从生态系统和社会技术角度看软件维护与发展

(特邀论文)

汤姆-曼斯
比利时蒙斯大学 COMPLEXYS 和 INFORTECH 研究所软件工程实验室
电子邮件： tom.mens@umons.ac.be

摘要 *在*这篇特邀论文 *中*，我将重点讨论作为更大生态系统一部分的 软件系统在维护和发展过程中遇到的困难。虽然并非每个软件系统都属于这一类别，但由于开源软件无处不在，软件生态系统正变得无处不在。我提出了在软件生态系统的维护和演化过程中出现的几个挑战，并论证了应如何通过采用社会技术视角和多学科混合研究方法来应对其中的一些挑战。我的论点还附有大量该领域最新研究的参考文献，尽管这些参考文献难免不完整。

***关键词--软件*生态系统；社会技术网络；跨学科研究；混合方法研究；协作软件工程；实证软件工程**

I. 导言

本文介绍了我在跨学科项目 ECOS（"开源软件生态系统生态研究"，2012-2017 年）背景下开展的软件生态系统维护实证研究中获得的一些见解，以及其他研究人员在软件生态系统演化领域报告的一些最新成果。

2015 年，我们就软件生态系统研究中的公开挑战进行了一项调查[1]。26 位受访者（活跃在该领域的研究人员）回答了与当前研究、未来趋势、工具缺乏以及软件生态系统研究人员面临的具体挑战相关的开放性问题。针对软件进化，我将更详细地报告一些已确定的挑战。许多研究人员对软件生态系统提出了不同的定义。软件生态系统通常从商业角度进行研究 [2]。例如，Jansen 等人[3] 将软件生态系统定义为 "作为一个单元运作并与软件和服务共享市场互动的一组企业，以及它们之间的关系"。然而，从软件进化的角度来看，技术性更强的观点似乎更为合适。因此，Lungu [4] 将软件生态系统定义为 "在同一环境中共同开发和演进的软件项目集合"。Manikas [5]将所有这些观点合并为一个单一的软件生态系统定义，即 "在一个共同的技术平台上，一组参与者之间的互动会产生许多不同的结果"。

业模式的驱动，并通过共生关系与其他参与方相互作用生态系统相连接，而技术平台的结构则允许不同参与方的参与和贡献"。本文将采用后一种定义。

本文其余部分的结构如下。第二节首先介绍了软件生态系统演化过程中可能出现的重要问题的一些众所周知的例子。第三节指出，为了理解和应对这些问题，我们需要从社会技术角度来看待生态系统。第 IV 节说明了使用跨学科混合方法分析软件生态系统并提出促进进化和提高可维护性的解决方案的理由。第 V 节介绍了我认为软件生态系统研究中最重要的挑战。最后，第六节为结论。

II. 当事情出错时

复杂多变的依赖关系是许多开发人员的负担[6]，有时被称为 "依赖关系地狱"。正如本节中的几个例子所示，某个阶段都经历过重要的维护能力问题。

通过对 CRAN 生态系统的贡献者进行访谈，我们发现开发人员正在努力解决他们所依赖的软件包向后不兼容更新的问题[7]。Bogart 等人通过采访 Eclipse、CRAN 和 npm 的贡献者，探讨了与破坏性更改相关的类似问题[8]。

据报道，JavaScript 软件包生态系统，尤其是由 npm 软件包管理器管理的运行环境 Node.js 的软件包，存在一些引人注目的问题[9]。自 2009 年创建以来，npm 发展非常迅速，到 2016 年 8 月已托管超过 24 万个软件包。其中一些软件包是许多其他软件包所需要的，有时甚至是极端需要。例如，软件包 isarray[1] 实质上包含三行代码：

1https://www.npmjs.com/package/isarray

978-1-5090-3806-0/16 $31.00 © 2016 IEEE
doi 10.1109/icsme.2016.19

```
var toString = {}.toString;
module.exports = Array.isArray || function (arr) {
  return toString.call(arr) == '[对象数组]';
  };
```

然而，仍有 150 多个软件包依赖于它（2016 年 8 月统计）。大卫-哈尼（David Haney）在其博客[2] 中批评了与依赖相关的问题。一个更严重的例子是软件包 leftpad，它只包含几行源代码，却有数千个依赖项目，包括 Node 和 Babel。当该软件包的开发者决定取消为 npm 发布所有模块时，这造成了严重后果，"几乎破坏了互联网"[3]。

R 生态系统是一个非常流行且发展迅速的统计计算开源环境[10]，它依赖于一个名为 CRAN[4] 的中央看门人系统。该系统正在稳步增长，包含近 9000 个软件包（2016 年 8 月统计）。有人批评它变得过于庞大[11]，其依赖管理系统也存在问题[12]。由于越来越多的人使用 GitHub 发布 CRAN 上没有的软件包[13]、[14]，这也带来了额外的困难。实证研究证实了这些可维护性问题[15]，并提出了自动化解决方案。

Gentoo 是开放源码 Linux 发行版之一，它是另一个流行生态系统的例子，在其发展历程中也曾出现过重大问题。Zanetti 等人通过分析 Gentoo 错误跟踪系统十年来的协作结构和动态，研究了社会组织结构[16]。一个中心贡献者的日益集中化，以及该贡献者的意外离职，导致了社区错误处理性能的严重破坏。本案例研究表明，除了分析生态系统的技术方面（如软件包的依赖性），解决社会方面的问题也同样重要。

社区建议或强加的期望值、价值观、工具和（版本）政策可能会因生态系统的不同而大相径庭。此外，生态系统的贡献者不一定完全了解这些实践，或者没有正确地遵守这些实践。例如，Raemakers 等人在对 Maven Central Repository 的研究中发现，推荐的语义版本化政策[5] 并不总是得到遵守，导致了许多意外的破坏性更改[17]。Businge 等人对 Eclipse 第三方插件也有类似的观察[18]。因此，重要的是要使 "社区价值观和公认的权衡明确、透明，以便解决冲突和协商与变更相关的成本"[8]。

然而，一个社区在多大程度上愿意应对可维护性问题和破坏性变更，在很大程度上取决于生态系统。以 Eclipse 生态系统为例、

2http://www.haneycodes.net/npm-left-pad-have-we-forgotten-how-to-program/
3http://uk.businessinsider.com/npm-left-pad-controversy-explained-2016-3
4https://cran.r-project.org
5详见 http://semver.org。

npm 的目标是长期稳定和保留向后兼容的变更。Eclipse API Tools 组件[6] 为这一政策提供了支持，减少了意外破坏性更改的几率。另一方面，npm 没有中央看门人，而是以快速发展为目标。其开发人员 "只要通过版本号清楚地标明，就不会太在意破坏性更改"[8]。Greenkeeper[7] 等工具有助于识别和减少破坏性变更的影响。

虽然分析和减少依赖关系和变更影响的解决方案通常是自动化工具形式的*技术*解决方案，但*社会*解决方案可能同样有效。例如，de Souza 等人提出了软件开发人员管理软件项目中的依赖关系的几种策略[19]。其中一种策略是让特定的团队成员负责与为团队提供软件组件的外部开发人员进行沟通和同步。

另一个不同性质的挑战是如何对软件生态系统本身进行重大升级或迁移，同时限制这种对生态系统的 重大干扰所带来的负面影响。一个可行的策略是在迁移过程中让两个版本的生态系统共存（Python 3.0 就是这样一个例子，它是有史以来第一个故意向后不兼容的 Python 版本，在 2020 年之前一直与 Python 2 共存[8]）。不过，有时重大升级可能会让生态系统社区产生幻灭感。Gnome 3 Linux 桌面环境（Gnome 2 的替代品）就是这种情况，它导致许多 Linux 用户停止使用 Gnome 并选择 Xfce 作为替代。[9]

III. 社会技术观点

从第二节介绍的案例研究中可以清楚地看出，为软件生态系统的可维护性提供适当的自动支持并非易事，需要同时考虑*技术*和*社会*两个维度。因此，为了充分了解软件生态系统的动态，我们应将软件生态系统视为一种*社会技术*网络，它是一种图结构，其中包含两类节点：生态系统的贡献者（人员）和这些贡献者产生的技术成果（如软件包、文档、错误报告、补丁）。这样一个图将在所有类型的节点之间产生依赖关系：贡献者将与其他贡献者交流或合作，技术人工制品可能相互依赖，贡献者将生产或修改技术人工制品。

社会技术一致性

早在 1968 年，梅尔文-康威[20] 就提出了这样的假设："任何设计系统（广义上的系统）的组织都会产生一种设计，而这种设计的结构就是对系统的复制。

6http://www.eclipse.org/pde/pde-api-tools/
7https://greenkeeper.io
8http://python3porting.com/strategies.html
9http://www.pcworld.com/article/2691192

组织的沟通结构"。[10]Cataldo 等人提出了*社会-技术一致性的*概念，以反映项目中技术依赖与社会协调之间的一致性[21]。在公司环境中，他们研究了社会技术一致性对开发人员生产率[21]和软件系统失败可能性[22]的影响。Kwan 等人 – [23] 分析了其对软件构建成功的影响。McCormack 等人提出了 "*镜像假说*"（*mirroring hypothesis*）一词来反映社会-技术一致性，并为单个公司开发的软件找到了经验证据[24]。不过，他们还与[25]一起观察到，协作式开源软件项目似乎采用了一种更加模块化且与组织结构脱钩的方法。造成这种情况的原因之一似乎是 "数字技术使新的协调模式成为可能，从而使团体偏离公司内部的传统镜像"。关于在协作式开源软件生态系统中是否可以观察到社会-技术一致性，以及这是否有任何益处，仍需进一步研究。

社会-技术软件开发网络往往具有*双重性质*：为理解或支持*技术*依赖网络而定义或使用的技术或工具往往也适用于*社会*依赖网络，反之亦然。让我通过两个具体例子来说明这一点。

技术债务与社会债务

在技术方面，*技术债务*（由 Cunningham [26] 提出）是敏捷开发社区倡导的一个著名概念。它可以被定义为 "编程中的一个概念，反映了当使用短期内易于实现的代码而不是采用最佳整体解决方案时所产生的额外开发工作"。软件重构技术可用于减少技术债务。SQALE 质量模型和方法旨在管理技术债务，工具支持也很普遍。[11]一些生态系统（如 Eclipse）更容易受到技术债务的影响，因为它们强烈要求保持向后兼容性。这种要求与快速适应新变化的需求并不总是那么容易调和。

与*技术债务*类比的是*社会债务*。Tam- burri 等人将其定义为 "与次优组织-社会结构相关的不可预见的项目成本"[27]。在公司环境中，他们发现了一系列可能成为社会债务指标的*社区气味*或社会技术反模式。例如，团队之间缺乏沟通，团队成员以自我为中心或反应迟钝，以及与文化或经验差异有关的问题。就像重构技术有助于减少技术债务一样，社会和组织结构的重组也有助于减少社会债务。不过，它应该

[10] 引自 http://www.melconway.com/Home/Conways Law.html

[11]参见 www.sqale.org。SQALE 是作为 SonarQube 的插件实现的。

显然，不能孤立地看待社会债务和技术债务。由于错综复杂的社会技术关系，以及可能存在的社会技术一致性，社会债务很可能对技术债务产生影响，反之亦然。

社会债务不仅出现在单个软件项目中，也出现在软件生态系统中。在这种高度协作的系统中，可能成为社会债务指标的气味类型可能会有所不同。例如，在第二节提到的 Gentoo 案例中，人们看到了对单一中央贡献者的日益依赖，以及 "破坏性的社会环境 "和 "日益严重的官僚主义"[16]。

社会因素与技术总线因素

所谓的巴士系数（又称卡车系数）是协同软件开发中一个众所周知但研究不足的概念。它指的是在项目陷入困境之前，需要 "被公共汽车碾过 "的团队成员数量。在极端情况下，失去一个关键成员就可能导致严重的问题，第二节中的 Gentoo 案例就说明了这一点[16]。Cosentino 等人提出了一种工具，用于自动测量存储在 Git 仓库中的项目的总线因子[28]。Avelino 等人提出并验证了一种测量总线因素的新方法[29]。采用结对编程和共享代码所有权等技术可以降低与开发人员流失有关的风险。Rigby 等人建议确定继任者（并让他们作为共同所有者参与其中），以降低与开发人员更替相关的风险[30]。他们还提出了一种方法，用于预测拥有最相关专业知识的开发人员作为继任者。Garcia 等人提出了一种测量在线贡献者动机的机器学习分类器，用于预测贡献者的流失[31]。

与社会公共汽车因素现象相对应的技术因素指的是，由于意外拆除（被公共汽车 "碾 压"）技术人工制品而导致项目出现严重问题。第二节中已经介绍了几个这样的案例。例如，从 npm 中移除软件包 leftpad 导致了大量依赖关系的中断。顺便提一下，这次移除的原因也是社会公共汽车因素的一个实例，因为 leftpad 的所有者在移除所有软件包后放弃了 npm。

IV. 跨学科研究

研究人员已经认识到软件生态系统固有的社会技术性质，许多实证分析同时考虑了技术和社会因素。这些研究成果往往具有跨学科性质，从其他领域汲取灵感或使用源自其他领域的技术。

复杂的社会技术网络

一些研究人员分析了软件项目和生态系统的社会技术网络中*复杂网络特性的*存在和演变 [32]-[35]。这些特性似乎是软件项目和生态系统开发过程*中产生*的副作用。

在这一过程中，没有任何特定的软件设计原则明确规定它们的存在。这类特性的例子包括*幂律行为*、*无标度*网络拓扑[12]和*小世界*结构[13]。Myers 和 Concas 等人提出了一些模型，试图解释复杂网络特性的形成过程[32]、[34]。如何提出与软件生态系统的社会技术网络中的协作开发过程最自然地相似的解释模型，仍然是一个公开的挑战。

*社会网络分析（SNA）*技术[36]经常被用于研究开源软件开发者社区[37]、[38]。一个著名的成功案例是使用 SNA 改进软件故障预测。例如，Pinzger 等人[39] 通过实证研究了开发者贡献碎片化对发布后故障数量的影响。他们对社会-技术网络采用了所谓的*中心度量*，发现中心度越高的模块越容易出现故障。结果表明，紧密性中心度对预测发布后失败的数量最有意义。Bird 等人[40]也做了类似的研究，他们*将*技术网络属性（哪些软件组件依赖于其他组件）与社会网络属性（谁参与了哪些组件的工作以及参与的程度）*相结合*。这样做的理由是，社会和技术两方面相互作用，会影响最终软件的质量。研究发现，与单独使用每组属性相比，结合使用技术和社会属性能更准确地预测软件故障。

在一篇教程中，Madey 提议借用*复杂系统理论*的技术来研究大规模软件开发[41]。该理论为研究难以解决的问题和因果关系不明显而难以理解的系统提供了一个科学框架。特别是，他提出将软件（生态）系统视为*复杂自适应系统*，即 "由大量组件和子系统之间的非线性时空相互作用而产生行为的系统"。除国民账户体系外，还可使用多种技术和相关工具来研究此类系统：动态系统理论、细胞自动机和基于代理的模拟。

生态、经济和其他统计措施

有人根据自然生态系统和软件生态系统之间的类比，提出了生态建模技术。在 [42] 中，我更详细地探讨了这种类比。有人提出了*生态测量方法*来研究软件生态系统的社会技术网络。在生态学研究中，人们提出了一系列*多样性指标*来衡量生态系统中物种的生物多样性。例如 Pie´lou 指数（又称物种均匀度）、

[12] 如果无论观测尺度如何，分布都是相同的，则该分布被认为是*无*尺度分布。

[13] 这意味着任意两个节点之间的平均路径长度非常小，而且存在大量的群集。

香农多样性（又称熵）和辛普森多样性指数。我在 [43] 中概述了这些多样性指数及其在软件生态系统中的应用。Posnett 等人[44]使用相对熵（又称 Kullback- Liebler 分歧）的概念来定义和衡量（开发人员和软件模块的）关注焦点及其对模块中发现的缺陷的影响。他们观察到，注意力更集中的开发人员引入的缺陷更少，而受到狭隘关注的文件比其他文件更有可能包含缺陷。Vasilescu 等人[45]探讨了软件项目中的*语言多样性*以及使用编程语言的相关风险。同一作者还使用多样性测量方法来衡量 GitHub 团队中的*性别多样性*和*任期多样性*[46]。对于性别多样性，他们使用了布劳指数（辛普森指数的变体）；对于任期多样性，他们使用了变异系数。他们发现，多样性的增加与更高的生产力相关。然而，遗憾的是，在大多数开源软件生态系统中，女性的比例仍然偏低。

除了使用多样性度量外，还从经济学研究中借鉴了*计量经济学指数*（用于评估分布中的不平等）来研究软件生态系统的社会技术特征 [47]、[48]。例如，Vasilescu 等人根据基尼指数定义了一系列指标，用于评估 Gnome 软件生态系统社会技术网络中项目和贡献者的专业化程度 [49]。在许多其他结果中，他们观察到贡献者中存在两个截然不同的亚群体（主要是编码者和翻译者），具有不同的活动模式和不同的需求。这意味着，为了向生态系统社区提供适当的工具支持，必须识别和支持其子社区的特定需求。甚至可以根据自动识别的特定贡献者档案（在其特定活动、交流和贡献模式方面），为生态系统贡献者个人提供个性化支持。

生存分析[50]就是从另一门科学借鉴来的统计技术。它起源于生物医学，用于研究影响病人或实验动物死亡时间的因素。更广泛地说，它也广泛应用于社会科学领域，用于分析各种事件（如孩子出生、转换工作、结婚或离婚等）的*发生时间*。生存分析使用诸如*删减*等概念来处理信息不完整的观察结果（如研究期间受试者辍学）。在软件生态系统方面，Samoladas 等人将这一技术用于评估开源软件项目的预期持续时间[51]。在按类型或领域划分项目后，他们使用生存函数的 Kaplan-Meier 估计来比较属于不同领域的项目的存活率。他们还观察到，项目规模越大，存活率越高。

自然语言处理

*情感分析*源自自然语言处理 (NLP) 研究领域，是研究软件生态系统社会技术方面的一种新兴技术 [52]。将情感分析技术应用于软件生态系统领域需要根据该领域调整工具和词典，以避免产生不可靠的结果[53]。在 Apache 的 JIRA 问题跟踪系统中，Ortu 等人观察到人类的情感（以存在积极或消极情绪来衡量）对生产率（以修复问题所需的时间来衡量）有影响[54]。在 Gentoo 生态系统中，Garcia 等人分析了电子邮件档案和错误跟踪器贡献者的情绪[31]。他们观察到，当贡献者在错误跟踪器中表达强烈的积极或消极情绪时，或者当他们在邮件列表中的情绪偏离预期值时，他们更有可能变得不活跃。根据这些观察结果，他们提出了一个贝叶斯分类器来预测贡献者离开项目的风险。

总之，由于软件生态系统工程在很大程度上是一种社会活动，因此需要将人类和软件工程方面结合起来。这催生了一个蓬勃发展的新研究领域，即 *协作软件工程* [55] 或 *行为软件工程* [56]。

分析和预测软件生态系统如何随时间演变，必然需要采用*混合方法*进行实证研究 [57]。通过引入社会学和心理学等其他学科的理论，并将定量方法与定性方法（如用户调查和访谈）相结合，混合研究方法综合了所使用的互补研究方法的优势，从而提高了研究结果的可信度。

V. 重温软件生态系统的挑战

本节重温了之前确定的软件生态系统研究中的一些挑战 [1]。与前几节一样，我们将主要从社会技术角度探讨这些挑战。

任何软件社区面临的主要挑战都在于*如何设计和构建*其生态系统，使其既 能促进活跃的社区，又能保持生态系统的质量、稳定性和可靠性。我们在第二节中注意到，所提出的解决方案往往具有很强的*生态系统针对性*。它们取决于社区成员的价值观和习惯，但也取决于社区使用的特定工具。例如，选择 GitHub 作为主要版本库托管服务的决定将不可避免地影响开发人员的协作方式。事实上，GitHub 偏爱（但不强求）基于拉动的开发流程，这种流程有其特定的优势和不足 [58]，[59]。在分析从 GitHub 资源库中挖掘出的软件开发数据时，还应考虑 Kalliamvakou 等人[60]所报告的诸多危险。

在许多情况下，软件生态系统的边界并不明确。有些生态系统会随着时间的推移自然*出现*和发展。Blincoe 等人在研究 GitHub 托管项目时观察到了这种现象[61]。他们提出了一种方法，根据项目间的技术依赖关系来识别 GitHub 中的生态系统，并发现大多数已识别的生态系统都聚集在单个项目周围。此外，这些生态系统往往相互关联。*检测和支持此类"临时"新兴生态系统的*自动化工具支持是非常可取的。还需要更好地了解此类生态系统的社会技术网络，以了解生态系统是如何出现并随着时间的推移变得越来越受欢迎和成功的。

软件生态系统一旦建立（或自然形成），下一个挑战显然就是*如何发展和维护*它。本文前面的章节已经详细讨论了这一挑战。重点主要是管理属于同一生态系统的依赖软件组件之间的断裂变化，以及如何将生态系统迁移到新的主要版本。

*比较软件生态系统*以了解它们之间的差异是一项仍未解决的挑战。正如 Vasilescu 等人[49]所观察到的，这种差异甚至会出现在同一生态系统的不同子社区之间。了解这些差异对生态系统的所有贡献者都很有价值。对于研究人员来说，研究每个生态系统的特殊性如何影响其可维护性和进化非常重要。

大数据

从研究角度看，最重要的无疑是*大数据*挑战[62]。传统上，这一挑战被分为四个方面，通常称为大数据的 4 个 V：数量（Volume）、种类（Variety）、真实性（Veracity）和速度（Velocity）。[14]关于第一个 V，软件生态系统包括海量数据（通常达到 TB 级）值得分析。因此，软件生态系统分析面临着与许多其他科学领域的大数据类似的问题。

软件生态系统研究人员还需要应对数据源的*多样性*或异质性，这些数据源可以是结构化的（如程序）、半结构化的（如电子邮件）或非结构化的（如未格式化的文本）。能说明所用数据源种类繁多的例子包括：版本控制库（包含源代码和其他软件制品）、问题跟踪器（包含错误报告、功能再查询及其处理方式）、邮件列表（捕捉生态系统社区成员（如开发人员和用户）之间的交流）、问答网站（回答开发人员或用户提出的问题）、Twitter 交流、调查和访谈。Socha 等人甚至更进一步，提出了广域人种学，将视频记录等各种传感器的数据结合起来、

14www.ibmbigdatahub.com/infographic/four-vs-big-data

屏幕捕捉软件、录音、照片以及软件开发人员现场合作的实地记录[63]。

大数据的第三个层面涉及其*真实性*。需要分析的数据往往部分不完整、不确定或不一致。例如，Bruntink 报告了他在分析大型开源项目在线索引和分析平台 Ohloh 时遇到的一系列与不可信、不一致或缺失数据有关的问题[64]。其他数据源也存在类似问题。

在某些情况下，*速度*（最后一个 V）也是一个问题，因为新数据出现的速度超过了处理速度。例如，GitHub 上的新提交每秒都会出现数次。对于实证研究来说，这个问题可能较小，因为在实证研究中，数据通常是离线分析的。但对于支持软件生态系统社区活动的自动化工具（如基于网络的仪表盘）来说，为了做出明智的决策，依赖最新数据可能非常重要。

大数据也可以从*深度学习*算法等技术中获益，为更好地理解和支持不断发展的软件生态系统带来新的机遇。White 等人[65]提出了深度学习软件库方向的第一步。Wang 等人将深度学习用于改进软件缺陷预测[66]，Corley 等人将深度学习用于提高开发人员在特征定位方面的效率[67]。

隐私与可复制性

另一个非常重要的问题是*保护隐私*。软件生态系统数据通常包含参与开发和维护生态系统组件的贡献者信息。研究人员对此类数据很感兴趣，例如，他们想了解个人贡献者的专业知识和工作模式会如何影响整个软件生态系统。有人提出了身份合并等技术来促进这种分析 [68]、[69]。不过，必须保护这些个人的隐私。例如，2016 年 5 月，欧盟发布了一项新的个人数据保护指令，赋予公民对其个人数据使用的更多控制权。收集和管理个人信息的个人或组织必须保护个人信息不被滥用，并且必须尊重欧盟法律所保障的数据所有者的某些权利。要满足这些法律的要求是相当具有挑战性的，因为从软件生态系统（如电子邮件通信）收集的数据中推断出个人信息（如性别、个性特征，甚至信仰和观念）在技术上是可能的。因此，需要开发并采用适当的技术来保证匿名性。Fung 等人介绍了隐私保护数据发布方面的研究成果和未来方向[70]。Malik 等人概述了隐私保护数据挖掘工具和技术，并提出了未来的研究方向[71]。

既要保护隐私，又要满足研究人员的需要，使他们的数据和研究成果能够为公众所知晓，这仍然是一项挑战。

为达到*重现*目的，其他研究人员可公开访问研究结果[72], [73]。一个直接相关的实际挑战是*共享这些数据*的困难，因为要使用各种不同的格式、工具和操作系统，而且由于我们处理的是海量数据，还存在存储问题。研究界正在提出解决这些问题的部分方案，如使用 Docker 容器 [74] 或专用的实验软件工程研究工作台，如 TraceLab [75]。

VI. 结论

在本文中，我探讨了软件生态系统这一活跃的研究领域，特别是其维护和演进所面临的挑战。典型的问题包括：由于软件组件的依赖关系中存在向后不兼容的更新，导致软件组件被破坏；由于所有这些相互依赖关系，很难对整个生态系统进行重大升级。每个软件生态系统都有自己的特点，需要定制工具来解决这些问题。

然而，实证研究表明，要提供这样的解决方案，不仅要考虑生态系统动态的技术方面，还要考虑其贡献者群体的社会或人文方面。这种社会技术观点开辟了协作软件工程研究的新领域，将传统软件工程研究与计算机支持的协作工作的思想结合起来。

除此之外，有关软件生态系统的社会技术动态的实证研究正日益成为跨学科研究，借鉴了许多其他科学学科的技术和理念。例如，社会科学中社会网络分析技术的使用、生态学中多样性指标的使用、经济学中计量经济学指数的采用、医学中生存分析技术的使用等等。软件生态系统的社会技术性质还要求采用混合研究方法，将定量实证方法与定性方法（如社会学和心理学中使用的方法）相结合。

虽然有关软件生态系统进化的社会技术研究非常活跃，本文引用的许多最新参考文献也证明了这一点，但我也发现了许多尚未解决的挑战。我希望该领域的研究人员能从本文中得到启发，接受其中的一些挑战，从而进一步推动这一重要研究领域的发展。

鸣谢

本研究是在澳大利亚研究理事会研究项目 AUWB-12/17-UMONS- 3 的背景下进行的。我感谢我的研究合作者 Bogdan Vasilescu、Alexander Serebrenik、Alexandre Decan、Maelick Claes 和 Mathieu Goeminne 对本文早期版本提出的非常有用的反馈意见。

参考资料

[1] A.Serebrenik 和 T. Mens，"软件生态系统再搜索的挑战"，*欧洲软件架构研讨会*，2015 年，第 40:1-40:6 页。

[2] S.S. Jansen, M. Cusumano, and S. Brinkkemper, Eds., *Software Ecosystems：分析和管理软件业的业务网络》*。Edward Elgar，2013 年。

[3] S.Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community：A research agenda for software ecosystems," in *Int'l Conf.软件工程》*，2009 年 5 月，第 187-190 页。

[4] M.Lungu, "Towards reverse engineering software ecosystems," in *Int'l Conf.软件维护》*，2008 年，第 428-431 页。

[5] K.Manikas 和 K. M. Hansen，"软件生态系统：系统文献综述》，《*系统与软件》*，第 86 卷，第 5 期，第 1294- 1306 页，2013 年 5 月。5, pp.

[6] C.Artho, K. Suzaki, R. D. Cosmo, R. Treinen, and S. Zacchiroli, "Why do software packages conflict?" in *Int'l Conf.*2012 年 6 月，第 141-150 页。

[7] T.Mens, "Anonymized e-mail interviews with R package maintainers active on CRAN and GitHub," University of Mons, Tech.Rep., 2015.[在线]。Available: http://arxiv.org/abs/1606.05431

[8] C.Bogart, C. Ka¨stner, J. Herbsleb, and F. Thung, "How to break an API：三个软件生态系统中的成本协商和社区价值"，*国际软件工程基础研讨会，2016 年。软件工程基础》*，2016 年。

[9] E.Wittern, P. Suter, and S. Rajagopalan, "A look at the dynamics of the JavaScript package ecosystem," in *Int'l Conf.Mining Software Repositories*.ACM，2016 年，第 351-361 页。

[10] D.M. Germa´n、B. Adams 和 A. E. Hassan，"R 软件生态系统的演变"，《*欧洲*软件维护与再工程会议》，2013 年，*第 243-252* 页。2013 年，第 243-252 页。

[11] K.Hornik, "Are there too many R packages?" Austrian Journal of Statistics, vol. 41, no.*奥地利统计杂志*》，第 41 卷，第 1 期，第 59-66 页，2012 年。

[12] J.Ooms, "Possible directions for improving dependency versioning in R," *R Journal*, vol. 5, no. 1, pp.

[13] A.Decan, T. Mens, M. Claes, and P. Grosjean, "On the development and distribution of R packages：对 R 生态系统的实证分析》，《*欧洲软件架构研讨会*》，2015 年，第 41:1-41:6 页。

[14] --, "When GitHub meets CRAN: An analysis of inter-repository package dependency problems," in *Int'l Conf.IEEE, Mar*. 2016*, pp*.IEEE，2016 年 3 月，第 493-504 页。

[15] M.Claes, T. Mens, and P. Grosjean, "On the maintainability of CRAN packages," in *Int'l Conf.IEEE, 2014, pp*.IEEE, 2014, pp.

[16] M.M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "The rise and fall of a central contributor：Gentoo 社区中社会组织和绩效的动态"，*软件工程的合作与人性化方面国际研讨会*，2013 年 5 月、pp.49-56.

[17] S.Raemaekers, A. van Deursen, and J. Visser, "Semantic versioning versus breaking changes：对 Maven 代码库的研究"，《源代码分析与*处理工作会 议*》，2014 年 9 *月，第 215-224 页。源代码分析与操纵》*，2014 年 9 月，第 215-224 页。

[18] J.Businge, A. Serebrenik, and M. G. J. van den Brand, "Eclipse API usage: the good and the bad," *Software Quality*, vol. 23, no. 1, pp.

[19] C.R. B. de Souza 和 D. F. Redmiles，"软件开发人员对依赖关系和变更管理的实证研究"，*国际*软件工程*大会。Software Engineering*.ACM, 2008, pp.

[20] M.Conway, "How do committees invent?" Datamation Journal, pp.*Datamation Journal》*，第 28-31 页，1968 年 4 月。

[21] M.Cataldo, J. D. Herbsleb, and K. M. Carley, "Socio-technical congruence：A framework for assessing the impact of technical and work dependencies on software development productivity," in *Int'l Symp.Empirical Software Engineering and Measurement*. ACM , 2008、pp.2-11.

[22] M.Cataldo, A. Mockus, J. A. Roberts, and J. D. Herbsleb, "Software dependencies, work dependencies, and their impact on failures," *IEEE Transactions on Software Engineering*, vol. 35, no. 6, pp.

[23] I.Kwan, A. Schroter, and D. Damian, "Does socio-technical congruence have an effect on software build success?软件项目协调研究"，*IEEE Trans.Soft.Eng.*, vol. 37, no.3，第 307-324 页，2011 年 5 月。

[24] A.MacCormack, C. Baldwin, and J. Rusnak, "Exploring the duality between product and organizational architectures：镜像 "假设的检验》，《*研究政策*》，第 41 卷，第 8 期，第 1309 - 1324 页，2012 年。

[25] L.L. J. Colfer 和 C. Y. Baldwin，"镜像假设：理论、证据与例外"，哈佛商学院，Tech.Rep. Finance Working Paper No. 16-124, May 2016.

[26] W.Cunningham, "The WyCash portfolio management system - experience report," in *OOPSLA '92*, Mar. 1992.

[27] D.D. A. Tamburri, P. Kruchten, P. Lago, and H. van Vliet, "Social debt in software engineering: insights from industry," *J. Internet Services and Applications*, vol. 6, no. 1, pp.

[28] V.Cosentino, J. L. C. Izquierdo, and J. Cabot, "Assessing the bus factor of Git repositories," in *Int'l Conf.软件进化、分析和再工程*。IEEE，2015 年 3 月，第 499-503 页。

[29] G. Avelino、L. Passos、A. Hora 和 M. T. Valente，"估算卡车系数的新方法"，*国际程序理解大会*，2016 年。*程序理解*，2016 年。

[30] P.P. C. Rigby, Y. C. Zhu, S. M. Donadelli, and A. Mockus, "Quantifying and mitigating turnover-induced knowledge loss: case studies of Chrome and a project at Avaya," in *Int'l Conf.软件工程*。ACM , 2016, pp.

[31] D.Garcia, M. S. Zanetti, and F. Schweitzer, "The role of emotions in contributors activity：Gentoo 社区案例研究"，*国际*云计算与绿色计算*大会*，2013 年 9 月，第 410-417 *页。云计算与绿色计算》*，2013 年 9 月，第 410-417 页。

[32] C.R. Myers, "Software systems as complex networks：Structure, function, and evolvability of software collaboration graphs," *Physical Review E*, vol. 68, no.4, 2003.

[33] A.Potanin, J. Noble, M. Frean, and R. Biddle, "Scale-free geometry in oo programs," *Commun.ACM*, vol. 48, no.5, pp.

[34] G.Concas、M.Marchesi、S.Pinna 和 N.Serra，"大型面向对象软件系统中的幂律"，*IEEE Trans.Soft.工程*》，第 33 卷，第 10 期、pp.687-708, 2007.

[35] P.Louridas, D. Spinellis, and V. Vlachos, "Power laws in software," *ACM Trans.*18, no. 1, pp.

[36] J.斯科特，《*社会网络分析*》。 SAGE, 2012.

[37] G. Madey, V. Freeh, and R. Tynan, "The open source software development phenomenon：An analysis based on social network theory," in *Americas Conf.信息系统*》，2002 年。

[38] J.Xu, Y. Gao, S. Christley, and G. Madey, "A topological analysis of the open source software development community," in *Hawaii Int'l Conf.系统科学*，2005 年。

[39] M.Pinzger, N. Nagappan, and B. Murphy, "Can developer-module networks predict failures?" in *Int'l Symp.ACM*, 2008, pp.ACM, 2008, pp.

[40] C.Bird, N. Nagappan, H. Gall, B. Murphy, and P. Devanbu, "Putting it all together：Using socio-technical networks to predict failures," in *Int'l Symp.软件可靠性工程*。IEEE 计算机学会，2009 年，第 109-119 页。

[41] G. Madey 和 S. Kaisler，"[教程：]复杂自适应系统：Emer- gence, self-organization, tools, analysis and case studies," in *Hawaii Int'l Conf. on System Sciences*, 2009.

[42] T.Mens, M. Claes, P. Grosjean, and A. Serebrenik, "Studying evolving software ecosystems based on ecological models," in *Evolving Software Systems*.Springer, 2014, pp.

[43] T.Mens, "Evolving software ecosystems A historical and ecological perspective," in *Dependable Software Systems Engineering*, ser.北约科学促进和平与安全丛书，D：信息与通信安全。IOS Press, 2015, vol. 40, pp.

[44] D.Posnett, R. D'Souza, P. Devanbu, and V. Filkov, "Dual ecological measures of focus in software development," in *Int'l Conf.软件工程*。IEEE，2013 年，第 452-461 页。

[45] B.Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "The Babel of software development： Linguistic diversity in open source," in *Int'l Conf.Social Informatics*.Springer, 2013, pp.

[46] B.Vasilescu, D. Posnett, B. Ray, M. G. van den Brand, A. Serebrenik、P.Devanbu 和 V. Filkov，"GitHub 团队中的性别和任期多样性"，*国际计算机系统*中*的人为因素大会。ACM*, 2015, pp.ACM, 2015, pp.

[47] R.Vasa, M. Lumpe, P. Branch, and O. Nierstrasz, "Comparative analysis of evolving software systems using the Gini coefficient," in *Int'l Conf.软件维护*》，2009 年，第 179-188 页。

[48] E.Giger, M. Pinzger, and H. Gall, "Using the Gini coefficient for bug prediction in Eclipse," in *Int'l Workshop on Principles of Software Evolution*.ACM，2011 年，第 51-55 页。

[49] B.Vasilescu, A. Serebrenik, M. Goeminne, and T. Mens, "On the variation and specialisation of workload：Gnome 生态系统社区案例研究》，《*实证软件工程*》，第 19 卷，第 955-1008 页，2014 年 8 月。

[50] D.G. Kleinbaum and M. Klein, *Survival Analysis：自学教材*》，第 3 版，施普林格出版社，2012 年。

[51] I.Samoladas, L. Angelis, and I. Stamelos, "Survival analysis on the duration of open source projects," *Information & Software Technology*, vol. 52, no.

[52] B.Liu, "Sentiment analysis and opinion mining," *Synthesis Lectures on Human Language Technologies*, vol. 5, no.

[53] N.Novielli, F. Calefato, and F. Lanubile, "The challenges of sentiment detection in the social programmer ecosystem," in *Int'l Workshop on Social Software Engineering*. ACM , 2015, pp.

[54] M.Ortu, B. Adams, G. Destefanis, P. Tourani, M. Marchesi, and R.托内利："恶霸的工作效率更高吗？情感与问题解决时间的经验研究"，*国际软件库挖掘大会。Mining Software Repositories*.IEEE, 2015, pp.

[55] I.Mistr´ık, J. Grundy, A. Hoek, and J. Whitehead, Eds.施普林格出版社，2010 年。

[56] P.Lenberg, R. Feldt, and L. G. Wallgren, "Behavioral software engineering：定义与系统文献综述》，《*系统与软件*》，第 107 卷，第 15 - 37 页，2015 年。

[57] R.R. B. Johnson 和 A. J. Onwuegbuzie，"混合方法研究：A research paradigm whose time has come," *Educational Researcher*, vol. 33, no. 7, pp.

[58] G. Gousios, A. Zaidman, M.-A. Storey, and A. van Deursen, "Work practices and challenges in pull-based development: A. van Deursen.Storey, and A. van Deursen, "Work practices and challenges in pull-based development：The integrator's perspective," in *Int'l Conf.软件工程*。IEEE Press, 2015、pp.358-368.

[59] G. Gousios, M. -A.Storey, and A. Bacchelli, "Work practices and challenges in pull-based development：The contributor's perspective," in *Int'l Conf.软件工程*。ACM, 2016, pp.

[60] E.Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining GitHub," in *Int'l Conf.Mining Software Repositories*.ACM, 2014, pp.

[61] K.Blincoe, F. Harrison, and D. Damian, "Ecosystems in GitHub and a method for ecosystem identification using reference coupling," in *Int'l Conf.挖掘软件库*》，2015 年。

[62] Y.Demchenko, P. Grosso, C. De Laat, and P. Membrey, "Addressing big data issues in scientific data infrastructure," in *Collaboration Technologies and Systems*, May 2013, pp.

[63] D.D. Socha、R. Adams、K. Franznick、W.-M.Roth, K. Sullivan, J. Tenen- berg, and S. Walter, "Wide-field ethnography： Studying software en- gineering in 2025 and beyond," in *Int'l Conf.软件工程*。ACM , 2016, pp.

[64] M.Bruntink，"Ohloh 软件进化数据的初步质量分析"，*ECEASST*，第 65 卷，2014 年。[Online].Available: http://journal.ub.tu-berlin.de/eceasst/article/view/906

[65] M.White, C. Vendome, M. Linares-Va´squez, and D. Poshyvanyk, "Toward deep learning software repositories," in *Int'l Conf.Mining Software Repositories*.IEEE Press，2015 年，第 334-345 页。

[66] S.Wang, T. Liu, and L. Tan, "Automatically learning semantic features for defect prediction," in *Int'l Conf.软件工程*。ACM, 2016、pp.297-308.

[67] C.S. Corley, K. Damevski, and N. A. Kraft, "Exploring the use of deep learning for feature location," in *Int'l Conf.软件维护与进化》*，2015 年 9 月，第 556-560 页。

[68] M.Goeminne and T. Mens, "A comparison of identity merge algorithms for software repositories," *Science of Computer Programming*, vol. 78, no. 8, pp.

[69] E.Kouters, B. Vasilescu, A. Serebrenik, and M. G. J. van den Brand, "Who's who in Gnome: using LSA to merge software repository identities," in *Int'l Conf.软件维护*。IEEE, 2012, pp.

[70] B.B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: *A survey* of recent developments," *ACM Comput.Surv.*, v o l . 42, no.4, pp.

[71] M.M. B. Malik, M. A. Ghazi, and R. Ali, "Privacy preserving data mining techniques：当前形势与未来前景"，*国际计算机与通信技术大会，2012 年 11 月，第 26-32 页。计算机与通信技术》*，2012 年 11 月，第 26-32 页。

[72] F.J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in *empirical* software engineering," *J. Empirical Software Engineering*, vol. 13, no. 2, pp.

[73] J.M. Gonza´lez-Barahona and G. Robles, "On the reproducibility of *empirical* software engineering studies based on data retrieved from development repositories," *J. Empirical Software Engineering*, vol. 17, no.

[74] J.Cito, V. Ferme, and H. C. Gall, "Using Docker containers to improve reproducibility in software and web engineering research," in *Int'l Conf.Web Engineering*.Springer, 2016, pp.

[75] B.Dit, E. Moritz, M. Linares-Va´squez, D. Poshyvanyk, and J. Cleland-Huang, "Supporting and accelerating reproducible empirical research in software evolution and maintenance using TraceLab Component Library," *J. Empirical Software Engineering*, vol. 20, no.5, pp.