DevOps 自动化支持工具调研
（技术报告）

李杉杉

顾胜晖

张贺

南京大学 软件学院

2018.06.10

摘要

作为敏捷文化的延续，DevOps 强调自动化并在实践中依赖自动化工具的支持。DevOps 发展至今，自动化支持工具数量与日俱增，且种类繁多，因此理解和认识其现状有助于提高 DevOps 自动化实践水平。通过对 DevOps 自动化支持工具进行分类并表征工具之间的关系，本文旨在刻画工具支持下的 DevOps 自动化实践现状。为了尽可能保证证据的全面性，本研究采用了多元文献评价（MLR）的方法，即首先通过进行系统化文献评价（SLR）的方法来识别并分析学术文献中的相关证据；然后采用灰色文献评价（GLR）的方法，在业界实践者较为活跃论坛——Stack Overflow，通过数据挖掘技术收集相关数据并进行分析。另外，本研究还补充收集并分析了来自该领域权威组织发布的相关报告以及从工具的官方网站收集的相关数据。通过对数据的整合分析，本文设计了工具元模型，并选择了具有代表性的工具集合给出概览图谱来刻画其特征属性及其之间的关系映射，为实践者在对比并选择工具时提供参考。此外 ,本文还选择了两个典型的工具案例来阐述它们是如何支持 DevOps 实践并实现 DevOps 目标的。本文通过生成的映射图谱和案例研究共同报告了 DevOps 自动化支持工具的现状，在理解 DevOps 自动化实践方面实现了突破性进展。

关键词: DevOps; 自动化; 工具; 多元文献评价; Stack Overflow

Abstract

Emerging from the agile culture, DevOps extremely emphasizes automation and heavily relies on tools in practice. Given the rapidly increasing number and diversity of the tools for DevOps, systematic understanding of the-state-of-art of DevOps-friendly tools will help to improve the automation practice of DevOps. This study aims to portray a landscape for understanding the state-of-the-practice of DevOps by categorizing the supporting tools and characterizing their relationships. To help collect as much evidence as possible, we employed a Multivocal Literature Review (MLR) by conducting an adapted version of Systematic Literature Review (SLR) to identify and synthesize academic publications and performing a Gray Literature Review (GLR) for data mining in a practitioner's forum, Stack Overflow. This study is supplemented by the reports from professional organizations and the confirmed data from the official website contents of tools for the generation of the state report. On the basis of a metamodel, we present a landscape with a selective set of DevOps tools to holistically portray their characteristics and relationships, develops mappings between DevOps tools and different attributes to provide practitioners with a reference for preliminary comparison of these tools. Two representative cases were selected to elaborate how they support DevOps practices and achieve the DevOps goals. This study is able to offer a breakthrough for understanding the practical DevOps through the generated landscape, mappings and cases which jointly reports the state of DevOps tooling.

Key words: DevOps, Automation, Tool, Multivocal Literature Review, Stack Overflow

目录

1 背景及相关工作介绍

DevOps 是一种文化理念，业界对于该文化的范围尤其是自动化实践有着不同的理解与解释[1]。它旨在将开发（Development）无缝延伸到运维（Operation）阶段，并打破传统软件过程中这二者之间的壁垒，简化软件生命周期的各个环节，促进软件的健康、快速、持续交付。

DevOps 的发展可以追溯到敏捷文化的产生。为了应对快速且不可预测的业务需求变动，适应激烈的市场竞争，在过去的数十年中，有多种敏捷方法被提出，如 Scrum， XP，DSDM，Lean，Crystal 等，并被应用于软件开发实践中[2][3]。敏捷的宗旨是快速迭代，它能够加快发展进程(包括需求分析，设计，编码和测试)，降低项目失败率，提高客户满意度。特别是，敏捷方法通常以持续实施和反馈的迭代过程为特征，使开发团队能够跟上动态需求，最终使企业能够及时沟通和响应利益相关者的需求[4][5] 。

然而，敏捷方法只关注开发过程的迭代，而忽略了整个软件生命周期中的其他阶段。传统软件企业中普遍存在开发和运维团队之间的信息鸿沟问题，开发团队希望快速交付，而运维团队则要求更高的质量，两个团队之间缺乏沟通与协作，团队之间存在的目标差异往往会导致软件交付过程在开发和运维之间壁垒高筑，严重影响软件产品高质量快速交付的业务目标实现。具体来讲，开发团队主要关注的是敏捷所覆盖的狭义的开发过程，而运维团队则负责配置，部署和管理前者开发的软件系统或服务，此外，还可能会奔赴客户现场处理各种问题[6]。开发和运维团队之间的这种分工上的割裂极有可能产生不可调和的矛盾[7]。例如，一旦琐碎耗时的运维工作的进度无法跟上敏捷开发的步伐，整个软件项目将不得不延期，更不用说去填补随着项目规模增长而致使的日益增大的差距。事实上，从实践者的角度来看，按结构和功能将开发和运维区分开会对软件项目产生明显的负面影响[8]。

继瀑布开发、敏捷开发之后，近年来，衍生于企业系统管理（Enterprise Systems Management, ESM）和敏捷系统管理的 DevOps[10],[11]的 DevOps 文化成为又一新兴的软件开发理念和愿景。DevOps 是敏捷和精益原则在整个软件交付过程上的延伸[12]， 换句话说，它旨在通过一系列文化及技术手段(尤其是自动化 IT 工具链的打通)打破开发和运维团队之间的壁垒，改善团队之间的协作关系，实现更加频繁快速、可靠的软件产品交付[6]。DevOps 的影响正在工业界和学术界中蔓延开来。在实践中，DevOps 社区开始构建像 Vagrant 这样的开源工具，它能够和现有的配置管理工具协作，如 Puppet 和 Chef；在理论上，自 2012 年发布了《什么是 DevOps？》以来，出现了许多与 DevOps 相关的书籍和报告[17]。

目前的 DevOps 已经不单纯是一个口号，很多知名企业已经着手实践 DevOps 并从中获益，如 Facebook, Yahoo 和 Netflix 等[6,8,13,14]。根据 CA Technologies 的一项关注应用经济和 DevOps 在 2014 年的作用研究[15]，在 1425 名受访高管中有 88％声称他们已经采用了 DevOps 或计划采用 DevOps。参考 Puppet Labs 的《2017 State of DevOps Report》[16]，采用 DevOps 的高绩效团队比低绩效团队在 IT 性能上有显著提升，部署频率平均高 46 倍，变更交付速度平均提高 440 倍，故障恢复时间（MTTR）平均缩短 96 倍，变更失败率评价降低 5 倍。其中高绩效团队每天部署超过 4 次，变更交付时长和故障恢复时间均少于 1 小时，变更失败率低于 7.5%。而根据《DevOps 中国•2018 年度调查报告》□，就单项指标而言，国内已有 33.8%的团队可以实现一天多次部署，30.1%的团队可以在 1 小时内恢复故障，仅有 14.7%的团队的变更交付时长少于 1 小时，73.5%的团队可以将变更失败比例控制在 10%以内。此外，高绩效组织的员工对他们的公司更满意，并且由于更高的 IT 绩效和更少的问题，他们可以将更多的时间花在新工作上。DevOps 现在不再仅仅是一个流行语。从业者转向 DevOps 不仅是为了提高组织绩效、增加收益[17]，还因为这种优秀的文化能够提升他们的工作质量[9]。

DevOps 概念早在 2009 年就被提出来，伴随着越来越多的新技术(如微服务结构、容器)和自动化工具的普及，近两年才逐渐受到企业重视并开始被广泛实践。研究表明，DevOps 持续高效高质量的交付有赖于高度的自动化（工具）的支持[18]，自动化也是获得快速反馈的关键[19]。鉴于 DevOps 自动化支持工具涉及多个

□ http://softeng.nju.edu.cn/devops/

阶段，数量和种类繁多，理解和选择合适的工具往往对 DevOps 实践者来讲至关重要，但往往具有挑战性[9]，[20]-[22]。目前学术界中对于 DevOps 自动化工具的研究仍然很少，且都存在一定的不足。Akshaya 等人根据开发和运维涉及的阶段划分(构建、测试、监控、日志等)，给出了一系列 DevOps 相关工具的简单介绍[23]。Vaasanthi 等人仅仅对比研究了 DevOps 自动化构建工具。在这些针对 DevOps 某个或多个阶段工具的研究中，都仅仅是从功能和特点上对工具进行简单介绍[19]，缺乏深入的分析。Ghantous 等人[24]采用了系统文献评价的方法，从相关研究中筛选了 18 种 DevOps 工具。 然而，该研究都给出了初步分类和较少的工具集，未对工具特性及其之间的关系进行详细分析。工业界中，XebiaLabs 给出了 DevOps 相关工具的元素周期表[20]，包含一百多种不同类型的工具，但并没有针对工具的类型换分和工具之间关系等问题给出详细介绍。

相比之下，本文采用了系统化文献评价（SLR）和灰色文献评论（MLR）相结合的多元文献评价方法[25]，对识别和筛选的相关工具进行分类，对其之间的关系进行梳理，突破性地通过元模型和图谱来初步描绘工具的特征及关系，大大降低了实践者理解 DevOps 工具的复杂度。此外，本文从所选的工具集合中选择了比较典型的两个来分析其演变过程，以展示这些工具如何实现 DevOps 目标的。

本研究的贡献主要有以下四个方面：

- 基于工业界和学术界所收集的证据确定并生成了具有代表性的 DevOps 自动化支持工具集。
- 提出 DevOps 自动化支持工具元模型，以帮助理解本研究要呈现的 DevOps 工具属性和关系。
- 基于证据的整合与分析，构建 DevOps 自动化支持工具属性及关系映射图谱，为实践者在 DevOps 工具比较和选择过程中提供初步参考，其中所选的 DevOps 工具被映射到不同的属性维度，例如，阶段（Phase）和集成度（Integrations）。
- 对 DevOps 工具集中的两个典型工具进行案例分析，详细说明在 DevOps 实践中其是如何演化并实现自身价值的。

该研究的目标受众包括 DevOps 实践者，研究人员和教育工作者。除了筛选的典型的 DevOps 工具集合，DevOps 实践者将通过本研究提供的工具属性和关系图谱更深入地了解 DevOps 自动化支持工具的特性和其之间关系，并根据自己的需求，在 DevOps 工具选择与比较时将其作为初步参考；此外，本报告通过多元文献评价的方法来研究 DevOps 自动化支持工具的现状和演变过程，从而可以帮助研究的人员发现潜在的研究机会；同时，在本报告的帮助下，教育工作者不仅可以将其作为 DevOps 的教学参考材料，还可以从中选择合适的工具构建 DevOps 教学的实践环境。

本文剩余章节的结构组织如下。第 2 节阐述了开展本研究所采用的方法，包括研究问题和详细的研究方法介绍。第 3 节给出本研究的结果，包括识别并选择的 DevOps 工具集合，所设计的元模型以及工具特性关系映射图谱。在第 4 节分析了本研究选择的两个典型的工具案例。第五 5 节讨论了关于 DevOps 工具的一些其他发现，同时讨论了本研究的局限性。第 6 节总结本文工作以及未来计划开展的工作。

2 研究方法

本小节详细描述了进行本研究并生成该研究报告所遵循的研究方法。

2.1 研究问题

DevOps 旨在重塑软件工程，实现的手段是通过高度自动化、高质量的集成方法和工具来打通开发和运维之间的壁垒。本研究的主要目的是从工具的角度来刻画 DevOps 支持工具及其实践的现状。为了实现这一目标，本研究拟解决以下四个更具体的研究问题。

- 研究问题一：DevOps 的自动化支持工具有哪些？
- 研究问题二：从 DevOps 自动化工具中可以识别出哪些属性和特征？
- 研究问题三：所识别的 DevOps 自动化支持工具之间存在什么样的关系？
- 研究问题四：DevOps 自动化工具是如何演化以支持 DevOps 目标的？

由于 DevOps 运动起源于实践者的推动，因此迄今为止与 DevOps 相关的研究文献并不是很多。为了全面

解决研究问题，我们采用了一种混合的研究方法（如图一所示），即多元文献评价（Multivocal Literature Review，MLR），结合了系统文献评价和灰色文献评价[25]，并深入研究了所选定的工具。为了更加全面、真实地理解 DevOps 工具现状，本研究搜索并分析了学术领域发表的相关研究以及三个主要来源的（在线）灰色文献，包括论坛，专业机构的技术报告和特定工具的官方网站发布的报告。

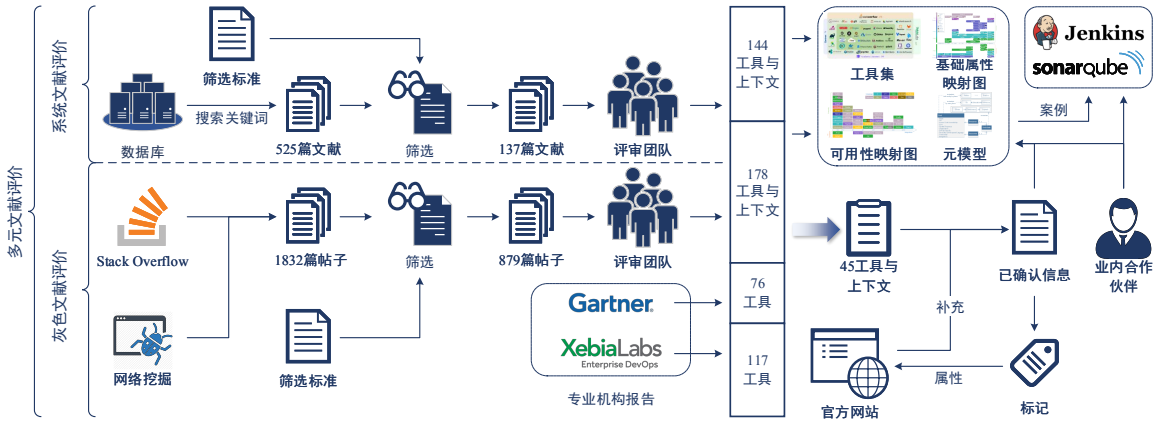


Fig 1 Research method.

图一 研究方法

2.2 系统化文献评价

系统化文献评价（Systematic Literature Review, SLR）由 Kitchenham 等人提出，是实证软件工程中最主要的研究方法[21]之一。本研究采用轻量级的系统化文献评价方法[27]，从学术文献中收集与 DevOps 自动化实践相关的证据。

具体来讲，首先使用自动搜索策略在软件工程领域四个主要的电子文献库 (IEEE Xplore, ACM Digital Library, SpringerLink 和 ScienceDirect) 中搜索同时包含 DevOps 和 tool 关键字的文献。为了保证搜索结果的全面性，本研究使用定义的搜索字符串如下所示。其中，需要注意的是“tool*”表示包含 tool 的词，如 tooling, toolset 和 toolchain 文献结果都符合检索的要求。

((DevOps) in title or keyword or abstract AND (tool) in full-text)*

上述搜索字符串在四个电子文献库适配检索后初步得到 525 篇相关文献，然后通过预先定义的筛选标准进对检索到的文献进行筛选，比如选择使用或分析了 DevOps 工具的论文，没有提到特定的 DevOps 工具的论文被排除。最终，该阶段有 137 篇文献被确定为相关文献，每个文献库筛选的相关文献如表 1 所示：

Table 1 The distribution of selected papers in four databases

表一 四个电子文献库筛选的文献分布

文献库	筛选的文献数量
IEEE Xplore	63
ACM Digital Library	29
SpringerLink	15
ScienceDirect	10

在得到了筛选的相关文献后，由三名研究人员对所筛选的相关文献进行数据抽取，抽取的内容包括文献基本信息(论文名称、年份)、工具名称、工具特征、详细程度(仅提及、详细介绍)、使用情况(使用、未使用)、问题与挑战。数据抽取过程中，每人负责一个文献库， SpringerLink 和 ScienceDirect 由于相关文献相对较少，由一名研究人员同时负责。为了尽可能减少误差，每篇经过数据抽取的文献都需要由另一名研究人员独立检查。当遇到分歧时要召开会议进行讨论，直到达成一致意见。该阶段结束后，初步生成了从学术文献中识别的 DevOps 实践自动化支持工具集合。

2.3 灰色文献评价

学术文献中的信息通常会滞后于灰色文献[25]，尤其是面对从技术社区发展并活跃起来的 DevOps，传统的系统化文献评价方法会因为局限于从学术领域发表的文献中客观地收集、整合分析证据，而忽略工业实践领域关于 DevOps 的声音，继而可能会导致大量未在学术领域应用的 DevOps 工具无法被识别。基于以上原因，除了系统化文献评价方法之外，本研究还采用了灰色文献评价（Grey Literature Review, GLR）方法来收集证据，补充对 DevOps 自动化实践的理解。

2.3.1 Stack Overflow

为了更好地倾听来自业界的声音，本研究以 Stack Overflow 论坛作为主要的灰色文献数据源。作为一种基于社区的问答网站（Community based Question Answering, CQA)，Stack Overflow 是目前最流行的软件开发者 CQA 网站，其主题主要与软件开发相关，除了其目标人群程序员外，Stack Overflow 还吸引了大批的研究者[26]。本研究借助 Web 挖掘技术在该论坛上使用字符串——"*DevOps AND tool**"搜索并自动筛选出包含 DevOps 和 tool 及其变型关键词的问题帖，检索到了 1832 篇问题帖中，通过人工识别并选择后，最终确定了 879 篇讨论了具体的 DevOps 工具的问题帖。所选择的问题帖平均分配给三名科研人员进行数据抽取，抽取的的内容包括问题帖基本信息(链接、问题标题、问题内容)、 回答基本信息(点赞数、回答内容)、工具信息(工具名称、功能、优缺点、与其他工具之间的关系) 等。其中抽取是点赞数是为了以计分的方式对工具相关信息进行质量评估。与系统化文献评价方法类似，每人负责的问题帖必须由另一名科研人员进行核对，检查抽取的信息是否有误，在该阶段定期同经验更为丰富的研究人员召开会议，消除数据抽取过程中产生的分歧。最后生成从 Stack Overflow 识别到的 DevOps 工具集合。

2.3.2 专业报告

为了避免错过任何重要的 DevOps 工具，同时也为了对已经从学术文献和论坛识别出的工具进行确认，我们从已发布的相关专业报告，即 Gartner 报告和 XebiaLabs 报告，确定了另外两个 DevOps 工具集合。这两个机构是业界最有有影响力的研究和分析组织，所发布的工具集合是迄今为止最全面的。

最终，本研究获得了来自学术文献、Stack Overflow 论坛、Gartner 报告、XiebiaLabs 报告的四个 DevOps 工具集，涉及到了将近三百个与 DevOps 相关的工具。 为了控制当前研究的范围、复杂性以及工作量，研究人员将前面所识别的这四个工具集进行了比较，并筛选出至少在三个工具集中出现的工具，以此来保证所选的工具集能够反映学术领域研究人员和产业界实践者的共同兴趣。最后将所选的工具作为 DevOps 自动化支持工具的典型工具集，以进行更有针对性的研究。

2.3.3 官方网站

为了识别所选 DevOps 工具之间的复杂关系，研究人员进一步从所选工具的官方网站上发布的内容中寻找并充分挖掘更多信息来作为补充，同时也是对之前抽取的数据的进一步确认。两名研究人员分别选择了其中五个工具浏览其官网发布的内容并进行标记，通过比对各自整理的标记内容，识别出了 DevOps 工具的共同属性并对其进行分类，这些属性可能能够揭示工具之间的关系，例如协作关系、可替代关系（这将在第 3 节详细介绍）。对于前面所选择的典型工具集中所包含的所有工具，本研究针对特定的属性从其官方网站收集相应的证据并进行整合分析。

2.4 案例研究

本研究采用多元文献评价的方法是为了获得对 DevOps 工具的静态理解，例如 DevOps 工具属性以及之间的关系。为了进一步探索 DevOps 工具如何不断发展演化以实现 DevOps 的目标和价值，本研究选择了其中的两个代表性工具案例来进行分析。基于多元文献评价所获得的支持证据，本研究从目标工具案例的官方网站以及业界的合作伙伴处收集了补充信息，例如与 DevOps 目标的典型特征相对应的演变过程。通过不断地对比分析所收集的数据，本研究从中确定了工具实现 DevOps 目标价值的不同方式。

3 DevOps 工具图谱

为了帮助实践者更好地理解 DevOps 自动化支持工具的现状，本研究选择了具有代表性的 DevOps 工具集，根据相关数据的分析整合进行了 DevOps 工具全貌图谱描绘。本研究首先给出了 DevOps 工具元模型，以帮助阐明在图谱中呈现的 DevOps 工具的属性和关系；然后基于相关数据绘制了工具及其属性之间的全貌映射图谱，为实践者提供 DevOps 工具比较和选择的初步参考。

3.1 DevOps工具集

如图一所示，本研究通过系统化文献评价的方法学术文献中识别出了 144 种 DevOps 工具，通过灰色文献评价的方法分别识别出了来自 Stack Overflow 的 178 种、XebiaLabs 的 117 种和来自 Gartner 的 76 种工具。在删除了重复的工具之后，最初获得了大约 300 个支持 DevOps 的自动化工具，面对如此庞大的工具集，很难去梳理其之间的关系。 因此，为了控制研究的范围，同时确保本研究的代表性，最终选择了至少在三个集合中出现的 DevOps 工具（图二）。如图二所示，所选的工具集包括了 45 个 DevOps 工具，其中 21 个工具在四个候选工具集合中均有出现（图中心）， 剩余的 24 个在其中的三个集合中出现过（图外围）。需要注意的是，本研究后续的所有分析都是基于图二所选的 DevOps 工具集。

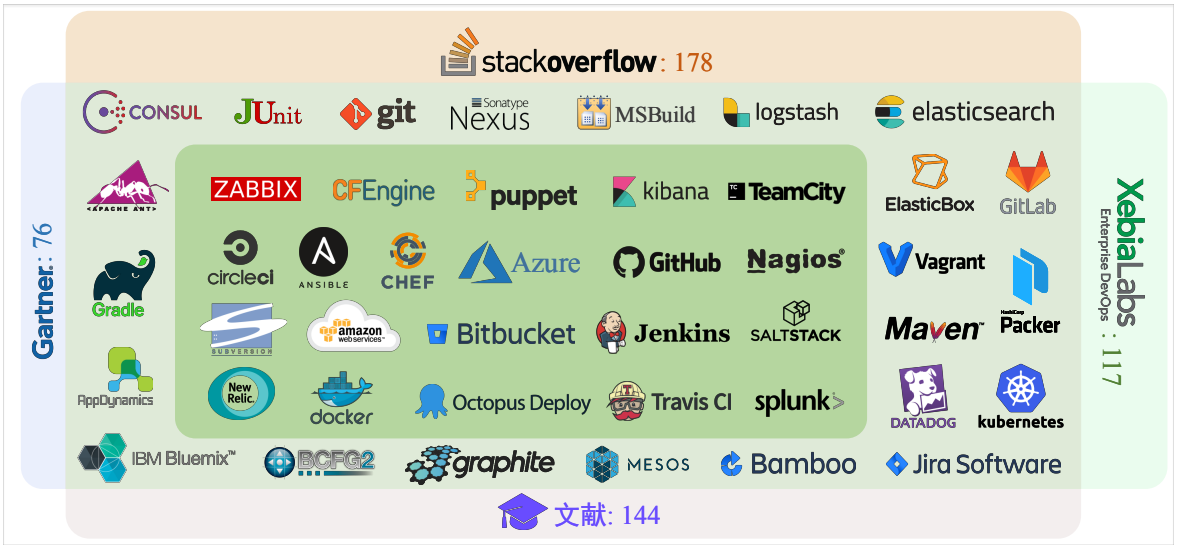


Fig 2 A selective set of DevOps tools.

图二 所选的 DevOps 工具集

3.2 DevOps工具关系元模型

在确定了要深入分析的 DevOps 工具集（参见图二）后，针对学术文献和 Stack Overflow 所获取的数据和工具上下文，本研究还使用了从工具官方网站上获取的数据来对其进行确认（如第 2 节所述），然后通过标签化的方式对工具相关的数据进行分析，从而识别其共同属性及之间关系。最后，根据属性的分类结果及关系

数据，本研究设计了元模型，用于描述并帮助实践者理解具有不同属性的 DevOps 工具之间的关系 （图三）。

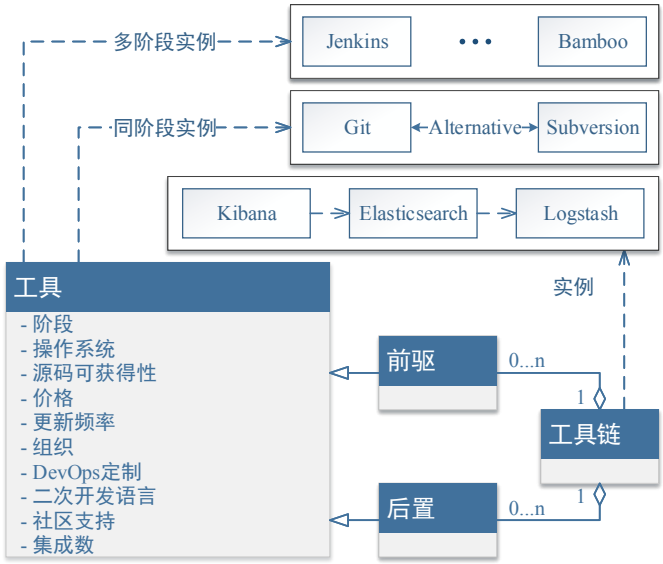


Fig 3 A metamodel of DevOps tools.

图三 DevOps 工具元模型

3.2.1 基类

在所设计的元模型中，以“工具（Tool）”作为基类，该基类抽象出十个属性，如图三所示。 以阶段（Phase）属性为例，它表示工具支持 DevOps 实践的哪个阶段。 阶段属性的取值以及其他九个属性的含义将在 3.3 节中详细说明。

在该元模型中，工具基类可以通过具有特定属性的相应工具进行实例化。比如，对于阶段属性来说，所选择的工具集合中有的工具可以支持多个阶段，例如支持三个阶段（构建，测试和部署）的持续集成工具 Jenkins，Bamboo 等；支持两个阶段（部署和配置）的 Ansible 和 Chef 等。工具基类与其实例之间存在一对多的关系。

3.2.2 工具链（Toolchain）

DevOps 主张打破开发团队和运维团队之间的壁垒，并实现整个交付流水线的自动化，那么工具不可避免地需要相互协作。本研究从所选工具集中发现了一种典型的协作关系——工具链，即可以由工具基类的子类前驱工具（Precursor） 和后置工具（Successor）相互合作，共同实现 DevOps 某个甚至多个阶段的工作。在协作关系中，前驱工具与后置工具之间存在特定的关联，本研究从证据中识别出了两种类型的关联：

- 数据关联

工具链中的协作工具可以通过不同类型的数据相互关联，例如请求或分析结果。数据可以在协作工具之间自动或通过媒介（人或某些中间工具）传输。比较典型的一个工具链的实例是 Kibana、Elasticsearch 和 Logstash。搜索和分析引擎 Elasticsearch 接受由数据处理管道 Logstash 提取、转换和发送的数据，然后通过 Kibana 可视化 Elasticsearch 的分析结果。 在该工具链中，Elasticsearch 不仅是 Logstash 的后置工具，也是 Kibana 的前驱工具。

- 制品关联

制品是用于连接相互写作的工具之间特殊数据类型。它可以从前驱工具传输到后置工具或决定某个 DevOps 工具的执行，比如测试工具 JUnit 对构建工具（如 Maven）构建好的应用程序进行测试；又比如通过所有测试的应用程序将由部署相关的工具（如 Octopus Deploy）进行部署。此时连接构建、测试、部署工具的制品分别是已构建和已测试的应用程序。

3.2.3 可替代的工具

可替代工具表示具有相同属性值的一组工具，彼此之间可以相互替代。对于特定的阶段属性，有多种工具可供选择，例如，Git，GitLab，GitHub 和 Subversion 在源码管理（SCM）阶段可以相互替代。 实践者可以根据自己的需求通过限制十个属性的值来过滤选择可替代的工具。

需要注意的是，所设计的元模型是可以随着 DevOps 工具的发展而不断来更新扩展，例如添加新的新属性或工具之间新的关系。此外，本研究基于现有的证据构建了 DevOps 工具属性及关系图谱（详见 3.3 节），后续可以将元模型中添加的工具实例持续映射到关系图谱中。

3.3 DevOps工具及属性映射图谱

基于 3.2 节给出的工具关系元模型，本研究对 3.1 节展示的所有 DevOps 工具建立了工具及其属性关系的全貌映射图谱。所建立的全貌映射图谱包含了本研究识别出的十个属性维度（3.3.1 节和 3.3.2 节），本研究根据属性维度的不同特点将其分为两类——基本属性和可用性属性，并针对两类属性分别创建映射图谱。需要说明的是，目前创建的全貌映射图谱只能从这十个维度为实践者提供 DevOps 工具比较与选择时的初步参考。

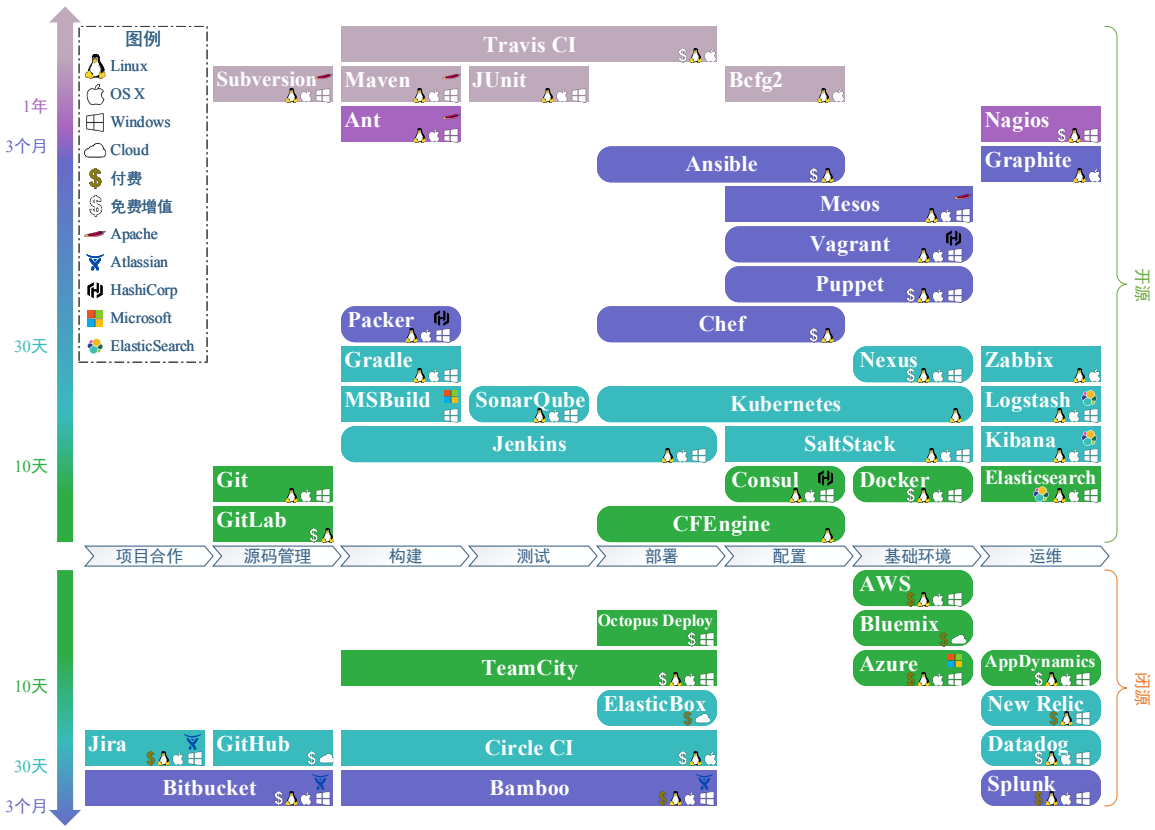


Fig 4 A landscape of DevOps tools in basic attribute dimensions.

图四 DevOps 工具基础属性全貌图谱

3.3.1 基础属性

基本属性是与 DevOps 工具的基本信息相关的属性，如图四所示，包括阶段（Phase），更新频率（Update Frequency），源代码可获得性（Source Code Accessibility），价格花费（Cost），操作系统（OS），所属组织（Organization）和 DevOps 定制（DevOps specific）。在所创建的基础属性图谱中，每个 DevOps 工具都由一个带有工具名称的矩形表示，上述的七个基本属性在图谱中有各自的表示形式。

平均更新频率通过左侧纵轴上五种不同的颜色来表示五个级别的更新频率；源代码可获得性即 DevOps 工

具是否为开源，该属性在实践者进行二次开发或考虑安全性相关的问题时会关注，代码是否开源通过右侧的纵轴来区分，其中正轴和负轴分别对应于开源和不开源；价格花费，支持的操作系统以及所属的组织由相应的图标显示，如图四图例所示。最后其余的两个属性，即阶段和 DevOps 定制，可能对于实践者来讲比较难理解，因此接下来对其进行详细解释。

- 阶段

本文所创建的基础属性映射图谱中，阶段属性的值是粗粒度的，以描述工具在 DevOps 中的主要作用，而不是精确显示它支持的详细子阶段。横轴表示支持 DevOps 工具的关键阶段。 我们对有关阶段属性的证据进行了分类，并获得了从开发到运维的七个阶段，例如构建，测试，部署和运维。 如图四所示，每个阶段可以通过多个 DevOps 工具相应地实现，一些 DevOps 工具也可以支持多个阶段，例如，Jenkins 和 TeamCity 都支持三个阶段，包括构建，测试和部署。

- DevOps 定制

此属性表示工具是否具有 DevOps 实践中不可或缺的功能或能力。 通常，该属性可以通过工具官网是否明确声明与 DevOps 的关系来确定。当然，不可否认没有此属性的工具也可能有助于 DevOps 实践，但是它可以作为一个很重要的标志，来明确表明工具本身是否是 DevOps 的支持者或工具可以为 DevOps 提供什么样的支持。如图四所示，本研究使用不同形状的矩形来帮助轻松区分该属性，其中 DevOps 定制的工具通过圆角矩形来表示，例如 Jenkins、kubernetes、AWS 和 Datadog，而除此之外的其他工具则属于非 DevOps 定制，未在官网中明确表示与 DevOps 的关系，如 Travis CI 和 Circle CI。

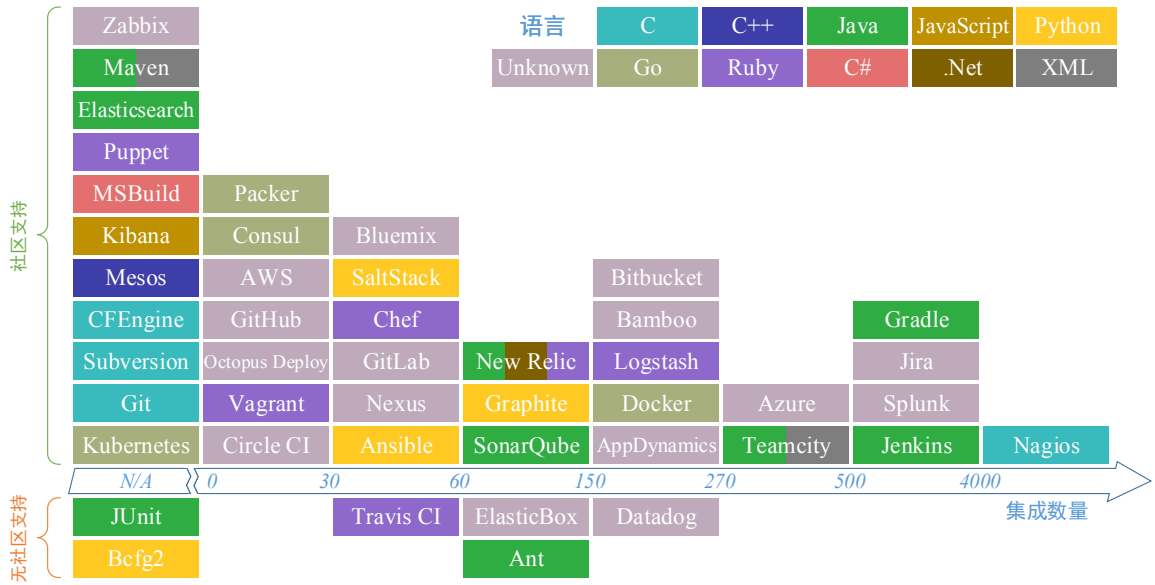


Fig 5 The mapping between DevOps tools and the usability attributes.

图五 DevOps 工具可用性属性映射图谱

3.3.2 可用性相关属性

本研究使用可用性来表示 DevOps 初学者在遇到问题时是可以方便地获取信息或帮助，或者在需要定制时如何方便地改造工具。从确定的十个属性中，本研究识别出三个可以组合在一起来判断 DevOps 工具的可用性的属性，即插件集成数，社区支持和二次开发语言。我们将归类为可用性相关的属性，并创建了如图五所示的 DevOps 工具可用性属性映射图谱。所识别的 DevOps 工具位于图谱中央区域，三个属性的解释如下：

- 插件集成数

插件集成数是衡量特定 DevOps 工具是否可以或易于和其他工具集成使用的重要属性。横轴代表不同的插件集成数量，其中“N/A”代表插件集成数量信息不详。在所选择的 45 个 DevOps 工具中，Nagios 集成了超过 4000 个插件，是所选工具中插件集成数量最多的。

- 社区支持

通常，具有社区支持的工具意味着它拥有足够丰富的文档支持和来自官方团队以及其他用户的强大技术支持。本研究使用左侧纵轴分割的上下区域分别表示 DevOps 工具是否具有社区支持，具有社区支持的工具对应于正轴。从图中可以看出，大约 13%（6/45）的 DevOps 工具是没有足够的社区支持的。

- 二次开发语言

在使用 DevOps 工具进行二次开发时，此属性至关重要，它反映了工具的可扩展性。如图五所示，该属性通过工具矩形中不同的颜色来表示。DevOps 工具矩形中填充了多种颜色意味着它可以支持不同的二次开发语言，例如，如图所示 TeamCity 和 Maven 都支持 Java 和 XML。

需要说明的一点是，本研究所创建的全貌映射图谱（图四和图五）支持实践者根据其需求从十个维度对 DevOps 工具进行初步地比较和筛选，图谱中所包含的十个属性维度没有明确的顺序之分。另外，除了这十个属性，其他可能存在的属性超出了现阶段本研究的范围，本文中不对其进行讨论。

4 案例研究

作为当前所选工具集的快照，第 3 节中的元模型和全貌映射图提供了所选 DevOps 工具的常见属性和关系的分类，比较以及分析结果的展示。但是，它们无法帮助理解特定工具是如何动态发展并支持 DevOps 实践的目标。因此，本研究进一步从工具集中选择了两个代表性工具，Jenkins 和 SonarQube，作为案例进行深入分析。之所以选择这两个工具案例是因为他们在学术领域和工具界中被研究和使用的频率均相对较高。这两个工具在实践中被广泛用于 DevOps 的不同场景，并且它们支持 DevOps 目标的方式大有不同，例如，Jenkins 用于提高部署频率，而 SonarQube 用于产品的质量保证。此外，本研究所选的两个案例（工具）的代表性也得到了业界合作伙伴证实。

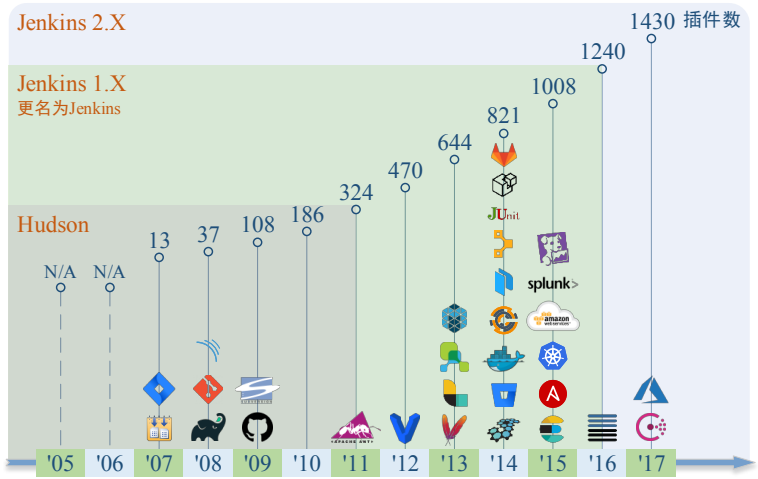
4.1 Jenkins

Jenkins 作为一个持续集成工具，诞生于 2005 年，迄今已经存在了 12 年[28]。目前，拥有 1400 多个插件的 Jenkins 已经成为业界领先的持续集成和持续交付的开源平台[29]。图六展示了过去 12 年里 Jenkins 插件演变和急剧增长的过程。由于无法检索到确切的信息，我们将前两年的插件数量标记为“N/A”。同时图六也展示了 Jenkins 开始为工具集中的 DevOps 工具提供插件的时间点。

4.1.1 演化过程

Jenkins 的整个演化过程可分为三个阶段，如图六所示。

- **Hudson**：在 2005 年，Jenkins 首次作为 Hudson 项目发布[28]。它可以支持当时不连续的前沿实践，例如，在每次提交时构建和测试，自动捕获更改日志和分析测试报告等。作为 CruiseControl 以及其他开源构建服务器的更好替代方案，Jenkins 逐渐被人们所熟知[29]。
- **Jenkins 1.x**： 2011 年 1 月，社区投票将项目从“Hudson”更名为“Jenkins”。一个月后，Oracle 表示他们打算继续开发 Hudson，并认为 Jenkins 是一个分支而不是更名。同月，Jenkins 的第一个版本发布了[31]。从此以后，Jenkins 和 Hudson 作为两个独立的项目继续发展，然而，Jenkins 更受欢迎。直到 2016 年 4 月，Jenkins 1.x 拥有近 1100 个插件并获得了超过 100K 的安装数量[30]。
- **Jenkins 2.x**：Jenkins 发布了重要版本 2.0，这个版本默认启用了 Pipeline 插件，目的是在持续交付中占有一席之地，让开发者能够在软件开发和运维过程中做他们想做的事情[32]。在此版本中诞生了一些新功能，例如，构建持续部署流水线，持续部署到服务器，利用容器，在 Pull 请求合并之前进行测试，这都符合 DevOps 的目标。



■ Fig 6 Evolution of Jenkins.

■ 图六 Jenkins 演化过程

4.1.2 支持 DevOps 目标的方式

DevOps 旨在打破开发和运维之间的壁垒，实现高质量的“更快的持续部署”。为了实现这一目标，团队需要以一种灵活的方式来模拟、协调和可视化他们的整个交付流水线。随着新功能的诞生，如内置交付流水线以及日益增多的插件[33]，Jenkins 2.x 逐渐展现了实现 DevOps 目标的愿望和能力。

1) **内置交付流水线**：Jenkins 2.0 借助 Pipeline 插件为交付流水线提供支持，具体流程如图七所示。Pipeline 插件允许开发者使用领域特定语言（DSL）编写 Jenkins 文件，将其软件交付流水线建模为代码，把诸如构建，测试和部署之类的作业链接在一起，使创建流水线变得更加容易和快捷。通过持续交付流水线，开发团队和运维团队可以在更高的部署频率下和更短的交付时间内协同工作[34]。

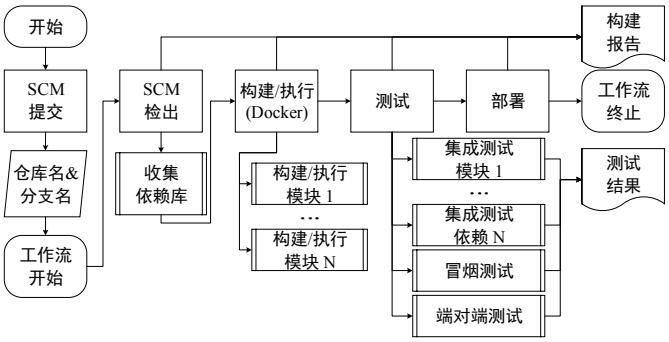


Fig 7 Jenkins delivery pipeline.

图七 Jenkins 持续交付流水线

2) **大量可持续交付插件**：除了 Pipeline 插件之外，社区贡献的 1400 多个 Jenkins 插件协同实现了持续交付中的所有流程，包括 SCM，构建，部署和自动化项目。最近，Jenkins 可以支持像 Git，GitLab 和 GitHub 这样的 SCM。每个分支都可以定义自己的交付流水线，Jenkins 可以检测到这些流水线，自动管理何时创建或删除分支。在 SCM 检出之后，Jenkins 可以协调应用程序的构建过程，其中源代码将被组装，编译或打包。Jenkins 提供了几乎所有常用的构建工具的插件，例如 Ant，Packer，Gradle 和 MSBuild。自动化测试是成功持续交付流程中的关键步骤。为了帮助提高测试效率，Jenkins 发布了许多插件，可以产生测试报告并在可视化界面中展示，例如 JUnit 插件。发布流水线的最后

一步是将构建的制品发布到 Artifactory 服务器或将代码推送到生产系统。Jenkins 插件库也收录了许多热门工具的插件，如 Docker，Kubernetes 和 Azure。

4.1.3 可替代工具

由于 Jenkins 的成功和受欢迎程度，一些同类型的 CI/CD 工具也随之诞生。他们也应用了和 Jenkins 类似的逻辑，即通过将 DevOps 任务链接在一起来实现持续交付的流水线并缩短项目的交付周期。

同类 CI/CD 工具分为两类[35]：第一类是以 Bamboo 和 TeamCity 为代表的工具，这类工具主要依靠其的 WebUI 来完成整个流程，例如创建工作；而以 Travis CI 和 Circle CI 为代表的第二类工具则通过自动发现项目类型或仅需要指令文件中的几行代码（例如 travis.yml 或 circle.yml）来提高维护效率。但是，第二类 CI/CD 工具仅在简单的流水线或小型项目中有效，其他情况下则会失去灵活性，因为实际交付流水线通常比这些工具提供的功能更复杂。相比之下，除了 WebUI 之外，Jenkins 允许用户在 Jenkinsfile 中定义流水线，从而提供更灵活，更有效的方法。凭借强大的社区支持和丰富的插件库，可以更灵活地与其他工具协作，这使得 Jenkins 仍然是大型或小型项目中最受欢迎和广泛使用的 CI/CD 工具。

4.2 SonarQube

SonarQube 是一个用于持续检查代码质量，执行自动代码分析，检测不同编程语言中错误、代码异味和安全漏洞的开源平台，此外提供记录指标历史并生成演化图的功能。目前，SonarQube 已成为代码质量持续检测的领先产品。

4.2.1 演化过程

SonarQube 演化过程中的里程碑可以追溯到 2007 年，为了使持续检测成为现实，SonarQube 的创始人梦想每个开发人员都有能力衡量其项目的代码质量。他们认为持续检测必将成为持续集成的主流[36]，因为在理想情况下，无论是可用版本，候选发布版还是里程碑版本，持续集成都可以随时以最小的风险触发任何类型的发布。高质量将成为必不可少的而非尽量达到的。不幸的是，许多工具仍然无法确保总体质量。像 Jenkins 这样的持续集成工具通常在特定工具的帮助下执行测试，例如 JUnit，FindBugs。这些测试工具分别用于不同的目的，例如单元测试，代码检查和覆盖测试。这些工具只是单纯生成测试报告，而持续集成工具也只是将不同的报告放在一起而不进行二次处理。但是，从宏观角度来看，没有一个系统的方法来衡量整个项目的源码质量。更通俗来说，重要的是要控制整体技术债务，只让它可控的增加，这就是持续检测的概念[37]。

受 DevOps 影响，SonarQube 目前已经发展成为一个以持续的方式保证源代码质量的工具。它不仅是持续集成的主流，也是 DevOps 自动化的重要组成部分。SonarQube 拥有同其他 DevOps 阶段工具协作的能力：

1）支持众多源配置管理工具，如 Git 和 Subversion;

2）内置集成大多数构建工具，在大多数情况下可以实现零配置;

3）集成 Jenkins，TeamCity 和 Bamboo 等持续集成工具[36]。 同时，SonarQube 一直致力于持续检测，以实现数据收集自动化，指标报告并高亮热点和缺陷[38]。

4.2.2 支持 DevOps 目标的方式

尽管 Jenkins 和 SonarQube 都是 DevOps 工具，但它们集成其他工具的方式却截然不同。 Jenkins 将其他工具集成到一个流水线中，通过工具链连接其他工具，但不处理它们生成的数据。例如，Jenkins 可以将 GitHub，Maven 和 JUnit 集成到流水线中。Maven 使用 GitHub 提取的源代码，然后 JUnit 根据 Maven 生成的制品创建测试报告。而 Jenkins 只是使用脚本定义它们的执行流程。

与 Jenkins 不同，SonarQube 提出了“一体化”概念，强调整体质量[39]：

- **统一指标**：多种测试工具可用于不同的测试目的，然而，通过不同工具生成的测试报告对代码质量进行统一的分析并非易事。虽然集成了不同的测试工具，但 SonarQube 提供了查看和度量代码质量的集中视图。
- **统一质量阈值**：质量阈值定义了源代码符合预定义质量要求的条件，SonarQube 提供开箱即用的默认质量阈值，以简单的方式表示源代码的健康状况。

- **语言一致性**：当从一种语言切换到另一种语言时，用户体验保持一致，使用相同的方式来描述和标记规则，报告问题，评估修复工作量。此外，当模式通用的时候，它可以自定义模式规则。

通常，SonarQube 不会简单的直接集成其他测试工具。它会先处理被集成工具生成的原始数据，然后统一二次处理这些数据，以提供整体代码健康报告。例如，由不同语言分析器创建的报告可以统一为一个错误报告和一个覆盖报告。错误数量和覆盖百分比可能成为质量阈值的组成部分，这反映了源代码的总体质量。

5 讨论

本节主要讨论了研究人员在整合数据以及生成该报告过程中的发现以及本研究存在的局限性。

5.1 发现1：DevOps自动化支持工具生态系统呈爆炸趋势

Jenkins 的案例说明了 DevOps 工具生态系统的规模在不断扩大，且扩展趋势显著。本研究从 Stack Overflow 的相关数据中发现了大量的 Jenkins 插件，并对 Jenkins 的插件进行了搜索。图六显示了近年来插件的快速增长情况，预计其将会发布更多的插件来支持持续增多的 DevOps 工具。

尽管越来越多的工具有利于支持 DevOps 实践，但 DevOps 相关工具和插件的数量爆炸可能带来一些副作用，例如维护的复杂性。比如，不只一个 Jenkins 的用户曾经抱怨过，“插件太多，如果发生崩溃，很难维护并从历史中恢复，而且插件的支持文档少之甚少”[42]。 幸运的是，对于新用户，Jenkins 2.0 启动了建议的插件，以帮助他们从选择正确的工具集开始来有效地使用 Jenkins。

5.2 发现2：DevOps工具个体向DevOps目标的支持不断发展演化

在整个 DevOps 工具生态系统不断扩张的背景之下，DevOps 工具个体也在向着 DevOps 目标的支持不断发展演化，这一点在第 4 节中的案例研究中得到了很好的证实。两个案例集成其他工具的差异反映了目前工具个体实现和支持 DevOps 目标的两种主要方式：

1) 扩展功能的多样性，例如 Jenkins 旨在为了建立一个全功能流水线而集成尽可能多的工具;
2) 加强特定的功能，正如 SonarQube 案例中，其功能仅仅是测试，但其所有的发展都在为自身不断加强该功能而努力[40,41]，而非纳入其他的工具；另外一个不断发展并尽可能囊括更多功能的工具案例是 Bitbucket。Bitbucket 近期推出了自己的流水线平台以更好地支持 DevOps [43]。但与 Jenkins 通过不断添加新插件支持的方式不同，Bitbucket 体现了原本只支持某些工具的工具，本身演化发展成为流水线工具、而非通过插件支持更多功能的趋势。

5.3 本研究局限性

本研究通过系统化文献评价方法以及灰色文献评价方法识别出大量的 DevOps 自动化支持工具，鉴于其之间存在的关系复杂且难以描绘，因此为了降低复杂性并保证研究的可行性，研究人员最终选择了学术界和工业界共同感兴趣的工具来进行分析。这样就导致本研究的一个可能的局限是所选择的 DevOps 支持工具集合不完整。为确保所选工具集的代表性和受欢迎程度，本研究将学术文献以及 Stack Overflow 所识别的工具，与该领域专业咨询机构给出的工具集进行对比筛选（参见 2.3 节）。此外，最终选择的 DevOps 工具也和本研究的行业合作伙伴进行了确认。

为了尽可能减少研究人员在数据抽取过程中产生的潜在误差，在多元文献评价过程中，所有数据在经过一名研究人员抽取之后，会由另一名研究人员独立进行检验。任何分歧都会和更高级的研究人员以及领域专家通过定期会议或电子邮件进行讨论，直至问题解决。

本研究选择了两个代表性的工具作为案例来探究工具在 DevOps 目标上的不同实现策略，但是对于它们的选择不是随机的。尽管其代表性已经从灰色文献评价获取的数据以及行业合作伙伴处得到了印证，但是选择这两个工具之后可能会限制对于其他工具的理解，导致本文的讨论不完全适用于其他工具。

6 总结

在 DevOps 已被全球产业界接受、DevOps 工具生态系统快速增长的背景之下，DevOps 的定义和范围无论在学术界还是产业界都仍然模糊不清[44]，这可能会给实践者在理解和选择 DevOps 自动化工具的过程中造成困扰。由于自动化被认为是实现 DevOps 目标的关键方法，如果没有工具的支持，旨在实现持续高效高质量交付的 DveOPs 是无法实现的。因此，为了帮助实践者梳理 DevOps 工具之间的关系，本研究采用多元文献评价方法（系统化文献评价和灰色文献评价）方法获取相关证据并进行整合分析，突破性地构建了 DevOps 工具全貌映射图谱。所采用的灰色文献评价方法的数据来源主要是业内流行的论坛（Stack Overflow）、业内专业机构的技术报告以及工具的官方网站。作为 DevOps 全貌图谱的补充，本研究还选择了两个代表性工具来进一步探究特定工具的演变过程以及其实现 DevOps 实践目标的方式。本研究使用不同的研究方法可以实现方法与数据的交叉核验，以确保本研究的质量以及研究结果的可信度。本文报告了现阶段与行业合作伙伴合作而得到的关于 DevOps 自动化支持工具的研究成果。

本研究的产出主要包括四点：首先根据学术界和工业界的共同兴趣以及工具的受欢迎程度，确定了具有代表性的 DevOps 自动化支持工具集合；继而通过对数据进行整合分析，抽象出一个元模型来描述 DevOps 工具的基本属性以及它们之间存在的的重要关系；然后创建了 DevOps 工具和不同属性维度之间映射的全貌图谱，为实践者在工具比较与选择时提供初步参考；最后从选定的 DevOps 工具中挑选了两个代表性案例进行详细分析，以帮助实践者理解相关工具是如何为支持 DevOps 实践而发展演化的。

本研究的局限是所采用的 DevOps 工具集不完整，并且所构建的全貌图谱中涉及的属性仍然是粗粒度的。未来研究将致力于把更多的工具纳入到现有的 DevOps 工具集，描绘更细粒度工具属性和关系的全貌图谱。

参考文献:

[1] B. S. Farroha and D. L. Farroha, “A framework for managing mission needs, compliance, and trust in the DevOps environment,” in Proceedings of the 2014 IEEE Military Communications and Information Systems Conference (MilCIS 2014). IEEE Computer Society, 6-8 October 2014, pp. 288–293.

[2] M. Virmani, “Understanding DevOps & bridging the gap from continuous integration to continuous delivery,” in Proceedings of the 5th International Conference on the Innovative Computing Technology (INTECH2015). IEEE Press, 20-22 May 2015, pp. 78–82.

[3] B. Boehm, “Making a difference in the software century,” Computer, vol. 41, no. 3, pp. 32–38, Mar. 2008.

[4] K. Kaur, A. Jajoo, and Manisha, “Applying agile methodologies in industry projects: Benefits and challenges,” in Proceedings of the 1st International Conference on Computing Communication Control and Automation (ICCUBEA2015). IEEE Computer Society,26-27February 2015, pp. 832–836.

[5] M. Olszewska and M. Walde´n, “DevOps meets formal modelling in high-criticality complex systems,” in Proceedings of the 1st International Workshop on Quality-Aware DevOps (QUDOS 2015). ACM Press, 1 September 2015, pp. 7–12.

[6] L. E. Lwakatare, P. Kuvaja, and M. Oivo, “Dimensions of DevOps,” in Proceedings of the 16th International Conference on Agile Processes in Software Engineering and Extreme Programming (XP 2015). Springer International Publishing, 25-29 May 2015, pp. 212–217.

[7] J. Waller, N. C. Ehmke, and W. Hasselbring, “Including performance benchmarks into continuous integration to enable DevOps,” ACM SIG- SOFT Software Engineering Notes, vol. 40, no. 2, pp. 1–4, Apr. 2015.

[8] F. Erich, C. Amrit, and M. Daneva, “A mapping study on cooperation between information system development and operations,” in Proceedings of the 15th International Conference on Product-Focused Software Process Improvement (PROFES 2014). Springer International Publishing, 10-12 December 2014, pp. 277–280.

[9] M. Hu¨ttermann, DevOps for Developers, ser. Expert’s Voice in Web Development. New York: Apress, 11 September 2012.

[10] E. Mueller, J. Wickett, K. Gaekwad, and P. Karayanev. (2016, Dec.) What is DevOps? [Online]. Available: https://theagileadmin.com/what- is- devops/

[11] R. Naegle. (2016, 6 July) Hype cycle for I&O automation, 2016. [Online]. Available: https://www.gartner.com/doc/3365831/hype-cycle- io- automation-

[12] M. Loukides, What is DevOps? O'Reilly Media, Jun. 2012.

[13] S. K. Bang, S. Chung, Y. Choh, and M. Dupuis, "A grounded theory analysis of modern Web applications: Knowledge, skills, and abilities for DevOps," in Proceedings of the 2nd Annual Conference on Research in Information Technology (RIIT 2013). ACM Press, 10-12 October 2013, pp. 61–62.

[14] D. G. Feitelson, E. Frachtenberg, and K. L. Beck, "Development and deployment at Facebook," IEEE Internet Computing, vol. 17, no. 4, pp. 8–17, Jul. 2013.

[15] CA. (2014, Oct.) DevOps: The worst kept secret towinning in the application economy. [Online]. Avail- able: https://www.ca.com/us/modern- software- factory/content/devops- the-worst-kept-secret-to-winning-in-the-application-economy.html

[16] Puppet. (2017) 2017 state of DevOps report. [Online]. Available: https://puppet.com/resources/whitepaper/state-of-devops-report

[17] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," IEEE Software, vol. 33, no. 3, pp. 94–100, May 2016.

[18] M. Callananand W. Alexandra Spillane,"DevOps: Making it easy to do the right thing," IEEE Software, vol. 33, no. 3, pp. 53–59, May 2016.

[19] M. Kersten, "A cambrian explosion of devops tools," IEEE Software, vol. 35, no. 2, pp. 14–17, March 2018.

[20] A. Dyck, R. Penners, and H. Lichter, "Towards definitions for release engineering and DevOps," in Proceedings of the 3rd International Workshop on Release Engineering (RELENG 2015). Florence, Italy: IEEE Press, 19 May 2015, pp. 3–3.

[21] B. B. N. de Franc ̧a, H. Jeronimo, Junior, and G. H. Travassos, "Characterizing DevOps by hearing multiple voices," in Proceedings of the 30th Brazilian Symposium on Software Engineering (SBES 2016). Maringa ́, Brazil: ACM Press, 19-23 September 2016, pp. 53–62.

[22] L. Zhu, L. Bass, and G. Champlin-Scharff, "DevOps and its practices," IEEE Software, vol. 33, no. 3, pp. 32–34, May 2016.

[23] R. Vaasanthi, V. P. Kumari, and S. P. Kingston, "Analysis of devops tools using the traditional data mining techniques," International Journal of Computer Applications, vol. 161, no. 11, 2017.

[24] G. B. Ghantous and A. Gill, "Devops: Concepts, practices, tools, benefits and challenges," 2017.

[25] V. Garousi, M. Felderer, and M. V. Ma ̈ntyla ̈, "The need for multivocal literature reviews in software engineering: Complementing systematic literature reviews with grey literature," in Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering (EASE 2016). Limerick, Ireland: ACM Press, Jun. 2016, pp. 26:1–26:6.

[26] D. Correa, A. Sureka. "Chaff from the wheat: characteriza- tion and modeling of deleted questions on stack overflow," in Proceedings of the 23rd international conference on World wide web. ACM, 2014: 631-642.

[27] B. A. Kitchenham, T. Dyba ̊, and M. Jørgensen, "Evidence-based software engineering," in Proceedings of the 26th International Conference on Software Engineering (ICSE2004). Edinburgh, UK: IEEE Computer Society, 23-28 May 2004, pp. 273–281.

[28] K. Kawaguchi. (2015, Jul.) Hudson. [Online]. Available: https: //www.java.net//blog/kohsuke/archive/20070514/Hudson%20J1.pdf

[29] Jenkins. (2017) Jenkins plugins index. [Online]. Available: https://plugins.jenkins.io/

[30] K. Kawaguchi. (2015, 25 September) Jenkins 2.0 proposal. [Online]. Available: https://groups.google.com/forum/#!msg/jenkinscidev/vbXK7JJekFw/BlEvO0UxBgAJ

[31] R. T. Croy. (2012, Feb.) Happy birthday Jenkins! [Online]. Available: https://jenkins.io/blog/2012/02/02/happy-birthday-jenkins/

[32] Jenkins. (2016, 27 April) Jenkins 2.0. [Online]. Available: https: //wiki.jenkins.io/display/JENKINS/Jenkins+2.0

[33] ——. (2017, 19 October) Jenkins user handbook. [Online]. Available: https://jenkins.io/user-handbook.pdf

[34] V. Armenise, "Continuous delivery with Jenkins: Jenkins solutions to implement continuous delivery," in Proceedings of the 3rd International Workshop on Release Engineering (RELENG 2015). Florence, Italy: IEEE Press, 19 May 2015, pp. 24–27.

[35] V. Farcic. (2016, 14 January) The short history of CI/CD tools. [Online]. Available: https://technologyconversations.com/2016/01/14/the-short-history-of-cicd-tools/

[36] K. Beck, "Extreme programming: A humanistic discipline of software development," Fundamental Approaches to Software Engineering, pp. 1–6, 1998.

[37] F. Mallet. (2010, Jun.) Continuous inspection practice emerges with Sonar. [Online]. Available: https://blog.sonarsource.com/continuous-inspection-practice-emerges-with-sonar

[38] SonarSource S.A, Switzerland. (2017) About SonarQube. [Online]. Available: https://www.sonarqube.org/about/

[39] ——. (2017) Centralize quality. [Online]. Available: https://www. sonarqube.org/features/centralization/

[40] F. Mallet. (2011) Differential services in Sonar for a complete support of continuous inspection. [Online]. Avail- able: https://blog.sonarsource.com/differential- services- in- sonar- for- a- complete- support- of- continuous- inspection

[41] ——.(2012)WhatiscomingupforSonarin2012?[Online].Available: https://blog.sonarsource.com/what- is- coming- up- for- sonar- in- 2012- 2

[42] G2Crowd. (2017) Jenkins reviews. [Online]. Available: https://www.g2crowd.com/products/jenkins/reviews?filters% 5Bcomment answer values%5D=&order=most recent&page=4

[43] IBM. (2017) Delivery pipeline. [Online]. Available: https://console.ng. bluemix.net/catalog/services/delivery- pipeline

[44] A. Plancius. (2017) DevOps a definition please? [Online]. Available: https://www.linkedin.com/pulse/devops-definition-please-anne-plancius