arXiv:2602.03017v1 [cs.CY] 3 Feb 2026

从闲逛到弄明白

在线社交：通往计算思维的途径

Samantha Shorey (sshorey@utexas.edu)[1]
* Benjamin Mako Hill (makohill@uw.edu)[2]
Samuel C. Woolley (swoolley@utexas.edu)[1]

[1]德克萨斯大学奥斯汀分校
[2]华盛顿大学

虽然社交是推动青少年在线参与的一股强大力量，但各个平台却难以利用这种参与来促进学习。我们通过对 Scratch 上 14,000 多条评论进行多阶段分析，力求理解这一动态。Scratch 是一个旨在支持编程学习的在线平台。首先，我们通过归纳法提出“参与式调试”这一概念——用户通过协作进行技术故障排查，并在此过程中学习。其次，我们运用内容分析，确定这一实践在 Scratch 上有多普遍。第三，我们对用户活动进行纵向定性分析，并找出促成参与式调试的三个社会因素：(1) 持续性社群，(2) 可识别的问题，以及 (3) 我们称之为“主题渗透性”的特征，用以描述能够跨越多个主题的对话。我们将这些发现整合到一个理论框架中，凸显了促进学习的愿望与兴趣驱动型子社区之间富有成效的张力；在许多新媒体环境中，正是这些子社区推动着用户参与。

引言

作为程序员，也作为普通人，我们很少能在第一次尝试时就做对。失败是构建事物的一部分。面对挑战时，我们通常会寻求知识丰富的他人的指导和帮助。在日常生活中，社会联系为问题解决提供了途径——教会我们如何构建事物，以及如何做得更好。支持问题解决技能的发展，已成为教育工作者的一项优先任务，他们希望为学生配备应对现代世界的工具。新的教育项目正专注于发展问题解决策略，例如协作和提问——这些技能必然与同他人的互动相关（21世纪技能合作组织

*通讯作者。

，2019年）。尽管问题解决的协作性质隐含在许多关于数字媒体和学习的研究中，但直到最近，才出现了一批关于“关联式学习”的研究，将这种社会行为与年轻人高阶思维技能的学习联系起来（伊藤等，2013年）。

本研究通过分三个阶段考察 Scratch 用户的学习与评论之间的关系，为新兴的关联式学习研究议程作出了贡献。Scratch 是一个拥有数百万用户的在线社区，或许也是年轻人学习基于积木的编程最受欢迎的平台。我们对 640 个项目和 53 份用户项目历程开展扎根理论分析，并对另外 600 个项目进行内容分析，以考察并阐明社会互动所带来的、以学习为导向的益处。我们提出了参与式调试这一概念：Scratch 用户（“Scratch 用户”）利用项目中的评论区，排查与项目设计相关的问题，并培养更高阶的计算思维技能。我们还指出了三个我们认为对参与式调试的发展和持续实践具有独特重要性的情境因素：（1）持续性社群；（2）可识别的问题；（3）低主题密度，或我们所称的“主题渗透性”。我们认为，第三项尤为重要，因为它阐明了关联式学习中的一个核心矛盾：促进特定概念（如与计算思维相关的概念）学习的愿望，与参与兴趣驱动型子社群之间的权衡。Ito 等人（2009年b）及其他学者已将参与这类子社群视为实现有意义学习的一条路径。最后，我们综合总结研究发现，并探讨其对不同情境下关联式学习设计的启示。

背景

社交与学习

有影响力的教育方法表明，学习实践深深植根于社会环境之中（Lave and Wenger, 1991）。Seymour Papert（1976）等建构主义学者长期以来倡导这样的学习环境：将教育融入“更宏大、更丰富的文化—社会体验”（第7页）。计算机是建构主义学习的理想工具，为自主探索和技能发展提供了集体空间（Resnick, Bruckman, and Martin, 1996）。参与式媒体尤其如此。参与式媒体环境的特点是参与门槛较低、强调分享、资深成员与新手之间的非正式指导，以及群体成员之间的联结感（Jenkins, 2009年：6）。这些环境本质上以社交为主，是主动学习的空间；孩子们在彼此共同创作、并为彼此创作媒体的过程中发展新技能。研究表明，参与这些社交社群对学习的贡献，与软件等技术工具不相上下（Kafai and Burke, 2014）。

在参与式媒体环境中，社会互动往往推动年轻人开展创意活动和教育活动。正如 Mimi Ito 和合著者（2009年b）在《闲逛、瞎混与极客化》，中所述，年轻人以各种方式在线社交。他们会“闲逛”，参与友谊驱动型活动，与线下的朋友和同龄人聊天。他们也会“极客化”，参与由共同兴趣或爱好驱动的兴趣型活动。当社会关系催生出新活动的萌芽兴趣时

，用户会参与一种富有成效的中间状态，他们称之为“瞎混”。这些文化世界有能力改变人们对同龄人的看法：用户在兴趣驱动的在线空间中闲逛，社会关系由此催生新的兴趣。然而，回顾此前的研究，Ito 等人（2018年）指出，尽管他们在各个田野调查点都看到了大量社交闲逛和兴趣驱动型极客活动的证据，“很少有 [年轻人] 充分利用数字网络的学习潜力”（第 2 页）。

随后，一组研究数字时代的青少年的研究人员提出了“关联式学习”模型，将这些活动结合起来，并指出新媒体与教育的三个关键因素：(1) 同伴支持，(2) 个人兴趣的参与，以及 (3) 与学术机会的联系（Ito 等人，2013年）。在这些因素交汇之处，年轻人与志趣相投的同伴互动，以拓展并深化他们的正式知识。Ito 等人写道：“有效的学习既需要个人兴趣，也需要社会支持来克服逆境”（第4页）。在我们的研究中，我们考察这一过程，以阐释年轻人如何从“闲逛”走向“弄明白”——年轻程序员面对创造性问题时，如何运用兴趣驱动的子社群中的社交功能来克服问题。

以教育为导向的组织，例如21世纪技能合作组织（2019年），已将“问题解决”认定为当今年轻人最重要、最实用的技能之一。学生需要培养策略，运用已有知识创造性地寻求解决方案。在日益数字化的世界里，问题解决技能往往是在网络环境中习得的；年轻人在那里学习使用计算机工具，也从这些工具中学习。通过这些活动，他们培养了Janet Wing（2006年）所称的“计算思维”技能。计算思维是一种处理问题的方法，运用计算机科学的广泛核心概念。计算思维不仅仅是使用计算机的能力，它还植根于一系列实践和视角，这些实践和视角可应用于实体世界或数字世界。将事物拆分为更小的部分（模块化）、通过测试周期不断创建（迭代），或在他人作品的基础上继续创作（再混合）等实践，都是计算思维的体现（Brennan 和 Resnick，2012年）。我们的研究尤其关注“调试”实践：用户用来“应对并预见问题”的策略（Brennan 和 Resnick，2012年：7）。

调试是计算思维的关键方面，其本质也使它与问题解决的理念密切相关。在学校等传统教育环境中，学生往往被置于这样的情境：要么“弄懂某件事”，要么就弄错了（Papert，1980：21）。作为一名计算机程序员，你很少能一次就做对。程序员通过失败和成功调试，学会如何解决问题，并制定可应用于线上和离线情境的策略。调试可以表现为尝试不同的代码脚本、重写脚本、寻找可用脚本的示例，甚至还包括更常见的调试行为，例如休息一下或向他人求助（Brennan 和 Resnick，2012）。

建构主义学习学者一直认为，社交是在线学习不可或缺的方面。同伴社群通过提供榜样、项目示例以及分享创作的受众，既激励学习，也支持学习。在游戏制作平台上，互相帮助的能力是社会互动与乐趣的催化剂（Bruckman，1998年）。他们并不把编程视为一种孤立的活动，而是认为教育倡导者若将其视为一种“共享的社会实践”将大有裨益——这不仅要求具备进行 计算思考的技能，还要求

能够参与 计算实践（Kafai，2016年；Kafai 和 Burke，2014年）。尽管人们承认二者相互交织，但在建构主义文献中，成为更优秀的程序员和成为更出色的合作者，主要仍被概念化为不同的成果。学习代码被视为个人成就，即通过获得技能而实现的个人成果。社会维度则主要聚焦于软技能的培养，例如良好的团队协作能力（Kafai 和 Burke，2015年）。

在线空间中的问题解决往往是一项集体事业。例如，研究多人在线游戏魔兽世界 的研究者观察到，该游戏的社会互动能力有助于玩家在共同形成共享理解的过程中寻求并评估信息（Martin 与 Steinkuehler，2010年；Steinkuehler 与 Duncan，2008年）。在 GitHub 等在线软件开发社区中，社交功能为程序员提供了协作组建团队以及构建软件的新方法等独特机会（Begel、DeLine 与 Zimmerman，2010年）。Dabbish、Stuart、Tsay 与 Herbsleb（2012年）认为，GitHub 上信息的透明共享使程序员能够更有效地协调项目并提升其技术技能。这些在线编程社区的社交要素使群体能够通过团队合作与集体学习推进软件开发。

我们的工作推进了对此类社区中集体问题解决的研究，进而考察年轻人与初学程序员的实践。具体而言，我们探究 Scratch 在线社群中的调试这一计算实践。由此，我们提出一个框架，用以识别并培育富有成效的社会互动，将其作为在面向青年的参与式媒体环境中获取分析性高阶知识的途径。

实证背景：Scratch 在线社群

Scratch 面向年轻人开发，是一种可视化编程语言，用户可以通过拼接编程积木块来构建互动媒体。麻省理工学院媒体实验室终身幼儿园研究小组的 Scratch 开发者将这种语言置于一个在线社群中，让年轻人能够参与“以建构为导向、对个人有意义的创造性表达活动”（Brennan 等，2010年）。Scratch 在线社群的社交属性是刻意设计的：用户在创建在线“项目”的同时，可以彼此互动并学习基础编程；这些项目通常以动画和游戏的形式呈现。

Scratch 在线社群的创建者 Andres Monroy-Hernández（2012年）解释说，他希望通过打造“一个让同伴们创作、分享、改编，甚至只是‘消磨时光’的空间”（第38页），将社交活动置于核心。为此，Scratch 提供了诸如“喜欢”按钮等社交功能，并允许用户改编项目、将项目整理到作品集中、在论坛发帖以及发表评论。用户在创建项目时遇到问题，常会转向评论串和论坛，从其他用户那里寻找有用的信息。

在关于Scratch的大量实证研究中，社交与学习之间的关系存在争议。Brennan等（2010）、Brennan 等（2011年）、Fields等（2015）、Kafai与Burke（2014）以及Aragon等（2009年）的定性分析与案例研究均支持如下论点：社交在学习使用Scratch编程的过程中可以发挥重要作用。在一项定量研究中，

Dasgupta等（2016）表明，评论的增加（用作控制变量）对Scratch用户编程词汇中的积木块数量具有微弱但正向的影响。
其他实证研究对 Scratch 在线社群中社交活动与计算机运算学习之间的关系，描绘了不那么乐观的图景。Dasgupta等（2016）的研究还表明，评论增加与用户首次使用六种与计算思维相关的复杂 Scratch 积木块的可能性降低有关。Gan、Hill 和 Dasgupta（2018）指出，“love its”形式的正向反馈与用户决定分享后续 Scratch 项目的可能性降低有关，尤其是在男孩中。这项研究揭示了社区开展社交活动的能力与通过项目创作促进学习的能力之间的张力。

Brennan 等（2010）在文章《制作项目，结交朋友》的结尾段落中指出，Scratch 用户有两种原型：“社交型用户”和更专注于计算的“创作者”。他们认为，这两种极端类型都面临挑战：“社交型用户缺乏专注于交互媒体创作并以新形式表达自我的机会”，而“创作者则无法受益于与他人共同学习、向他人学习，也无法发展社区及自身的创作能力”（第78页）。综合来看，实证研究表明，尽管在 Scratch 上进行社交活动可以成为培养计算思维的一条途径，但这种潜力往往未能实现。

Data

我们的研究数据来自 Scratch 用户创建的逾 4600 万个项目和 2.21 亿条评论。这些数据是在一项关于非正式学习环境的大型研究项目中收集的，该项目由麻省理工学院人体实验对象使用委员会监督。本文分析的所有项目和评论均由创作者发布，他们知晓这些内容可能会在网上公开显示。为尊重 Scratch 用户的自主权，已将用户后来删除或设为私密的项目从数据集中剔除。此外，Scratch 不允许使用真实姓名，因此用户在 Scratch 平台上及我们的数据中均保持匿名。本研究仅使用公开数据中的信息，研究人员没有与用户进行任何互动，也未实施任何干预。

项目级评论数据

为了识别有意义的同伴反馈实例，并开始了解它们在 Scratch 上出现的条件，我们构建了一个数据集，其中包含社区中分享的 1,240 个项目的随机子集。“抽样单位”是单个 Scratch 项目。我们的项目样本取自收到三条或更多评论的 Scratch 项目总体。之所以设定这一限制，是因为我们的研究旨在了解通过在线社交活动开展的协作式问题解决。我们认为，三条评论足以代表讨论和社交活动的合理最低水平。数据集包含每个项目的元数据，包括指向 Scratch 上“在线”项目的超链接，以及所有评论的完整文本表示；这些评论按时间顺序排列，并以与其在 Scratch 上显示方式完全相同的线程形式呈现

按时间顺序排列，并以与 Scratch 上显示方式完全相同的方式串联起来。

用户项目历史评论数据

由于我们的目标之一是了解协作解决问题技能的发展，我们构建了第二个数据集，用于记录用户随时间推移的经历。我们希望这些数据能够揭示用户发展或未能发展计算思维技能的过程。为了构建合适的数据集，我们根据下一节所述的方法论，选取了53名用户，并编写了一个 Python脚本，收集这些用户创建的项目上的所有评论活动。我们为每位用户分别创建了一个文件，其中包含他们在 Scratch 上期间创建的每个已发布项目所收到的评论。虽然53名用户并不算多，但由于每位用户都可以分享许多项目，而每个项目又可以收到许多评论，因此这仍然构成了一个大型数据集。这些用户分享的项目数量从1个到超过1,500个不等，中位数为29个项目。与 Scratch 中的总体活动一样，不同用户之间的活动分布高度偏斜。总体而言，用户轨迹构成了一个数据集，其中包含在5,213个项目上留下的3,779条评论。

方法论

我们的分析分为三个阶段。第一阶段对项目级评论数据进行了归纳分析，从中识别出一种我们称为 参与式调试 的构念。第二阶段对 Scratch 项目的随机样本进行了简单的内容分析，以确定参与式调试在 Scratch 上的普遍程度。第三阶段再次进行归纳分析，通过考察 Scratch 用户共享项目的项目历史，识别参与式调试产生的若干潜在因素。

由于第一阶段和第三阶段的目标是识别社会互动的反复出现的特征，因此它们非常适合采用扎根理论（Glaser和Strauss，1967年）。扎根理论是一种迭代式程序，采用“持续比较法”（Glaser和Strauss，1967年：101）。持续比较始于在“开放式编码”期间，将数据编码为尽可能多的分析类别（Lindlof和Taylor，2011年：246）。随后使用备忘录创建编码手册，其中包含各编码的定义、它们在主题类别中的相互关系，以及文本中的示例（Lindlof和Taylor，2011年：251）。随着新的类别和编码不断显现，它们会塑造后续数据和先前数据的分析，并应用于其中（Glaser和Strauss，1967年：109）。

第一阶段包括由两位作者并行对 640 个项目及其完整评论串进行扎根理论分析。第一阶段的成果是第一作者撰写的一份备忘录，其中描述了参与式调试这一构念。备忘录的内容在研究结果部分的第一部分中进行了说明，并用于指导后续分析阶段。第二阶段包括对随机抽取的 600 个 Scratch 项目评论串进行内容分析，旨在根据第一阶段备忘录中的描述，判断是否存在参与式调试活动。按照 Neuendorf（2017）的做法，我们的内容分析包括一个“规范化”步骤，两位作者对 100

个项目是否存在参与式调试进行编码，并讨论所有分歧（阶段 II-A）。完成这一步后，两位编码员对 300 个项目进行了编码，并评估了编码员间信度（II-B）。在确认信度后，其中一位作者又对另外 200 个项目进行了编码（II-C）。

尽管第一阶段和第二阶段使用的项目随机样本提供了参与式调试的实例，并证实这种做法相当普遍，但我们发现很难理解促成这一做法出现的社会情境。第三阶段试图通过对特定用户随时间分享的所有项目，以及其评论串中形成的“文化”进行第二轮扎根理论分析，来解决这一问题。第三阶段使用的用户级数据集包含我们在第二阶段确认有参与式调试迹象的53位用户所分享的所有项目（以及这些项目相关的评论）。重要的是，这意味着每位用户的项目历史中至少有一次情境化社会问题解决的实例。第三阶段使用的数据集包含5,213个项目和3,779条评论。用户历史被随机分成两组，每组由一位作者编码。研究团队标出了具有代表性或难以处理的案例，并将其分享和讨论。在整个迭代式理论生成过程中，研究团队全体成员都会阅读并讨论评论实例和备忘录草稿。数据集的分析聚焦于体现并塑造参与式调试的持续互动。我们旨在理解人与人、情境与事件之间的过程，以及它们如何相互影响——这种方法称为过程理论。过程理论淡化了个体参与者的重要性，转而强调理解“参与者行动的特定情境，以及这一情境对其行动的影响”（Maxwell，2013年：30）。

表 1 展示了我们方法论各阶段及阶段内各步骤的情况，包括涉及的项目数量、评论数量和编码人员数量。总的来说，这项工作基于对 6,453 个项目中留下的超过 14,733 条评论的分析。

表 1
定性数据分析步骤

阶段	描述	项目	评论	编码人员
I	构念定义	640	2,846	2
II-A	规范化	100	1,168	2
II-B	可靠性测试	300	3,989	2
II-C	其他项目	200	2,951	1
III	用户项目历史记录（53 位用户）	5,213	3,779	1
总计		6,543	14,733	

研究发现

我们在第一阶段的研究发现是参与式调试这一概念构念。我们在第二阶段的研究发现是内容分析的结果，描述了参与式调试在 Scratch 上的普遍程度。我们在第三阶段的研究发现包括参与式调试能够最有效开展的三种条件。下面我们将逐一介绍。

第一阶段：参与式调试实践

第一阶段对 640 个项目进行扎根理论分析后，浮现出的最显著、最重要的主题，是我们最终称为<样式 id='1'>参与式调试。</样式>参与式调试描述了一种计算实践：用户借助社会互动来克服挑战并实现目标。从最基本的意义上说，当项目创作者发现自己的项目存在问题，并明确地将这一事实告知他人时，就发生了参与式调试。这种情况经常出现在项目描述中，例如项目创作者会承认不足（“我知道它有一些错误”），或给出具体警告（“这个按钮无法使用”）。在这些情况下，项目创作者表明，他们意识到与更大的在线社区互动，是获得帮助的重要第一步；而其他查看该项目的 Scratch 用户（因而也会遇到这个错误）可能同样拥有修复它所需的知识。当创作者发现自己的项目存在故障时，他们便为对话和评论创造了空间，而这些对话和评论会进一步引发其他解决问题的行为，最终产生解决方案。

项目创作者也可能明确请求帮助或协助。创作者通常不会向单个用户寻求帮助，而是面向更广泛的社区提出开放式问题，或表达困扰。请求可能较为笼统（“我需要帮助！”），也可能更加具体（“怎样制作云列表？”）。作为回应，其他 Scratch用户可能会发表评论，针对问题提供解释性的解决方案：“要改善跳跃效果，可以使用这个脚本……”或“如果你想减少永远循环，可以……”

在最基本的参与式调试表现形式中，Scratch 用户可能只是通过“它有故障”或“它坏了”之类的评论，说明他们正在查看的项目存在故障。Scratch用户还可能指出项目如何 出现故障（“这些命令什么也不做”），或者更有帮助地说明故障的原因及修复方法：“太棒了！你可以不使用 [when (key) key pressed]，而尝试使用 [forever [if [key (key) pressed?]]]……这样会让移动更加流畅！”即使创作者没有发现自己项目中的问题，其他Scratch用户也会主动识别问题并提供解决方案。这些对话的出现展现了一种参与规范：Scratch用户将他人和自己都视为改进项目的资源。

重要的是，参与式调试并非只是项目创建者与一位专业用户之间发生的活动。相反，它会在单个项目的评论区内，调动多位用户的资源，修复多个项目中的多个问题。接下来，我们将看到四位 Scratch 用户之间的一段对话，他们都不是项目创建者。前两条评论都指出了一处故障（文本显示速度），其中第一位 Scratch 用户还提供了解决方案。随后，第三位 Scratch 用户提出了一个看似无关的编程问题，第四位用户则给出了详细回复，告诉他该怎么做——添加一个额外的编程积木，以及该积木如何与其他积木配合工作：

> P1：你应该设置成这样：有人读完文本后，可以按空格键听下一段，而不用让 Scratch 用户匆忙读完。
>
> P2：文字速度太快了……而且你的拼写也不太好……抱歉。

> P3：救命！Control 里有个按钮写着“当接收到_____”，我输入了一些内容……但还是不明白这条消息是什么意思。救命！（附注：这条消息要怎么用？）
>
> P4 [回复 P3]：你必须使用广播积木。可以这样理解：广播积木会让角色说出其他所有角色都能听到的话。「当接收到」积木的真正意思是「当我听到时」。如果这还帮不上忙，就在我的某个项目下留言，我可以进一步解释。

Scratch 用户遇到问题时，会寻找其他人成功的项目，并向经验更丰富的 Scratch 用户留言。他们可能会留言说明自己的情况并寻求解答，也可能针对正在查看的项目发表评论，询问“你是怎么做到的？”

这种情况会以独特的方式呈现，因为 Scratch 在线社群允许用户“查看内部”，了解动画是如何编写代码的。查看项目代码的功能使项目能够展示成功的编程概念。尽管其他 Scratch 用户始终都能看到这些编程积木，但通过评论区进行社交互动仍然是一种很有价值的解释方式。例如，在一个以外太空为主题的项目中，一位 Scratch 用户问道：“能请你解释一下你是如何确定行星的位置的吗？”创作者回答：“好的，首先创建太阳。无论它在哪里，都为它的 x 和 y 位置创建一个变量……”随后继续逐步解释如何制作这个项目。

参与式调试关注的是计算实践，描述的是与项目功能有关的问题解决行为，例如识别故障或代码问题，而非与游戏外观或玩法有关的审美建议。尽管这些行为有所不同，我们发现它们与参与式调试高度共现：用户讨论游戏外观的项目，也很可能会讨论游戏运行方式。在这两种情况下，项目创作者和 Scratch 用户都会彼此交流，将对方视为一种社交资源。

第二阶段：参与式调试有多普遍？

在第二阶段，我们对 Scratch 项目进行了内容分析，以确定参与式调试在 Scratch 上有多普遍。我们的内容分析遵循 Neuendorf（2017）提出的方法论，由人工编码员查看项目和评论串，并判断其中是否包含参与式调试的实例；我们对参与式调试的定义来自第一阶段形成的备忘录（上一节已对此作了详细总结）。与内容分析中的常见做法一致，我们的方法依赖人工编码员的判断，并不要求出现特定关键词。为了评估这一构念的编码员间信度，我们对第二阶段B中独立编码的300个项目样本计算了克里彭多夫α系数，发现我们能够可靠地对项目是否存在参与式调试进行编码（α=0.79）。

在第二阶段编码的600个项目样本中，我们发现了53个参与式调试的实例，占样本项目的8.8%。通过二项分布“精确”计算，我们得出的95%置信度

区间显示，从中抽取样本的总体（即至少有3条评论的项目）中的实际比例估计在6.7%至11.5%之间。尽管我们的团队和 Scratch 设计团队都对这个比例如此之高感到惊讶，但这也表明，参与式调试实践在 Scratch 上并不常见。

第三阶段：社会情境与评论文化

在分析的最后一步，我们试图了解，在以群体为导向的问题解决文化中，参与式调试是如何以及何时得到促进的。为此，我们对第二阶段中有证据表明参与式调试行为的53名用户分享的所有项目，进行了新一轮扎根理论分析。我们发现，在线评论空间能够让人们以多种形式参与其中：既可以围绕共同兴趣或个人感受进行深入的社交互动，也可以讨论编程相关内容。尽管这些社交环境的技术特性——评论和改编的功能——在不同用户之间是一致的，但用户使用这些功能的方式却并不相同。促成参与式调试的并非评论空间本身的存在，而是特定的评论文化。我们识别出三个促进或阻碍参与式调试的因素，按从最基础到更复杂的顺序排列：（1）持续性社群；（2）已识别的问题；（3）不局限于单一话题、因而具有“渗透性”的对话。

持续性社群。 持续性社群由两个特征界定：他人的在场，以及持续的参与。虽然他人的在场看似参与式调试中显而易见的方面，但在 Scratch 上，其他用户会评论创作者的项目并非理所当然。例如，在一位用户的历史中，我们看到一系列八个项目（均为同一项目的修订版本），他每次都问：“请为完整游戏发布建议”，但没有得到任何回应。总体而言，这位用户在其各个项目中的评论出现次数非常少。没有他人的参与，他的求助不会形成任何真正的对话，也没有学习的机会。当另一位经常收到大约十几位Scratch用户评论的用户发出极为相似的反馈请求时，却得到了回应和解决方案。当这第二位用户问“还有其他人遇到滑块故障吗”时，另一位Scratch 用户指出了代码中出错的确切位置。创作者用对代码的详细解释回复道：“是的，我真的没法修复，原因是这样的：repeat until <(answer) > [0]> ask [Hour] .....它会跳过分钟部分，因为它已经满足了传感器模块”，而另一位Scratch 用户再次回复，帮助重写这段代码。

有些用户群体会以某种方式反复互动，从而形成一种社会动能。这些用户往往会积极且持续地参与某位用户的各个项目。持续的互动对于促成参与，并最终修复项目中基于代码的问题，尤为重要。这些用户的评论串十分活跃，充满互惠与支持。对话的热度似乎会吸引更多用户加入。随着这些群体不断壮大，参与式调试实践也蓬勃发展。

尽管没有其他用户参与就无法进行参与式调试，但先前研究表明，高强度社交性和受欢迎程度往往可能与后续参与度（Gan 等，2018年）和学习（Brennan 等，2010年）相互冲突。持续性社群常常通过将成员的作品添加到用户策展画廊中，提高这些作品的可见度，从而增加项目评论，并巩固持续性社群。

拥有一小群彼此关心对方项目的用户——这类用户会采取诸如将彼此的项目加入作品集等行动——为以面向代码的问题解决为目的的评论创造了可能性。最有力地促成“弄明白”的，是这种共享社群的存在，而不一定是评论数量多。事实上，一些评论最多的项目既没有问题解决的迹象，也没有持续群体互动的证据。例如，出现在 Scratch 首页上的项目收到了许多用户的大量正面评论；而属于主题更集中的粉丝群组的项目则有着大量评论记录——但这些评论数量并未促成参与式调试的出现。

已识别的问题。 存在已识别问题的项目会促使其他用户提供实质性反馈。实质性反馈包含信息和见解，与充斥在 Scratch 评论区中那些较为笼统的赞美（“项目很棒！”或“做得不错”）不同。当用户直接针对项目或技术的特定方面发表评论时，就会产生实质性反馈，其中常包含“如果……会更好”“我认为你应该……”或“你需要……”之类的说法。这类反馈既可能是主动提出的（即未经提示就给出的建议），也可能是对如何实现某种编程技术的问题所作的回应。当出现实质性反馈时，一群用户更有可能参与到协作式调试中。

能否提供实质性反馈，取决于 Scratch 用户识别项目中问题的能力。无论是发现自己项目中问题的项目创作者，还是通过与他人项目互动而发现问题的访客，都是如此。创作者指出项目中的某些内容无法正常运行，或访客指出存在程序错误时，都可能出现这种情况。即使 Scratch 用户缺乏专业术语，无法明确说明问题是什么或该如何解决，Scratch 项目的互动性也能让其他访客亲自体验故障，随后查看项目内部进行调查。

Scratch 的一项重要功能是能够“改编”项目。用户借助这一功能，超越纯文字说明，示范如何修复项目。在下面的例子中，项目创作者正尝试编写一个脚本，将项目的访客添加到一个持续更新的列表中。然而，该项目需要一个能够调用存储在云端的 Scratch 数据的积木——这项功能只有经验更丰富的 Scratch 用户才能使用。访客使用混编功能展示了一种变通方法，通过“解码”来说明列表积木的运作方式：

> P1：嗯……很遗憾，这样行不通，因为它不是云列表……我会发一个示例，说明要怎样才能让它运行……:D
>
> 创作者：哦。要怎么创建云列表？没有任何选项可以
>
> P1：目前还不行……:P 你得用一套解码云变量的系统来实现……等一下……我快弄好了……:P
>
> P1：好了，给你

Scratch 展现了一种学习社群：即使没有专业知识，也能获得专家的帮助。

相反，Scratch 用户能够发现有些地方 出了问题，而其他 Scratch 用户则可以利用 Scratch 内置的社交功能——例如查看项目内部或进行改编——提供实质性反馈。

低主题密度（“主题渗透性”）。最后，从我们的用户项目历史数据中可以清楚看出，无论是用户数量还是已识别的问题，都不能确切表明存在参与式调试。互动内容高度围绕某个特定主题（例如角色扮演游戏、《美少女战士》或其他粉丝圈）展开的群体，通常会有活跃且篇幅很长的评论串，其中涉及多个用户，但往往并未体现计算思维实践。这些群体中的所有或大多数互动都围绕感兴趣的主题展开，而非改进项目。此外，即使确实出现了问题解决（例如某位用户寻求帮助），也会被粉丝相关的对话淹没。在探索参与式调试时，讨论的情境和内容至关重要。例如：

> P5：图片切换得不够流畅，图片数量也不够，它会直接从开头跳到中间部分。
>
> P6：我喜欢《凉宫春日》的忧郁
>
> P7：我超爱《凉宫春日的忧郁》！！！

这场讨论可以与更开放式的对话相比较；后者不太聚焦于游戏的抽象主题，而是留有空间讨论游戏本身：游戏如何运作、怎么玩，以及看起来是什么样子。

不过，这里存在一种隐含的张力。参与粉丝子社区是与其他用户建立联系的好方法，尤其是那些非常活跃、社交网络广泛的用户；这些网络可以成为吸引他人参与并使其持续投入的资源。然而，参与这些子社群的 Scratch 用户会利用评论区来“尽情畅谈自己的兴趣”。当明确可识别的问题出现在这些主题密集型社交空间中时，它们很容易淹没在拥挤的对话里。

因此，我们将对广泛对话的开放性——我们称之为“主题渗透性”——视为参与式调试的一个重要方面。我们隐喻性地使用渗透性 这一术语：正如多孔物质能让物质轻易穿过一样，渗透性主题是那些容易被其他类型的对话渗透或与之结合的主题。主题渗透性并非衡量一个群体所涵盖主题数量的指标（类似于广度这一概念），而是描述评论文化的一种开放性与回应性。具有渗透性的评论文化不会过于密集，以至于新主题被阻隔在对话流之外。开放式对话使新提出的想法能够改变当前对话，并对其他用户变得显著。这些评论文化不仅可以包含关于粉丝文化的讨论，也可以包含关于编程的对话，甚至像我们阅读的许多 Scratch 项目一样，包含对个人生活与信念的真挚分享。这些对话空间体现了建构主义学习学者的最高期望——有意义的世界催生有意义的学习。我们的分析表明，在 Scratch 上平衡兴趣驱动的参与与学习激励是一项核心挑战。

讨论：支持参与式调试

第二阶段的结果表明，参与式调试在 Scratch 上很少见。我们对参与式调试不常出现的解释是：上文所识别的各项因素都是该实践得以出现的必要但不充分的要素。这三项因素——持续性社群、已识别的问题以及开放式对话——必须同时存在，才能最大限度地提高 Scratch 的社交功能促进学习的可能性。例如，我们发现许多拥有持续性社群的评论文化从未进行参与式调试，因为它们缺少另外两项因素中的一项或两项。图1提供了一幅简单示意图，用以呈现这三项因素全部七种可能的组合。尽管我们数据中所识别的最丰富的学习机会位于图中以星号标示的中心位置，但全部七种组合的例子都出现在我们的数据中。具备三项特征中两项的评论文化（图中编号区域）反映出支持参与式调试的最大机会，因此是开展设计干预最有成效的场所。然而，每种组合都带来独特的挑战。我们将依次讨论这三种组合。

在区域 1 中，我们发现了一些具有持续性社群和开放式对话特征的评论文化。这些评论文化具有很强的社交性和开放性，但用户从未指出问题或寻求帮助。尽管有人倾听，互动也广泛多样，他们却没有共同分享知识来克服问题。这类用户群体最有潜力开展参与式调试，因为他们的开放态度和积极参与意味着，问题解决的尝试能够获得良好的基础。Scratch 等在线社群的设计者可以通过鼓励这些环境中的创作者公开反思他们所面临的挑战，来促进学习，并由此在持续进行的对话中开启问题解决。

在区域 2 中，我们发现了一些评论文化，其中的问题界定明确、对话开放，但缺乏持续性社群。在这里，项目创作者意识到了问题，也愿意着手解决，但就是没有其他 Scratch 用户在场提供帮助。Scratch 这样的环境可以通过促进这些情况下用户之间的联系来鼓励学习。要参与参与式调试，用户需要能够找到并吸引愿意且有能力共同寻求解决方案的其他用户。“需要帮助”工具可以让遇到困难的项目更广泛地被人看到，例如将这些项目收集到“需要帮助”作品集中，从而帮助这些环境中的项目建立持续性社群。

在区域 3 中，有些评论文化既拥有持续性社区，也能识别出问题，但主题密度很高。当主题是编程时，这些文化中会出现一些最鼓舞人心的参与式调试案例。在这里，具备技术知识的用户社群聚集在一起，彼此推动，共同掌握并学习复杂的编码流程。尽管如此，我们的数据集中，大多数主题密度高的群组都有这样的风险：为了围绕粉丝文化或角色扮演展开更多主题性对话，而淹没问题解决的尝试。虽然粉丝子社群也有其独特的钻研方式，但他们通常钻研的并不是编码或计算机。这并不意味着以粉丝文化为中心的 Scratch 子社群不会为了创作和构建粉丝主题项目而学习计算思维概念。然而，在我们的数据中，他们通常不会在社会互动中学习计算实践。即使问题解决作为一个话题被提出来，也很容易被忽视，因为用户更专注于一个更.....

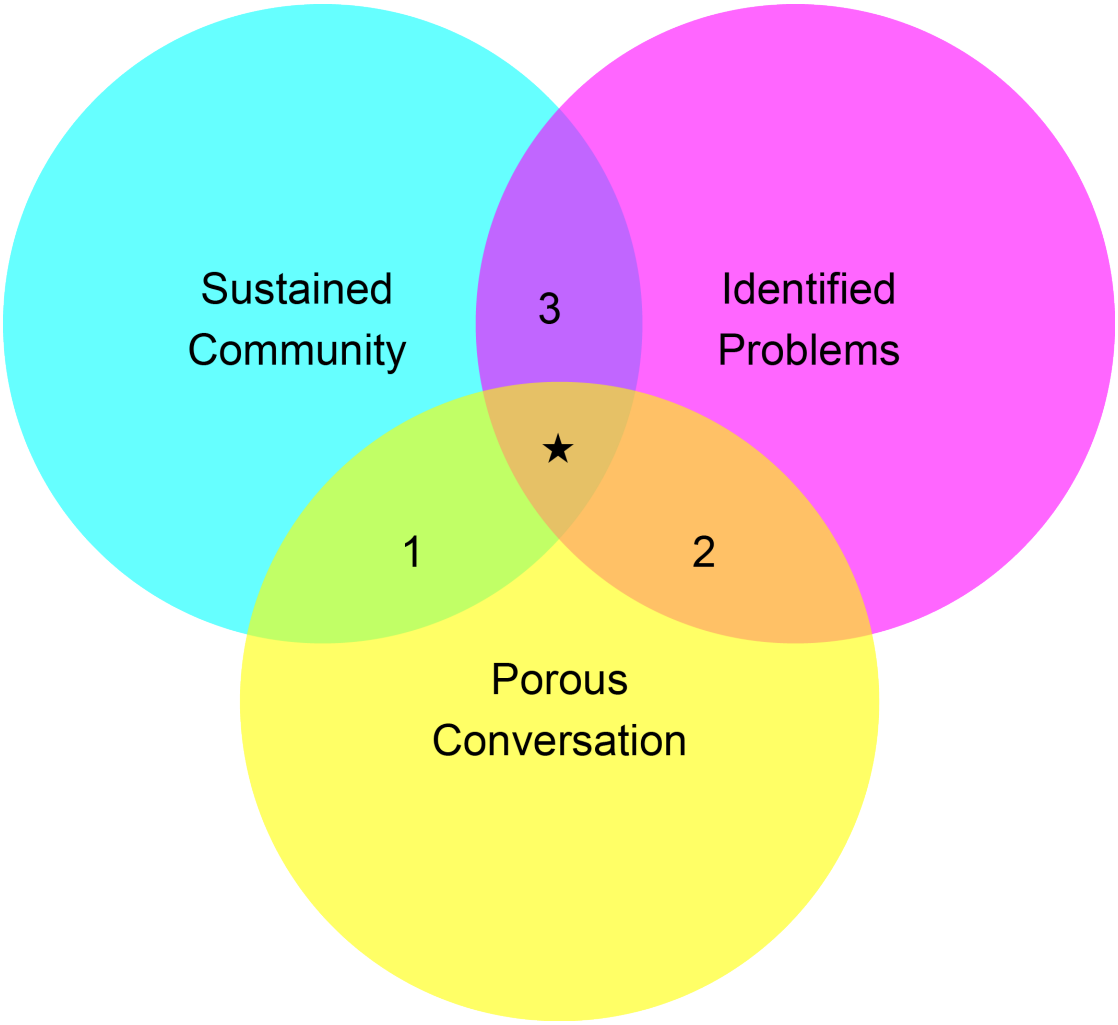


图1：维恩图，展示了促进参与式调试的三种社会因素之间的关系。

主题性对话。尽管在这些情境中支持参与式调试颇具挑战，但在专注于解决问题的评论上附加“需要技术帮助”标记，有助于引起人们对已识别问题的关注，并打破冗长而复杂的对话。

个人兴趣与更具针对性的生产性活动之间的张力，体现了参与式学习环境中一种广泛存在却此前缺乏理论阐释的权衡。建构主义学习方式的核心，是致力于为了“可识别的个人目的”而获取知识（Papert，1980：21）。在 Scratch 中，这意味着构建项目，而不是死记硬背编程概念。学习超越了非个人化与抽象化，变得富有表现力并对个人具有意义（Resnick、Bruckman 和 Martin，1996）。对流行文化的主题兴趣往往会推动项目设计，并将用户联系到社会性社区中。例如，在对 MOOSE Crossing 的微观分析中，Bruckman 发现，“重要的共同兴趣”（在此案例中是《星际迷航》）能够在成员之间建立纽带，并显著促进他们融入社区（第 49 页）。项目的主题、故事和内容在 Scratch 中也尤为重要（Kafai 和 Burke，2014：114）。在我们的分析中，缺乏持续评论文化的项目，可能表明项目创作者没有充分利用兴趣驱动型社区的力量。然而，我们的分析也表明，围绕共同主题兴趣而形成的社区所蕴含的活力，需要通过能够促进以编程为重点的互动的工具来加以平衡。

虽然我们的分析将这三个因素视为有利于参与式调试的评论文化中的静态要素，但我们发现，它们往往构成复杂反馈回路网络的一部分。例如，已识别的问题可能会引发持续的讨论，用户群体通过反复互动来测试、迭代并找出问题的解决方案。或者，涵盖个人话题、兴趣驱动型话题和计算机相关话题的开放式对话，也能促进持续性社群的形成，让用户彼此建立有意义的联系。因此，我们发现，这三个因素难得地同时出现时，往往能够在一段时间内反复支持用户在 Scratch 项目中的参与式调试，而非仅仅发生一次。尽管参与式调试相对少见，但当评论文化呈现这三个特征时，它可以成为用户在 Scratch 上反复进行的一种活动。

结论

故障、程序错误和错误是编程中自然会出现的情况，但对于那些根深蒂固地认为自己能力不足的人来说，这些最初的失败会成为他们无法胜任的证明（Papert，1980）。对于某些群体而言，最初失败的影响可能尤其强烈，例如女孩，她们常被刻板地认为天生不擅长数学，也不擅长计算机运算等与数学技能相关的活动（Spencer、Steele 和 Quinn，1999）。在《头脑风暴》中，Papert（1980）指出，只有少数特殊事件才能让人们克服对自身能力根深蒂固的看法。使用 Scratch 创建项目，正是促成这些事件的契机。通过研究用户解决问题的社会过程，我们希望更好地了解人们在学习过程中克服失败的途径。

即将发表于《新媒体与社会》预印
本制作于2020年4月18日

我们的项目表明，年轻人确实会求助于社会资源，以克服挑战并解决问题。尽管如此，在这些项目的创作者中，我们发现他们的发展轨迹截然不同，也有许多用户并未培养和磨练更高阶的计算思维实践。持续性社群、已识别的问题以及开放式对话，都是参与式调试的重要因素。要支持线上学习，就需要这类环境的创作者在兴趣驱动型子社区中现有的社会资源，与能够促进概念学习和分析思维技能的工具之间取得平衡。关联式学习议程旨在将“深度垂直专长”（例如兴趣驱动型社区中所具备的专长）与被视为职业发展机会来源的实践（例如编程）联系起来（Ito 等，2013年：56）。Ito 及其研究伙伴（2018年）指出，这种联系蕴含着关联式学习尚未实现的潜力（第170页）。

我们希望，未来的研究能够证明，我们所描述的动态远不止存在于 Scratch 这类编程社群中。我们认为，我们所发现的参与式调试实践，只是新媒体所促成的重要计算机媒介的社会问题解决实践中的一个例子。尽管还需要进一步研究来确认我们的发现是否具有普遍性，但我们认为，参与式调试中的社会性是更广泛意义上健康的在线问题解决文化的特征。我们相信，我们的发现以及所提出的框架，将与众多致力于支持关联式学习的情境相呼应。我们的研究表明，参与式媒体领域的文献已准备好超越这样的讨论：青少年的社交是否能够促进高阶计算思维技能的发展。我们将这项研究视为一个富有成效的起点，借此探讨如何有效支持并鼓励学习与社会互动。

致谢

我们谨向 Scratch 在线社群的成员致谢，感谢他们以富有创意的问题解决方式启发了这项研究。我们由衷感谢 Sayamindu，他与我们合作构建了本文使用的数据集，向我们推荐了相关研究，并在整个项目期间提供反馈与支持。我们还要感谢 Mitchel Resnick、Natalie Rusk，以及 New Media and Society 的编辑和匿名审稿人，感谢他们提出的宝贵反馈并给予支持。我们曾向社区数据科学集体以及国际传播协会儿童、青少年与媒体分会介绍本研究的早期版本，并感谢这两个团体的成员提供宝贵的反馈和建议。本研究获得了美国国家科学基金会的资助（拨款编号 DRL-1417663 和 DRL-1417952）。

参考资料

Aragon CR、Poon SS、Monroy-Hernández A 和 Aragon D（2009年）两个在线社群的故事：促进科学家与儿童之间的协作和创造力。第七届 *ACM* 创造力与认知会议论文集：9–18。https://doi.org/10.1145/1640233.1640239

Begel A、DeLine R 和 Zimmermann T（2010年）面向软件工程的社交媒体。*FSE/SDP*软件工程研究未来研讨会论文集：33–38。https://doi.org/10.1145/1882362.1882370

Brennan K、Monroy-Hernández A 和 Resnick, M（2010年）。《制作项目，结交朋友：在线社区作为交互媒体创作的催化剂》。《青年发展新方向》，2010年（128）：75–83。https://doi.org/10.1002/yd.377

Brennan K 和 Resnick M（2012年）。研究和评估计算思维发展情况的新框架。载于：加拿大温哥华教育研究协会。

Brennan K、Valverde A、Prempeh J、Roque R 和 Chung M（2011年）。不止是代码：社会互动对年轻人成长为交互媒体创作者的意义。载于：ED-MEDIA，葡萄牙里斯本。

Bruckman A（1998年）建构主义学习的社区支持。*ACM*计算机支持协同工作会议论文集，第47–86页。https://doi.org/10.1023/A:1008684120893

Dabbish, L、Stuart C、Tsay J 和 Herbsleb J（2012年）GitHub 中的社交编程：开放软件代码库中的透明度与协作。*ACM 2012*年计算机支持的协同工作会议论文集，第 1277–1286 页。https://doi.org/10.1145/2145204.2145396

Dasgupta S、Hale W、Monroy-Hernández A 和 Hill BM（2016年）再混合作为计算思维的途径。*2016*年 *ACM* 计算机支持的协同工作与社交计算会议论文集，第 1438–1449 页。https://doi.org/10.1145/2818048.2819984

Fields DA、PanticK、Kafai YB（2015年）“我有一个相关教程”：Scratch编程社群在线同伴支持的语言。论文集 *14th* 国际交互设计与儿童会议，第 229-238 页。https://doi.org/10.1145/2771839.2771863

Gan EF, Hill BM 和 Dasgupta S。（2018年）。性别、反馈与学习者分享其创意计算项目的决策。*ACM*计算机支持的协同工作会议论文集，第54:1–54:23页。https://doi.org/10.1145/3274323

Glaser BG 和 Strauss AL（1967年）。扎根理论的发现：定性研究策略。芝加哥：Aldine PublishingCompany。

Ito M、Baumer S、Bittanti M、boyd d、Cody R、Herr-Stephenson B、Horst H、Lange P、Mahenran D、Martínez K、Pascoe C、Perkel D、Robinson L、Sims C 和 Tripp L（2009年b）。闲逛、瞎折腾与极客范儿：生活在新媒体中的孩子。马萨诸塞州剑桥：MIT出版社。

Ito M、Gutiérrez K、Livingstone S、Penuel B、Rhodes J、Salen K，……Watkins SC（2013年）。关联式学习。加利福尼亚州欧文：数字媒体与学习研究中心。

Ito M, Martin C, Pfister RC, Rafalow MH, Salen K 和 Wortman A（2018年）在线亲和力：连接与共同兴趣如何促进学习。纽约：纽约大学出版社。

Jenkins H（2009年）直面参与式文化的挑战：*21*世纪的媒体教育。马萨诸塞州剑桥：麻省理工学院出版社。

Kafai YB（2016年8月）从计算思维到K–12教育中的计算参与。《*ACM*通讯》，*59*（8）。

Kafai YB与Burke Q（2014年）互联代码：儿童为何需要学习编程。马萨诸塞州剑桥：MIT出版社。

Kafai YB 和 Burke Q（2015）建构主义游戏：理解制作游戏对学习的益处。《教育心理学家》，*50*（4）：313–334。https://doi.org/10.1080/00461520.2015.1124022

Lave J 和 Wenger E（1991）情境学习：合法的边缘性参与 （第1版）。英国剑桥：剑桥大学出版社。

Lindlof TR 和 Taylor BC（2011）定性传播研究方法 （第3版）。加利福尼亚州千橡市：SAGE。

Martin C 和 Steinkuehler C（2010）大型多人在线游戏中的集体信息素养。《电子学习与数字媒体》，*7*（4）：355–365。https://doi.org/10.2304/elea.2010.7.4.355

Monroy-Hernández, A.（2012）。为再创作而设计：支持业余创作者在线社区。硕士论文，麻省理工学院，马萨诸塞州剑桥。取自 http://dspace.mit.edu/handle/1721.1/78202

Maxwell JA（2013年）定性研究设计：一种互动方法（第3版）。加利福尼亚州千橡市：SAGE。

Neuendorf, KA（2017）内容分析指南（第2版）。加利福尼亚州千橡市：SAGE .

Papert S（1976）一些诗意 *c*与教育设计的社会标准。检索自 http://dspace.mit.edu/handle/1721.1/62

Papert S（1980）《头脑风暴》：儿童、计算机与强有力的思想。纽约州纽约：Basic Books。

21世纪技能合作组织（2019）*21*世纪学习框架。报告。检索自 http://www.p21.org/about-us/p21-framework/260

Resnick M、Bruckman A 与 Martin F（1996）钢琴而非立体声音响：创建计算构造套件。《Interactions》，*3*（5）：40–50。

Steinkuehler C 和 Duncan S（2008）虚拟世界中的科学思维习惯。《科学教育与技术杂志》，*17*（6）：530–543。https://doi.org/10.1007/s10956-008-9120-8

Spencer SJ、Steele CM 和 Quinn DM（1999）。刻板印象威胁与女性数学表现。《实验社会心理学杂志》，*35*（1）：4–28。

Wing JM（2006）计算思维。《ACM通讯》，*49*（3）：33–35。https://doi.org/10.1145/1118178.1118215