自己的问题尚未发布并得到回答）。例如，有人在这条架构相关问题下发布了一条评论："*stackoverflow.com/questions/15142386/* 的重复问题。. ."[49]。

RQ4 的主要发现

发现 6：问题中信息不足（39%，318 个架构相关问题中有 125 个）和结构不佳的架构问题（22%，318 个架构相关问题中有 69 个）是仅有一个答案的架构相关问题中最常见的两种特征。

[45] https://tinyurl.com/btukhpzx
[46] https://tinyurl.com/3m6fzjbe
[47] https://tinyurl.com/rmpfarjc
[48] https://tinyurl.com/nrz66x3w
[49] https://tinyurl.com/2p9382hh

表 8：回答较少的架构相关问题的特征及其数量和百分比 s

特征	数量
问题中信息不足	125（39%）
结构不佳的架构问题	69（22%）
被视为离题的架构	51（16%）
问题中涉及专有技术	32（10%）
重复的架构问题	29（9%）

4.5. 被认为有用的架构解决方案分类体系（研究问题*5*）

此研究问题旨在构建一个分类体系，归纳 Stack Overflow 上被认为有用的架构解决方案。正如第 3.2.2 节所述，为回答此研究问题，我们首先考察了 Stack Overflow 用户如何讨论归属于相关架构问题的架构解决方案的有用性。我们需要了解 Stack Overflow 用户可能会用哪些方式（例如，使用哪些术语）来表达架构解决方案在 Stack Overflow 上的有用性。了解 Stack Overflow 用户对这些解决方案有用性的讨论，有助于引导问答平台所有者创建相应机制，帮助 Stack Overflow 用户高效、有效地搜索和（重新）使用此类有用的架构解决方案。我们发现，在评论串中（见图 2），Stack Overflow 用户经常使用两个与有用性相关的词（即“有用”和“有帮助”），以及另外六个词（即“肯定”“非常”“确实”“超级”“极其”和“难以置信地”），明确表达他们对某些架构解决方案有用程度的反馈，这些解决方案是针对相应架构相关问题提供的。请注意，Stack Overflow 用户还可能通过其他方式表达架构解决方案在 Stack Overflow 上的有用性，因此我们不能声称已找出所有表示有用性的词语。其次，正如第3.2.2节所详述的，我们对包含有用知识的 324 个架构相关帖子（968 个架构相关帖子中的一个子集）的解决方案内容进行了全面、深入的考察（见图 1），以构建这些解决方案的分类体系。此次考察得出了一个分类体系，包含 7 个主要类别、20 个子类别（其中 1 个被编码为“其他”，即指不属于已生成子类别的编码）以及 85 种类型（见图4）。

ary for linear algebra
ication

ary for Web-based
ication

ary for Android application
ementation

ry for vector graphics
cation implementation

ry for computer vision
cation implementation

ry for game system
ementation

orary for architecture
mplementation **(12)**

or video game system
ementation

or facial recognition application
mentation

or cryptocurrency
cation implementation

or Web-based application
ementation

or iOS application
ementation

architecture implementation
(18)

mentation

5 (2%)

lution for architecture documentation

or architecture
ntation **(2)**

ment viewpoint
itecture
ntation

ment viewpoint
itecture
ntation

Tool for architecture
documentation **(3)**

- Tool for database architecture documentation
- Tool for Web application architecture documentation
- Tool for desktop application architecture documentation

(1) 架构配置解决方案：这是我们分类体系中最大的架构解决方案类别（见图 4）。该类别中的解决方案提供了方法和工具，用于配置计划开发的软件系统中的组件和连接器。在此类别确定的三个子类别中，领域配置解决方案子类别涵盖了该类别中讨论的过半解决方案（127 个 ARP解决方案中有 75 个，见图 4）。

- **领域配置解决方案**：此子类别探讨了为各种领域的应用程序配置系统的方法和工具，例如分布式系统配置解决方案、银行系统配置解决方案、电子商务系统配置解决方案和游戏系统配置解决方案（见图 4）。关于分布式系统配置解决方案这一类型，有用户询问如何使用 Java 设计和配置一个包含特定组件的二层或三层分布式应用程序，其中包括一个中央共享数据库和多个胖客户端（基于 Swing 的图形用户界面客户端（图形用户界面））。用户需要一种更简单的方法，帮助配置这些客户端，使每个客户端都能获知其他客户端提交到数据库的数据变更。该 ARP 中的第一个解决方案[50] 基于问题中描述的应用领域（即 分布式应用程序）提出。该解决方案建议遵循面向企业层的 Java EE 分布式容器／基于组件的架构来配置应用程序的组件（例如数据库和客户端），并指出：*"(...) Java EE* 是一种面向企业层的分布式容器／基于组件的架构 *(...)* 你可以设计一个同时采用主题/订阅模式和直接队列的消息传递领域。这些配置可以通过声明式方式设为持久或非持久，等等。*(...)*。"

- **平台配置解决方案**提供了一系列方法和工具，使应用程序的组件和连接器能够根据这些应用程序将在生产环境中运行的平台（例如 Windows操作系统）进行配置。我们在这一类别中识别出七种常见的解决方案类型，例如 *Android*应用配置解决方案、*Windows* 应用程序配置解决方案、*iOS*应用配置解决方案以及跨平台配置解决方案（见图4）。关于跨平台配置解决方案这一类型，一位 SO 用户需要设计并配置一个在两台 iOS设备（即 iPad 和 iPhone）之间发送数据的应用程序，并由 iPad 充当 iBeacon。该架构相关帖子中的第一个解决方案[51] 说明了如何配置应用程序的组件（包括 iPad 和 iPhone），并指出可以采用一种同样支持 Android 的方法：*"(...)* 我不得不采用一种同时支持 *Android* 的架构，因此改用了蓝牙。充当 *iBeacon* 的 *iPad* 也有蓝牙代码，用于查找具有特定特征码的'外围设备'。*iPhone* 检测到 *iBeacon* 后，应用程序便开始传输具有相应特征码的蓝牙外围设备信号 *(...)* "。

（2）架构实现解决方案：此类别中的架构相关帖子解决方案提供技术解决方案，例如框架和库（见图4），用于实现多种架构设计，以满足系统需求（例如质量属性）。根据我们的数据集，我们将这些技术解决方案分为三个子类别，其中架构实现框架包含了 Stack Overflow 用户在此类别中讨论的大多数解决方案（59 个架构相关帖子解决方案中有 21 个）（见图4）。

- 架构实现框架：这些解决方案汇集了用于实现架构设计的不同类型的框架，例如基于 *Web* 的应用程序框架(如 Laravel、Django、Express.js 和 Play 框架）、*iOS* 应用程序框架(如 SwiftUI、Flutter 和 React Native 框架），以及*Windows* 应用程序框架(如 WinForm、WPF 和 UWP 框架）（见图4）。关于基于 *Web* 的应用程序框架这一类型，有用户询问了一种框架（以及其他问题），该框架能够帮助在基于 Web 的应用程序中实现 REST API，并向移动应用提供内容。该架构相关帖子中的第三个回答[52] 推荐使用*Play* 框架来解决这个问题，并指出："用 *Play*！就能完成所有工作。在 *Play* 中编写 *REST* 服务非常容易（.....）"。

- 架构实现 *API* 汇集了不同类型的 API，作为回答相关问题的解决方案。这些问题询问的是用于实现架构设计的 API，例如视频游戏系统 *API*

[50] https://tinyurl.com/jfnuke2w
[51] https://tinyurl.com/ytw52k6p
[52] https://tinyurl.com/2p8pm9rs

（例如 Pokeapi、Chicken Coop、Dota2 和 Minecraft API）、基于 *Web* 的应用程序 *API*（例如表述性状态转移、简单对象访问协议、远程过程调用和地理定位 API），以及人脸识别应用程序 *API*（例如 Lambda labs 和 Microsoft 计算机视觉 API）（见图4）。关于基于 *Web* 的应用程序 *API*这一类型，一位开发人员需要一个 API，以面向服务的架构（SOA）实现请求-响应式 Web 应用程序，从而满足某些要求（包括高性能）。这条架构相关帖子[53] 中的第二个回答建议使用表述性状态转移 *API*而不是简单对象访问协议 *API*，并指出："（.....）也可以考虑使用表述性状态转移 *API*，它的开销比简单对象访问协议小，而且可以使用 *JSON* 作为文档格式，这种格式也比 *XML* 更紧凑，从而降低网络吞吐量要求（.....）简单对象访问协议有更多花哨的功能，但并非所有语言都能很好地支持；如果使用表述性状态转移 *API*，在这方面会更稳妥（.....）"。

- 架构实现库：这些解决方案推荐了各种用于实现架构的库，例如，线性代数应用程序库（如 JBLSA、MTJ、OjAlgo 和 EJML）、计算机视觉应用程序库（如 OpenCV 库）、矢量图形应用程序库（如 DISLIN 库），以及基于 *Web* 的应用程序库（如 HPPC、Trove 和 FastUtil）（见图4）。关于矢量图形应用程序库类型，这篇架构相关帖子中的第一个回答[54] 推荐将 Raphaël 库作为解决方案，以回答有关矢量图形应用程序的问题，并指出：*"(...)* 我选择了 *RaphaëlJS*，必须说使用它非常愉快，而且它的帮助也非常出色 *(...)*"。

（3）架构说明：此类别中的 ARP解决方案提供架构的理论解释、用途或功能，而非提供如何执行某项操作的具体说明（例如，如何配置系统中的某些架构组件）。架构说明类别包含四个子类别，其中基于云的架构说明是讨论最多的子类别（53 个 ARP解决方案中有 28 个）（见图4）。

- 基于云的架构说明 介绍云计算服务架构的理论解释、差异和功能，例如 *Azure App Service* 功能说明、*EC2* 功能说明和*AppHarbor* 功能说明 (见图4)。例如，有用户询问了 *Azure App Service* 与 A*zureService Fabric* 在软件开发功能方面的区别。该架构相关帖子中的第四个回答[55] 详细说明了这两个 Azure 平台在软件开发功能方面的区别，以此回答该问题，并指出：*"(...)* 它们是两个独立的平台，遵循不同的开发范式。*App Service* 能提供 *Service Fabric* 开箱即用时不具备的功能，例如自动扩缩、身份验证、速率限制、与 *SaaS* 应用程序集成等。*(...)* "。

- *CMS* 架构说明 介绍内容管理系统（CMS）架构的功能，例如 *Drupal* 功能说明 和*TYPO 3* 功能说明 (见图 4)。例如，该架构相关帖子中的第一个回答[56] 介绍了 *Drupal* 的架构概述（附有架构图），以此回答有关 Drupal 功能（例如控制流以及页面的生成方式）的问题。回答指出：*"(...)* 尽管它采用过程式 *PHP*，但其架构完全由事件/监听器驱动，在主 *PHP* 脚本中没有简单的'流程'可供查看。*(...) Drupal* 的 *index.php* 文件充当前端控制器。*(...)*"。

- 数据库架构说明 将解释或描述数据库系统架构的架构相关帖子解决方案归为一组，例如 *NoSQL* 数据库功能说明 (例如 Apache Cassandra) 和 图数据库功能说明 (例如 Nebula Graph)（见图 4）。例如，这篇架构相关帖子中的第一个回答[57] 从数据复制的角度解释了 *Cassandra* 如何应对数据故障场景，并将其作为对相关问题的解决方案。该问题询问，当包含用户正在查询的记录（数据）的一个节点宕机时，Cassandra 会如何处理这种情况。回答指出："*Cassandra* 集群会在各个节点间复制数据。副本的具体数量可以配置，但生产集群通常会使用 *3* 的复制因子。这意味着，给定的行将存储在三台不同的机器上，位于

[53] https://tinyurl.com/pakzw2yk
[54] https://tinyurl.com/259h4f6e
[55] https://tinyurl.com/2p8hnkmm
[56] https://tinyurl.com/2a9hb3ek
[57] https://tinyurl.com/wpvw6j5h

集群中（.....）。在处理请求方面，如果某个节点收到的数据请求所需的数据不在该节点上，它就会将请求转发给拥有该数据的节点”。

- *Web* 服务器架构说明介绍 Web 服务器架构的功能或差异，例如*Web* 服务器功能说明（如 XAMPP、IIS 和 WAMP 服务器）（见图4）。例如，这篇架构相关帖子[58] 详细说明了*XAMPP*、*WAMP*和*IIS* 服务器之间的差异，作为对询问这三种 Web 服务器差异的问题的解答，并指出：“*(...)* 它们（*XAMPP* 和 *WampServer*）的区别在于软件包的格式／结构、配置等。*(...) IIS* 是一种 *Web* 服务器应用程序，就像 *Apache* 一样，只不过它由微软开发，且仅适用于 *Windows*（*Apache* 同时支持 *Windows* 和 *Linux*）*(...)* ”。

(4) 架构策略：此类 ARP 解决方案会提供并说明架构策略，以实现软件系统的特定质量属性（如性能和安全性）。该类别中识别出四个架构策略子类别，其中性能策略是讨论最多的子类别（37 个 ARP 解决方案中有 21 个涉及该子类别）（见图4）。

- 性能策略介绍并解释有助于实现系统性能需求的架构策略。我们识别出四种性能架构策略，例如资源调度策略和内存缓存策略（见图4）。关于内存缓存策略，一名开发人员想在两种数据处理设计技术之间选择一种合适的方法（即直接操作数据库，或使用对象并让 ORM 负责存储），以提升库存系统的性能。该系统需要处理存储在数据库中的数千种商品类型及其各自的数量。根据问题中描述的场景，这篇架构相关帖子中的第一个回答[59] 建议采用内存缓存架构策略并结合 ORM 来提升系统性能，并指出：*“(...)* 大多数时候，执行 *SQL* 查询更简单，但内存缓存确实能大幅提升性能。没错，它会占用内存。谁在乎呢？如今的工作站可以配备 *64GB* 内存了 *(...)* ”。

- 可维护性策略涵盖了能够满足系统可维护性需求的架构策略。我们识别出三种可维护性策略，例如低耦合策略和通过引入 *API* 实现封装的策略（见图4）。关于低耦合策略，一名开发人员需要为所有 Web 应用程序构建一个可扩展、易于维护且低延迟的单点登录系统。这篇架构相关帖子中的第一个回答[60] 建议在设计应用程序时采用低耦合策略，以提高其可维护性，并指出：“我不会在数据库层实现身份验证 *(...)* 这可能会变得难以维护。我更倾向于采用松耦合方法，在中央服务器上开放一个简单的服务，让其他应用服务器能够执行身份验证请求 *(...)*”。

- 安全战术提供有助于实现系统安全需求的架构策略。该子类别包含三种架构策略： 身份验证战术、限制访问战术和通信链路加密战术（见图4）。关于身份验证战术，这篇架构相关帖子中的第一个回答[61] 给出了并解释了身份验证战术，以回答如何在基于微服务的应用程序中设置“用户 JWT”的双级身份验证方法这一问题，并指出：*“(...)* 您可以通过使用单个用户令牌和基于声明的授权来实现所需的两个级别的安全性。如果使用用户令牌向网关发起调用，网关会根据用户令牌对调用进行身份验证，并提取*“userId”*声明 *(...)*”。

- 可用性战术 汇集了能够满足系统可用性需求的架构策略。我们收集了三种可用性战术，例如数据复制战术 和 功能冗余战术(见图 4)。关于数据复制战术，一位开发人员询问，如何满足某应用程序的可用性需求：该应用程序需要使用两个 Amazon EC2 实例，且每个实例都配有 Cassandra 数据库。该架构相关帖子中的第一个回答[62] 介绍并解释了可应用于其应用程序的复制机制，

[58] https://tinyurl.com/ysacs8zd
[59] https://tinyurl.com/289ffurv
[60] https://tinyurl.com/22ckrdv4
[61] https://tinyurl.com/bdhk4uud
[62] https://tinyurl.com/2p8db7mu

（根据问题中描述的设计场景），可以通过以下方式应用于其应用程序：*"(...)* 在你的场景中（由于你位于单个数据中心），可以使用 *SimpleStrategy* 作为复制策略，并将复制因子（*RF*）设为 *2*。采用此配置后，所有数据都会复制到两个节点上。这样，无论从哪个节点访问，数据都将可用，且具有较高的"。

（5）架构模式：此类别中的架构相关帖子解决方案提供了用于满足多项系统质量属性的架构模式，同时也介绍了某些应用领域中常用的架构模式（见图4）。在该类别确定的两个子类别中，满足质量属性的架构模式建议子类别包含了 Stack Overflow 用户在此类别中讨论的大多数解决方案（30 个架构模式类别的架构相关帖子解决方案中有 17 个）（见图架构模式类别的解决方案中有 17 个）（见图4）。

- 满足质量属性的架构模式建议收集了用于满足系统质量属性的架构模式，例如用于提高可修改性、可复用性和可移植性的模式（中介者、MVC 和面向服务的架构）（见图4）。例如，一位 Stack Overflow 用户询问了适用于多租户网站、能够实现各个独立插件之间通信的最佳 C# 架构模式，其中主要关注可修改性、可复用性和灵活性。该架构相关帖子中的首个回答[63]建议采用面向服务的架构模式来解决这个问题，并指出："（.....）我可能会建议采用面向服务的架构。主要是因为它能以非常快速、灵活的方式适应业务需求。这种架构有很多优点：轻量、灵活、代码可复用（.....）"。

- 架构模式的使用汇总了针对特定应用领域中常用架构模式提出的问题（见图4），例如时间关键型系统的模式（抢占式多任务处理）、*Android*应用的模式（MVP、MVVM、MVC、观察者）以及分布式系统的模式（SOA、中介者）。例如，一位用户询问了时间关键型应用程序的架构模式。该架构相关帖子的第一个回答[64] 建议抢占式多任务处理模式，并回答说："*.....*这种模式称为抢占式实时操作系统，能够立即处理事件（.....）"。

（6）架构部署解决方案汇总了讨论系统在托管设备（云或本地服务器）上的架构部署、以满足系统需求的架构相关帖子解决方案。该类别包含两个子类别，其中部署流程是讨论最多的子类别（13个架构相关帖子解决方案中有9个）（见图4）。

- 架构部署流程汇集了讨论如何部署应用程序架构以满足应用程序需求（例如功能或非功能需求）的 ARP解决方案。我们发现了几种架构部署流程，例如使用 *Azure App Service* 的部署流程、使用 *Kubernetes* 的部署流程以及使用 *AppHabor* 为 *.NET* 应用程序部署的流程（见图4）。关于使用 *Kubernetes* 的部署流程，一位回答者向一位提问者提供并说明了使用 Kubernetes 的部署流程。该提问者希望将一个由 15 个 Spring Boot 微服务构成的微服务架构部署到五个 Kubernetes 节点上，并设置一个集群主节点。根据问题中描述的场景，该 ARP 中的第一个回答[65] 建议至少使用三个集群主节点，而不是一个，以避免数据丢失，并据此满足系统的可用性需求。回答中提到：*"(...)* 一个主节点是不够的。如果该虚拟机、底层硬件或主节点上的服务发生故障，所有客户都将无法使用服务，并可能导致灾难性的数据丢失。至少运行 *3* 个主节点"。

- 架构部署工具汇集了用于部署系统架构、以满足系统需求的工具。我们收集了几种工具，例如架构部署模拟器工具和持续部署工具（见图4）。关于持续部署工具，这篇架构相关帖子中的第一个回答[66] 推荐将 Argo CD 工具作为该问题的解决方案。该问题询问的是用于在 Kubernetes 上进行微服务架构持续部署的工具。回答指出：*"(...) ArgoCD* 工作流提供了这一功能 *(...)*"。

[63] https://tinyurl.com/42m8ts56
[64] https://tinyurl.com/25y4b9e6
[65] https://tinyurl.com/483en2ts
[66] https://tinyurl.com/yc7j8z5j

（7）架构文档解决方案：此类别中的架构相关帖子解决方案提供了用于记录架构的方法和工具（见图4）。此类别包含两个子类别，其中架构文档工具是讨论最多的子类别（5 个架构相关帖子解决方案中有 3 个涉及此子类别，见图4）。

- 架构文档工具指能够促进架构文档编写的工具，例如*Web* 应用架构文档工具和数据库架构文档工具（见图4）。关于*Web* 应用架构文档工具，该架构相关帖子中的第一个回答[67] 将 NJsonSchema 工具作为解决方案，回答了有关用于记录基于微服务的应用的工具的问题，并指出：“*(...) there isNJsonSchema tool https://github.com/NJsonSchema/NJsonSchema*”。

- 架构文档视点提供了多种架构文档视点，例如架构文档的开发视点和架构文档的部署视点。例如，该架构相关帖子中的第一个回答[68] 提供了两个架构文档视点（即架构文档的开发视点和架构文档的部署视点），用于记录以 Java 实现的架构。

研究问题5的主要发现

发现 7：Stack Overflow 用户经常在评论串中使用两个与有用性相关的词语（即“useful”和“helpful”），以及其他六个词语（即“definitely”、“very”、“really”、“super”、“extremely”和“incredibly”），明确表达他们对特定架构解决方案有多大用处的反馈；这些解决方案是针对相应架构相关问题提出的。

发现 8：我们构建了一个有用架构解决方案分类体系，由 7 个类别、20 个子类别和 85 种类型组成，表明 SO 提供的有用架构解决方案具有多样性。

发现 9：架构配置解决方案（39%，324 个 ARP解决方案中的 127 个）、架构实现解决方案（18%，324 个 ARP解决方案中的 59 个）、架构说明（16%，324 个 ARP解决方案中的 53 个）和架构策略（11%，324 个 ARP解决方案中的 37 个）是讨论频率最高的四类有用架构解决方案。

4.6. 有用架构解决方案的特征（研究问题*6*）

如图 2所示，Stack Overflow 用户偶尔会在架构解决方案下方发表评论，以表示该解决方案有用。因此，我们开展了这项研究，以探讨被认为有用的架构解决方案的特征。与研究问题5类似，我们使用了 324 个具有有用知识的架构相关帖子（968 个架构相关帖子的子集）（见图 1），对架构解决方案及其附带的评论进行分析，并研究这些解决方案的特征。定性数据分析（见第 3.2.2节）确定了 Stack Overflow 用户认为有用的架构解决方案的四个共同特征。图 5展示了这些特征及其数量，其中 完整且全面的架构解决方案是 Stack Overflow 用户认为有用的架构解决方案中出现频率最高的特征（34%，324 个 ARP解决方案中的 111 个）。

（1）完整且全面的架构解决方案：开发人员可能会在 Stack Overflow 上的一个架构相关问题中提出多个问题（子问题）。如果某个解决方案涵盖了问题中的所有子问题，并针对这些子问题提供了全面的回答（例如，说明所提供架构解决方案的决策依据，如其优缺点），那么该解决方案就被认为是有用的。例如，某位开发人员在这条架构相关帖子[69] 中的第一个回答下发表了这条评论：“*+1* 这是我见过最完整、最全面、最有用的回答（.....）”。该回答全面涵盖了问题中的所有子问题（例如，难以将以 Java 和 Python 实现的系统架构中的数据可视化）。

（2）配有架构图的简明解释对架构解决方案的关键要素作出简要说明。这些关键要素的示例包括最佳

[67] https://tinyurl.com/4zm82snw
[68] https://tinyurl.com/2p8ezw7j
[69] https://tinyurl.com/dfw27h38

架构模式、策略和技术（例如数据库），以解决问题中描述的设计关注点。此外，提供架构图（如组件图）来呈现并概述解决方案的实际适用性，也有助于使架构解决方案被认为是有用的。例如，一位开发人员询问，在架构中，“命令处理器”和“命令总线”应属于应用层还是领域层，或应在哪一层实现。起初，在这篇架构相关帖子的第一个回答中[70]，，回答者提供了简洁且切题的解决方案。但提问者并不满意，于是发表评论，要求提供时序图（后来补上了），以便将其与解决方案结合起来，使方案更有用：“谢谢你，*David*。如果能分享一张时序图，那就太有帮助了。感激不尽”。

（3）详细架构解决方案：这些解决方案会提供并完整描述所有必要的架构元素（如模式、组件）以及在解决问题中提出的设计关注点时需要考虑的其他各种方面（例如解决方案权衡、约束和替代方案）。例如，一位开发人员在这篇架构相关帖子[71] 的第二个回答下发表了这条评论：“感谢您提供如此详细的回答。这确实很有帮助”。该回答列出并详细说明了在解决问题中提出的设计关注点（例如，将外部模块（外部 Web 应用程序）集成到 Drupal 中，或反向集成）时需要考虑的所有必要架构元素（例如质量属性）和其他方面（例如解决方案的优缺点以及替代解决方案）。

（4）外部相关内容摘要：提问者不喜欢回答仅将外部链接（URL）作为解决方案发布在问题下，因为链接可能失效，导致解决方案无法访问且毫无用处。在数据分析过程中，我们观察到，对于架构解决方案，提问者更希望看到 URL 中相关内容的摘要，而不是只有 URL。例如，这篇架构相关帖子[72] 的第一个回答总结了三个 URL 中的内容，以回答这个问题。该问题主要询问，为了解决 Cassandra 数据库的系统可用性问题，应采用何种设计方法。一位开发人员在该回答下评论道：“非常感谢您的信息。您的解释已经足够，您提到的链接也非常有用”。

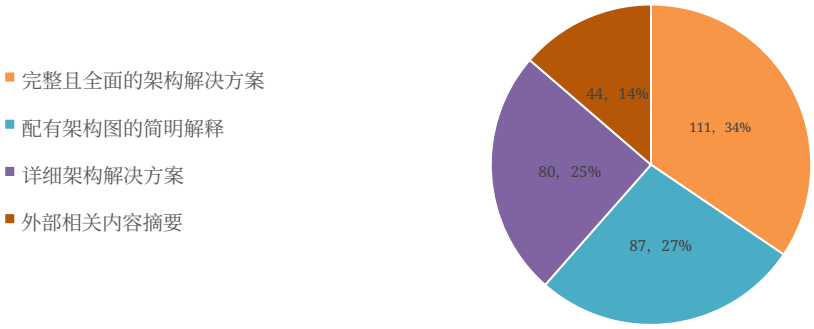


图 5：被认为有用的架构解决方案的常见特征

研究问题6的主要发现

发现 10：完整且全面的架构解决方案 是 Stack Overflow 用户认为有用的架构解决方案中最常见的特征（占 34%，即 324 个架构相关帖子解决方案中的 111 个）。

发现 11：所提供的架构解决方案中包含架构图（例如组件图），会增加这些解决方案被认为有用的可能性。

5. 讨论

在本节中，我们将重新审视本研究的发现，并解读第5.1 节中的结果，随后在第5.2节讨论其对各利益相关者的影响。

[70] https://tinyurl.com/yh292pn8
[71] https://tinyurl.com/pfezm7nn
[72] https://tinyurl.com/2p87vu99

5.1. 结果分析

5.1.1. 我们的结果与先前研究结果之间的差异

与我们的研究类似，已有多项研究分析了来自 SO 的架构相关帖子，以挖掘 Stack Overflow 用户讨论的架构知识，从而支持架构设计活动。本节将讨论我们的研究结果与以往研究（即 Soliman 的三项研究 等人 [6][51][10]）结果之间的联系与差异；这些研究与我们的工作密切相关。

Soliman 等人 [6]识别并分析了来自 SO 的架构相关帖子，这些帖子讨论了架构知识，重点关注技术决策（一种架构决策 [46]）。他们依据两个维度对这些架构相关帖子进行了分类： 问题目的 和 问题的解决方案类型。他们进一步将目的维度 分为三种子类型：解决方案综合、解决方案评估和多种目的，并将解决方案类型维度 分为三种子类型：技术特性、技术组合和架构配置。总体而言，他们的分析得出了 6 种架构相关帖子类型。我们的分析得出了 9 个类别和 21 个架构相关问题子类别（见表 5中的研究问题1结果），例如架构配置、架构决策、架构概念、架构实现、架构演化和架构重构。Soliman 等人 在 [6] 中发现的部分架构相关帖子类型（例如，解决方案综合、解决方案评估、架构配置）与我们的一些架构相关帖子类型相吻合，而 [6] 中介绍的大多数架构相关帖子类型都可以归入我们研究报告的主要类别的子类别。例如，我们设置了一个名为 架构决策的主要类别，该类别可涵盖 [6]报告的三种架构相关帖子类型（解决方案综合、解决方案评估和多种目的）。此外，我们的分析还得出了新的架构相关帖子类别（例如架构概念、架构工具、架构演化、架构重构、架构部署和架构文档）。

Soliman 等人 [51] 使用了与其先前研究中相同的架构相关帖子样本（即 [6]），并构建了一个涵盖 Stack Overflow 中架构知识概念的本体。该本体包含三个主要本体类： 简单本体类、复合本体类和词汇触发本体类。简单本体类由多个子类组成，例如技术方案、架构模式、质量属性、架构组件和架构连接器。复合本体类包含若干子类，例如架构配置、技术特性、技术优缺点、技术用例、用户请求和设计规则。词汇触发本体类包含一些子类，例如难度形容词、建议动词、价值形容词、愿望动词、支持动词和对比介词。在 [51], 的分析中发现的一些子类，例如架构配置和架构模式，与我们的研究问题5结果相符（见图4）。然而， [51] 的分析基于一组架构相关帖子样本，这些帖子主要讨论技术信息（例如技术解决方案的需求与约束、技术优缺点以及技术特性）。我们的分析在 [51] 的研究基础上新增了若干类别，例如架构策略、架构说明、架构文档解决方案和架构部署解决方案，从而更全面地涵盖了 Stack Overflow 中提供的架构相关帖子解决方案的类别和子类别。

Soliman 等人 [10] 开发了一种搜索方法，该方法依赖分类方法，为 Kazman 和 Cervantes 提出的每个设计步骤提供合适类型的架构相关帖子 [12]。Soliman等人 [10] 开展的分析也基于他们先前工作中的架构相关帖子样本 [6],以及从 SO 提取的其他一些帖子。具体而言，他们的搜索方法将 SO 帖子分为四种类型：技术识别、技术评估、特性与配置以及编程帖子。前三种帖子类型是他们先前研究中报告的架构相关帖子类型（即[6]），在本节第一段中，我们已经描述了这些架构相关帖子类型与我们研究中的架构相关帖子类型之间的异同（参见表5中研究问题1的结果）。

除了上述我们研究结果与[6][51][10], 中报告的结果之间的差异外，我们还探讨了一组新的研究问题（研究问题2、研究问题3、研究问题4、研究问题5和研究问题6）。我们探究了其他类型的架构知识，例如架构相关问题中讨论的设计情境（研究问题2）、架构相关帖子的特征（问题与解决方案）（研究问题3、研究问题4和研究问题6），以及架构相关帖子解决方案的实用性（研究问题5）；上述研究（即 [6][51][10] ）并未关注这些内容。此外，我们的分析涵盖了整篇帖子，包括问题及其相关评论（研究问题3、研究问题4），以及对问题的回答及其相关评论（研究问题5、研究问题6）。Soliman等人 上述研究中的分析仅关注问题和回答。因此，我们的研究结果为研究现状补充了新信息，从业者和研究人员可以从我们的研究结果和发现中受益（例如，被认为有用的架构解决方案分类体系）。

5.1.2. SO 中识别出的架构相关帖子的类别可支持架构设计活动

本研究的重要结果包括架构相关帖子的类别（问题（即研究问题1）和解决方案（即研究问题5））。本研究发现，Stack Overflow 用户会提出范围广泛（共九类）的架构相关问题，例如有关架构配置、架构决策、架构概念、架构实现和架构工具的问题（见第4.1节中的表5）。此外，我们将被认为有用的架构解决方案分为七类，例如架构配置解决方案、架构实现解决方案、架构说明和架构策略（见第 4.5 节中的图4）。值得注意的是，我们所确定的这些架构相关帖子（问题与解决方案）类别，几乎涵盖了系统生命周期中从架构创建初期阶段（例如，架构分析、综合和评估 [18]）到后期阶段（例如，架构实现以及维护与演化 [19]）的所有架构设计活动。因此，所确定的架构相关问题和解决方案类别能够在架构生命周期中支持上述架构设计活动。这些结果也支持了 Soliman 等人在 [6] 中的发现，即应将 Stack Overflow 视为架构知识的重要来源之一。此外，根据我们近期的工业调查 [4]，从业者表示，搜索架构信息时，问答网站（例如 Stack Overflow）最有用。因此，从业者可以借助 Stack Overflow，了解特定应用领域中架构解决方案的优缺点；例如，可以参考我们的分类体系（见图4）中关于架构实现的 iOS 应用框架优缺点的信息。

5.1.3. 设计上下文在架构设计中的重要性

研究问题2的结果表明，在所研究的架构相关问题中，大多数（71%，即 687 个，占 968 个）问题的 Stack Overflow 用户在描述架构相关问题中的设计关注点时，都会考虑设计情境（即关于系统预期运行环境的知识 [27]）（**第 2 项发现**，见第4.2节）。其中一个原因可能是，Stack Overflow 用户倾向于简要介绍项目背景，然后期望回答者根据给出的设计关注点和设计情境，提出潜在的架构解决方案及其决策依据。此外，研究问题2的结果表明，大多数 Stack Overflow 用户确实将设计情境视为推动系统架构设计不可或缺的要素之一 [27]。

5.1.4. 已识别的架构相关帖子的特征及其对提升质量的作用

根据表7、表8和图5，我们发现 Stack Overflow 上的架构相关帖子（问题和回答）具有多种特征。例如，我们观察到，清晰阐述架构信息的架构相关问题往往会获得多个回答（见表7），而缺少某些重要信息且结构不佳的架构相关问题（见表8）通常只会获得一个回答。原因如下：表述清晰的架构问题会概述计划中的系统，并清楚说明其设计关注点，这有助于潜在回答者充分理解问题的目的，从而给出回答。我们还发现，提问者高度认可完整且全面的架构解决方案，并认为这类方案很有用。一个原因是，这些架构解决方案会全面回应所有子问题中提出的设计关注点（如果一个问题包含子问题），例如提供设计情境以及所提架构解决方案的优缺点，从而帮助提问者理解为何这些架构解决方案会是现在的样子。对 Stack Overflow 上架构相关帖子（问题和回答）特征的识别表明，Stack Overflow 用户对架构相关帖子（问题和回答）的描述方式及详细程度有着不同需求。这些发现有助于提升 Stack Overflow 上所发布的架构相关问题和回答的质量。

5.2. 启示

5.2.1. 对 *Stack Overflow* 而言

提升 Stack Overflow 在用户中的认知度：Stack Overflow 将自身定位为一个社区问答平台，供用户在软件开发过程中提问和收集编程相关知识。因此，大多数 Stack Overflow 用户仅将该平台视为分享和学习编码相关知识的场所。然而，自该网站开始发展并广受欢迎以来，架构师便开始通过提出架构相关问题或提供架构解决方案（例如架构模式 [52]）来分享他们的能力、经验和架构问题。正如用户会搜索并（重新）使用 Stack Overflow 上现有的代码示例来解决编程相关问题一样，Stack Overflow 用户也会搜索并（重新）使用 Stack Overflow 上现有的架构解决方案（例如架构策略 [5] ），以解决架构设计关注点（例如满足质量属性的架构设计）。因此，Stack Overflow 不仅整理

编程相关知识，还积累了针对各种架构问题或设计问题提供的架构解决方案 [6, 5]。然而，在我们的研究中，我们发现，Stack Overflow 用户会将架构相关问题视为离题，认为不应在该网站上提出这些问题（见第8 节中的表 4.4）——这源于他们对 Stack Overflow 用途的认知（即，这是一个讨论编程相关问题的网站）。在这种情况下，人们提出的有趣架构相关问题可能得不到解答，甚至可能被网站版主删除。尽管一些 Stack Overflow 用户认为架构相关问题离题，但我们认为这类问题会持续存在，并在 Stack Overflow 上不断发展。根据我们研究的数据集（318 个架构相关问题），针对研究问题4（仅有一个答案的架构相关问题的特征），我们观察到，被评论为离题的架构相关问题并不多（318 个架构相关问题中有 16%，即 51 个）。这一发现表明，架构相关问题在 Stack Overflow 上具有良好的长期发展前景。此外，我们认为，传达架构知识 [23] 的架构相关问题是一类重要的问题，对软件开发具有系统层面的影响。许多架构相关问题是在开发过程中，为解决特定设计关注点（例如质量属性）及其权衡而产生的。因此，架构相关讨论（例如通过架构相关问题进行的讨论）不应被视为 Stack Overflow 上的离题内容；Stack Overflow 应考虑提升用户对其用途的认知度（即，将其定位为讨论开发相关问题的网站，而不仅仅是讨论编程相关问题的网站），并欢迎用户在网站上讨论架构问题。

调整当前的答案和评论组织机制，以改进架构解决方案的搜索和（再）利用：SO 吸引了大量具有不同背景、技能、专业知识和观点的用户。因此，在我们的数据分析过程中，我们观察到，SO 中的架构相关问题与其他问题（例如编程相关问题）一样，可能会收到多个（或替代性的）回答。Wang 的研究等人 [53] 报告称，近 650 万个问题（占 SO 中所有问题的 37%）拥有多个回答，而回答的平均长度为 789 个字符。在当前 SO 的回答组织机制下，当一个帖子中的某个问题有多个回答时，提问者最多只能接受或标记其中一个回答，以表明该回答是最有用的回答 [3]。该提问者必须是 SO 上拥有至少 15 点声誉的注册用户 [54]。网站上不具备所需声誉（即声誉少于 15 点）的注册用户，无法接受回答，也无法通过投票（赞成票或反对票）来表明这些回答是有用的 [54]。因此，SO 中有大量回答虽然尚未被接受或标记为有用回答，却确实具有实用价值，仅仅因为用户（提问者）不具备执行这些操作所需的最低声誉。在我们的数据分析过程中，我们观察到，并非所有有用的架构解决方案都在 SO 中被明确标记（即被接受为有用回答），从而不利于这些解决方案的搜索和（再）利用（例如，参见图 2）。此外，我们发现，SO 用户可能会在评论串中使用“有用”和“有帮助”等与有用性相关的词语，明确表达他们对 SO 中针对相关问题所提供的某些架构解决方案有用程度的反馈（**发现 7**，见第 4.5节）。我们的发现与 Zhang 等人在 [42] 和 [41] 中的研究发现一致，即评论提供了支持回答的补充信息，例如对回答的改进 [42]以及已过时的回答 [41]。以往研究批评了 SO 的评论组织机制（例如，[55]）。为了使每个回答帖保持紧凑，SO 采用评论组织机制，仅显示排名前 5 的评论 [55]。为了展示信息最丰富的评论并隐藏信息较少的评论，该机制首先根据评论的评分对其进行排序。当多个评论的评分相同时，则按照评论的创建时间进行排序 [55]。隐藏的评论不会被 Google 编入索引[73]。因此，由于当前的评论排序机制，信息丰富的评论可能会被隐藏，从而降低人们检索或为其投票的机会。尽管 SO 取得了成功并广受欢迎，但浏览 SO 仍然是一项挑战；SO 如何引导用户检索信息丰富的评论仍不够充分 [56]。阐述回答（包括架构解决方案）有用性的评论，是最重要的信息丰富评论之一。因此，我们向 SO 提出以下建议：

- 评论组织机制不应仅按评分、再按创建时间对评论进行排序 [55], ，而应提高信息丰富的评论的优先级。SO 可以考虑调整其评论组织机制，例如开发专门的分析技术（如机器学习方法），以筛选和排列评论

[73] https://tinyurl.com/2p87yyfr

例如，指出回答的改进之处 [42], 回答的有用性（例如，有用的架构解决方案）。

- Stack Overflow 用户也可以参考并扩展我们提出的有用架构解决方案（被评论为有用的解决方案）分类体系，开发一种自动化工具，帮助 Stack Overflow 用户识别现有架构解决方案中的有用知识，例如。

5.2.2. 面向 Stack Overflow 用户

在对研究问题3、研究问题4和研究问题5进行定性分析的过程中，我们确定了 Stack Overflow 中架构相关帖子（问题和解决方案）的各种特征。在这些特征中，我们发现，提供清晰描述并附有架构图的问题更有可能获得多个回答（见表7），而结构不佳的架构问题（见表8）往往只能获得一个回答。此外，我们发现，提供简洁解释并附有架构图是被认为有用的架构解决方案中第二常见的特征（见图5）。一项观察结果是，Stack Overflow 用户希望在问题和解决方案中都能看到架构图，例如组件图，因为这些图对双方都有帮助。对于问题，提供架构图会提高获得更多回应（例如多个回答）的可能性（**研究发现5**，见第4.3节）。另一方面，解决方案中的架构图会提高其被认为有用的可能性（**研究发现11**，见第4.6节）。因此，提问者和回答者都应在其架构相关帖子（问题和回答）中尽量提供图示。原因之一是，架构的抽象层次较高，仅用文本描述架构问题就很困难，而解释架构解决方案则更加困难。架构图能让架构更易于理解 [57], ，利益相关者也能借助架构图更轻松地交流架构问题和解决方案。此外，本研究发现的架构相关帖子的各种特征（见表7、表8和图5）表明，Stack Overflow 用户在架构相关问题和架构解决方案的构成及详细程度方面有着不同需求。因此，有必要为 Stack Overflow 用户提供指南，供他们发布架构相关问题和解决方案时遵循。

面向 Stack Overflow 提问者：下文将为 Stack Overflow 提问者提供一些指南，供他们在发布架构相关问题时遵循，以提高其他 Stack Overflow 用户作答的可能性，并获得多个 Stack Overflow 用户的回答：

- 在问题中附上说明清晰的架构图：我们建议提问者在架构相关问题中添加架构图（例如组件图），并明确说明设计关注点，以帮助其他 Stack Overflow 用户更好地理解问题的目的。

- 撰写表述清晰的架构问题，并详细说明上下文：我们建议提问者在问题中清晰地描述架构信息。例如，可以概述系统的整体情况，并详细说明其关注范围内的组件，以及这些组件与其他组件之间的接口和关系。此外，我们建议提问者补充设计情境信息，因为设计情境对于其他 Stack Overflow 用户正确理解架构相关问题至关重要。

面向 Stack Overflow 回答者：正如本研究全文所述，我们不仅分析了架构相关问题，还考察了 Stack Overflow 用户认为有用的架构解决方案的特征。因此，下文将为 Stack Overflow 用户提供一些指南，供他们在发布解决方案时遵循，以提高其解决方案被其他 Stack Overflow 用户认为有用的可能性：

- 使用架构图撰写简洁的架构解决方案：建议回答者在解决方案中只陈述要点，撰写简洁的架构解决方案，并在必要时添加架构图，以呈现和阐明例如架构实现视图等内容。

- 在架构解决方案中附上 *URL*，并提供充分且相关的架构知识：提问者不喜欢回答者仅将外部链接（URL）作为解决方案发布在问题下方 [58]。如果回答者想让架构解决方案简短一些，可以提供指向外部网站的链接，以便读者获取更多说明或复杂示例，但其解决方案本身应当完整自洽。换句话说，解决方案应提供某些重要且相关的架构知识，使其具备可解释性，例如设计决策及其决策依据，

上下文、假设以及其他共同决定为何采用特定解决方案的因素。

5.2.3. 面向研究人员

开发创新工具，以搜索和（重）用 SO 中的架构知识：我们的研究结果（例如，第 4.1 节中的架构相关问题类别，以及第4.5节中的有用解决方案）揭示了 Stack Overflow 用户在 SO 中关于架构设计的讨论性质。此外，本研究结果再次印证了 Soliman等人的结论： [51] SO 应被视为架构知识的重要来源之一。然而，SO 的帖子中包含大量信息，而这些信息主要以非结构化文本的形式呈现。此外，架构概念的抽象性使基于关键词的搜索难以找到相关的架构信息，因此 SO 用户可能难以从 SO 中获取并（重）用架构知识（例如，在特定应用领域使用特定架构模式的优点、缺点和权衡）。因此，研究人员可以聚焦于创新技术和工具，以促进 SO 中架构知识的搜索与（重）用；这些技术和工具能够高效、有效地引导人们获取和使用这类知识，从而支持架构设计活动（例如架构分析与综合 [18]）。例如，研究人员可以参考我们提出的 SO 有用架构解决方案分类体系，以此为指导，开发自动化方法和工具，从而挖掘并定位架构解决方案（例如，架构配置解决方案，这是 SO 中最常见的有用架构解决方案类别，见图 4），以解决类似的设计关注点（例如，询问有关架构配置的问题，见表5）。这有助于 SO 用户查找与其设计关注点（例如银行系统配置）相关的问题和解决方案。此外，我们观察到，架构配置（27%）、架构决策（19%）和架构概念（15%）是最常被提出的架构相关问题中的三大类别（**第 1 项发现**，见第 4.1 节）；研究人员可以进一步探索与这些最常见的架构相关问题类别有关的挑战（即 Stack Overflow 用户面临的挑战）。

研究问答网站中的设计情境，以支持架构知识管理：根据第6 节表4.2，我们发现，Stack Overflow 用户在 Stack Overflow 上提出架构相关问题时，会同时讨论设计情境和设计关注点。我们从所研究的架构相关帖子样本中识别出设计情境的三个类别（应用程序、平台和组织情境）以及八个子类别（应用领域、外部服务、软件、硬件、开发进度、利益相关者和资源情境）（见表6）。虽然我们知道，Stack Overflow 用户在提出架构相关问题时会同时讨论设计情境和设计关注点，但针对问答社区网站（如 Stack Overflow）中设计情境挖掘的研究仍然很少，而这有助于支持架构知识管理 [59], ，因此这是未来研究值得探索的一个有趣领域。

6. 效度威胁

本节遵循 Wohlin等人提出的指南，讨论研究结果的效度威胁，以及我们在研究中如何缓解这些威胁。

内部效度涉及用于检索 Stack Overflow 中架构相关帖子的搜索词选择。我们使用了“architecture”和“architectural”等搜索词来识别 Stack Overflow 中的相关帖子（见第3.2节）。这会对本研究的内部效度构成威胁，因为我们可能遗漏了 Stack Overflow 用户用来表达架构概念的其他词语，例如“design”。因此，本研究使用的搜索词可能无法识别 Stack Overflow 中的全部架构相关帖子。为降低这一威胁，我们首先进行了试点搜索，并发现 Stack Overflow 用户使用“design”一词时，大多处于编程语境中（例如“singleton design pattern”[74]）。此外，如第3.2节所述，仅通过标签使用搜索词（例如“architecture”和“architectural”）进行搜索可能效果不佳，因为标签有时提供的信息较少 [36] （见第3.2.1节中的示例）。因此，我们决定将问题的标题和正文也纳入搜索，以尽量降低因标签不正确或不相关而遗漏架构相关帖子的风险。最后，我们通过搜索收集了 10,423 篇架构相关帖子，构成了一个相当大的数据集。若要依靠人工彻底分析如此规模的数据集，以获得准确而全面的结果，可能并不现实。因此，

[74] https://tinyurl.com/8yks7nhm

针对该数据集。为此，我们计算了这 10,423 篇架构相关帖子的具有统计代表性的样本 [16] ，并随机抽取 968 篇架构相关帖子，作为本研究中待分析的数据集。然而，为了进一步缓解这一威胁，我们下载并使用了最新的 Stack Overflow 数据转储（即 10 月 5, 2022[75]日的 Stack Exchange 数据转储）。该数据转储是 Stack Overflow 底层数据库的快照，其中以 XML 文件的形式存储了有关问题、回答、标签、评论、投票和用户历史记录的全部信息（例如， `Posts.xml`）。我们使用了 `Posts.xml` 文件，其中存储了所有 Stack Overflow 帖子的提问和回答，并以此为基础估算由于本研究将搜索范围限制在“architect*”搜索词而遗漏的架构相关帖子数量。根据 2022 年 10 月 5 日的 Stack Overflow 数据转储，共有 2,300 万个问题（帖子）和 3,400 万个回答。随后，我们利用统计功效计算出这 2,300 万个帖子的代表性样本量。在 95% 的置信水平和 3% 的误差幅度下，计算得出的代表性样本量为 1,069 个帖子。之后，我们从这 2,300 万个帖子中随机抽取 1,069 个，并对其进行人工检查，以计算搜索架构相关帖子时因将搜索范围限制在“architect*”搜索词而可能遗漏的帖子数量。具体而言，第一作者对这 1,069 个帖子进行了标注，以确定其中哪些属于架构相关帖子。第二作者检查并验证了标注结果。研究人员通过会议解决分歧，以提高标注结果的可靠性。根据人工标注结果，我们发现，在这 1,069 个帖子中，只有 21 个是架构相关帖子（即真阳性）；其中，14.3%（即 21 篇架构相关帖子中的 3 篇）不含“architect*”搜索词，85.7%（即 21 篇架构相关帖子中的 18 篇）含有“architect*”搜索词。因此，我们承认，可能遗漏了一定数量（即 14.3%）不含“architect*”搜索词的架构相关帖子。我们在复现包中加入了 [39] 随机抽取的帖子（即 1,069 个帖子）及其标注结果（即 18 篇含有“architect*”搜索词的架构相关帖子，以及 3 篇不含“architect*”搜索词的架构相关帖子），以供复现。

构念效度是指研究对其声称要测量的内容进行测量的程度[60]。本研究的构念效度面临的一项威胁与对所选 Stack Overflow 帖子的人工分析有关。这是因为人工分析可能因多种解读和／或疏忽而引入个人偏见。为缓解这一威胁，我们采用了扎根理论中的两种定性技术（开放式编码和持续比较） [17] 来分析提取的数据并回答研究问题。此外，我们在正式数据编码之前进行了试点数据编码，以尽量减轻这一威胁。如第3.2.2节所述，在试点数据编码期间，第一作者随机选取了 100 个架构相关帖子，并根据各研究问题的目的（见表4）对提取的数据进行编码（见表1）。我们与第二作者安排了数次线下会议，以解决第一作者在试点数据编码期间遇到的任何疑惑。此外，本研究的其他三位作者（第二、第三和第四作者）对试点数据分析所得的最终结果（即概念、类别和子类别）进行了检查和验证。我们在一次会议上采用协商一致法解决了分歧 [45] ，从而提高试点数据分析结果的可靠性。另一项威胁与通过检查帖子所附的评论，识别包含有用知识的架构相关帖子（解决方案）有关，以回答研究问题5和研究问题6（见第3节的第二阶段）。为缓解这一威胁，如第3节所述，我们并未依据评论中“有用”等词语（及类似说法）的出现频率，来衡量本研究中针对某个架构相关问题给出的架构解决方案的有用性。相反，我们参考解决方案所附评论中与有用性相关的信息，以考察 Stack Overflow 用户对架构解决方案有用性的讨论。换言之，我们根据 Stack Overflow 用户看到并使用架构解决方案后的反应来判断其有用性（见图2中的一条评论）。此外，在将此类包含有用知识的帖子（解决方案）纳入分析之前，我们（本研究的四位作者）先阅读了研究代表性样本中被评论为有用的解决方案，以确认它们是否确实有助于解决相关问题 [15] 。因此，我们认为已充分缓解了这一威胁。

外部效度 是指研究结果能够推广到其他情境的程度 [60]。本研究仅将 SO 作为研究架构相关帖子及其有用性的来源。尽管 SO 是一个广泛使用且颇受欢迎的开发者问答网站，但仅依赖这一独特来源仍会对研究结果的多样性构成威胁。为减轻这一威胁，我们可以通过纳入更多来源（例如 GitHub），并查看架构相关问题，进一步完善研究，以了解架构师和开发人员面临的架构设计问题。此外，研究人员还可以深入实际工作场景，直接向架构师和开发人员征求反馈，以更好地了解他们在架构设计方面面临的问题，以及哪些架构解决方案可被视为有用。

[75] https://archive.org/details/stackexchange_20221005

可靠性 是指其他研究人员复现研究时，研究能否得出相同的结果和发现 [60]。在本研究中，这一威胁主要与手动数据收集和分析的过程有关。为减轻这一威胁，我们（本研究的作者）遵循了严格的流程，其中包括数据收集和分析活动（见第 3.2 节）。此外，我们邀请本研究的四位作者交叉核查分类和特征描述阶段的结果。为确保结果和发现的可靠性，我们公开提供了一个复现包，其中包含本研究使用的数据集和生成的编码数据 [39], ，以便其他研究人员评估设计的严谨性并复现本研究。采取这些措施后，这一威胁在一定程度上得到了减轻。

7. 相关工作

与本研究相关的工作来自调查问答网站（如 Stack Overflow）中软件开发知识的研究。本节将相关工作归纳为两类：（1）对问答网站中架构知识的研究；（2）对问答网站中知识的质量评估。

7.1. 问答网站中的架构知识

现有少数研究从不同角度探讨了 Stack Overflow 的架构相关帖子所提供的架构信息。Bi等. [5] 采用半自动的基于词典的挖掘方法，从 Stack Overflow 帖子中提取与质量属性（QA）和架构策略（AT）相关的讨论。具体而言，他们应用基于词典的分类器支持向量机（SVM），自动识别 Stack Overflow 帖子中与质量属性和架构策略相关的讨论。此外，作者还手动整理了实践中使用的架构策略（AT）与质量属性（QA）之间的设计关系，并根据相关讨论构建了知识库，说明开发人员如何针对质量属性相关问题使用架构策略。这类知识有助于架构师更好地制定架构策略的设计决策。Chinnappan等.[61] 从五个开放的软件代码库来源提取数据，其中包括 Stack Overflow，并定性挖掘了从业者在实际机器人项目中应用的高能效机器人软件架构策略。为促进这些策略的适用性（包括在机器人软件社区之外的适用性），他们借鉴 UML 组件图和时序图表示法，以通用且独立于具体实现的方式描述这些策略。所提出的能耗感知策略可为机器人学家以及其他有意进行能耗感知软件架构设计与实现的开发人员提供指导。Soliman等. [11] 对 50 名软件工程师开展了一项实证研究；这些工程师使用 Google，按照属性驱动设计的步骤 [12]制定设计决策。他们根据软件工程师指定的相关性和架构知识（AK）概念，评估了网络搜索引擎从各种来源（包括 Stack Overflow）查找相关架构信息并获取架构知识的效果。在另一项工作中，Soliman等. [51] 开发了一个涵盖 Stack Overflow 中架构知识概念的本体。该本体对架构相关信息进行了描述，以表示和组织 Stack Overflow 中的架构知识。

Soliman 等人。 [10] 也利用 Stack Overflow，旨在改善架构师在在线开发者社区中搜索架构相关信息的方式。他们开发了一种新的搜索方法（即领域特定搜索方法），利用 Stack Overflow 搜索架构相关信息。他们发现，这种新搜索方法优于传统的基于关键词的搜索方法（通过 Google 等搜索引擎进行搜索）。Tian等人。 [14] 通过分析 Stack Overflow 中架构异味相关帖子的讨论，对 Stack Overflow 用户对架构异味的理解开展了一项实证研究。他们发现，Stack Overflow 用户经常用一些笼统的词语来描述架构异味，例如“糟糕”“错误”“脆弱”，或违反架构模式。Li 等人[62] 从八个最受欢迎的在线开发者社区（包括 Stack Overflow）中提取数据，以调查开发人员如何理解架构侵蚀这一概念、其原因和后果，以及识别和控制架构侵蚀的工具和实践。在研究报告的其他主要发现中，Li 等人发现，开发人员要么关注架构侵蚀的结构性表现，要么关注其对运行时质量属性、维护和演化的影响；除了技术因素，架构侵蚀在很大程度上也是由非技术因素造成的。Zou 等人 [63] 使用主题模型技术研究 Stack Overflow 帖子文本内容中与非功能需求相关的内容，以了解开发人员的实际需求。本研究对上述工作进行了补充，因为我们通过对架构相关帖子进行特征描述与分类，调查 Stack Overflow 中的架构知识。此外，我们还从 Stack Overflow 用户的角度考察这些帖子的有用性（即回答是否有助于解决问题? [15]）。

与本研究密切相关的是 Soliman 等人的研究。 [6], 该研究利用 Stack Overflow，根据帖子中提供的技术相关信息对架构相关帖子进行分类。本研究与他们工作的主要区别在于，我们以更广泛的视角审视这些问题。换言之，我们旨在通过考察帖子中提供的各种架构信息（例如架构策略），对 Stack Overflow 中的架构相关帖子进行分类，而不局限于某一种特定信息（即技术信息）。因此，我们通过深入探讨架构相关帖子（例如，识别架构相关帖子的其他类别，并探索架构相关问题的设计情境），对 [6] 中的工作进行了补充。此外，我们还对这些帖子的有用性进行了特征描述与分析，以供从业者和研究人员参考。

7.2. 问答网站中的知识质量评估

问答平台（如 Stack Overflow）在知识共享中发挥着重要作用；然而，如何确保其知识质量仍面临重大挑战。越来越多的研究从不同角度（如代码和文本）分析 Stack Overflow 编程相关帖子的内容质量，这充分体现了这一点。

Dagenais 和 Robillard [64]对 Stack Overflow 中 API 与其学习资源之间的可追溯性链接开展了一项实证研究。他们发现，在线论坛代码片段中的大多数 API 名称（89%）都含义模糊，而且由于代码片段不足，难以轻松解决相关问题。An 等人 [65] 研究了 399 个 Android应用程序，并发现 GitHub 与 Stack Overflow 之间存在 1,279 起代码复用版权侵权案例。Fischer 等人 [66] 评估了 Stack Overflow 代码片段中的安全相关问题，发现其中 29% 存在安全隐患。Zhang 等人 [41] 调查了 Stack Overflow 中的答案过时问题，发现其中 31% 可能存在潜在的 API使用违规，进而导致系统崩溃和资源泄漏等意外行为。Ragkhitwetsagul 等人 [67] 调查了 Stack Overflow 中的 Java代码片段，发现有 153 个克隆被复制到 Stack Overflow，其中 66% 已经过时。Zagalsky 等人 [68]推出了 Example-Overflow，这是一款利用 Stack Overflow 中的知识推荐高质量代码的代码搜索和推荐工具。Zhang 等人 [8] 对 Stack Overflow 中 API 误用的普遍程度和严重性开展了一项探索性研究。Treude 和 Robillard [69]对 GitHub用户进行了调查，以全面研究 Stack Overflow 中代码片段的理解难度。他们发现，不到一半的 Stack Overflow 代码片段能够自解释。Ragkhitwetsagul 等人 [70] 开展了一项在线调查，研究 Stack Overflow 中的答案过时问题。调查结果表明，半数顶级答主意识到代码示例已经过时。然而，用户很少、甚至从不修复过时代码示例。Treude 和 Robillard [71]开发了一款工具，利用从 Stack Overflow 中提取的“洞察句”改进 API文档。Wong 等人[72] 提出了 AutoComment 工具，可自动为问答网站中带有 Java 和 Android 标签的问答帖子生成评论。结果表明，准确、充分、简洁且实用的生成式评论有助于用户理解代码。Gao 等人 [73] 调查了具有相似崩溃跟踪信息的问题，以自动修复问答网站中反复出现的崩溃漏洞。McDonnell 等人 [74]利用 GitHub 中的版本历史数据，调查了 Android生态系统中的 API演化。结果显示，Android 的演进速度很快，平均每月更新 115 个 API。Dalip等人 [75]提出了一种根据回答所获反馈为回答排序的方法。他们发现，用户特征和评价特征对于评估回答质量都至关重要。Xu等人 [76] 提出了一种名为 AnswerBot 的方法，可自动总结 Stack Overflow 中与技术问题相关的回答帖子。Zhu 等人 [15] 提出了一种多维模型，用于评估电子学习背景下社交问答网站（如 Answerbag 和 Yahoo! Answers）中回答的质量。Calefato 等人 [77]开展了一项实证研究，旨在评估 Stack Overflow 中的 26 种最佳答案预测模型。

这些研究与我们的工作有关，因为它们考察了 Stack Overflow 帖子中所提供代码示例的质量，而我们的工作考察的是 Stack Overflow 帖子中所提供架构解决方案的质量（即有用性）。尽管如此，我们的工作与上述研究不同：它们关注低层级的源代码（例如 API），而我们的研究关注高层概念（例如，将提出的架构模式作为解决设计关注点的方案），以考察其有用性。我们认为，通过分析架构相关帖子，我们的研究补充了有关 Stack Overflow 帖子质量的现有研究。

据我们所知，尚无研究从 Stack Overflow 用户的角度，考察架构相关帖子中所提供的架构信息的类别、特征及其有用性（即这些回答是否有助于解决问题）；这正是本研究的关注重点。

8. 结论与未来工作

研究在线开发者社区（如 Stack Overflow）中提供的架构解决方案（例如架构策略和模式）这一重要类型的架构知识至关重要，因为这类知识是最重要的开发知识之一 [22]。架构知识在开发过程中帮助架构师和开发人员做出明智的架构设计决策方面发挥着重要作用 [24]。架构解决方案是现代软件设计的基本构件[22]。与实现变更（例如低级源代码）不同，架构解决方案（例如架构模式）一经采用并实现，就很难且代价高昂地进行更改[22]。通过分析和了解 Stack Overflow 用户如何在 Stack Overflow 等在线开发者社区中处理架构问题，可以获得三项优点：（1）深入了解 Stack Overflow 用户在架构设计过程中面临的设计问题类型、所讨论的架构解决方案类型及其有用性；（2）帮助了解提出架构问题时所处的设计情境；（3）帮助了解所讨论的架构问题和解决方案的特征。这些优点为开发新的方法和工具创造了机会，使其能够帮助 Stack Overflow 用户搜索和（重新）利用在线开发者社区中分享的架构知识。为此，本研究考察了 Stack Overflow 上的架构相关问题及其相关架构解决方案。具体而言，我们采用定性分析方法，对从人工识别的 10,423 条架构相关帖子中抽取的、具有统计代表性的 968 条随机样本进行了分析。我们旨在识别架构相关问题及其解决方案的类别和特征。我们还探究了提出这些问题时所处的设计情境。最后，我们研究了 Stack Overflow 用户关于架构解决方案有用性的讨论。我们的主要结果和发现总结如下：

- Stack Overflow 用户提出了范围广泛的架构相关问题，涉及架构工具到架构配置、架构实现到架构部署等方面。此外，Stack Overflow 用户主要讨论架构配置解决方案（39%），其次是架构实现解决方案（18%）、架构说明（16%）和架构策略（11%）。我们观察到，架构相关帖子（问题和回答）几乎涵盖了所有架构设计活动。
- Stack Overflow 用户提出的架构相关问题中，关于架构配置的问题最多（27%，968 个问题中有 261 个）。
- 大多数 Stack Overflow 用户（占 71%，即 968 个架构相关问题中的 687 个）在提出架构相关问题时考虑了设计上下文。
- 提供清晰描述及架构图的架构相关问题，更有可能获得多个回答；而结构不佳的架构问题往往只能获得一个回答。
- 在我们提出的分类体系中，面向配置的架构解决方案是 Stack Overflow 上最常见且被认为有用的架构解决方案类型。
- Stack Overflow 用户主要认为，架构解决方案应当完整且全面，并且应有配有架构图的简明解释，才有帮助。

我们的结果和发现有助于研究人员和从业者了解 SO 提供了哪些类型的架构知识，例如架构相关问题和解决方案的类别，以及优质的架构相关问题和有用的架构解决方案具有什么特点。此外，我们的结果可以促使研究人员和从业者将 SO 视为宝贵的架构知识来源（例如架构模式和策略），并开发新的方法和工具，从 SO 中挖掘有用的架构知识，以支持架构设计活动和开发。

下一步：（1）我们计划对 Stack Overflow 和其他平台（如开发者邮件列表和问题跟踪系统）提供的架构解决方案开展比较研究，这有助于深入了解当前架构解决方案应用的重点及其优缺点。（2）我们希望从从业者的角度，通过工业调查验证并扩展 Stack Overflow 上有用架构解决方案的分类体系（见图 4）。（3）我们还计划设计并采用（半）自动化方法，从检索到的架构信息中提取和总结架构信息，并建立架构问题—解决方案对。例如，从多个架构信息来源（如问答网站、GitHub、问题跟踪系统、技术博客）中收集特定任务的架构问题对应的某些架构解决方案（如模式和策略）的优缺点，从而利用同行和社区中的架构知识，帮助架构师进行决策。

致谢

本研究部分由国家自然科学基金委员会（NSFC）项目（编号：62172311）资助。作者还感谢国家留学基金管理委员会提供的经费支持。

参考文献

[1] C. Sadowski, K. T. Stolee, S. Elbaum，开发人员如何搜索代码：一项案例研究，载于：第10届欧洲软件工程大会与 ACM SIGSOFT 软件工程基础研讨会联合会议论文集（ESEC/FSE），意大利贝加莫，2015年，页码 191–201。[2] A. Zagalsky, D. M. German, M.-A. Storey, C. G. Teshima, G. Poo-Caamaño，R 社区如何创建和整理知识：对 Stack Overflow 和邮件列表的扩展研究，《实证软件工程》23 (2) (2018) 953–986。[3] C. Treude, O. Barzilay, M.-A. Storey，程序员如何在 Web 上提问和回答问题？（NIER 专题），载于：第33届国际软件工程大会（ICSE）论文集，美国夏威夷州檀香山，2011年，页码 804–807。[4] M. J. de Dieu, P. Liang, M. Shahin，开发人员如何搜索架构信息？一项工业调查，载于：第19届国际软件架构会议（ICSA）论文集，美国夏威夷州檀香山，2022年，页码 58–68。[5] T. Bi, P. Liang, A. Tang, X. Xia，在 Stack Overflow 上挖掘架构策略与质量属性知识，《系统与软件杂志》180 (2021) 111005。[6] M. Soliman, M. Galster, A. R. Salama, M. Riebisch，开发者社区中用于技术决策的架构知识：一项基于 StackOverflow 的探索性研究，载于：第13届 IEEE/IFIP 软件架构工作会议（WICSA）论文集，意大利威尼斯，2016年，页码 128–133。[7]T. Diamantopoulos, A. Symeonidis，利用源代码信息改进 Stack Overflow 上的问题回答，载于：第12届 IEEE/ACM 软件仓库挖掘工作会议（MSR）论文集，意大利佛罗伦萨，2015年，页码 454–457。[8] T. Zhang, G. Upadhyaya, A. Reinhardt, H. Rajan, M. Kimm，在线问答论坛上的代码示例可靠吗？一项关于 Stack Overflow 上 API 误用的研究，载于：第40届 IEEE/ACM 国际软件工程大会（ICSE）论文集，瑞典哥德堡，2018年，页码 886–896。[9] D. Liu, Z.-L. Ren, Z.-T. Long, G.-J. Gao, H. Jiang, 挖掘设计模式的使用场景及相关设计模式对：一项基于在线帖子的案例研究，《计算机科学与技术学报》35 (5) (2020) 963–978。[10] M. Soliman, A. R. Salama, M. Galster, O. Zimmermann, M. Riebisch, 改进在线开发者社区中的架构知识搜索，载于：第15届 IEEE 国际软件架构会议（ICSA）论文集，美国华盛顿州西雅图，2018年，页码 186–195。[11] M. Soliman, M. Wiese, Y. Li, M. Riebisch, P. Avgeriou，探索网络搜索引擎以查找架构知识，载于：第18届 IEEE 国际软件架构会议（ICSA）论文集，德国斯图加特，2021年，页码 162–172。[12] H. Cervantes, R. Kazman，《软件架构设计：实用方法》，Addison-Wesley Professional，2016年。[13] I. Malavolta, K. Chinnappan, S. Swanborn, G. A. Lewis, P. Lago，挖掘 ROS 生态系统中的机器人学绿色架构策略并进行实证评估，载于：第18届 IEEE/ACM 软件仓库挖掘国际会议（MSR）论文集，西班牙马德里，2021年，页码 300–311。[14] F. Tian, P. Liang, M. A. Babar，开发人员如何讨论架构异味？一项基于 Stack Overflow 的探索性研究，载于：第16届 IEEE 国际软件架构会议（ICSA）论文集，德国汉堡，2019年，页码 91–100。

[15] Z. Zhu, D. Bernhard, I. Gurevych，用于评估社交问答网站回答质量的多维模型，载于：第14届国际信息质量会议（ICIQ）论文集，德国波茨坦，2009年，页码 264–265。[16] G. D. Israel，确定样本量，事实表 PEOD-6，美国佛罗里达州（1992年）。[17] K. S. P. Ralph, F. Brian，软件工程研究中的扎根理论：批判性综述与指南，载于：第38届 IEEE/ACM 国际软件工程大会（ICSE）论文集，美国德克萨斯州奥斯汀，2016年，页码 120–131。[18] H. Christine, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, P. America，源自五种工业方法的软件架构设计通用模型，《系统与软件杂志》80 (1) (2007) 106–126。[19] A. Tang, P. Avgeriou, A. Jansen, R. Capilla, M. A. Babar，架构知识管理工具比较研究，《系统与软件杂志》83 (3) (2010) 352–370。[20] C. Hofmeister, P. Kruchten, R. L. Nord, H. Obbink, A. Ran, P. America，源自五种工业方法的软件架构设计通用模型，《系统与软件杂志》80 (1) (2007) 106–126。[21] Z. Li, P. Liang, P. Avgeriou，基于知识的方法在软件架构中的应用：一项系统性映射研究，Information and Software Technology 55 (5) (2013) 777–794。[22] L. Bass, P. Clements, R. Kazman，《实践中的软件架构》，第3版，Addson-Wesley Professional，2012年。[23] C. Rafael, A. Jansen, A. Tang, P. Avgeriou, M. A. Babar，软件架构知识管理十年：实践与未来，《系统与软件杂志》116 (2017) 191–205。[24] A. Jansen, J. Bosch，作为一组架构设计决策的软件架构，载于：第5届 IEEE/IFIP 软件架构工作会议（WICSA）论文集，美国宾夕法尼亚州匹兹堡，2005年，页码 109–120。[25] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, A. Tang，工业界对架构语言的需求：一项调查，《IEEE 软件工程汇刊》39 (2013) 869–891。[26] T. Bi, W. Ding, P. Liang, A. Tang，两个 OSS 项目中的架构信息交流：原因、参与者、时间和内容，《系统与软件杂志》181 (2021) 111035。[27] A. Bedjeti, P. Lago, G. A. Lewis, R. D. D. Boer, R. Hilliard，利用架构视点对上下文建模，载于：第14届 IEEE 国际软件架构会议（ICSA）论文集，瑞典哥德堡，2017年，页码 117–120。[28] A. Tang, F.-C. Kuo, M. F. Lau，迈向独立的软件架构评审，载于：第2届欧洲软件架构会议（ECSA）论文集，塞浦路斯帕福斯，2008年，页码 306–313。[29] K. E. Harper, J. Zheng，探索软件架构上下文，载于：第12届 IEEE/IFIP 软件架构工作会议（WICSA）论文集，加拿大魁北克省蒙特利尔，2015年，页码 123–126。[30] K. Petersen, C. Wohlin，工业软件工程研究中的上下文，载于：第3届国际实证软件工程与度量研讨会（ESEM）论文集，美国佛罗里达州布埃纳维斯塔湖，2009年，页码 401–404。[31] I. Groher, R. Weinreich，情境中的架构决策研究，载于：第12届 IEEE/IFIP 软件架构工作会议（WICSA）论文集，加拿大魁北克省蒙特利尔，2015年，页码 11–20。[32] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, P. Sommerlad, M. Stal，《模式导向的软件架构》，第1卷，John Wiley & Sons，1996年。[33] V. R. Basili, G. Caldiera, H. D. Rombach，目标—问题—度量方法，《软件工程百科全书》（1994）528–532。

[34] M. Soliman, M. Riebisch, U. Zdun，利用技术设计决策丰富架构知识，载于：第12届 IEEE/IFIP 软件架构工作会议（WICSA）论文集，加拿大魁北克省蒙特利尔，2015年，第135–144页。[35] F. Calefatoa, F. Lanubileb, N. Novielli，如何寻求技术帮助？在 Stack Overflow 上撰写问题的循证指南，Information and Software Technology 94 (2018) 186–207。[36] A. Barua, S. W. Thomas, A. E. Hassan，开发人员在讨论什么？对 Stack Overflow 中主题与趋势的分析，Empirical Software Engineering 19 (3) (2014) 19–32。[37] A. Tahir, A. Yamashita, S. Licorish, J. Dietrich, S. Counsell，能看出代码是否有异味吗？一项关于开发人员如何在 Stack Overflow 上讨论代码异味和反模式的研究，载于：第22届国际软件工程评估与评价会议（EASE）论文集，加拿大魁北克省蒙特利尔，2018年，第68–78页。[38] A. Anderson, D. Huttenlocher, J. Kleinberg, J. Leskovec，从专门问答网站的社区活动中发现价值：Stack Overflow 案例研究，载于：第10届软件仓库挖掘工作会议（MSR）论文集，中国北京，2013年，第53–56页。[39] J. de Dieu Musengamana, P. Liang, M. Shahin, A. A. Khan，《Stack Overflow 中架构相关帖子的特征及其有用性》一文的复现包，https://doi.org/10.5281/zenodo.4683744，2022年。[40] L. Ponzanelli, A. Mocci, A. Bacchelli, M. Lanza，理解并分类技术论坛问题的质量，载于：第14届 IEEE 软件质量国际会议（QSIC）论文集，美国得克萨斯州艾伦，2014年，第343–352页。[41] H. Zhang, S. Wang, T. P. Chen, Y. Zou, A. E. Hassan，Stack Overflow 上过时答案的实证研究，IEEE 软件工程汇刊 47 (4) (2019) 850–862。[42] H. Zhang, S. Wang, T.-H. Chen, A. E. Hassan，阅读 Stack Overflow 答案：远远不够！，IEEE 软件工程汇刊 47 (11) (2021) 2520–2533。[43] H. O. Obie, I. Ilekura, H. Du, M. Shahin, J. Grundy, L. Li, J. Whittle, B. Turhan，移动应用中的诚信违规问题：自动检测与类别划分，载于：第19届软件仓库挖掘工作会议（MSR）论文集，美国宾夕法尼亚州匹兹堡，2022年，第321–332页。[44] J. Cohen，名义量表的一致性系数，Educational and psychological measurement 20 (1) (1960) 37–46。[45] J. L. Campbell, C. Quincy, J. Osserman, O. K. Pedersen，深度半结构化访谈的编码：单元化、编码者间信度与一致性问题，Sociological Methods & Research 42 (3) (2013) 294–320。[46] P. Kruchten，软件密集型系统中架构设计决策的本体，载于：第2届格罗宁根软件变异性管理研讨会（SVM）论文集，格罗宁根大学，2004年，第54–61页。[47] B. Foote, J. Yoder，大泥球，程序设计模式语言 4 (1997) 654–692。[48] R. de Freitas Bulcao Neto, M. da Graca Campos Pimentel，构建领域无关的上下文感知计算语义模型，载于：第3届拉丁美洲网络大会（LA-WEB）论文集，阿根廷布宜诺斯艾利斯，2005年，第10–19页。[49] P. Petrov, U. Buy, R. L. Nord，需要一种多层次上下文感知软件架构分析与设计方法，将企业架构和系统架构关注点视为一等实体，载于：第9届 IEEE/IFIP 软件架构工作会议（WICSA）论文集，美国科罗拉多州博尔德，2011年，第147–156页。[50] M. Asaduzzaman, A. S. Mashiyat, C. K. Roy, K. A. Schneider，回答有关 Stack Overflow 未回答问题的问题，载于：第10届软件仓库挖掘工作会议（MSR）论文集，美国加利福尼亚州旧金山，2013年，第97–100页。

[51] M. Soliman, M. Galster, M. Riebisch，构建开发者社区架构知识本体，载于：第14届 IEEE 国际软件架构会议（ICSA）论文集，瑞典哥德堡，2017年，第89–92页。[52] T. Bi, P. Liang, A. Tang，架构模式、质量属性和设计情境：开发人员如何运用它们进行设计？，载于：第25届亚太软件工程会议（APSEC）论文集，日本奈良，2018年，第49–58页。[53] S. Wang, T. P. Chen, A. E. Hassan，用户如何修改技术问答网站上的答案？Stack Overflow 案例研究，IEEE 软件工程汇刊 46 (3) (2020) 1024–1038。[54] S. M. Nasehi, J. Sillito, F. Maurer, C. Burns，怎样才算优质代码示例？一项关于 StackOverflow 编程问答的研究，载于：第28届 IEEE 软件维护国际会议（ICSM）论文集，意大利特伦托，2012年，第25–34页。[55] H. Zhang, S. Wang, T. P. Chen, A. E. Hassan，Stack Overflow 上的评论是否组织得当，便于开发人员检索？，ACM 软件工程与方法学汇刊 30 (2) (2021) 文章编号22。[56] S. Nadi, C. Treude，便于浏览 Stack Overflow 答案的关键句，载于：第27届 IEEE 国际软件分析、演化与再工程会议（SANER）论文集，加拿大安大略省伦敦，2020年，第229–239页。[57] T. Haitzer, U. Zdun，关于架构组件图对新手架构师设计理解的支持作用的受控实验，载于：第7届欧洲软件架构会议（ECSA）论文集，法国蒙彼利埃，2013年，第54–71页。[58] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, J. Lun，想要获得好答案？先提出一个好问题！（2013），arXiv:1311.6876。[59] L. Wijerathna, A. Aleti, T. Bi, A. Tang，从 Stack Overflow 中挖掘并关联设计情境与设计模式，实证软件工程 27 (1) (2022) 1–53。[60] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén，《软件工程实验》，Springer，2012年。[61] K. Chinnappan, I. Malavolta, G. A. Lewis, M. Albonico, P. Lago，面向能耗感知机器人软件的架构策略：初步研究，载于：第15届欧洲软件架构会议（ECSA）论文集，线上活动，瑞典，2021年，第164–171页。[62] R. Li, P. Liang, M. Soliman, P. Avgeriou，理解架构侵蚀：从业者的视角，载于：第29届 IEEE/ACM 国际程序理解会议（ICPC）论文集，西班牙马德里，2021年，第311–322页。[63] J. Zou, L. Xu, M. Yang, X. Zhang, D. Yang，从开发人员视角理解非功能需求：运用主题分析探索 Stack Overflow，Information and Software Technology 84 (2017) 19–32。[64] B. Dagenais, M. P. Robillard，恢复 API 与其学习资源之间的可追溯性链接，载于：第34届 IEEE 国际软件工程大会（ICSE）论文集，瑞士苏黎世，2012年，第47–57页。[65] L. An, O. Mlouki, F. Khomh, G. Antoniol，Stack Overflow：代码洗白平台，载于：第24届 IEEE 国际软件分析、演化与再工程会议（SANER）论文集，奥地利克拉根福，2017年，第283–293页。[66] F. Fischer, K. Böttinge, H. Xiao, C. Stransky, Y. Acar, M. Backes, S. Fahl，Stack Overflow 是否有害？复制粘贴对 Android 应用安全性的影响，载于：第38届 IEEE 安全与隐私研讨会（S&P）论文集，美国加利福尼亚州圣何塞，2017年，第121–136页。[67] C. Ragkhitwetsagul, J. Krinke, M. Paixao, G. Bianco, R. Oliveto，Stack Overflow 上的有害代码片段，IEEE 软件工程汇刊 47 (3) (2019) 560–581。

[68] A. Zagalsky, O. Barzilay, A. Yehudai, 《示例溢出：利用社交媒体进行代码推荐》，载于：第3届软件工程推荐系统国际研讨会（RSSE）论文集，瑞士苏黎世，2012年，第38–42页。[69] C. Treude, M. P. Robillard, 《理解 Stack Overflow 代码片段》，载于：第33届 IEEE 软件维护与演化国际会议（ICSME）论文集，中国上海，2017年，第509–513页。[70] C. Ragkhitwetsagul, J. Krinke, R. Oliveto, 《开发人员对 Stack Overflow 上过时及违反许可证的代码的认知与经验：一项在线调查》（2018年），arXiv:1806.08149。[71] C. Treude, M. P. Robillard, 《利用 Stack Overflow 的洞见扩充 API文档》，载于：第38届国际软件工程大会（ICSE）论文集，美国得克萨斯州奥斯汀，2016年，第392–403页。[72] E. Wong, J. Yang, L. Tan, 《AutoComment：挖掘问答网站以自动生成评论》，载于：第28届 IEEE/ACM自动化软件工程国际会议（ASE）论文集，美国加利福尼亚州硅谷，2013年，第562–567页。[73] Q. Gao, H. Zhang, J. Wang, Y. Xiong, L. Zhang, H. Mei, 《通过分析问答网站修复反复出现的崩溃漏洞》，载于：第30届自动化软件工程国际会议（ASE）论文集，美国内布拉斯加州林肯，2015年，第307–318页。[74] T. McDonnell, B. Ray, M. Kim, 《Android生态系统中 API稳定性与采用情况的实证研究》，载于：第29届 IEEE 软件维护国际会议（ICSM）论文集，荷兰埃因霍温，2013年，第70–79页。[75] D. H. Dalip, M. Cristo, P. Calado, 《利用用户反馈学习问答论坛中的回答排序：以 Stack Overflow 为例》，载于：第36届国际 ACM SIGIR 信息检索研究与开发大会（SIGIR）论文集，爱尔兰都柏林，2013年，第543–552页。[76] B. Xu, Z. Xing, X. Xia, D. Lo, 《AnswerBot：自动生成开发人员技术问题的答案摘要》，载于：第32届 IEEE/ACM自动化软件工程国际会议（ASE）论文集，美国伊利诺伊州厄巴纳，2017年，第706–716页。[77] F. Calefato, F. Lanubile, N. Novielli, 《技术问答网站中最佳答案预测模型的实证评估》，《实证软件工程》24（2）（2019年），854–901。