面对全球化的挑战，新进者的能力如何增长

（原文：Growth of Newcomer Competence: Challenges of Globalization）

【摘要】

软件项目开发向境外的转移、遗产软件核心开发者的新旧交替、快速成长中的互联网公司的职员招聘以及开源项目的参与，这些问题呈现出了一个相似的挑战：如何使软件项目新进者的能力快速增长。特别是文化差异、交流的困难以及几乎没有甚至没有任何项目经验的开发者的大量涌入在实践和研究上都给软件工程提出了一些问题。例如，文化差异如何影响在项目中的学习？关于能力提升的交流存在最佳实践吗？怎样从顶级开发者的经验中学习以改进新进者的训练过程？有什么资源和工具能够帮助新进者学习得更快，变得更有效率？这些问题描述了关注项目学习的议事日程，这是要解决当前所面临的挑战所必需的。一些新兴的测量方法利用了软件代码库里面的丰富数据和基于认知科学及组织科学的理论框架，我们提出了这些方法在解决锁面临的挑战和对人们如何学习的理解上是如何可以被应用的。

【关键词】

项目能力，文化差异，交流，学习轨迹，推荐系统，通用代码库（universal repository）

1. 引言

把一个已有的项目转移到境外引发了如何提高项目新人的学习速度的问题。“所有（外包）队伍都有过相似的经历，都有过毕业生的大量涌入，他们也都在想方设法地让他们跟上脚步”一个外包经理如是说到。

同样的，一个始于90年代或更早的成熟遗留项目的核心开发者的退休，使得这个项目失去了原有的创始人，这让新人如何学习这个项目成为了一个问题。一个具有30年历史的呼叫中心（call-center）项目的顶级设计师表达了他的焦虑：“我们必须确保人们能理解我们日渐减小的专家组的技术。原开发者也许能理解每个模块（features）怎么工作，以及模块（features）之间怎么交互，但负责增添代码（adding to the code）的后继开发者并不需要完整地了解这些上下文。”同时，互联网应用的爆炸发展创造了许多成功的互联网公司，他们的快速增长要求他们的员工能快速地学习。“我们的许多项目只持续几个月，这确实要求学习者们足够的高效”淘宝（中国顶级电商）的首席设计师如是说到。

以及，开源软件开发的爆炸式增长提出了这样的问题：个人和企业如何学习去参与到开源项目当中。例如，本文的第一作者带领了一个北京大学的团队，他们在过去十年间利用JOnAS（另一个开源AS）搭建（built and later merged）了一个应用服务器（application server）。期间认识和发现了许多的问题，例如：文化、语言、时区、网络带宽，最重要的是：获得关于代码和开发实践的知识的途径。

正如Fischer[13]注意到的那样，诸如全球化、越来越多的高层次认知任务外包、针对复杂世界问题的高效参与的需求（the need to participate effectively in addressing complex world problems）此类的现象，正在改变我们思考、学习、工作以及合作的方式。这些现象提出了一个至关重要的挑战：带领项目新人跟上脚步以及提高他们的项目能力。因此，我们需要对开发者如何学习一个软件项目、如何使项目更加具有可读性（self-documenting）、如何创造革命性的资源和工具来提高新人的学习能力和生产力有更深入的理解。

人类学习已经在认知科学、发展心理学、组织管理科学有所研究。更确切地说，意识和脑、思考和学习的方式、思考和学习过程中神经的行为、能力的提升都被集中地研究。也许当前最具冲击性（striking）的是许多被发明出来的研究方法和技术以及多个分支学科发现的证据开始有交集的方式[4]。因此，我们不仅能借用这些学科的视角去理解开发者的学习（例如：Curtis[9]论证的，认识科学就是这样一个范例，它提供了最好的方式去学习和掌控影响项目表现的最大因素来源），还能帮助我们普适性地理解人类学习。

另一方面，更完善的数据和更好的分析工具已经使得这些进步能够达成。开发者项目能力的研究中最显著的方面是软件项目仓库中丰富的数据来源，从发行质量的总体度量到被IDE记录的开发者的个人行为。许多分析软件数据的方法和工具正被软件工程的社区开发出来，但许多工作都是外围的。日益丰富的用户和消费者数据收集使之成为可能，相比软件工程本身庞大的广告业、零售业及其他经济部门驱动着它的发展。显然，这些领域的方法和工具对量化和解决个别项目能力问题有帮助，例如，利用社交网络的分析方法找出最有影响力的开发者，总结开发者在解决问题时的交流途径，以及文化差异的研究。

2. 全球化视野下的学习

合法的边缘性参与方法（Legitimate Peripheral Participation approach）[18]提出：学习过程（比如，参与者在实践中的学习），一开始是边缘的，但是慢慢地在参与程度和复杂性方面增加。在软件项目中，实践就是执行常规的项目任务。它在全球化的软件开发中有许多挑战，比如，当开发者打算去整合异源工具（heterogeneous tool）产生的构件（artifacts），或当他们和在不同时区或说不同语言的人们交流。这些挑战常会导致不利的结果。比如，Mockus[22]发现：一个经验丰富的专家在境外的大型软件项目中可能需要最多六个新的开发者来代替，Herbsleb和Mockus[15]发现，同样的任务，任务分布不同地方花费的时间几乎是同一地方的三倍。

解决下面的问题需要对开发者怎样在实践中提高开发项目的能力有更好的理解。文献上，存在两种提出了的方式去应对它们。一是根据经验理解什么因素阻止了或帮助了开发者的学习。比如，Dagenias etal.[10]通过对在IBM工作的18个开发者的研究，列举了大量开发者在加入新项目时面对的困难。Begel and Simon发现：对于新开发者来说交流和了解产品的程度也形成了严峻的挑战，因为在他们正式的学习中这样的任务没有训练。一些研究甚至去找到促进全球化和学习的方式。比如，Mockus和Weiss提出将一个大型系统分离成独立的，可改变的小块，目的是减少在这些小块上开发的团队之间的依赖性。Sim和Holt【28】认识到，在项目移植的过程中，指导者教导新开发者也是很有效果的一种方式。

二是研究调查高级别的开发者和新手之间的不同。项目人员的不同是影响项目质量【9,3】最重要的一个因素。因此 ，解释差异的发展和改变，目的是希望找到方法去减小新手和高级别程序的差距，似乎是一个有前途的措施。认知科学中知识形成和发展的理论能解释这些差异的本质。因此，认知科学常被用作理解在编程方面新手和专家在脑力结构方面的不同。此外，差异被发现可能受文化【19】和激励【7】的影响。

更多地来说，各种各样的工具【28,8,1,14】通过可视化或者其他能帮助发现开发者需要的知识的方法，去增加开发者关于项目的知识。

然而，对开发者学习的理解还处于初期阶段，我们将会看到许多新的方法去加速开发者的学习，和对人类学习更深的理解。正如Fischer【13】暗示的那样，学习和教育正在经历一个意义深远的变化时期，问题的本质其实是系统性的，将通过跨学科的合作构件和解决。

3. 理解学习

为了理解并提升开发者的项目能力，我们考虑了三种类型的挑战。

第一，在境外或者外包项目中，交流、协调以及文化差异给开发者的项目能力增长带来的巨大的阻碍。因此，交流和文化是两个关键的因素，在软件项目上下文中需要对它们加以研究。

第二，研究那些最高产的开发者的学习轨迹，看他们如何获得技能、解决关键而复杂的任务，了解是什么激励了他们，也许能够将他们获得技能的方法具体化，并进一步地用于训练其他人上。

第三，从理解到实践的转换需要推荐方法、工具、资源来帮助开发者学习。

3.1 文化与交流

社会心理学、认知心理学和人类学的研究工作表明，所有的学习都发生于那些带有特定文化和社会规范及期望的设定当中，这些设定以强大的方式影响了学习和知识的转换。身处不同文化中的人在对自我、对他人、对这二者的相互依赖这几个方面的解释上有惊人的差别。这些解释能够影响，在许多情况下甚至能够决定，关于包括认知、情感和动机在内的个人经验的真正本质。

在领域内经验丰富的人已经认识到实践应当随文化而变。例如，一个把项目从英国外包到印度的外包经理说道“在应对印度团队的态度和文化时，我们必须注意到存在的不同的效果，同时又很难把这些因素考虑在内。例如，**我们希望人员裁剪得多一些，这样所有权就必须分散于开发者间，但在英国我们是不会这么做的。”**“从印度那边来看，以方便的方式迅速开发创新性产品是一个比较不现实的能力。”

交流占据了开发者的大部分时间。如Begel等人在微软的发现，绝大多数的开发者需要**探索**、参加会议、和其他人保持联系，而不仅仅是代码。但是，许多人认为协调提出了一个严重的问题。正如中国的一个从日本接外包项目的经理所抱怨的“一个非常简单的问题就（因为误解）经常要在双方之间多次问来问去。”开发者到底要什么信息？例如，Ko等人总结了两类经常被那些依赖于搭档的开发者问及的一些问题：“我的搭档们在什么呢？”和“这些错误什么情况下会发生？”开发者如何交流？Schmidt等人**提出的交互的社会机制强调了在有多个人参与的分布式活动中产品本身对工作接合的支持。** 能够支持项目的接合工作的不只是代码库，

缺陷报告的格式同样能够发挥作用。类似的，Zhou和Mockus发现一些类似**MR**/变化版本库的软件制品是很重要的，有时候这些东西甚至是供开发者交流的唯一可行机制，对与境外项目或跨代项目更是如此，因为对于许多境外项目传统的交流机会不复存在，而对于跨代项目，新一代的开发者可能无法与最初的开发者交流，因为后者可能已经辞职或者早已不存于世。交流的本质是什么？Nakakoji等人认为**交流不应被视为某种自增自长的东西，真正重要的是交流过程中体验的质量。**特别地，这一工作催生了**注意力保护**的概念，这一概念的灵感源于在互联网和**gadget**步伐下人们深陷信息与媒体之洪泛的生活方式。一个关于交流质量提升的例子如Cataldo等人所描述，此时，随着时间的变化，最高产的开发者改变了他们对电子交流工具的使用，如此一来，在协调需求和协调活动二者之间达到了更高级别的一致。

总而言之，对在全球化和多文化的软件开发环境中交流及其最佳实践的研究是一个将会产生能够提升项目能力的更好策略的话题。

3.2 高手的学习之路

我们把对于高手的认识置于重要的位置，这不是因为所有学习的人都要成为高手，而是因为这些认识为我们摸索什么才是有效的学习提供了有价值的洞见。

首先，我们需要理解高手的项目实践轨迹。具体的问题包括，一个开发者是如何从初学新手成长为行家里手，他是怎么获取专门知识的，要想成为主要贡献者他需要掌握什么的知识、又是以一种什么顺序掌握的。特别地，开源软件项目的开发者如何从初进社区新手的角色变为核心团队成员这一问题非常有趣，因为在开源项目中，并无层级组织、训练计划、集中式的环境给予他们必要的熏陶和培育。我们从任务的复杂度和主要程度为开发者提供了一些关于如何提升能力的参考，但更加广阔和深入的研究还很有必要，比如，是什么导向了那一轨迹。

然后，知道了学习者如何发展关于信息的内在连贯的结构对组织知识本质的理解有巨大的好处，而组织知识为有效的理解和思考筑立了基石。例如，正如我们在文献【29】提议的，高手和新手之间的差别可能就在于组合和应用已经学到的知识创造性地或者在新形势下去做一些更加复杂的事情的能力。从20世纪50年代起，心理学家尝试通过编程者选择测试来帮助软件工程。例如，McKeithan等人观察到高手们能够基于语言命令在语言结构中的位置而记住他们。对语言结构还不具足够心理表示的新手通常使用记忆性的技巧来记住命令的名字。Curtis认为一个人解决复杂编程任务的表现和他关于问题域的知识丰富长度有关。但是，最初的测试惨败了，原因并不是心理学的原理和技术无法胜任这个这个任务，而是心理学家**在选择测试来衡量之前**对心理和编程行为的建模就失败了。学习理论现在能够对学习者如何获得技能来搜索问题空间然后在其他问题中使用这些通用的策略做出解释。

最后，最能影响学习的因素可能是开发者的动机。特别是外包、内部和开源项目的开发者可能是被完全不同的原因所激励然后参与其中的。在外包项目内，动机可能也是千差万别的。例如，完全外包表情时间长达多年的项目可能会让外包团队中的开发者做更多的贡献，而对于只外包全部任务的一部分或者外包时间只持续几个月的项目情况则可能会不同。例如，一个来自印度某网站的、接境外项目的开发者，当他被问及什么能让他更快乐时，他说：“更多新的、有趣的功能”。这一差别似乎在激励人们学习这方面有至关重要的作用，例如，一个罗马尼亚的外包经理说“我们招那些想在一个特定地方工作的工程师，这样对他们来说是一种激励。”

总的来说，对程序员的知识基础和他们为什么学习、如何学习有了更好的理解后，在个人差异的建模上才可能取得进步。

3.3 为学习设计的推荐系统

推荐系统现在吸引了世软件工程研究者和实践者的注意。这些系统帮助开发者和管理者更好地处理在今天的软件项目中出现的巨量信息，正如专注于软件工程推荐系统的研讨会的声明中所提及的。

在提供工具以改进开发者在软件项目中的学习和表现这方面， 已经有大量实质性的工作。例如，Expertise Browser呈现了开发者和他们修改的代码之间的关系，以此帮助确定和这部分代码相关的高手、开发者技能概要；Hipikat提供了**对群组的记忆的访问，**这些群组是在软件项目开发过程中隐式形成的；Ariadne，一个Eclipse插件，给开发者提供了代码的作者和可能出现的协调问题的可视化。最近，**知识模型的分级调整了那些之后被其他人修改的代码的开发者技能，**Codebook通过挖掘各种类型的软件版本库，试图探索在人、代码、缺陷、测试用例、说明书以及相关产品的关系。

我们需要更好的工具来做项目知识的**外化与内化，**因为：

# 通常来讲，在软件项目中，知识的外化并不是最紧要的事，因此需要非入侵的个人后项目活动记录工具使得从未受激励的人的活动记录中抽取出有价值的东西成为可能。

#对一个项目进行长期学习的动机可能是缺乏的，这也使能够让新进者在数天或者数周之内而不是在数年之内保持效率的工具和方法成为必需。

按照Curtis的说法，直到所有的个人差异来源在一个数据集中结果了比较，才有可能精确地确定是哪一个差异来源是在训练项目或者工作中能否取得成功的最重要的预测指标。同时，如Mockus认为的，一些关键问题可以通过考量所有公开可用的源代码及其历史而获得解答。所有类型的代码库以及分析它们的方法在最近涌现出来，例如，一个项目的版本历史，这一切给我们提供了机会。更加多样、细致的代码库以及分析它们的方法在将来一定会出现。这个领域内的研究将使得以更加细致、更加相关的方法来衡量人类认知成为可能，将开创新的可能，以此去理解、提升开发者的项目能力，满足这个迅速变化、文化多元、全球化的软件开发所提出的不断增长的需求。

4. 结论

本文提出了：线上开发、外包开发、开源开发的流行对开发者提升他们的项目能力提出了关键的挑战。理解文化差异和沟通的本质、研究专业人士如何学习和提升他们的专业能力，以及为帮助新人学习提供资源和工具，是应对这一趋势提出的挑战的最有利方式。

从那些短时间内（at a moment’s notice）要启动新项目的跨国公司对新人训练的期望的变化上可以看到，提升这一理解有十分重要的意义。而且，提出的理解不仅仅使用软件工程中已发明的概念和方法，而且能对一个更广泛的领域做出贡献——人类如何学习。